CIT237

PROGRAMMING AND ALGORITHMS

NATIONAL OPEN UNIVERSITY OF NIGERIA

FACULTY OF SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

COURSE CODE: CIT237

COURSE TITLE: PROGRAMMING AND ALGORITHMS

CIT237 COURSE GUIDE

COURSE GUIDE

CIT237

PROGRAMMING AND ALGORITHMS

Course Developer Prof. A.S.Sodiya
Dept.of Computer Science
Universityof Agriculture
Abeokuta

Course Reviewer Prof Olumide Longe
American University,
Yola

NATIONALOPENUNIVERSITYOFNIGERIA

National Open University of Nigeria
Headquarters
91, Cadestral Zone,
Nnamdi Azikwe Express Way,
Jabi, Abuja.
Nigeria.

National Open University of Nigeria
Lagos Office
14/16AhmaduBelloWay
VictoriaIsland
Lagos

e-mail: centralinfo@nou.edu.ng
URL: www.nou.edu.ng

Publishedby
National Open University of Nigeria

FirstPrinted 2008

Reprinted 2020

ISBN:ISBN:978-058-581-8

CONTENTS PAGE

Introduction

CIT237– Programming andAlgorithms isathreecreditunitcourseof twenty-oneunits.Thiscoursepresentsan overview ofthemethods and conceptofprogramming andtheroleofalgorithmsinprogramming.It coversaspectsonprogramming conceptssuch as basicdatatypes, algorithms,performanceanalysis,fundamentaldatastructures,P,NP andNP-CompleteProblems andsomesorting algorithms.

Thiscourseisdividedintothreemodules.Thefirstmoduledealswith the basic introduction to the concept of programmingand algorithms; such asdefinitionandcharacteristicsof algorithms,basicdatatypes and fundamental data structures,programdevelopmentlife cycle,typesof programminglanguages,languagetranslators and their characteristics, toolsfor programdesign,etc.

Thesecond modulefocuseson theperformanceanalysisofalgorithms discussing issues such as efficiencyattributes(i.e.timeandspace efficiency), measuring the running time of an algorithm, measuring inputsize,worst-case,best-caseandaverage-caseefficiencies,P,NP andNP-Completeproblems, etc.

Thethirdmodule dealswithsortingand some specialproblems.It introducesyou to some sorting and divide-and-conquer algorithms after which it goeson to discuss some sorting techniques such asMergeSort, Bubble Sort, Selection Sort, etc. giving their algorithms and performance analysis.

Theaimof this courseisto equipyouwiththebasicknowledgeof writingefficientprogramsthroughtheuseof conciseand efficient algorithms.By the end of thecourse,you should beableto confidently tackleanyprogrammingproblembreakingit intoits component parts, write efficient algorithms to solvetheproblem and implementthe algorithmusing anyprogramminglanguageofyour choiceaswellas being abletoevaluate andmeasuretheperformance efficiencyof any algorithm.

ThisCourseGuidegivesyoua briefoverviewof thecoursecontent, courseduration, andcoursematerials.

A courseon computerscanneverbecompletebecauseof theexisting diversitiesofthecomputer systems.Therefore,you are advisedtoread throughthefurtherreadingsto enhancethebasicunderstanding you will acquire fromthecourse material.

WhatYou will Learnin thisCourse

Themainpurposeof thiscourseistointroduceyouto conceptsrelating to problem solving through the efficient use of algorithms and subsequent implementation ofthe algorithminanylanguageof choice that is suitable to the application area. This, weintend to achieve throughthe following:

CourseAims

1. Introduce the basic concepts relating to algorithms and programming;
2. Exposethebasicrelationshipsthatexistbetweenalgorithmsand programdevelopment.
3. Discuss the basic features of algorithms and components of programs.
4. Discussthefundamentaldatastructures,datatypes,arithmetic operations,etc.
5. Discuss features of programming languages, programming methodologies and application areas, language translators, programming environment, etc.
6. Exposethebasicsofmeasuringtheefficienciesofalgorithmsand howtoidentifybasicoperationswithin analgorithm.

CourseObjectives

Certainobjectiveshavebeensetouttoensurethatthecourseachieves itsaims.Apart fromthecourseobjectives, everyunitofthiscoursehas setobjectives.Inthe courseof the study, youwill needtofindout,atthe endofeachunit,ifyouhavemettheobjectivessetatthebeginningof theunit. Bytheendof thiscourseyoushouldbe ableto:

1. Defineanalgorithm,stating itsbasiccharacteristics
2. Enumeratetheroleofanalgorithminproblemsolvingandhowit relatestoaprogram
3. Define the concept of programming and describe the basic featuresof a program;
4. Explaintheprogramdevelopmentlife cycle
5. Discusstheconceptoforderofgrowthandexplainthedifferent asymptoticnotations
6. Operatethehillclimbingtechniqueandshowhowhillclimbing is usedtosolveproblems.
7. ResolvetheKnight’sTourproblem,describeandresolveann*n tour problem
8. Explainthemeasures of algorithm efficiency
9. Explaintheidentification ofbasicoperationswithinanalgorithm

10. Distinguishbetweena polynomialandnon-polynomialproblem
11. Discuss (extensively)P, NP,NP–completeproblems
12. Developalgorithmstoperformsomebasicsorting,suchasMerge Sort,SelectionSsort, BubbleSort,Quick Sort, etc.on some data, and evaluatetheperformanceofeach algorithm.

WorkingThrough ThisCourse

Inorderto haveathoroughunderstandingofthecourseunits,youwill need toreadandunderstandthecontents, and practisethestepsby solvingsome simpleproblemsbybreakingtheminto smallerproblems anddevelopingalgorithms for each. Youmay then implement your algorithmsusinganyprogramminglanguageof your choicethatis suitablefor theapplication area.

Thiscourseisdesignedtocoverapproximatelysixteenweeks,andit willrequire yourdevoted attention.You should dothe exercisesin the Tutor-MarkedAssignments andsubmit toyour tutors.

CourseMaterials

These include:

1. TheCourseGuide
2. StudyUnits
3. Recommended Texts
4. A file for your assignments and for records to monitor your progress.

Study Units

Thereare21studyunitsinthiscourse:

Module1 Introduction toProgramming andAlgorithms

Unit 1 Introduction toProgramming
Unit 2 ProgrammingConcepts
Unit 3 Algorithms
Unit 4 BasicData Types
Unit 5 FundamentalData Structure
Unit 6 Practical Exercise I
Unit 7 FundamentalData Structures
Unit 8 Exercise I

Module2 PerformanceAnalysis of Algorithms

Module3 SortingandSomeSpecial Problems

Textbooksand References

GonnetandRicardoBaeza-Yates(1993). *HandbookofAlgorithmsand DataStructures.*InternationalComputerScienceSeries

Holmes,B.J. (2000). ***PascalProgramming****Continuum*(2nd ed).

www.doc.ic.ac.uk/~wjk/C++Intro/

Levitin,A.(2003).*IntroductiontotheDesign&AnalysisofAlgorithms.* Addison-Wesley.

www.personal.kent.edu/~muhama/Algorithms

Tucker, A.B and Noonan, R.(2006).*Programming Languages – Principles andParadigms*. (2nd ed).McGraw–HillCollege.

Algorithms (2019). Jeff Erickson. 1st Edition, available in Amazon

www.eslearning.algorithm.com

Cormen, T.H., Leiserson,C.F. Rivest, R.L.,Stein,C. (2001). *Introduction toAlgorithms*(2nd ed).Canbridge:M/T Press.

Goodrich, M.T., and Tamassia, R. (2002). *AlgorithmDesign. Foundations,Analysis, andInternetExamples*. NewYork:John Wiley& Sons.

http://mathworld.wolfram.com/QueensProblem.html.

Ahrens, W.(1910).*Mathematische UnterhaltungenundSpiele.* Leipzig,Germany:Teubner,p.381.

Ball,W.W. R. andCoxeter,H. S.M.(1987). *MathematicalRecreationsandEssays(13thed).*NewYork:Dover,pp.175-186,.

Chartrand,G."TheKnight's Tour."§6.2in *IntroductoryGraphTheory* (1985).NewYork:Dover, pp. 133-135.

Conrad,A.;Hindrichs,T.; Morsy, H.;andWegener,I.(1994). "Solution of theKnight's HamiltonianPath ProblemonChessboards." *Discr.Appl. Math.* **50**, 125-134.

de Polignac. *ComtesRendusAcad. Sci.Paris*, Apr. 1861.

de Polignac. *Bull.Soc.Math.de France***9**, 17-24, 1881.

Dudeney, H. E.(1970). *AmusementsinMathematics.*NewYork:Dover, pp.96and102-103.

Elkies, N. D. and Stanley, R.P. "TheMathematicalKnight." *Math. Intell.***25**, No.1,22-34,Winter 2003.

Euler,L. "Solution d'unequestion curieuse quineparoit soumise a aucune analyse."*Mémoiresdel'AcadémieRoyaledesSciences et BellesLettres de Berlin, Année1759***15**, 310-337,1766.

Euler,L. *CommentationesArithmeticae Collectae, Vol.1*.(1849) Leningrad,pp.337-355,.

Friedel, F."The Knight'sTour."

http://www.chessbase.com/columns/column.asp?pid=163.

Gardner, M. (1978)."Knightsof theSquare Table."Ch.14in *MathematicalMagicShow:MorePuzzles,Games,Diversions, IllusionsandOtherMathematicalSleight-of-Mindfrom ScientificAmerican.*NewYork:Vintage, pp. 188-202,.

5

Gardner, M. (1984). *TheSixthBookofMathematicalGamesfromScientificAmerican.*Chicago, IL:University of Chicago Press, pp.98-100.

Guy, R. K.(199)."The nQueensProblem."§C18in *UnsolvedProblemsinNumberTheory,2nded.*NewYork:Springer-Verlag, pp. 133-135.

Algorithms (2019). Jeff Erickson. 1st Edition, available in Amazon

Jelliss,G."Knight'sTour Notes."

http://www.ktn.freeuk.com/.Jelliss,G."ChronologyofKnight'sTours." http://www.ktn.freeuk.com/cc.htm.

Kraitchik, M.(1942)."The Problemof theKnights." Ch.11in *MathematicalRecreations.*NewYork:W. W.Norton, pp. 257-266.

Kyek,O.;Parberry, I.;andWegener,I."BoundsontheNumber of Knight's Tours."*Discr.Appl. Math.* **74**, 171-181,1997.

Lacquière.*Bull. Soc. Math. deFrance* **8**, 82-102and132-158, 1880.

Madachy, J. S.(1970). *Madachy'sMathematicalRecreations.*New York:Dover, pp. 87-89.

Murray, H. J.R. (1902)."TheKnight'sTour, AncientandOriental." *BritishChess Magazine*, pp.1-7.

Pegg, E.Jr."Leapers (ChessKnightsandthe Like)" http://www.mathpuzzle.com/leapers.htm.

Roget,P. M. (1840).*Philos.Mag.* **16**, 305-309.

Rose, C. "TheDistribution of theKnight." http://www.tri.org.au/knightframe.html.

Roth, A. "TheProblemoftheKnight:AFast and SimpleAlgorithm." http://library.wolfram.com/infocenter/MathSource/909/.

Rubin, F. (1974)."ASearch Procedure for Hamilton PathsandCircuits." *J. ACM* **21**, 576-580.

Ruskey, F."Information onthe Knight'sTour Problem."
http://www.theory.csc.uvic.ca/~cos/inf/misc/Knight.html.

Skiena, S. (1990). *ImplementingDiscreteMathematics:CombinatoricsandGraphTheorywithMathematica.*Reading, MA:Addison- Wesley, p. 166.

Sloane, N. J.A. Sequences A001230,A003192/M1369, A006075/M3224, A033996,and A079137in"TheOn-Line EncyclopediaofInteger Sequences."

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

Steinhaus,H.(1999). *MathematicalSnapshots,3rded.*NewYork: Dover,p.30.

Thomasson,D."The Knight'sTour." http://www.borderschess.org/KnightTour.htm.

vander Linde, A.(1874). *GeschichteundLiteraturdesSchachspiels,Vol.2.*Berlin: Springer-Verlag, pp. 101-111.

Vandermonde,A.-T."Remarquessur les Problèmes de Situation." *L'Histoire del'Académiedes Sciences avecles Mémoires, Année 1771.*Paris:Mémoirs,pp.566-574and PlateI, 1774.

Velucchi, M. "Knight'sTour:The UltimateKnight's Tour Pageof Links." http://www.velucchi.it/mathchess/knight.htm.

Volpicelli,P. (1872)."Soluzionecompletae generale, mediantela geometriadisituazione, delproblemarelativo allecorsedel cavallosopra qualunquescacchiere." *Atti della Reale Accad.dei Lincei***25**, 87-162.

Warnsdorff,H. C. (1823). von*Des Rösselsprungs einfachsteund allgemeinste Lösung.* Schmalkalden.

Watkins, J. (2004). *AcrosstheBoard:TheMathematicsofChessboard Problems.*Princeton, NJ: PrincetonUniversityPress.

AssignmentsFile

Theseareof twotypes:OnefortheSelf-AssessmentExercisesandthe other fortheTutor-Marked Assignments.Theself-assessment exercises will enableyou monitoryourperformanceby yourself,while thetutor-markedassignmentswill besupervised. Theassignmentstakeacertain percentageof your total scorein this course.Thetutor-marked assignmentswillbeassessedbyyourtutorwithinaspecifiedperiod.
Theexamination atthe endof this coursewill aim atdeterminingyour level ofmasteryofthesubject matter.Thiscourseincludes21tutor- marked assignments and each must bedone and submitted asstipulated Yourbestscoreshowever,willberecordedforyou.Besuretosend theseassignmentstoyour tutor beforethedeadlineto avoidlossof marks.

PresentationSchedule

ThePresentationScheduleincludedin your coursematerials givesyou the important datesforthe completion of tutormarked assignments and the schedule for attending tutorials. Remember,you arerequired to submitallyourassignmentsbythe duedate. Youshouldguard against laggingbehindinyour work.

Assessment

Therearetwo aspectstotheassessmentofthecourse. First arethetutor markedassignments;second, is awrittenexamination.

Intackling the assignments,you are expected to applythe information andknowledgeyouacquiredduringthiscourse.Theassignmentsmust besubmittedtoyour tutor forformal assessmentinaccordancewith the deadlinesstated intheAssignment File.Theworkyou submitto your tutor for assessmentwill count for 30% of yourtotalcourse mark.

Atthe end ofthe course,youwillneed tosit forafinalthree-hour examination.This will alsocountfor 70% ofyour totalcourse mark.

Tutor-MarkedAssignment

There are 21 tutor-marked assignments in this course. You need to submitallthe assignments.The totalmarksforthebestfour(4) assignments will be30% ofyour totalcourse mark.

Assignmentquestionsfortheunitsinthiscoursearecontainedinthe AssignmentFile. Youshouldbeabletocompleteyourassignments from theinformationandmaterialscontainedinyoursettextbooks, reading andstudyunits. However, youmaywishtouseother references tobroadenyourviewpointandprovidea deeperunderstandingofthe subject.

When youhave completed eachassignment, send ittogetherwithaform to yourtutor. Make surethat each assignmentreachesyour tutoron or beforethedeadlinegiven.Ifhowever,youcannotcompleteyourwork ontime,contact yourtutorbeforetheassignmentisdonetodiscussthe possibility of anextension.

Final Examinationsand Grading

Thefinal examination forthecoursewillcarry70%percentageofthe totalmarksavailablefor this course.Theexamination will cover every aspectof the course,soyou areadvisedtoreviseallyour corrected assignments before theexamination.

Thiscourseendowsyouwith thestatusof a teacher andthatof alearner. Thismeansthatyou teachyourself and thatyou learn,asyour learning capabilitieswouldallow.Italsomeansthatyouareinabetterposition to determine and to ascertain the what,the how, and the when ofyour learning. Noteacher imposes anymethodof leaming onyou.

Thecourseunitsaresimilarlydesignedwith theintroduction following thetableofcontents,thenasetofobjectivesandthenthedicourseand soon.

The objectivesguide you asyou go throughthe units to ascertain your knowledge of the requiredtermsand expressions.

CourseMarkingScheme

Thistableshows howtheactualcourse markingis brokendown.

Assessment	**Marks**
Assignment1- 4	Fourassignments,bestthreemarksofthe four countat30% ofcoursemarks
Final Examination	70% of overall coursemarks
Total	100% of course marks

Table1:Thecoursemarkingscheme

CourseOverview

Unit	Titleof Work	Weeks Activity	Assessment (End of Unit)
	CourseGuide	Week1	
	Module 1: Introduction to Programming andAlgorithm		
1	Introduction to Programming	Week1	Assignment1
2	Programming Concepts	Week1	Assignment2
3	Algorithm	Week2	Assignment3
4	BasicDataTypes	Week2	Assignment4
5	FundamentalDataStructure	Week2	Assignment5
6	PracticalExercises I	Week3	Assignment6
7	FundamentalDataStructures	Week3	Assignment7
8	Exercises I	Week3	Assignment8
	Module2: PerformanceAnalysisof Algorithms		
1	Performance AnalysisFramework	Week4	Assignment9
2	Orderof Growth	Week4	Assignment10
3	Worst-case, Best-case and Average-case Efficiencies	Week5	Assignment11
4	P, NP andNP-Complete Problems	Week6 -7	Assignment12
5	PracticalExerciseII	Week8	Assignment13
	Module 3: Sorting and Some Special Problems		
1	Introduction to Sorting and Divide-and-Conquer Algorithm	Week9	Assignment14
2	Merge Sort	Week10	Assignment15
3	QuickSort	Week10	Assignment16
4	BinarySearch	Week11	Assignment17
5	Selection Sort	Week12	Assignment18
6	BubbleSort	Week13	Assignment19
7	SpecialProblems andAlgorithms	Week14	Assignment20
8	Practical Exercise IV	Week15	Assignment21
	Revision	Week16	
	Examination	Week17	
Total		17weeks	

How to Get the BestfromthisCourse

Indistancelearningthestudyunitsreplacetheuniversitylecturer.This isoneofthegreatadvantagesofdistancelearning;youcanreadand workthroughspeciallydesignedstudymaterialsatyourownpace,and at atime and placethat suit you best. Think ofit asreading thelecture insteadoflisteningtoalecturer. Inthesamewaythatalecturermight setyousomereadingtodo,thestudyunitstellyouwhentoreadyour setbooksor other material.Justasalecturer mightgiveyou anin-class exercise,yourstudyunits provideexercisesforyouto doat appropriate points.

Eachof thestudyunitsfollowsacommonformat. Thefirstitemisan introductiontothesubjectmatter oftheunit andhowaparticular unitis integratedwiththeotherunitsandthecourseasawhole. Nextisaset oflearningobjectives.Theseobjectives enableyou knowwhatyou shouldbeabletodobythetimeyouhavecompletedtheunit. You should use these objectives to guide your study. When you have finished theunitsyou mustgo back and checkwhetheryou have achievedtheobjectives.Ifyoumakeahabitof doingthis youwill significantlyimproveyour chances of passingthecourse.

Rememberthat yourtutor’sjob isto assist you. Whenyou needhelp, don’thesitateto call andaskhim orher toprovideit.

1. Readthis Course Guide thoroughly.

2. Organizea studyschedule. Referto the CourseOverview formore details. Note the timeyou are expectedto spendon eachunitand how the assignments relate to the units. Whatever method you choosetouse,youshoulddecideonitandwriteinyourowndates for workingoneachunit.

3. Onceyou have created your own studyschedule,do everything you cantosticktoit. Themajorreasonstudentsfailisthattheylag behindintheir course work.

4. TurntoUnit1andreadtheintroductionandtheobjectivesforthe unit.

5. Assemblethestudymaterials. Informationaboutwhatyouneedfor aunitisgivenintheOverview atthebeginning ofeachunit. You will almost alwaysneedboththe studyunityou areworkingonand one of your setof booksonyour deskatthesametime.

6. Work through the unit. The content ofthe unit itself has been arrangedtoprovideasequenceforyoutofollow. Asyouwork throughtheunit youwillbe instructedtoreadsectionsfromyourset books or other articles.Use theunittoguideyour reading.

7. Review the objectivesfor each study unit to confirmthatyou have achievedthem.Ifyoufeelunsure about anyofthe objectives,review thestudymaterialor consultyour tutor.

8. When youareconfidentthat youhaveachieveda unit'sobjectives, youcanthenstartonthenextunit. Proceedunit byunitthroughthe course andtry to paceyour study sothatyou keep yourselfon schedule.

9. Whenyouhavesubmittedanassignmenttoyourtutorformarking, do not wait foritsreturnbefore starting on the next unit. Keep to yourschedule. When the assignmentisreturned,pay particular attention toyour tutor'scomments,bothonthetutor-marked assignmentformandonthe assignment. Consultyourtutor assoon as possibleifyouhaveanyquestionsor problems.

10. After completing thelastunit, reviewthecourse andprepareyourself forthe final examination. Check that you haveachieved the unit objectives(listed atthebeginningof eachunit)andthecourse objectives(listedinthisCourseGuide).

Facilitators/Tutorsand Tutorials

Thereare15hoursof tutorialsprovidedinsupportofthis course.You will be notified of the dates, times and location of these tutorials, together withthenameandphonenumberof your tutor,as soonas you are allocateda tutorialgroup.

Yourtutorwillmarkand comment onyourassignments,keepa close watchonyour progressandon anydifficulties youmightencounter and provideassistanceforyou duringthe course. Youmustmail orsubmit yourtutor-markedassignmentstoyourtutor wellbeforetheduedate(at leasttwo workingdays arerequired).Theywill bemarkedby yourtutor and returnedtoyouassoonaspossible.

Donothesitateto contactyourtutor bytelephone,or e-mail if youneed help. Thefollowingmight becircumstancesinwhichyouwouldfind helpnecessary. Contactyour tutor if:

i. Youdonotunderstandanypartofthestudyunitsortheassigned readings,

ii. youhavedifficultywiththeself-testsor exercises,
iii. you have a question or problem with an assignment, with your tutor’scommentson an assignmentor withthegradingof an assignment.

Youshouldtryyour besttoattendthetutorials. Thisistheonlychance tohavea facetofacecontactwithyour tutorandtoaskquestionswhich are answeredinstantly. Youcan raise anyproblemencountered inthe course of your study. To gain the maximum benefit from course tutorials,prepareaquestion list before attending the classes.You will learnalot fromparticipating indiscussionsactively.

Summary

Programming and Algorithms, as the title implies, will take you throughthefundamentalconceptsofproblemsolvingthroughtheuse of algorithms and efficient programming. Therefore,you should acquire the basicknowledgeoftheprinciplesofalgorithmdevelopmentand programwriting inthiscourse.Thecontentof the coursematerialwas plannedandwrittentoensurethatyouacquiretheproperknowledge and skills inorderto be able towrite efficientalgorithmsand implement them,using applicable programminglanguagesforthat areaof application. The essence is to get you to acquire the necessary knowledge andcompetence and equipyouwiththenecessarytools..

I wishyousuccesswiththecourse andhopethatyouwillfindit interestingandusefuI.

CONTENTS PAGE

MODULE1 INTRODUCTION TO PROGRAMMING ANDALGORITHM

UNIT1 INTRODUCTIONTOPROGRAMMING

CONTENTS

1.0 INTRODUCTION

This unit introduces methods and concepts of programming. It also explainshowprogramsareexecutedbythecompilers.

2.0 OBJECTIVES

Bytheendof thisunit youshouldbe ableto:

- listprogramsandprogramming languages
- outlinethedifferentlevelsofprogramminglanguagesandtheir characteristics

- outlinetheconventionalfeatures of programming languages outlinethemethodsof programming anditsapplication areas
- understand language evaluation criteria

explainlanguagetranslators, types, andtheir characteristics outlineand explaintheenvironments of programming.

3.0 MAINCONTENT

3.1 Meaning and Significance ofProgrammingLanguages

Programming languages are formal languages through which we caninstruct the computer to carry out some processes or tasks in order to produce a more accurate and meaningful results called outputs. Programming language consists of words whose letters are taken from set of alphabets called character set and obey a well-defined set of rules called syntax. In this way, programming languages are used to communicate explicit instruction between human beings and computer systems. These explicit instructions which are often expressed in a computer implementable notation are called algorithms. Programming languages can be usedtoexecute a wide range of algorithms, that is, an instruction couldbe executed through more than a procedure of execution. The full concept of algorithm will be explained later.

A computer program is a set of instructions(i.e. notations or codes)that can be executed by a computer to perform a particular task or process. The process of writing a computer program is called programming. Programming often involves a number of steps through which computer instructions are transformed into usable computer applications.

3.2 Levelsof ProgrammingLanguages

Programsandprogramming languages have beeninexistence sincethe invention ofcomputers,andthere are three levelsof programming languages.Theseare:

- **Machine Language:**Machinelanguageisasetbinary codedinstruction,whichconsistsof zeros(0) andones(1). Machinelanguageispeculiartoeachtypeofcomputer. Thefirstgeneration of computerswascodedinmachine languagethat was specific to each model ofcomputer. Someof theshortcomingsofthemachinelanguagewere:

1. Coding in machine language was a very tedious and boring job
2.. Machine language was not user-friendly.That is the user had to remember along list of codes, numbers or operation codes and know where instructions were stored in computer memory.
3. Debugging any set of codes is a very difficult task since it requires going through the program instruction from the beginning to the end.

Themajor advantageof machinelanguageisthatitrequiresno translation sinceit isalreadyinmachinelanguage and isthereforefaster to execute.

- **LowLevelLanguage:**Thisisalevelofprogramming language which isdifferent fromthe machine language. Thatis,theinstructionsarenot entirelyin binarycoded form.Italso consistsof some symbolic codes,which are easierto remember thanmachine codes.Inassembly language,memoryaddresses arereferenced bysymbols rather than addresses inmachinelanguage.Lowlevel programminglanguageisalsocalled a**ssemblylanguage,** becauseitmakesuseofanassemblertotranslatecodes into machinelanguage.An exampleofassembly language statementis:

MOVE A1,A2 Move thecontentsof Register A2toA1

JMPb Gototheprocesswithlabelb

Thedisadvantages of assemblylanguagearethat:

- Itis specifictoparticular machines
- Itrequires a translatorcalledanassembler.

Themajoradvantageoftheassemblylanguageisthatprogramswritten initareeasier toreadandmoreuserfriendlythanthosewrittenin machinelanguage, especially whencommentsareinsertedinthecodes

- **High Level Language:** This programming language consists of English-likecodes.High-levellanguageis independentofthecomputerbecausethe programmeronly needstopayattentiontothestepsorproceduresinvolved insolvingthe problemforwhich the programistobeused to execute the problem.High-level languageisusually broken into oneor more states such as: Main programs, sub-programs,classes,blocks, functions,procedures,etc. Thenamegiven toeach component differsfromone languagetotheother.

Someadvantagesofhigh-levellanguage:

- Itis moreuser friendly, thatis, easytolearnandwrite
- Itis veryportable, thatis, it canbe usedonalmostanycomputer
- Itsavesmuchtimeandeffortwhenusedcomparedtoanyother programming levellanguage.

- Codeswritteninthislanguagecaneasilybe debugged.

3.3 Features ofProgrammingLanguages

Therearesomeconventionalfeatureswhichaprogramminglanguage mustpossess,these featuresare:

- Itmusthave syntacticrulesfor forming statements.
- Itmusthave a vocabularythatconsistsofletters of thealphabet.
- Itmusthavealanguagestructure,whichconsistsofkeywords, expressions andstatements.
- It may require a translator before it can be understood by a computer.
- Programming languages are written and processed by the computerforthepurposeofcommunicatingdatabetweenthe humanbeingandthecomputer.

3.4 ProgrammingMethodologiesand Application Domain

3.4.1 Methodologies

Someprogramming methodologiesarestatedbelow:

- **Procedural Programming*:*** A procedural program is a seriesof steps, eachofwhichperformsa calculation, retrieves input, or producesoutput. Conceptslike assignments,loops, sequencesandconditional statements are thebuildingblocks of proceduralprogramming. Major procedural programminglanguages areCOBOL, FORTRAN, C, ANDC++.

- **Object-Oriented(OO)Programming*:***TheOOprogram isacollection ofobjectsthat interact witheach otherby passingmessagesthattransformtheir state.The fundamental building blocks of OO programming are
object modelling, classification and inheritance. Major object-orientedlanguages areC++,Java etc.

- **Functional Programming:** A functional program is a collectionofmathematical functions,each withan input (domain)and aresult(range).Interaction and combination offunctions iscarried outbyfunctional compositions, conditionalsandrecursion.Major functionalprogramming languages areLisp,Scheme, Haskell, andML.

- **Logic(Declarative)Programming:**Alogicprogramme isacollectionoflogicaldeclarationsaboutwhatoutcome a function should accomplish rather than how that outcome should beaccomplished.Logicprogramming providesanatural vehiclefor expressing non-determinism, sincethe solutionstomanyproblemsareoftennot unique butmanifold.Themajorlogicprogramming languageis Prolog.

- **EventDrivenProgramming:** Aneventdrivenprogram isacontinuousloop that respondstoeventsthat are generated in an unpredictable order. These events originatefromuseractions onthe screen (mouse clicksor keystrokes,forexample),or elsefromother sources(like readingsfrom sensorsona robot).Major event-driven programming languages includeVisualbasicandJava.

- **ConcurrentProgramming:**Aconcurrentprogramisa collectionof cooperating processes,sharinginformation with eachother fromtimetotimebutgenerallyoperating asynchronously. Concurrent programminglanguages include SR, Linda,andHighperformance FORTRAN.

3.4.2 Application Areas

Theprogramming communitiesthatrepresentdistinctapplication areas canbe groupedinthefollowingway:

- **Scientific Computing**: It is concerned with making complex calculationsvery fast and very accurately.The calculations aredefined by mathematical models,which represent scientificphenomena.Examplesof scientific programminglanguagesinclude Fortran 90,C, and High PerformanceFortran

- **ManagementInformationSystem(MIS)**:Programsfor usebyinstitutionstomanagetheir informationsystemsare probably themost prolificin theworld.These systems include an organisation’spayrollsystem,online sales and marketing systems, inventoryandmanufacturing systems, and so forth. Traditionally,MIShavebeen developed in programming languages likeCOBOL, RPG, andSQL.

- **Artificial Intelligence:** The artificial intelligence programmingcommunityhasbeenactivesincetheearly 1960s. Thiscommunityisconcernedaboutdeveloping programsthat model human intelligent behaviour,logical deduction,andcognition.ExamplesofAI programming languages are prominent functional and logic programminglanguageslike Prolog, CLP, ML,Lisp, SchemeandHaskell.

- **Systems:S**ystemprogrammersarethosewhodesignand maintain thebasic software that runssystems– operating systemcomponents,networkssoftware, programming language compilersand debuggers,virtual machines and interpreters,andsoon. Someoftheseprograms are written inthe assemblylanguageofthemachine,while many others arewrittenina language specificallydesigned for systemsprogramming.Theprimaryexampleofa system programming languageisC.

- **Web-centric:** The most dynamic area of new programmingcommunitygrowthistheWorldWideWeb, whichistheenablingvehicleforelectroniccommerceand awiderangeof applicationsin academia,government,and industry.Thenotion of Web-centric computing,and then Web-centric programming,ismotivated byan interactive model,inwhichaprogramremainsin an infiniteloop waitingforthenextrequestoreventtoarrive,responding to that event, and returning to itslooping state. Programminglanguagesthatsupport Web-centric computing requireaparadigmthat encouragessystem-user interaction,orevent-driven programming. Programming languages that support Web-centric computing include Perl,Tc1/Tk, Visualbasic, andJava

- **Mobile Computing**: This is one of the newer areas of programming technology. Mobile computing often involves technology which allow for transport of data, voice and video over a network via a mobile device. This new area of programming paradigm allows for connectivity, personalization and social engagement via availability of different applications or apps. Mobile computing supports a number of devices such as Tablets, smart phones, hand held gaming devices, wearable devices etc. The programming languages for mobile computing include python, Kotlin, SWIFT, Go etc.

- **Cloud Computing**: This is one of the newer emerging technologies that is rapidly changing the face of internet and programming. It is a programming technology in which programs are designed to provide on-demand availability of computer system resources over data storage without direct active management by the subscriber or user. Businesses are now employing cloud computing in different ways to maintain users' information on private servers or public servers on the internet. The most popular examples of cloud computing infrastructures include Google cloud, IBM cloud, Amazon web services etc. Most common programming languages for cloud computing include Python, Clojure, Erlang, Haskell etc.

3.5 Translators

Atranslator isa programthattranslatesanotherprogramwrittenin any programminglanguageother thanthemachinelanguagetoan understandable set of codesforthecomputer and insodoing producesa program that may be executed on the computer. The need for a translator arisesbecauseonlyaprogramthatisdirectly executableona computer isthemachinelanguage.Examplesofatranslator are:-

1. **Assembler:**Thisisaprogramthatconvertsprogramswrittenin assemblyor low-levellanguage tomachinelanguage.

2. **InterpretersandCompilers:** Theseconsistofprogramsthat convert programsin high level programminglanguageinto machinelanguage.Themajor differencebetweeninterpretersand compilersisthatacompilerconvertstheentiresourceprogram

intoobjectcodebeforethe entireprogram isexecutedwhile the interpreter translatesthe sourceinstructionsline by line.Inthe former, the computer immediately executes one instruction beforetranslating thenextinstruction.

3.5.1 Features ofTranslators

They existtomake programsunderstandablebythecomputer
Thereexistdifferenttranslatorsfordifferentlevelsandtypesof programming languages
Withoutthem,theprograms cannotbe executed.

3.6 **Element of Programming languages**

All programming languages have some basic building blocks for describing data and processes or transformations applied to them. There are two main elements of all programming languages namely syntax and semantics.

- Syntax: The syntax of a programming language describes the possible combinations of symbols from the language's character set that form a syntactically correct program. Programming language syntax is usually defined using a combination of regular expressions and Backus Naur Form.

- Semantics: The semantic of a programming language describes the meaning of languages. Semantics comes in different forms. For instance, the static semantics defines restrictions on the structure of valid texts that are hard or impossible to express in standard syntactic formalisms while the dynamic semantics defines how and when the various constructs of a language should produce a program behavior.

3.7 Language Evaluation Criteria

The following are the criteria that influences the evaluation of computer languages:

- Readability: This describes the ease with which programs can be read and understood and it is one of the most important criteria for evaluating a programming language.

- Simplicity: This describes the ability of programming languages to use familiar symbols for their basic operations and computations

- Reliability: This describes the ability of a programming language to perform to its specification under all conditions by providing for exception handling, type checking, error overthrow etc.

- Expressivity: This describes the ability of a programming language to clearly reflect the meaning intended by a programmer by using notations which are consistent with those used in the field for which the language is designed.

- Pedagogy: This is the ability to teach and learn programming language which is usually evaluated by its clarity and simplicity of instructions and programming constructs

3.8 TheProgrammingEnvironment

TheEditor: An editor allowsa programtoberetrievedfromthedisk and amended as necessary. In order to type any program on the keyboard andsavetheprogramonadisk,it willbenecessarytoruna program calledan editor.

TheCompiler: Thiswilltranslateaprogramwritteninhighlevel languagestoredinatextmode onadisk totheprogramstoredin a machine-orientedlanguageona disk.

TheLinker/Loader: Alinker/loaderpicksupthemachine-oriented programand combinesit withanynecessary software(alreadyin machineoriented form)toenabletheprogramtobe run.
Beforeacompiledprogram can berunor executed bythecomputer,it mustbeconvertedintoanexecutable form.

4.0 CONCLUSION

In thecourseof theseunityouwhereintroducedtotheconceptof programming,you alsolearnt abouttheideaof programminglanguages and the various types and methodologies involved in writing a programs. Conclusivelyyou learntaboutthevariousfieldsinwhich programminglanguagecouldbeimplemented.Wefinishedthiscourse bylooking atvariousinterpretersandthevariousfeaturesof a programming environment.

5.0 SUMMARY

Inthisunit youlearntthat:

- Programminglanguagesarelanguagesthroughwhichwe can instructthecomputer tocarryoutprocesses andtasks.
- Aprogramis asetof codesthatinstructsthecomputer tocarry outsomeprocesseswhileprogrammingistheactof writing programs.
- Therearefourlevels ofthe programminglanguage-machine language,lowlevellanguage,assemblylanguageandhigh level language.
- Therearevariousprogramming methodologies, of whichwe have procedural programming,object-

orientedprogramming, functional,logic (declarative),eventdriven and concurrent programming.

- Therearebasicallythreetypesof translators- assembler, interpreters andcompilers.
- Programming environment consists of the editor, translator and the linker/loader
- Language evaluation criteria are Readability, Simplicity, Reliability, Expressivity and Pedagogy

6.0 TUTOR-MARKEDASSIGNMENT

a. Writeoutanytenprogramminglanguagesstatingtheapplicationareas
b. How does pedagogy affect language selection?
c. Using a practical example, differentiate between syntax and semantic
d. State ten distinct programming application areas

7.0 REFERENCES/FURTHERREADINGS

GonnetandRicardoBaeza-Yates(1993). *HandbookofAlgorithmsand DataStructures*. InternationalComputerScienceSeries.Holmes,B.J. (2000).

Pascal ProgrammingContinuum(2nd ed).

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

Published by Morgan Kaufmann.

https://www.technotification.com/2018/08/programminglan

guages-cloud-computing.html

UNIT2 PROGRAMMINGCONCEPTS

CONTENTS

1.0 INTRODUCTION

Thisunitintroducesyouto programming concepts.Theseincludethe programmingdevelopmentcyclewhichconsistsofthestagesinvolved in developing an efficient program. It also introduces you to the program execution stages, as well as the conventional principles of goodprogramming

2.0 OBJECTIVES

Havinggonethroughthisunit,youshouldbeableto:

- ✓ explainthefivemajorstepsinvolvedindevelopinganefficient program
- ✓ outlinethefourstagesinvolvedintheexecutionofanormal program
- ✓ outlinetheprinciplesof a goodprogramming style.
- ✓ understand programming paradigms

3.0 MAINCONTENT

3.1 TheProgramDevelopmentCycle

Program Development cycle depicts the various stages involved in the lifespan of a computer program from its origin until completion or closure. Program development cycle addresses three main attributes namely stakeholders of the program (i.e. organization requesting the computer application), benefits acquired from the program (i.e. the deliverables) and rules governing the development of the program lifecycle (i.e. expectation and documentations). In program development, a program is constituted by two fundamental parts namely the objects and the operations. The object is a representation of the data relative to the domain of interest while the operations describe how the objects are to be manipulated in such a way to realize the desired outputs. The various stages of program development lifecycle are discussed relative to these two fundamental parts.

Themajorfive stages involvedindeveloping anefficientprogram are:-

- **ProblemAnalysis:**Thisiswheretheclearstatementof theproblemisstated.Theprogrammermustbesurethat heunderstandsthe problem and how to solve it.Hemust knowwhatis expected ofthe problem,i.e.whatthe programshoulddo,thenatureoftheoutputandtheinput toconsidersoasgettheoutput.Hemustalsounderstand the ways of solving the problem and the relationship betweentheinput andtheexpectedoutput.

- **Design:**Theplanningofthesolutiontotheprobleminthe firststagetakesplaceinthisstage.Theplanningconsists oftheprocessoffindingalogicalsequenceofprecise stepsthatsolvetheproblem. Suchasequenceofstepsis calledan**algorithm**.Everydetail,includingobvioussteps shouldappearinthealgorithm.Thethreepopularmethods usedtodevelopthelogicplanare:flowcharts,apseudo code, and a top-down chart. These tools help the programmer breakdown aprobleminto asequence of small tasks the computer can perform to solve the problem.Planningmayalsoinvolveusingrepresentative datatotestthelogicofthealgorithmbyhandtoensure thatitis correct.

- **Coding:**Translationofthealgorithminstagetwointoa programming languagetakes placehere. Theprocess for writing the program is called **coding.** The programmer usesthealgorithmdevised inthedesignstage alongwith thechoiceoftheprogramminglanguagehegot fromstage three.

- **Testing and Debugging:** The process involves the location and removal of error in the program if any. Testing is the process of checking if the program is workingasexpectedandfindingerrorsintheprogram, and debuggingisthe processofcorrecting errorsthat are found(An error ina programis calleda bug.).

- **Documentation: T**his is the final stage of program development.It consistsof organising allthematerialthat describestheprogram.Thedocumentationoftheprogram isintendedtoallow another personortheprogrammerata laterdate,tounderstand theprogram.Internal documentation remarks consist of statements in the programthat arenot executed,but pointout thepurposeof variouspartsoftheprogram.Documentation might also consistofadetailed descriptionofwhat theprogram does and howto usethe program.Othertypesofdocumentation areflowchartand pseudo codethat wereused to construct the program.Although documentation is listed asthelast step in the program development cycle, it should take placeastheprogramisbeingcoded**.**Itissometimesthe first step during program execution because the programmercanuseanotherprogramdocumentationin

 CIT237

developing a new program by just improving on the previous work.

3.2 Program Execution Stages

Thenormalprogramexecutionconsistsoffour(4)stages(Seefigure 2.1),thoughsomeprogramming languageslike BASICcombinetwoor threeofthesein one single process.Theprogram executionstages are explainedbelow:-

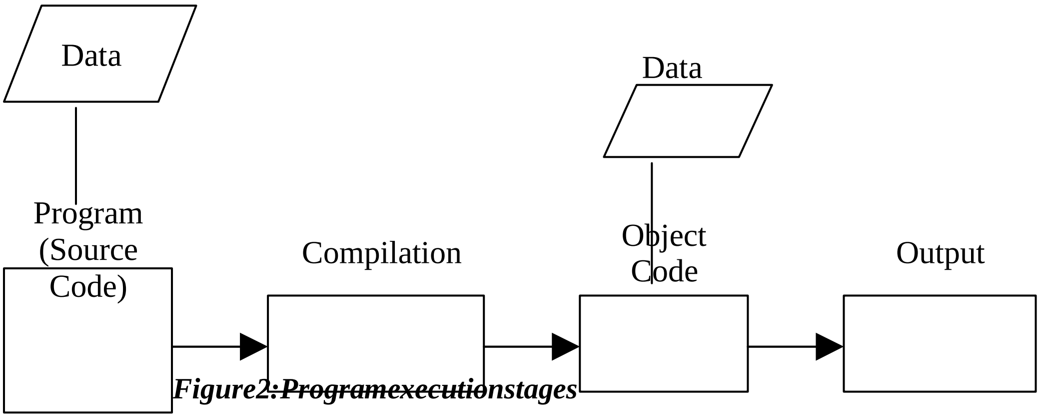


Figure2:Programexecutionstages

- **TheProgram (Source Code):** Thisis thecoded instruction giventothecomputer ina particular programming languageinorder toaccomplisha given task.The sourcecodemustobeythesyntacticand semanticrulesofthesource programming language.

- **TheCompilationProcess:**The source codeis suppliedto thecomplier, whichcoverts theobjectcode.The process of compilationinvolves readingthesource code,checking for errorsinthesource code andconverting ittoan

 18

executable format(machinecode) if noerror isdetected, elsetheprocess of compilationis abortedandan error is reported.

- **TheObjectCode:** The objectcode is theresult of the compilation processandit isalsocalledthetargetcode. Theobject codeisdependentontheprogramming languagechosen.For instance,theobject codeofJAVA compilation isa bytecode, thatof Fortranisan executable statementofthetargetmachine, whilethatBASICis that of thetargetmachinelanguage, butit isnotwrittentoany filelikethatofFortran andJAVA.

- **TheOutput:**The laststageisfor thecomputertogivethe result.Thecomputer executestheobject codeinorder to presentthedesiredoutput. Itis importanttonotethata valid ordesiredoutput might notbe givenifthelogicof theprogramis not correct.

3.3 Principlesofa Good ProgrammingStyle

The following represent the major considerations in writing good programs:-

1. **NamingConventions:** It is very important to give meaningful names to all your constructs. A name like `get_Height()` or `get_avg_height()` gives us much

moreinformationthan**`ctunde()`**.Also,avariablename-**total**-foradditionismoremeaningfulthan**pen.** Thenameofaclass should communicateitspurpose.Classnameshouldstartwithan uppercase letter, e.g. `class AddPrime.`

Major variables,which aresharedbymultiple functionsand/or modulesshould beidentified and named at the design stageitself. Variable name should start withalowercaseletter,e.g. `firstQuad;`

Functionsshouldbenamedsimilartovariables.Wecan always distinguish between them becauseof the parenthesis associated with functions.

2. **FileNamingand Organisation:**Filesshouldbeorganisedinto directoriesina module-wisefashioninsteadofhavinga monolithicstructurewhereallsourcecodefilesandallheader files arein a single directory.Thisshouldbepartof thedesign process.

3. **FormattingandIndentation:**Thelineswithinthecodeshould be clearlyorganisedina waythatit will beeasytoread and understandevenfor thewriter.Proper identification shouldbe usedtoshowsubordinatelines.

4. **Commentsand Documentation:**Introducing comments and proper explanations(documentation) of theprogram aidin understanding the code.Theyhelpusin followingthe program flow,andskippartsforwhichwearenotinterestedindetails. Thisallowsfor programamendmentandextensibility.

5. **Classes:**Ensurethatalltheclassesinyourapplicationhavea default constructor,copy constructorand overloadedoperator. Also ensurethat alltheclassdataitemsare appropriately initialisedinconstructorandassignedtoeachmemberofthe class.

6. **Functions:**Afunctionshouldnormallydoonlyonejobanddoit well.Avoidgeneric functionswith lots of conditional branchesto do everything.If afunctionissupposedtodomultiple jobs,then create helper functions and delegate responsibilities to them. Makefunctions simple and small.Theideal sizeof functions is around35- 40lines.

7. **UsingSTL:**UseStandardTemplateLibrary(STL)insteadof creatingyourowncontainerdatastructures.Donotusehash maps inSTL;they arenotportableacrossplatforms.

 CIT237

8. **PointersandReferences:**Usereferences,especiallyifcodingin C++,encouragestheuseofreferencesinsteadofpointers.Infact apointer should typicallybepassedto afunctiononlyincases whereyouneedto execute somethingonthepointer beinganull condition.

9. **Minimising Bugs by Testing:** Testing is an integral part of softwaredevelopment.Testshelpusnot onlyin makingsurethat what we have written is correct, but also in finding out if someonebreaksthe codelater.So,it isagoodprogrammingstyle tothoroughlytesta program.

3.4 Programming paradigms

There are several programming paradigms used in program life cycle development. Each programming paradigm differs in the emphasis put on the two fundamental aspects of programming which are objects and operations. The objects are entity that receive instructions to perform a particular method or actions while the operations are the events that direct the behavior of an object. The three main programming paradigms are:

- **Imperative:** This paradigm places emphasis on the operations intended as actions that change the state of the computations. The objects are functional to the computation.

- **Functional**: This paradigm puts emphasis or development on the operations intended as functions that compute results. In this case, the objects are functional to the computation like in the Imperative type.

- **Object-oriented**: This paradigm place emphasis or development on the objects which serves as the domain of interest in the overall picture of the system. In this case, the operations are functional to the representation.

 22

Imperative/Functional paradigm Object-oriented paradigm

Usually, every programming languages provide support for these three programming paradigms as a program may use different paradigms within the development life cycle of a program depending on the ease and functionality intended at hand.

3.5 **Object-oriented Modeling**

Object-oriented modeling (OOM) is a common approach to modeling programs by using object-oriented paradigm throughout the entire development life cycle. This is the main technique used in modern software engineering. The OOM typically divides programming life cycle into two aspects:

- Modeling of the dynamic behaviorslike processes and use cases
- Modeling the static structures like classes and components

The advantages of using OOM are

- Efficient and effective communications between the system and the real world
- Useful and stable abstractions that define essential structures and behavior within the system under development

4.0 CONCLUSION

Inthecourseoftheprogramthestudentshouldbeabletowritea good programfollowingagoodprogramming convention,apartfromlearning thecycleof programdevelopment.Theprogram executionstagesare alsonotleftoutof thisunit.

5.0 SUMMARY

- Problem Analysis - This is where the clear statement of the problemis stated.
- Design-Theplanningofthesolutiontotheprobleminthefirst
- stagetakes placeinthisstage
- Planningmayalsoinvolveusingrepresentativedatatotestthe logicofthealgorithm byhandtoensure thatitis correct.
- Coding - Translation of the algorithm in stage two into a
- programming languagetakes placehere
- Theprocess for writing theprogramis calledcoding.
- Testinganddebugging-Theprocessinvolvesthelocationand removaloferrors(ifany) intheprogram.
- Documentation-Thisisthefinalstageofprogramdevelopment;
- it consists of organising all the material that describes the program.
- Thenormalprogram execution consistsof four (4) stages.

- The programme (source code) –This is the set of coded instructionsgiventothecomputer toperforma particular task. Theprocessofcompilationinvolvesreadingthesourcecodeand checkingfor errors it.
- Theobjectcode-Theobjectcodeistheresultofthecompilation
- processanditisalsocalledthetargetcode.
- It is very important to give meaningful names to all your
- constructs. A name like **`get_Height()`** or **`get_avg_height()`** gives us much more information than **`ctunde()`**.

The variable name should start with a lowercase letter, e.g. **`firstQuad;`**
Thenameof a classshould communicateitspurpose.
Aclassnameshould startwith anuppercaseletter.
Files should be organised into directories in a module-wise fashion.
Introducingcommentsandproperexplanations(documentation) of theprogramhelpsinunderstandingthecode.
Ensure that allthe classes in yourapplication have adefault constructor.
Avoidgenericfunctionswithlotsofconditionalbranchestodo everything.
Use theStandard Template Library (STL) instead of creating your owncontainerdatastructures.
Programming paradigms are generally classified as imperative, functional and object-oriented

6.0 TUTOR-MARKEDASSIGNMENT

1. What are the major five stages involved in developing an efficientprogram?
2. Whatis thefinalstageofprogramdevelopment?
3. istheprocessofcheckingiftheprogramisworkingor is notworking aright.
4. Drawtheprogramexecutionchart.
5. Whatdoyouunderstandby compilation?
6. Whatdoyouunderstandbylogicalerror?
7. Thecomputerexecutestheobjectcodeinordertopresentthe desired_
8. What are theprinciples of a goodprogramming language?
9. Which of these is a good variable name following a good programming? conventions:
   (a) variable (b) variable2 (c) 2variable
   (d) math()
10. Differentiatebetweena function anda variable.
11. Howdoyoumakea clumsycode lookneat and readable?
12. Whatis theimportanceofcomments?
13. isanintegralpartof softwaredevelopment.
14. Whatdoyouunderstandbyagoodprogramming style?
15. State the main goal of object-oriented technology
16. What do you understand by object in object-oriented?

7.0 REFERENCES/FURTHERREADINGS

GonnetandRicardoBaeza-Yates(1993). *HandbookofAlgorithmsand DataStructures*. InternationalComputer ScienceSeries.

Holmes,B.J. (2000). Pascal Programming.*Continuum.* (2nd ed).

Levitin,A.(2003).*IntroductiontotheDesign&AnalysisofAlgorithms.* PublishedbyAddison-Wesley.

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

www.doc.ic.ac.uk/~wjk/C++Intro/

www.personal.kent.edu/~muhama/Algorithms

https://www.inf.unibz.it/~calvanese/teaching/04-05-ip/lecture-notes/uni01.pdf

https://en.wikipedia.org/wiki/Object-oriented_analysis_and_design

UNIT3 ALGORITHMS

CONTENTS

1.0 INTRODUCTION

Inthisunit, youwill beintroducedtoalgorithmsaswell asthenecessary conditions to design a good algorithm. The unit also highlights importantstagestoindesigning an algorithm. Thecharacteristicsofa good algorithmare also outlined, among which is the fact that a good algorithm musthave a beginning andan end.

2.0 OBJECTIVES

Bytheendof thisunit youshouldbe ableto:

- ✓ explainwhat analgorithm is
- ✓ differentiatebetweencomputationalproblems andalgorithms outlinethecharacteristics of analgorithm
- ✓ explainthestagesinthedesignof analgorithm
- ✓ understand the relationship between computer and algorithm
- ✓ explain types of algorithm

3.0 MAINCONTENT

3.1 Introduction to Algorithms

Whatis an algorithm?Although thereisno universallyagreed-on wordingtodescribethisnotion,thereisageneralagreement aboutwhat theconceptmeans:

 CIT237

An algorithm is a finite sequence of unambiguous instructions for solvingaproblem,i.e.,forobtainingarequiredoutput foranylegitimate input ina finiteamount oftime.

 28

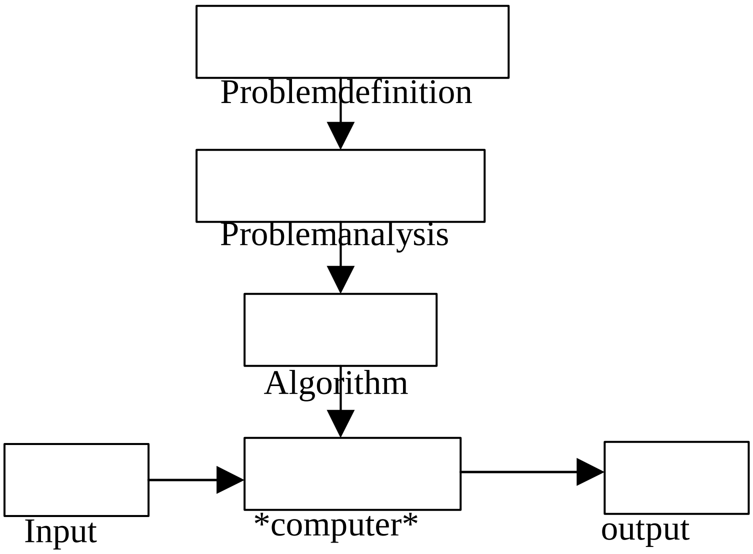


Figure.3: Thepositionof algorithmsinproblemsolving

As shown inthe abovefigure, afteraproblem hasbeenidentified,the problemisthen carefullyanalysedin orderto present a suitable algorithm.An algorithm isthendesignedandpresentedtothe computer inaparticular programminglanguage.Thecomputerwillthen generate theoutput,basedontheinput. Hence, an algorithm presents a well-defined computational procedure that takes some values as ***input*** and produces some value as ***output.***

In general, an effective algorithm has three main characteristics

- Explicit, complete and precise initial conditions (input)

- A finite, complete but not necessarily linear (i.e. looping, recursive, sequential) series of steps to arrive at the desired results (process)

- Explicit, complete and precise terminal (stopping) conditions (output)

Example: Converting Fahrenheit to Celcius
The example describe the relationship between the Celsius and Fahrenheit scales as follows:

$$5(F-32) = 9C \qquad (1)$$

where F = temperature in degrees Fahrenheit, and
C = temperature in degrees Celsius.

Formula (1) defines the relationship between temperatures in Celsius and Fahrenheit, but it doesn't give us an explicit algorithm for converting from one to the other. Fortunately, our understanding of algebra can easily allow us write such an algorithm.

First, we can use the basic operations of algebra to convert (1) into a form that expresses C as a function of F:

$C = 5(F - 32)/9$. (2)

Now it's a straightforward task to write an algorithm (based on the standard order of arithmetic operations) to convert from Fahrenheit to Celsius.

1. Start with a given temperature in degrees Fahrenheit.
2. Subtract 32 from the value used in step #1.
3. Multiply the result of step #2 by 5.
4. Divide the result of step #3 by 9.
5. The result of step #4 is the temperature in degrees Celsius.

Algorithm 1: Conversion from Fahrenheit to Celsius

The skills required to effectively design and analyze algorithms are entangled with the skills required to effectively describe algorithms. At least, a complete description of any algorithm has four components:

- What: A precise specification of the problem that the algorithm solves.
- How: A precise description of the algorithm itself.
- Why: A proof that the algorithm solves the problem it is supposed to solve.
- How fast: An analysis of the running time of the algorithm.

It is not necessary (or even advisable) to develop these four components in this particular order. Problem specifications, algorithm descriptions, correctness proofs, and time analyses usually evolve simultaneously, with the development of each component informing the development of the others.

3.2 Computational Problemsand Algorithms

Definition1:A **computational problem** is a specification of the desiredinput-output relationship.

Definition2: An**instance**ofaproblemisall theinputsneeded to computea solution totheproblem.

Definition3: An**algorithm**isawell definedcomputationalprocedure that transformsinputs into outputs, achieving the desired input-output relationship.

Definition4:A correct algorithm haltswiththecorrectoutputforevery inputinstance.Wecan thensimplysaythatan **algorithm**isaprocedure for solvingcomputationalproblems

3.3 CharacteristicsofAlgorithms

Thefollowing arethemajor considerationsinthedesignofalgorithms

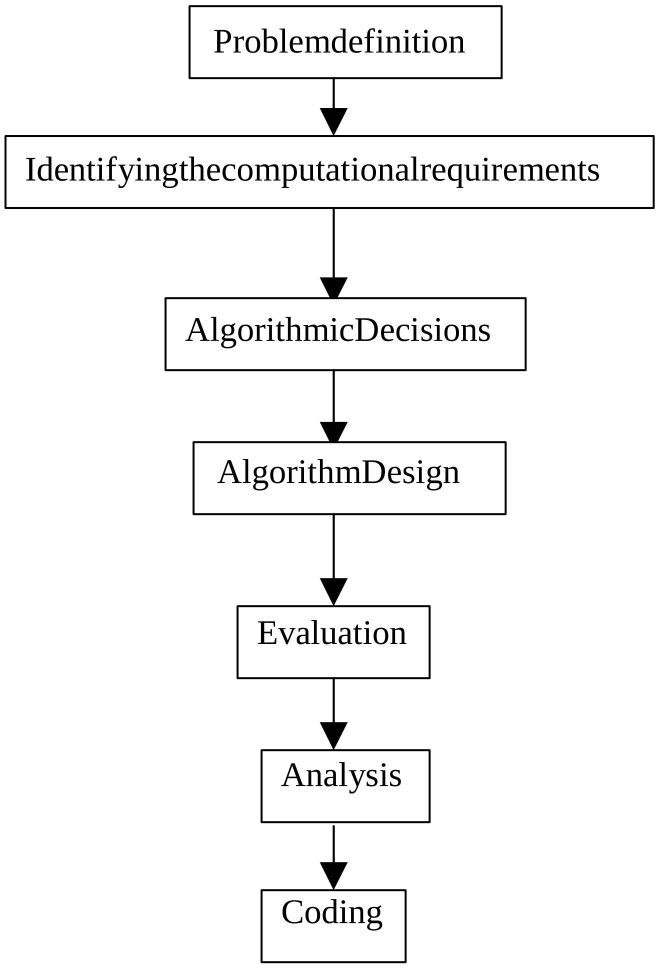
Analgorithm musthavea beginning andanend
Thenonambiguityrequirementforeachstepofanalgorithm cannotbe compromised.

 CIT237

Therangeofinputsforwhichanalgorithmworkshastobe specifiedcarefully
Thesamealgorithm canbe representedin severaldifferentways
Severalalgorithmsfor solving thesameproblemmayexist
Algorithmsforthesameproblemcanbebasedonverydifferent ideas and can solve the problem with dramatically different speeds
Itmustterminateata reasonableperiodof time.

3.4 AlgorithmDesignand AnalysisStages

Thediagrambelow representsthestagesinalgorithm design



Problem Definition

Conventionally, whenproviding solutionsfor anygivenproblem,the problem solver mustfullyhavetheunderstandingof theproblem,thatis he/ shemust think about the problem'sexceptional cases and must ask questions again and again in order to avoid doubt(s), and fully understandthesubjectmatter.

 32

IdentifyingComputationalRequirement(s)

Aftertheprogrammerhasfullyunderstoodtheproblem,all he/sheneeds do,istoidentifythecomputationalrequirement(s)neededto solve the problem.

AlgorithmicDecisions(Pre-Design Decisions)

Beforeaprogrammerdesignsanalgorithm,he/shedecidesthemethod toimplementin solvingtheproblem,whether itsexactor approximate, which arecalledexact algorithm or approximatealgorithm respectively. Alsoduringthis stagetheprogrammer decidesand choosesthe appropriatedatastructure neededtorepresenttheinputs.

Algorithm Design

Inthis phasetheprogrammerbattleswiththe problemofhowhe orshe shoulddesign an algorithm to solvethegivenproblem.Also,the programmerspecifiesthefashionwhichthealgorithmwill follow,either pseudocodealgorithm fashionorEuclid algorithm fashion.

Algorithm Evaluation

Thealgorithmevaluation phaseisthetesting phasewherebythe programmer confirms thatthe algorithmyieldsthedesiredresultforthe rightinputthatisinareasonableamountof time.Theprogrammer proves thecorrectness of thealgorithm.

Provingan Algorithm’sCorrectness

Since an algorithmhasbeen specified,you haveto proveits correctness. Thatis,youhavetoprovethatthealgorithmyieldsarequiredresult for everylegitimateinput ina finiteamount of time.

Algorithm Analysis

Inthisphase,we check the efficiency ofthe algorithmin termsoftime and spacewhicharetermed astime efficiencyand space efficiency respectively.

CodingtheAlgorithm

Mostalgorithmsfinallytransit intocomputerprograms.Thetransition (codingofanalgorithm)involvesachallenge and an opportunity.The challengeisthedevelopmentofthealgorithminto aprogram,either incorrectly or inefficiently, while the opportunity is that the coded algorithm eventuallybecomesan automatedsolutiontothe given problem.

3.5 Relationship between computer and algorithms

Computers are electronic devices whose amazing feats of calculation and memory depends largely on its internal and external programs. These programs consists of step-by-step procedures for the computer to follow to produce specific results. In other words, computer programs is almost about algorithms. For instance, when we write a line of Java or Python code to compose a formula for a cell in a spreadsheet, we are using algorithms that others have written as building blocks for performing the required operations

3.6 TypesofAlgorithms

1. RecursiveAlgorithms

Thesearealgorithmsthathavethesamefunctioncallingthemselves. For example,therecursivealgorithm for the Fibonacciexampleis.

```
Algorithm f (n)
F (0)=0;
F (1)= 1;
For I= 2toN
        F (I) = F (I-1) +F ( I-2);
RETURNF (N);
```

Inthis algorithm,wecan seefunctionslikeF(I-1),F(I),F(I-2)calling / referringtothesamealgorithm.This caseisalsoreferredto as a **recursion**.Factorial problem is another example of such algorithm

2. Non-recursiveAlgorithms

Thesearealgorithmsthatdonotrecallbackthesamealgorithm or function. For example,writea programtogenerate Fibonaccisequence.

```
    M = 1
    N= 2
    I = 2
    WRITEM
    WRITEN
30  L = N
    N= N+ M
    WRITEN
    M = L
    I = I+1
    IF I <= 30GOTO30
```

END

4.0 CONCLUSION

Inthis unit,youhavelearnthow todesignan efficient algorithm. You were also shown the difference between computational problems and algorithms. Thisunitalsoexplainedthestagesinthedesignofan efficientalgorithm.

5.0 SUMMARY

Thisunithasexplained what an algorithmis and the necessary considerations to design agood algorithm. It has also examined the importantstagestotakein the design of an algorithm, aswell asthe characteristicsof algorithms.

6.0 TUTOR-MARKEDASSIGNMENT

1. Inone sentence,definean algorithm.
2. Listandexplainfive(5)stagesinvolvedinthedesignofan algorithm.
3. Write a recursive algorithm for finding the factorial of any given number

7.0 REFERENCES/FURTHERREADINGS

Gonnet and Ricardo Baeza-Yates (1993). Handbook of Algorithms andData Structures. International Computer Science Series.

Holmes, B. J. (2000). Pascal Programming Continuum (2nd ed).

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition
www.doc.ic.ac.uk/~wjk/C++Intro/

https://jeffe.cs.illinois.edu/teaching/algorithms/book/Algorithms-JeffE.pdf

UNIT4 PROGRAM DESIGNTOOLS

CONTENTS

1.0 INTRODUCTION

Inthisunit, youwill beintroducedtoalgorithmsaswell asthenecessary conditionstodesignagoodalgorithm. Theunithighlightsimportant stagestodesign an algorithm and alsooutlinesthecharacteristicsofa good algorithm,oneofwhichisthatagood algorithm musthavea beginning andan end.

2.0 OBJECTIVES

Attheendof thisunit, youshouldbe ableto:

- ✓ identifyvarioustoolsusedtorepresentanalgorithmsotherwise knownas programming tools
- ✓ understand the symbols and functions of each pictorial componentof a flow-chart
- ✓ explainwhatpseudocodesareandtheiradvantag es differentiatebetween flowchartsandpseudocodes.

3.0MAINCONTENT

3.1Introduction to ProgrammingTools

Ithasbeen stated earlier thatan algorithmisasetofproceduresfor solving aproblem.Thetoolsusedto clearlyrepresent analgorithmare programming tools.

Example: Problem:Designanalgorithmtofindtheaverageoftwo numbers.

Discussion: Sinceanalgorithmisjustthesolutionstepsforaproblem, it canberepresentedbyordinaryEnglishexpressions.

Solution:

1. Start
2. Getthefirstnumber
3. Getthesecondnumber
4. Addthetwonumberstogether
5. Showthe result
6. Stop

3.2Flowcharts

Aflowchart consistsof specialgeometricsymbolsconnectedby arrows. Withineach symbolisa phrasepresentingtheactivityatthatstep.The shapeof thesymbolindicatesthetypeofoperationthatistooccur.For instance,theparallelogramdenotesinput or output.Thearrows connecting the symbols, called **flow lines**, show the progression in whichthestepstakeplace.Flowchartsshould“flow”fromthetopof the pagetothebottom. Althoughthe symbolsusedin flowcharts are standardised,no standardsexistfor theamount of detailrequiredwithin eachsymbol.

Atableof theflowchartsymbolsadoptedbytheAmericanNational StandardsInstitute(ANSI)follows(Figure4).Figure5 showsthe flowchartfor thepostage stampproblem.

Themain advantageof usingaflowcharttoplanataskisthatit provides apictorial representation of the task,which makesthe logic easier to follow.We can clearlysee everystep and how each step isconnected to the next. The major disadvantage with flowcharts is that when a programis verylarge, theflowchartsmaycontinuefor manypages, makingthemdifficult tofollowandmodify.

 CIT237

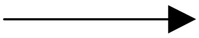
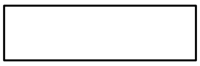
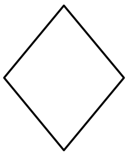
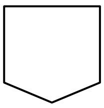
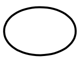
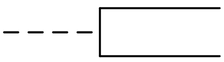
Symbol	**Name**	**Meaning**
	Flow line	Usedtoconnectsymbolsandindicate theflowof logic.
	Terminal	Used to represent the beginning (start) or theend (end) of a task.
	Input/Output	Usedforinputandoutput operations, such asreading and printing.Thedata toberead orprinted aredescribed inside.
	Processing	Used forarithmetic and data-manipulation operations.The instructions arelisted insidethe symbol.
	Decision	Used for anylogic orcomparison operations. Unlike the input/output and processing symbols,which have oneentryandoneexit flowline,the decision symbol has one entry and twoexitpaths.Thepath chosen dependsonwhether theanswertoa questionis “yes” or “no”.
	Off page	Used to indicate that the flowchart continuestoasecondpage.
	Connector	Usedtojoindifferentflowlines
	Predefined	Used to represent a group of statements that perform one processingtask.
	Annotation	Used to provide additional information aboutanother flowchart symbol.

Figure 4: Table of the flowchart symbols adopted by the American National StandardsInstitute(ANSI)

 39

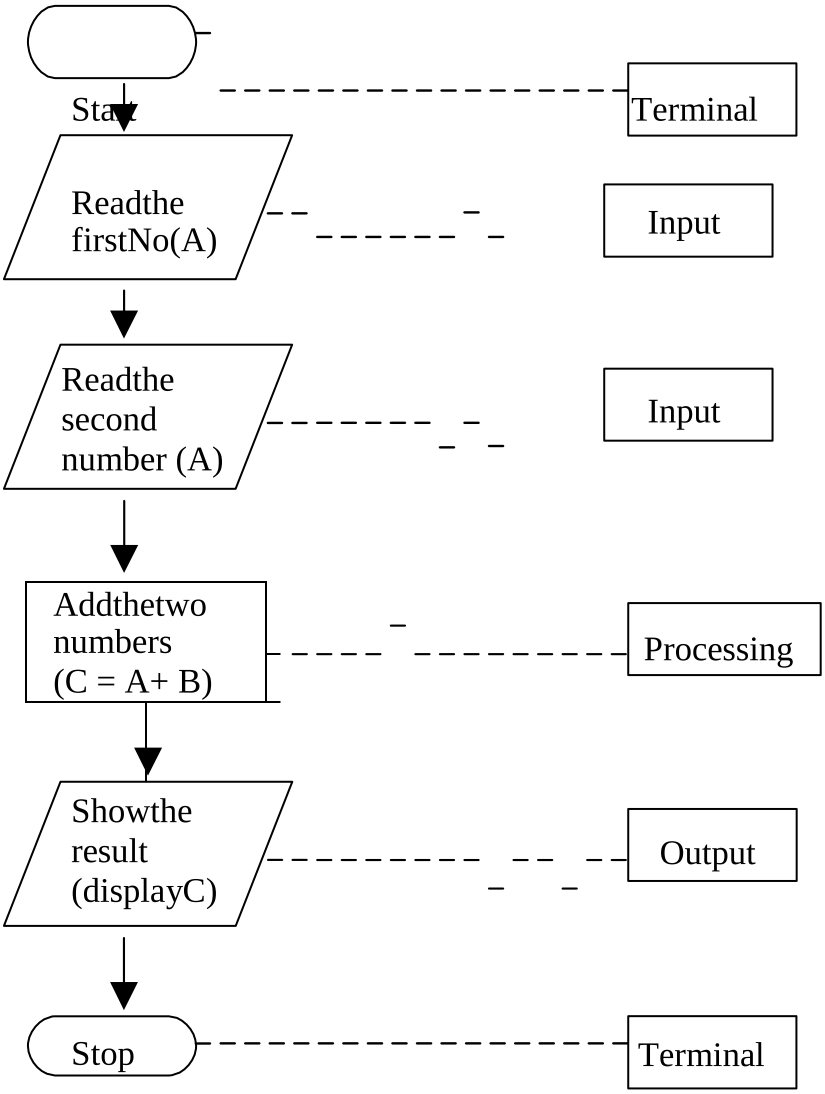


Figure5:Theflowchartforthepostagestampproblem

3.3Pseudocodes

Apseudo codeis anabbreviated version of an actualcomputer code (hence, the term pseudo code). The geometric symbols used in flowcharts are replaced by English-like statementsthat outline the process.Asaresult,apseudocodelooksmorelikeacomputercode thanaflowchartdoes.Thepseudocodeallowstheprogrammer tofocus onthe stepsrequiredto solveaproblemrather thanonhowtousethe computer language. The programmer can describe the algorithm in Visual Basic-likeformwithout beingrestrictedbytherulesofVisual Basic.Whenthepseudocodeiscompleted,itcanbeeasilytranslated into theVisual Basiclanguage.

Thepseudocodehasseveral advantages.Itis compact and probablywill notextendformanypagesasaflowchartwould.Also,theplanlooks likethecodetobe written andsois preferredbymanyprogrammers.

Thepseudocode for theexamplein3.2is givenbelow:

Step1 Start Step2
InputA Step3
InputB Step4
C=A+ B Step5
PrintC Step6 Stop

4.0 CONCLUSION

Theunitintroducedthe tools used torepresent an algorithm,known as programmingtools,suchasflowchartsandpseudocodes.Youwere alsointroducedtothe symbolnamesandmeaningof pictorial componentsof a flowchart.

5.0 SUMMARY

Thisunityhasdealt with programmingtools (flow-charts and pseudo code). Ithasalsoshowed thediagramsused and theEnglish-like statementsusedtorepresentanalgorithm. Theunitalsostatedthe advantages of bothpseudocodesandflowcharts.

6.0 TUTOR-MARKEDASSIGNMENT

a. Using theflowchartonly,designan algorithmto findthemean offive numbers.

b. Writethepseudocode of the flowchartin (a) above.

7.0 REFERENCES/FURTHERREADINGS

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

www.doc.ic.ac.uk/~wjk/C++Intro/

www.personal.kent.edu/~muhama/Algorithms

UNIT5 PROGRAM TESTING, DOCUMENTATION & MAINTENANCE

CONTENTS

1.0 INTRODUCTION

Thisunitintroducesyoutotesting,documentation,andmaintenanceof programsindetails.

2.0 OBJECTIVES

After completing thisunit youshouldbeableto:

- ✓ outlinewhatismeantbyprogramtestingandthe reasonwhyit maybe labour intensive
- ✓ explainprogramdocumentationandthetwomaj or reasonsfor programdocumentation
- ✓ explainwhyprogrammaintenanceisimportant.

3.0 MAINCONTENT

3.1 Program Testing

Programtestingisanintegral componentof softwaredevelopmentandit isperformedtodeterminetheexistence,quality, orgenuinenessofthe attributes of theprogramof application.

Programtestingisdoneina waythattheprogramisrunonsometest casesandtheresultsoftheprogram’sperformanceareexaminedto checkwhethertheprogramisworkingasexpected.Itisalsoimportant toperformatestprocessonevery condition or attributethatdetermines theeffective/correct functionality of the system. The testing process normallybeginswith selectingthetestfactor(s).Thetestfactors determinewhethertheprogramisworking correctly and efficiently.

Testing is generally focused on two areas: internal efficiency and external effectiveness.Thegoalof external effectivenesstestingisto verifythatthe softwareisfunctioning accordingtosystemdesign,and thatitisperformingallthenecessaryfunctionsorsub-functions.The goal of internal testing is to make sure that the computer code is efficient,standardised,andwell documented.Testingcanbea labor-intensiveprocess,duetoitsiterativenature.

1. **StructuralSystemTesting:** Thisisdesignedtoverify that the developed system and programsworkcorrectly. Itscomponents include:

Stress testing
Recoverytesting
Compliancetesting
Execution testing
Operationstesting
Securitytesting

TECHNIQUE	DESCRIPTION	EXAMPLE
STRESS	Determinethatthe system still performs with expectedvolumes	Sufficient disk space allocation Communication lines adequate
EXECUTION	System achieves desired level of proficiency	Transaction turnaround timeadequate Software/hardware use optimized
RECOVERY	System can be returned to an operational status after a failure	Induce failure Evaluate adequacy of backupdata
OPERATIONS	System can be executed in a normal operational status	Determine systems can runusingdocument JCL adequate
COMPLIANCE (TO PROCESS)	System is developed in accordance with standards and procedures	Standards followed Documentation complete
SECURITY	Systemisprotected inaccordancewith importance to organisation	Accessdenied Procedures inplace

2. **FunctionalSystemTesting**: Thisisdesignedtoensure that the system requirements and specifications are achieved. Its components include:

Requirementtesting
Error-handling testing
Inter-systemstesting
Paralleltesting
Regressingtesting
Manual-supporttesting
Controltest

TECHNIQUE	DESCRIPTION	EXAMPLE
REQUIREMENTS	System performs as specified	Prove system requirements Compliance to policiesregulations
REGRESSION	Verifies that anything unchanged still performs correctly	Unchanged system segments function Unchanged manual procedurescorrect
ERROR HANDLING	Errors can be prevented or detected, and then corrected	Error introduced into test Errorsre-entered
MANUAL SUPPORT	The people -computer interaction works	Manual procedures developed Peopletrained
INTER-SYSTEMS	Data is correctly passed from system tosystem	Intersystem parameterschanged Intersystem documentation updated
CONTROL	Controls reduce system risk to an acceptablelevel	File reconciliation procedureswork Manual controls in place
PARALLEL	Oldsystemandnew system are run and theresultscompared to detect unplanned differences	Oldandnewsystem can reconciled Operationalstatusof old system maintained

3.2 Program Documentation

Thisistheprocedureofincludingillustrationsor commentstoexplain linesorsegmentswithinthe program.Thisisnecessarysoasto understandtheprogram especially whentheprogramis long.

Thetwomajor reasons for documentation are:

Clarity: Itmakestheprogramtobeclear andunderstandableto theprogrammers.Even,theprogram writerwillfindit difficult understanding somepartsoftheprogramifit isnotproperly documented.
Extensibility: Documentation allowsfor easy amendment, extensionor upgradeoftheprogram.Documentation allowsother programmers(apartfromthewriter)tobeabletoworkonthe programmer.Weallknowthataprogrammer mightnotbe availableeverytime.

However, it is important to note that program documentation mustbeefficient.Thismeansthatcorrectdescriptions shouldbe attachedtothelinesand segmentswithin theprogram.Thisis necessarysoasnottomisleadotherprogrammersthatmight wanttoworkontheprograminthe future.

3.3 Program Maintenance

Programdevelopmentdoesnotreally end afterimplementation; itisstill importanttostillmonitor thesystemso astocontinually checkwhether theprogramisstill working accordingtoearlier specifications.Itisalso importanttocheck whetherthe programstillmeets currentneedsofthe user.Programmaintenanceisthe act of ensuringthe smoothand continuousworkingof theprograminthenatureof businessand dynamics of operation.The following are thereasonswhy program maintenance isnecessary:

1. Changes innature of business
2. Dynamicsof operation
3. Changes intechnology
4. Improvingthesize andefficiencyof code– refactoring

4.0 CONCLUSION

Thisunithasexplained someofthereasonswhyyou should maintain yourprogram. Ithasfurther explained twomajorreasonsforprogram documentation. Itintroduced programtestingaswellasthereasonsfor carryingoutinour program.

5.0 SUMMARY

Whatthis unitexplainsisthe reasonwhy youshouldmaintain your programsandwhyprogramtestingisalabourintensivetask. Italso explainedand gaveinstancesoftypesofprogrammingtestingand the techniquesusedincarryingthemout.

6.0 TUTOR-MARKEDASSIGNMENT

1. ExplainbrieflywhatyouunderstandbyProgramTesting.
2. Enumeratethecomponents of StructuralSystem Testing.
3. WhatdoyouunderstandbySystemRecovery?
4. Whatis thefunctionof Functional SystemTesting?
5. Givetwo examples of ErrorHandling.
6. What are thetwomajor reasonsfor documentation?
7. Whyisprogrammaintenance necessary?

7.0 REFERENCES/FURTHERREADINGS

GonnetandRicardoBaeza-Yates(1993).*HandbookofAlgorithmsand DataStructures*. InternationalComputer ScienceSeries.

Holmes,B.J (2000). Pascal Programming. Continuum (2nd ed).
Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

www.doc.ic.ac.uk/~wjk/C++Intro/

www.personal.kent.edu/~muhama/Algorithms

UNIT6 BASICDATATYPES

CONTENTS

1.0 INTRODUCTION

Inthisunit,youwillbeintroducedtothevariousformsinwhichdata can berepresentedDataStructures).You willalsobeintroducedtothe differentdatatypes,likeintegers,real numbers,characterdata type and stringdatatype.

2.0 OBJECTIVES

Bythetimeyoumusthavecompletedthisunit,youshouldbeableto:

- ✓ explainthefactthatdataexists ina varietyof forms outlinethedatatypes,which include numericand non-numericdatatypes
- ✓ outlinetheconstituentsof aninteger,real-numbers, character datatypeandstringdatatype.

3.0 MAINCONTENT

3.1Data and Programming

Mostprogramsaredesignedtomanipulatedatainorder toget anoutput. Dataexistinavarietyof forms.Examplesare20,000,000,whichmight be aday’ssales, simplified, limited, nameof anorganisation andsoon.

Dataserveasinputtomostprograms.Theformatorprocedureforinput specification withina programdependsonthenatureofdata.

3.2NumericData Types

They consistof wholenumericvalves.Examplesare:

1. **Integers:** Integers consistof positiveandnegativewholevalves.

Examples,are500,- 112,77etc.

Majorstandardintegerdatatypes are:

Bytes
Shortint
Integer
Word
Longint

Youcanfindout aboutall these inprogramming languages.

2. **RealNumbers:** Theseconsistofvalveswithfractional parts.Examplesare257.29, 20.10,11.00, etc.Floating-point numbers normally have two parts: the mantissa (thefractionalpart)andan exponent(thepowerto which thebaseofthenumber israised toinordertogive the correctvalveof thenumber).

For example: Thefloating-point representationof 49234.5is mantissa

Mantissa = 0.4923425
Exponent= 5
Sowe have 0.4923425E5.

Thestandardrealdatatypesare:

Real
Single
Double
Extended

Also readmoreaboutthese

3.3Non-NumericData Types

These are valvesthat are notnumbersinnature. Examples are

1. CharacterDataType

Thisconsistsofrepresentationsofindividualcharactersusing the American StandardCodeforInformationInterchange (ASCII).ASCII uses7-bitstorepresenteach character.

Character	ASCIICode
A	65
B	66
C	67

2. StringData Type

A string consists of a sequence of characters enclosed in single or doublequotation marks dependingontheprogramming language.

For example

- "Abiola"
- "Iama man"
- "1999"

4.0 CONCLUSION

Thisunithas examinedindetailthetypesof data inprogramming languages.Alsoyouhavebeenabletoknowmoreaboutthenumeric andnon-numericdatatypes, thestandardrealdatatypes,etc.

5.0 SUMMARY

Therearebasicallytwotypesofdatatypes,whicharenumericand nonnumericdatatypes.
Numericdatatypes are either integers or realnumbers.
Nonnumericdatatypesareeither characteror string.

6.0 TUTOR-MARKEDASSIGNMENT

Writeandexplainany exampleofa numericandnon-numericdatatype.

7.0 REFERENCES/FURTHERREADINGS

GonnetandRicardoBaeza-Yates(1993).*HandbookofAlgorithmsand DataStructures*. InternationalComputer ScienceSeries.

B.J. Holmes(2000). Pascal Programming. Continuum (2nd ed).

www.doc.ic.ac.uk/~wjk/C++Intro/

www.doc.ic.ac.uk/~wjk/C++Intro/

www.personal.kent.edu/~muhama/Algorithms

UNIT7 FUNDAMENTALDATA STRUCTURES

CONTENTS

1.0 INTRODUCTION

Thisunitisintendedtoshowyouthevariouswaysofrepresentingdata in analgorithm andinprogrammingas a whole.

2.0 OBJECTIVES

Attheendof thisunit youshouldbeableto:

- ✓ explaindatastructuresandgive relatedexamples outlinethedifferenttypesofdatastructures
- ✓ explainthedifferenttypesoflineardatastructuresandwhenthey are usedinthedesignof algorithms
- ✓ explaintheoperationsof thedifferenttypesof datastructures differentiatebetweentreesandgraphs.

3.0 MAINCONTENT

3.1 Introduction to Data Structure

Data structure is a means of organising related data items. Data structuresbecamenecessarytolearnthedesignofalgorithms.Since mostalgorithmsoperateon data,therefore,itis importanttounderstand thewaysoforganisingdatainthedesignandanalysisofalgorithms. Thedata structureto beused isdeterminedbytheproblem at hand. Data structure is a way to store and organize data in order to facilitate access and modifications. No single data structure works well for all purposes and so it is important to know the features of some of them

For instance,if youhavetoworkonalistof data, youwill needan arrayin thedesignof thealgorithm. Therearetwobasictypesofdatastructures; thesearelineardatastructuresandnon-lineardatastructures.Examples ofdata structuresarearrays (one-dimensionalor multidimensional), queues, stack, trees, linkedlistetc.

3.2 LinearData Structures

Array: An arraycanbedefined as sequencesofobjectsall ofwhich are ofthe sametype that are collectivelyreferredto bythe samename.Each individualarrayelement(thatiseachofthedataitems),canbereferred tobyspecifying the arrayname,followedbyan index(alsocalled a subscript),enclosedinparenthesis.ForinstanceLIST(1) LISTisthe name of the arraywhile 1 isthe index pointingto the data item on LIST. Therearetwotypesofarrays;onedimensionalarray(listorcolumn) andmulti dimensional arrays (table,matrix etc).

Arrays are used to implement mathematical vectors and matrices, as well as other kinds of rectangular tables. Many databases, small and large, consist of (or include) one-dimensional arrays whose elements are records. Arrays are used to implement other data structures, such as heaps, hash tables, queues, stacks, strings etc.

There are three ways in which the elements of an array can be indexed:
• 0 (zero-based indexing): The first element of the array is indexed by subscript of 0.
• 1 (one-based indexing): The first element of the array is indexed by subscript of 1.
• n (n-based indexing): The base index of an array can be freely chosen. Usually programming languages allowing n-based indexing also allow negative index values and other scalar data types like enumerations, or characters may be used as an array index.

Linkedlist: Alinkedlistisa sequenceofzeroormoreelementscalled **nodes,**eachcontainingtwo kindsof information: some data and oneor morelinks called**pointers**toothernodesof thelinkedlist.Inevery linkedlist,thereisaspecialpointerwhichiscalledthenullwhichis used toindicate theabsencesofanode successor.Also,it containsa special nodecalled the header; this node containsthe informationabout thelinkedlistsuchasitscurrentlength.

Stack: A stackisadata structureinwhichinsertionand deletion can onlybedoneatoneend(calledtheTOP).Ina stack,therearetwomajor processes called PUSH and POP. PUSH is the process of adding elementstothe stackwhilePOP istheprocessofdeleting elementsfrom thestack.This(stack's) scheme isreferredtoastheLast-In-First-Out (LIFO)scheme.Atypical/physicalillustrationofastackisapileof platesin acontainer.Stacksareusedin implementingrecursive algorithms.

Queue: Unlikethestack,aqueueisadata structurewithtwo ends,in which aninsertion ismadeataend(REAR) anda deletion isdoneatthe otherend(FRONT).A queue operatesaFirst-In-First-Out(FIFO) scheme. Atypicalandpracticalillustrationofthisdatastructureisa queue ina modernentry. Queuesareusedfor severalgraphproblems.

Heap: It is a partially ordered data structure that is used in implementingpriorityqueues. Apriorityqueueisa setofitemswith an orderable characteristic calledan item'spriority.Aheapcan also be definedas a binarytree with keys assignedtoitsnodes.

3.3 Non-Linear Data Structures

A**graph**consistsof twothings:

1. AsetVwhoseelementsare calledvertices,pointsor nodes and
2. AsetEof unorderedpairsofdistinctvertices, callededges.

AgraphisdenotedbyG(V,E)whenwewanttoemphasisethetwo parts of thegraphG.

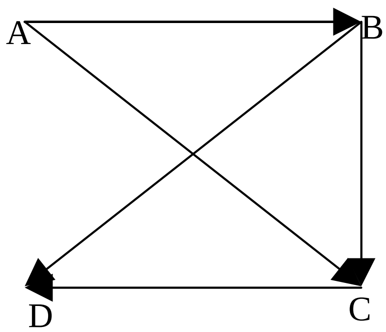
Therefore,agraph canbe pictorially(imaginarily)definedasa connectionofpointsina planecalledverticesor edges,someof which are connectedlinesegments called**edgesorarcs.**Itis formallydefined bya pairof twosets.

GraphG= (V, E).

Itismoreconvenienttolabeltheverticesofagraphwithletters,integer numbers or character strings.

ThefigurebelowrepresentsthegraphGwithfourverticesA,B,C&D
and five edges e1=(A,B), e2=(B,C),e3= (C,D),e4=(A,C), e5=(B,D).

Weusuallydenoteagraphbydrawingitsdiagramratherthanexplicitly listing itsvertices andedges.



3.4 TREES

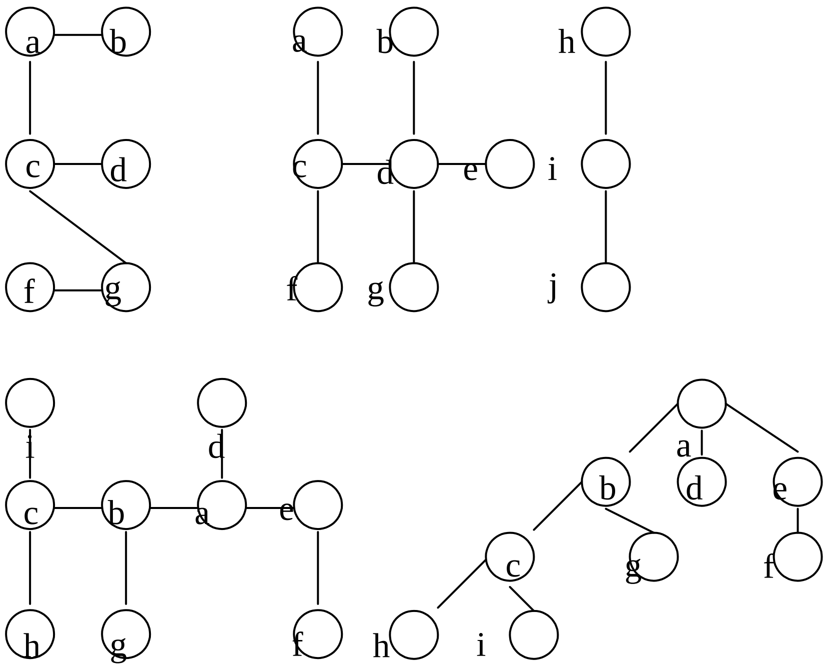
Agraphissaidtoby acyclicor cycle-free,sinceit containsno cycle, whileatreeisaconnectedacyclicgraph. Aforestisagraphwith no cyclehence,eachof itsconnectedcomponents is atree.

There aresome propertiespossessed by treeswhich graphsdo not have; forinstance,the numberofedgesinatreeis alwaysonelessthe number of itsvertices.

\E\= |V|-1.

 CIT237

The figures below are examples of trees with different numbers of vertices.



4.0 CONCLUSION

Inthecourseofthisunityouwereintroducedtotheconceptofdata structure andthevariousdatastructures that areavailable.

5.0 SUMMARY

Thisunit has shownthat:

- Datastructure is ameans of organisingrelateddataitems.
- Adatastructurecouldbe eitherlinear ornon-linear.
- Thebasiclineardatastructuresavailablearearray,linkedlist, stack, queue andheap.
- Thebasicnon-linear datastructures aregraph andtrees.

6.0 TUTOR-MARKEDASSIGNMENT

1. Describe how one can implement each of the following operationsonan array sothatthetimeit takes does notdependon thearray’ssize n.
   a. Deletetheith elementofanarray(1 i n).
   b. Deletetheithelementofasortedarray(theremaining arrayhas to staysorted, of course).

 58

2. If you have to solve the searching problem for a list of n numbers,how can youtakeadvantageofthefactthat thelistis knowntobe sorted?Giveseparateanswersfor
   a. Listsrepresentedasarrays
   b. Listsrepresentedaslinkedlists.

3. a. Show the stack after each operation of the following sequence thatstarts with theempty stack.
   Push(a),push(b), pop, push(c), push (d),pop
   b. Show the queue after each operation of the following sequence that starts with the empty queue:enqueue(a), enqueue(b),dequeue,enqueue(c),enqueue(d), dequeue

7.0 REFERENCES/FURTHERREADINGS

Levitin,A.(2003).*IntroductiontotheDesign&AnalysisofAlgorithms.* PublishedbyAddison-Wesley.

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

www.doc.ic.ac.uk/~wjk/C++Intro/

www.personal.kent.edu/~muhama/Algorithms

UNIT8 EXERCISEI

CONTENTS

1.0 INTRODUCTION

Thisunitisarecapofthemodulebecauseitwillexposeyoutothe practicalaspectof whatthemodulehas taught.

2.0 OBJECTIVES

At the end of this unit you should be able to develop a working algorithm anda corresponding flowchartfor thealgorithm.

3.0 MAINCONTENT

3.1 TheProblem

Usepseudocodesandaflowcharttorepresentanalgorithmtogenerate primenumbersbetween1 and200.

3.2 Solution

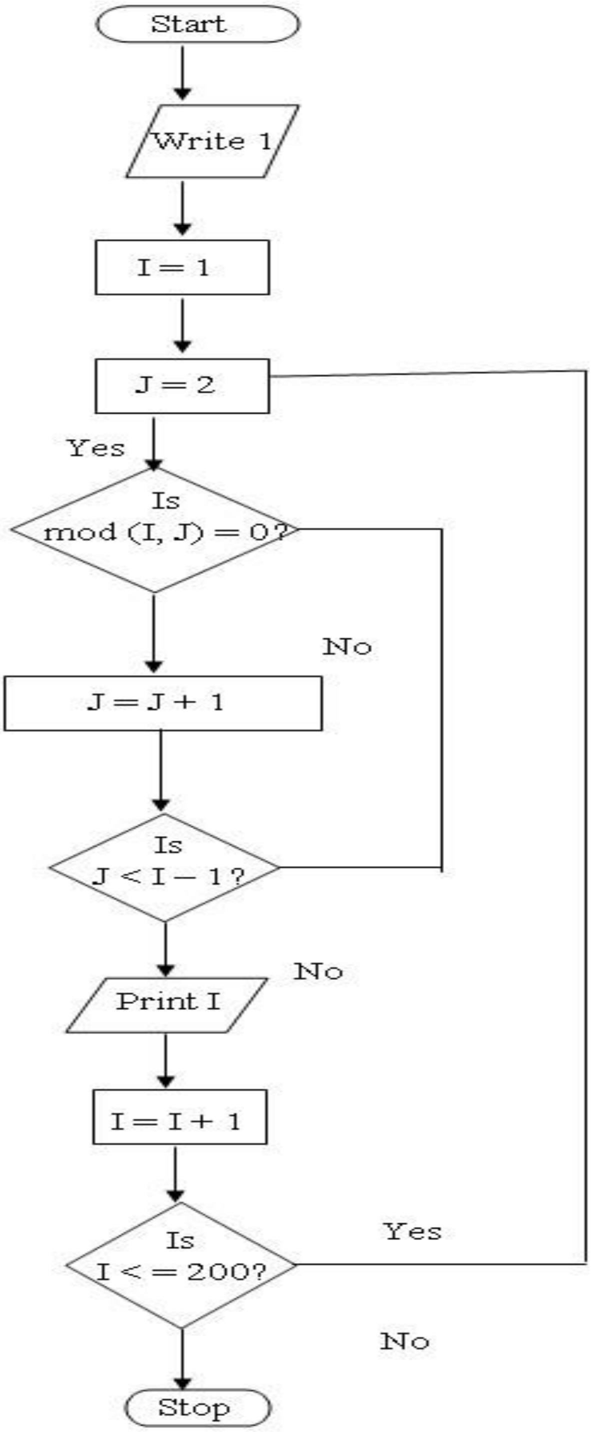
a. Pseudocodes

```
 I=1
WRITE1
FORI =2, I –1
    FORJ = 2,I–1
        IF MOD (I,J)= 0
        GOTO20
     ENDIF
  NEXT J
  WRITE“I”
20  NEXT I
```

Note

- Themodfunctionthatisusedinthepseudocodesisused inmostprogramminglanguagestogettheremainder when a number is dividedby another number.
- The facilitator should explain the pseudo codes to the students.

b. Flowchart


Start
Write 1
I = 1
J = 2
Yes
Is
mod (I, J) = 0?
No
J = J + 1
Is
J < I – 1 ?
No
Print I
I = I + 1
Is
I < = 200?
Yes
No
Stop

4.0 CONCLUSION

Theunitisapracticalapproachtothemodule andit isnecessaryfor you tohaveasolidexperienceinthedevelopmentofanalgorithmanda flow chart as a pre-requisite for the remaining partof thecourse.

5.0 SUMMARY

Youshouldbeabletodevelopanalgorithmandaflowchartforthe algorithm

6.0 TUTOR-MARKEDASSIGNMENT

Usepseudocodesandaflowcharttorepresentanalgorithmtofindthe averageofthefirst100numbers.

7.0 REFERENCES/FURTHERREADINGS

GonnetandRicardoBaeza-Yates(1993).*HandbookofAlgorithmsand DataStructures*. InternationalComputer ScienceSeries

Holmes,B.J (2000). Pascal Programming. Continuum, 2nd (ed).

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

www.personal.kent.edu/~muhama/Algorithms

MODULE2 PERFORMANCE ANALYSIS OF ALGORITHMS

UNIT1 PERFORMANCEANALYSISFRAMEWORK

CONTENTS

1.0INTRODUCTION

Thisunitwill introduceyoutothe criteriausedin analysingthe performance of analgorithm. As youwill see,thisunitisanintroduction tootherunitsinthismodule.

2.0OBJECTIVES

Attheendof thisunit, youshouldbe ableto:

- statethecriteria for estimatingthe runningtimeofanalgorithm estimatetherunningtimeof analgorithm
- list the efficiency attributes of an algorithm
- describe how time efficiency of an algorithm is measured.

3.0 MAINCONTENT

3.1 EfficiencyAttributes

The efficiencyattributes areused to analysethe performanceof algorithms. Therearetwotypes of algorithm efficiency attributes.

TimeEfficiency:- This indicateshow fastanalgorithm runs.

SpaceEfficiency: -Itdealswith the spacerequiredforan algorithmto runefficiently.Intheearlycomputing days,bothresources–time and space – werelimited.

3.2 MeasuringInputSize

Itiscertainthatallalgorithmsdonotrunatatimeonthesamenumber ofinput(s).For example,it takeslongertosortlarger arrays,multiply larger matrices,and so on.It isthen necessary to investigate an algorithm'sefficiency, as afunction of inputsizeparameter n. Selecting suchaparameterisnotdifficultinmostproblems.Forexample,itwill bethesizeof thelistfor problemsof sorting,searching,finding thelist's smallestelement,and most otherproblemsdealingwithlists.Forthe problemofevaluatingapolynomial$P(x)=a_nx^n +\ldots+a_0$ ofdegreen,it willbethepolynomial'sdegreeorthenumberofitscoefficients,which is larger byonethanitsdegree.

Of course, there are situations where the choice of a parameter indicating aninput sizeisnotreallyafactor.An exampleis computing theproductoftwo*n-by-n* matrices.Therearetwo naturalmeasuresof sizeforthis problem.Thefirst andmorefrequentlyused isthematrix order*n*.Buttheothernaturalcontenderisthetotalnumberofelements *N*inthematrices beingmultiplied.

Thechoiceof an appropriatesizemetric can beinfluenced byoperations ofthe algorithminquestion.For example,howshouldwemeasure an input'ssizefora spell-checking algorithm?Ifthe algorithmexamines individualcharactersofitsinput,then we should measurethe sizebythe number ofcharacters;if itworksbyprocessingwords,weshouldcount their numberintheinput.

3.3 Unitsfor MeasuringRunning Time

Inmeasuringtherunningtimeofalgorithm,itisnecessarytoidentify thebasic operationswithinthealgorithm. Thebasic operationsarethe mostimportantoperationsofthealgorithm.Afteridentifyingthebasic

operation,wethen computethenumber of timesthebasicoperationis executed.
As a rule, it is not difficult to identify the basic operation of an algorithm: it is usually the most time-consuming operation in the algorithm'sinnermost loop.For example,most sortingalgorithmswork bycomparingelements(keys)ofalistbeingsortedwitheachother;for such algorithms,thebasicoperationisakeycomparison.Asanother example, algorithms for matrix multiplication and polynomial evaluation require two arithmetic operations: multiplication and addition. On most computers, multiplication of two numbers takes longerthanaddition,makingtheformeranunquestionablechoicefor thebasicoperation.

Conclusively, an algorithm'stime efficiencycan be measuredby counting thenumberof times thealgorithm'sbasicoperation isexecuted oninputsofsize*n*. Thiswillbefullytreatedinunitthreeofthis module.

4.0 CONCLUSION

Inthis unit, youhavebeenintroducedto thefundamentalconceptsof analysingtheperformance of analgorithm.

5.0 SUMMARY

Havinggone throughthis unit, youareexpected tohavelearntthe following:

- The efficiency attributes of an algorithm are time efficiency andspace efficiency.
- Thetime efficiencyismeasured as afunctionoftheinput sizen
- The time efficiency is measured as a function of the numberoftimesbasic operationswereexecuted onan input.
- Therunning timeofanalgorithmisestimatedbasedon thebasicoperations.

6.0 TUTOR-MARKEDASSIGNMENT

1. What do you understand by the term "Running time of an algorithm"?Howisit measured?
2. What are the units for measuring the running time of an algorithm?

3. Describe space efficiency with respect to the running of an algorithm.

7.0 REFERENCES/FURTHERREADINGS

Holmes,B.J.(1997).*BASICProgramming–ACompleteCourseText.* Gp Publications.

www.eslearning.algorithm.com

Levitin, A. (2003). *Introduction to the Design and Analysis of Algorithms*. Addison- Wesley.

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

Tucker, A.B and Noonan, R.(2006).*Programming Languages* – Principles and Paradigms. (2nd ed). McGraw – Hill College.

www.personal,kent.edu/wmuhama/algorithms.

UNIT2 ORDEROFGROWTH

CONTENTS

1.0 INTRODUCTION

This unit introduces and explains order of growth and the different notationsthat areusedtorepresentorderof growthof algorithms.

2.0 OBJECTIVES

Bytheendof thisunit youshouldbe ableto:

- explainorder of growth
- explainthedifferentasymptoticnotations.

3.0 MAINCONTENT

3.1InformalIntroduction

Letusconsiderthetablebelowtoinformallydescribethegrowthof algorithmsbasedonsomestandardfunctions.

N	$\log_2^n$	n	$n\log_2^n$	n^2	n^3	2^n	n!
10	3.3	10^1	$3.3.10^1$	10^2	10^3	10^3	$3.6.10^6$
10^2	6.6	10^2	$6.6.10^2$	10^4	10^6	$1.3.10^{30}$	$9.3.10^{157}$
10^3	10	10^3	$1.0.10^4$	10^6	10^9		
10^4	13	10^4	$1.3.10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7.10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0.10^7$	10^{12}	10^{18}		

All thefunctions arebasedoninput sizen.Wecanseethatthefunction growingthe slowest among theseisthelogarithmic function.It grows so slowly, infact,thatweshouldexpectaprogramimplementingan algorithmwithalogarithmic basic-operationcounttorun practically instantaneouslyoninputsof all realisticsizes.

Ontheother endof thespectrum aretheexponentialfunction*2^n*andthe factorialfunction*n*.Boththese functions growso fast that theirvalues becomeastronomicallylarge, even for rather small values of*n*.

Algorithms performance is mostly represented by these functions becausethesefunctionsdescribethe performanceofthesemost algorithmsoninput size n.

3.1 o-Notation

Thisispronouncedas“littleohof”.Let (x)andg(x)betwofunctions of*x*.Each ofthefivesymbols aboveisintended tocomparetherapidity ofgrowthof and g.Ifwesay that (x)=o (g(x)),theninformallywe are sayingthat growsmore slowlythangdoeswhenxis verylarge.

Definition

We say that $f(x) = o(g(x))$ (f(x→ ∞) If limx → ∞, $f(x)/g(x)$ exists and is equal to 0.

Here are some examples:

1.X2 = o(x5)
2. Sin x = o (x)

3.14.709 √ x = o (x/2 + 7 cos x)

3.2O-Notation

Thisispronounced as“bigohof”.The secondsymboloftheasymptotic vocabularyisthe‘O’.Whenwesaythat (x)= O(g(x))wemean, informally,that certainlydoesnotgrowatafasterratethang.Itmight grow at the same rate or it might grow more slowly; both are possibilitiesthatthe ‘O’ permits.

Definition

We say that $f(x) = O(g(x))$ $(x \to \infty)$ if $\exists C$, xo such that $|f(x)| < Cg(x)$ $(\forall x > xo)$.
Thequalifier ‘$x \to \infty$’ will usuallybeomitted,sinceit willbeunderstood thatwewill mostoftenbeinterestedinlargevaluesof thevariablesthat are involved.

For example,it is certainlytruethat $\sin x = O(x)$,but even more can be said,namelythat$\sin x=O(1)$.Also$x^3+5x^2+77\cos x=O(x^5)$and$1/(1+x^2) = O(1)$.Nowwecan seehowthe‘o’ givesmoreprecise informationthatthe‘O’,forwe can sharpenthe lastexamplebysaying that$1/(1+x^2)=o(1)$.Thisissharperbecausenotonlydoesittellus that the function isbounded when x islarge,welearn that the function actually approaches 0as x .

ThisistypicaloftherelationshipbetweenOando.Itoftenhappensthat a‘O’ result issufficientforanapplication.However,thatmaynotbethe case,andwe mayneedthemoreprecise ‘o’ estimate.

3.3 **θ -Notation**

Thethirdsymbolof the languageof asymptotic isthe ‘**θ’.**Thisis pronounced,as“is thetaof”

Definition

Wesaythat $f(x)= \theta(g(x))$iftherearecontants$c_1>0, c_2>0, x_o$ such thatfor all $x> x_o$it is truethat $c_1g(x) <(x) <c_2g(x)$.

Wemight thensaythat *f*andg areof thesamerateof growth,onlythe multiplicative constantsareuncertain.Some examplesof the ‘**θ’**atwork are

$$(X+1)^2= \theta(3X^2)$$
$$(x^2+5x+7)/(5x^3+7x+2) = \theta(1/x)$$
$$3+ 2x= \theta(x¼)$$
$$(1+3/x)^x= \theta(1).$$

The‘**θ’**ismuchmoreprecisethaneitherthe‘O’orthe‘o’.Ifweknow that $(x)= \theta(x^2)$,thenweknowthat $f(x)/x^2$ staysbetweentwononzero constantsfor all sufficientlylargevaluesof x.Therateof growthof is established:it grows quadraticallywith x.

3.4 ~-Notation

Thisispronounced as“is asymptotically equal to”.Themost preciseof thesymbolsof asymptoticisthe~’‘ ’.Ittellsusthat notonlydo and go grow atthesamerate,butthatin fact /g approaches1as x- ∞.

Definition

We say that $f(x) \sim g(x)$ if $\lim_{x \to -\infty} f(x)/g(x) = 1$

Here are someexamples.

$$X^2 + x \sim x^2$$
$$(3x+1)^4 \sim 81x^4$$
$$\text{Sin}1/x \sim 1/x$$
$$(2x^3+5x+7)/(x^2+4) \sim 2x$$
$$2^x + 7\log x + \cos x \sim 2^x$$

Observetheimportanceof getting themultiplicative constantsexactly rightwhenthe~’‘ ’symbolisused.Whileitistruethat$x^2 = \theta(x^2)$,itis nottruethat$1x^2 = \theta(17x^2)$,but tomakesuch anassertionistouseabad style sinceno moreinformation is conveyed with the “17”than without it.

3.5 Ω-Notation

Thisispronounced,as “isomegaof”.Thelast symbolintheasymptotic setthatwewillneedisthe‘Ω ’whichisthenegationof‘o’.Thatisto say, $f(x) = \Omega(g(x))$ meansthatit isnottruethat (x) =o(g(x)).Inthe studyofalgorithmsforcomputers,the‘ Ω ’isusedwhenwewantto expressthethoughtthatacertaincalculationtakesatleastso-and-so longtodo.Forinstance,wecanmultiplytogether twonxnmatricesin time $\Omega(n^3)$.

4.0 CONCLUSION

Inthisunit,you havelearnt the five asymptotic symbolsforrepresenting order of growthof algorithms. .

5.0 SUMMARY

Thisunithasexplainedthefivefunctionsforcomparingthegrowth order of an algorithm. Thesefunctionsinascendingorder areo,O, , ,

6.0 TUTOR-MARKEDASSIGNMENT

Question:Usethe asymptoticsymbolsto comparethefollowing functions.Youmightneedtowritesmallprogramsbeforeyoucanget theorder of growth effectively.

1. f (x) andg(x) wheref(a) = $4a^2$+a+ 7andg (a) = a!
2. f (x) andg(x) wheref(x) = 4^xandg(x) = x^4
3. h(x) andi(x) whereh (x) = cos(x) andi (x) = sin (x)
4. f (a) andg (b) where f (x) = $(x^2)^2$andg(x) = x^4

7.0 REFERENCES/FURTHERREADINGS

Holmes,B.J.(1997).*BASICProgramming–ACompleteCourseText*. Gp publications.

www.eslearning.algorithm.com

Levitin, A.(2003) *Introduction to the Design and Analysis of Algorithms*. Addison- Wesley.

www.personal.kent.edu/wmuhama/algorithms.

Tucker, A.B and Noonan, R. (2006). *Programming Languages – Principlesand Paradigms*.(2nd ed).McGraw–HillCollege.

UNIT3 WORST-CASE, BEST-CASE AND AVERAGE-CASEEFFICIENCIES

CONTENTS

1.0 INTRODUCTION

Thisunit extends whatwe learntinUnit1 ofthismodule.Itdiscussesthe differenttechniquesthat can be used to measure the efficienciesof algorithms.

2.0 OBJECTIVES

Attheendof thisunit youshouldbeableto:

- explainthethreemethodsthatareusedtomeasuretheefficienciesof algorithms
- explainhowtoidentifybasicoperationswithin analgorithm.

3.0 MAINCONTENT

3.1Worst-Case

Analgorithmisitsefficiencyfortheworst-caseinputofsize*n*,which is aninput(orinputs)of size*n*for whichthealgorithmrunsthelongest amongallpossible inputsof thatsize.Thewaytodeterminetheworst-caseefficiencyofanalgorithmis,inprinciple,quitestraightforward:

Weanalysethe algorithm to seewhatkindof inputsyieldthelargest valueof thebasicoperations' countC(n) among all possibleinputsof sizenandthencomputethisworst-casevalueC_{worst}(n).Clearly,the worst-case analysisprovidesveryimportant informationabout an algorithm's efficiency by bounding its running time from above. In otherwords,itguaranteesthatforanyinstanceofsizen,therunning timewillnot exceedC_{worst}(n),its runningtimeontheworst-case inputs.

Asitwasmentionedin Unit1 ofthis module,weneed tocountthe number of**basicoperations**performedbythealgorithmonthe**worst- caseinput**

A*basicoperation*couldbe:

- Anassignment
- A comparison between two variables
- An arithmetic operation between two variables. The worst-case input is that input assignment for which the most basic operations are performed.

Example1

```
n:= 5;
loop
   get(m);
   n:= n -1;
until(m=0orn=0)
```

Worst-case:5iterations

Example2

```
get(n);
loop
   get(m);
   n:= n -1;
until(m=0orn=0)
```

Worst-case:niterations

Examples3of "inputsize":

a. Sorting:

n== Thenumber of itemstobesorted;
Basicoperation: Comparison.

 CIT237

b. Multiplication(of xand y):

n== Thenumber of digitsinxplus thenumber of digitsiny.
Basicoperations:singledigit arithmetic.

c. Graph “searching”:

n==thenumberofnodesinthegraphorthenumberofedgesinthe graph.
Counting theNumberof BasicOperations

Example4 Sequence:P andQare two algorithm sections:

Time(P ;Q) = Time(P)+ Time(Q)

Iteration:

```
while<condition >loop
  P;
end loop;
or
foriin 1..nloop
  P;
end loop
```

Time = Time(P)*(Worst-casenumber of iterations)

Conditional

```
if <condition >then
  P;
else
  Q;
end if;
```

Time = Time(P) if <condition >=**true**
Time(Q) if<condition >=**false**

Example5

```
foriin 1..nloop
  forjin 1..nloop
    if i< jthen
      swop (a(i,j), a(j,i));-- Basicoperation
    end if;
  end loop;
end loop;
```

 76

Time < n*n*1
= n^2

3.2 Best-Case

Thebest-caseefficiencyof an algorithm isaninput(or inputs) of size*n* forwhich thealgorithmrunsthe fastest amongallpossibleinputsofthat size.Accordingly,we cananalyse the best-caseefficiencyasfollows: First,wedeterminethekindofinputsforwhichthecount*C(n)*willbe thesmallestamong all possibleinputsofsize*n*.(Notethatthebest case doesnotmeanthesmallest input;itmeansthe inputofsize*n*forwhich the algorithm runsthefastest).Thenwe shouldascertainthevalueof *C(n)* on these most convenient inputs. For example, for sequential search,best-caseinputswillbelistsofsizenwith theirfirst elements equaltoa searchkey;accordingly. $C_{best}(n) = 1$.

Theanalysisofthebest-caseefficiencyisnotasimportantasthatofthe worst-case efficiency. Butit isnotcompletelyuseless either.

3.3 Average-Case

The**average-case**efficiencyseekstoprovideinformationonrandom input.Itiscalculatedbydividingallinstancesofsizenintoseveral classessothatforeachinstanceoftheclass,thenumberoftimesthe algorithm’sbasicoperation isexecutedis thesame. Thisthenmeansthat aprobabilitydistributionofinputsneedstobeassumedorobtainedso thattheexpectedvalueof thebasicoperation’scountcanbederived. Estimatingaverage-caseefficiencyisnotaneasytaskanditisdifficult forthislevel.Studentscanjustgetfamiliarwithknownaveragecase results.

4.0 CONCLUSION

This unit teaches the three methods of analysing the efficiency of algorithms.

5.0 SUMMARY

Thisunit has explainedhowtomeasure the efficienciesof algorithms.

6.0 TUTOR-MARKED ASSIGNMENT

Writethepseudocodesforthealgorithmtofindthelargestoutofn integers anduse worst-case todeterminetheefficiency.

7.0 REFERENCES/FURTHERREADINGS

Holmes,B.J.(1997).*BASICProgramming–ACompleteCourseText.* Gp Publications.

Levitin, A.(2003). *Introduction to the Design and Analysis of Algorithms.* PublishedbyAddison- Wesley.

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

Tucker, A.B and Noonan, R.(2006). Programming Languages – Principles and Paradigms. (2nd ed). McGraw – Hill College.

www.personal.kent.edu/wmuhama/algorithms.

www.eslearning.algorithm.com

UNIT4 P, NPANDNP-COMPLETE PROBLEMS

CONTENTS

1.0 INTRODUCTION

This unit will discuss the different categories of computational problems.

2.0 OBJECTIVES

Attheendof thisunit youshouldbeableto:

- distinguishbetweena polynomialandnon-polynomialproblem
- identifya non-polynomial–completeproblem
- understandsomebasicissues in algorithm timeefficiencies
- understand P, NP,NP–completeproblems.

3.0 MAINCONTENT

3.1 BasicDefinitions

Wesaythatan algorithmsolvesa probleminpolynomialtimeifits worst-case time efficiency belongs to $O(p(n))$ here $p(n)$ is a polynomialoftheproblem'sinputsizen.(Notethat sinceweareusing big-oh notationhere,problems solvablein,say,logarithmictime are solvablein polynomial time aswell).Problemsthat can be solved in polynomialtime are called**tractable**;problemsthat cannot be solved in polynomialtimeare called**intractable**.

Computational complexity classifies problems according to their inherentdifficulty.

3.2Pand NP Problems

Mostproblemsdiscussedinthisunitcanbesolvedinpolynomialtime by some algorithms. They include computing the product and the greatestcommondivisorof twointegers,sorting, searching(fora particularkeyin a listorforagivenpatterninatext string),checking connectivity andacyclicity of a graph,finding a minimum spanningtree, andfindingthe shortestpathsina weightedgraph.(You areinvitedto addmoreexamples tothislist.).

Informally, we can think about problems that can be solved in polynomialtime as the setthat computer sciencetheoreticianscall***P***.A moreformaldefinitionof*P*isthatit includesonly**decisionproblems**, whichareproblemswithyes/noanswers.

Class*P*isaclassof decisionproblemsthatcanbesolvedinpolynomial time by (deterministic) algorithms.This classof problemsiscalled **polynomial.**

The restriction of *P* to decision problems can be justified by the followingreasons.First,itissensibletoexcludeproblemsnotsolvable inpolynomialtimebecauseof theirexponentiallylargeoutput.Such problems arisenaturallye.g.,generatingsubsetsofagiven setorall the permutationsofndistinctitemsbutitisapparentfromtheoutsetthat they cannot be solved inpolynomialtime.Second,manyimportant problems that are not decision problems in their most natural formulationcanbereducedtoaseriesof decisionproblemsthat are easier to study. For example, instead of asking about the smallest numberofcoloursneededtocolourtheverticesofagraphsothatno two adjacent vertices arecoloured the same colour,we can ask whether there existssucha colouringof thegraph’svertices with nomorethan*m* coloursfor*m=1,2…*(Theproblemof vertexcolouringwithm colours iscalledthe**m-colouringproblems**)Thefirstvalueof*m*inthisseries forwhich thedecision problemof*m-colouring*hasasolution solvesthe optimisation versionofthegraph-colouringproblemaswell.

Itisnaturaltowonder whethereverydecisionproblem canbe solvedin polynomialtime.The answerto thisquestionturnsoutto beno.Infact, somedecision problems cannot be solved at all by any algorithm.Such problemsarecalled**undecidable**.AlanTuringgaveafamousexample in 1936.Theproblem in questionis called the**haltingproblem**: given a computerprogramandaninputtoit,determinewhethertheprogram will haltonthatinput or continueworkingindefinitelyonit.

Aretheredecidablebutintractableproblems? Yes,thereare,butthe number of known examples is small, especially of those that arise naturallyrather than beingconstructed for the sakeof atheoretical argument.

Therearealargenumber of importantproblems,however,for whichno polynomial-timealgorithmhasbeen found,nor hastheimpossibilityof suchan algorithm beenproved.TheclassicmonographbyM.Garey and D.Johnson(GJ79)containsalistofseveralhundredsuchproblems fromdifferent areasof computer science,mathematics,andoperations research. Here is just a small sample of some of the best-known problems thatfallinto thiscategory:

Hamiltoniancircuit: Determinewhether a givengraphhasa Hamiltonian circuit(apath thatstartsandends atthesame vertexand passesthroughall theother vertices exactly once).

Travelingsalesman: Findthe shortesttourthrough*n*citieswith known positive integerdistancesbetween them (find the shortest Hamiltonian circuit ina completegraphwithpositiveintegerweights).

Knapsackproblem: Findthemostvaluable subsetof*n*itemsof given positiveintegerweights andvaluesthatfitintoaknapsackof agiven positiveinteger capacity.

Partitionproblem: Given*n*positiveintegers,determinewhetherit is possibletopartition theminto twodisjoint subsets with thesamesum.

Binpacking: Givenn itemswhose sizes arepositive rationalnumbers notlargerthan*1*, putthemintothesmallestnumberof bins of size*1*.

Graphcolouring: Foragivengraph,finditschromaticnumber(the smallest number of coloursthat need to be assigned tothe graph’s verticessothatnotwoadjacentvertices are assignedthesamecolour.

Integerlinearprogramming: Findthemaximum (orminimum)value ofalinear functionofseveralinteger-valuedvariablessubjecttoafinite setof constrains inthe formof linear equalitiesand/or inequalities.

Anondeterministicalgorithmis atwo-stageprocedurethattakes asits input aninstance *I* of a decisionproblemanddoes thefollowing:

Nondeterministic(“guessing”)stage: An arbitrary string*S*isgenerated that canbethoughtofasa candidatesolution tothegiveninstance*I*(but maybecompletegibberishas well).

Deterministic(“verification”)stage: A deterministic algorithmtakes both*I*and*S*asitsinputandoutputsyesif*S*representsasolutionto instance*I*.(IfSisnotasolutiontoinstance*I*,thealgorithmeither returnsnoor is allowednottohaltatall.)

Wesaythata nondeterministicalgorithmsolvesadecisionproblemif andonlyifforeveryyesinstanceoftheproblemit returnsyes onsome execution.(Inotherwords,werequireanondeterministicalgorithmto becapableof “guessing”asolutionatleastonceandtobeabletoverify itsvalidity.And,ofcourse,wedonotwantittoeveroutputayes answer on aninstancefor whichtheanswer shouldbeno).Finally,a nondeterministicalgorithmissaidtobe**nondeterministicpolynomial** if thetime efficiencyofitscertification stage is polynomial.

Nowwecandefinetheclass of*NP*problems.

Class*NP* isthe classof decisionproblemsthatcanbesolvedby nondeterministicpolynomialalgorithms.Thisclassof problemsiscalled **nondeterministicpolynomial**.

Mostdecisionproblemsarein*NP*. Firstof all,thiscallincludesallthe problems in*P*;

$P \quad NP$,

Thisistruebecause,if aproblemisin*P*, wecanusethedeterministic polynomialtime algorithmthat solvesit inthe verification-stageofa nondeterministic algorithmthat simplyignoresstring*S* generatedin its nondeterministic(“guessing”) stage.But*NP* also containsthe Hamiltonian circuitproblem,thepartition problem,aswellasdecision versionsofthetraveling salesman,theknapsack,graphcolouring and manyhundredsof other difficult combinatorialoptimisation problems catalogedin (GJ79).Thehaltingproblem,onthe otherhand,is among therareexamples of decisionproblemsthat are knownnottobein*NP*.

3.3NP-CompleteProblems

Letusintroduceanother importantnotioninthecomputational complexitytheorythatof NP completeness.

Adecisionproblem*D*is saidtobe *NP*-completeif

1. it belongstoclass *NP*;
2. everyproblemin*NP*is polynomially reducibleto*D*.

Thefactthat closelyrelateddecision problemsarepolynomially reducibletoeachother isnotvery surprising.For example,letusprove thattheHamiltoniancircuit problemispolynomiallyreducibleto the decisionversionofthetravelingsalesmanproblem.Thelattercanbe statedastheproblem todeterminewhetherthere existsa Hamiltonian circuit inagivencompletegraphwithpositiveinteger weightswhose lengthisnotgreaterthanagivenpositiveinteger*m*.Wecanmapa graph*G*ofagiveninstanceof theHamiltoniancircuit problemtoa completeweightedgraph*G'* representinganinstanceof thetraveling salesman problembyassigning*1*astheweighttoeach edgein*G* and adding anedgeof weight*2*betweenanypair ofnotadjacentverticesin *G*.AstheupperboundmontheHamiltoniancircuitlength,wetake*m* *=n*,wheren isthenumberofverticesin*G* (and*G'*).Obviously,this transformation canbedoneinpolynomialtime.

Let*G*beayesinstanceoftheHamiltoniancircuitproblem.Then*G*has aHamiltoniancircuit,anditsimagein*G'* willhavelength *n*,makingthe imagea yesinstanceofthedecisiontraveling salesmanproblem. Conversely,ifwehaveaHamiltoniancircuitofthelengthnotlarger than*n*in*g'*,thenitslengthmustbeexactlyn(why?) andhencethe circuitmustbemadeupof edgespresentin *g*,makingtheinverseimage oftheyesinstanceofthedecisiontravelling salesmanproblembeayes instanceoftheHamiltoniancircuit problem. Thiscompletes theproof.

4.0 CONCLUSION

Thisunit showshowtodistinguishbetweendifferentcomputational problems.You have also been exposed to differentdecidable and undecidableproblems.

5.0 SUMMARY

Thisunit, teachesthat

- ✓ Problemsthat canbe solvedinpolynomialtime arecalled tractable while, thatwhichcannotbesolvedinpolynomialtimeare intractable The classof decisionproblemthat canbe solvedinpolynomialtime by(deterministic)algorithmsarecalledpolynomialproblemsand theyaremostly with yes/noanswer.
- ✓ Decisionproblemsthatcannotbesolved at all byany algorithmare calledundecidableproblems.

6.0 TUTOR-MARKEDASSIGNMENT

Agameofchesscanbeposedasthefollowingdecisionproblem:given alegalpositioningofchesspiecesandinformationaboutwhichsideis to move,determine whetherthatside can win.Isthisdecision problem decidable?

7.0 REFERENCES/FURTHERREADINGS

Holmes, B. J.(1997).*BASICProgramming–ACompleteCourseText.* Gp Publications.

Levitin, A.(2003). *Introduction to the Design and Analysis of Algorithms.* PublishedbyAddison- Wesley.

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

Tucker, A.B and Noonan, R.(2006). Programming Languages – Principles and Paradigms. (2nd ed). McGraw – Hill College.

www.personal.kent.edu/wmuhama/algorithms.

www.eslearning.algorithm.com

UNIT5 PRACTICALEXERCISE II

CONTENTS

1.0 INTRODUCTION

Thisunitpresentsanotherpracticalexerciseinorderforyoutofurther understandhowto analyse efficiencies of algorithms.

2.0 OBJECTIVES

Attheendofthisunit,youshouldhavehadfurtherunderstandinginto howtodetermineefficiencyof algorithms.

3.0MAINCONTENT

3.1 Exercise

Write the pseudocodes for the algorithm to find the average of the smallestandthelargestof nintegers.

3.2Solution

```
MIN=A( I )    1
For I =1ton
      If  MIN> A( I)
          TEMP = MIN                                   4n
           MIN=A( I ) A(
           I )= TEMP
      endIf
enddo
MAX= A( I)
For J = 2ton   4n
      If    MAX< A( I)
           TEMP1=MAX                              4n
            MAX=    A( J )
```

```
        A(J)=  TEMP
      endIf
  enddo
AVE = ( MIN+ MAX)/2
PRINT AVE

        = 1*4n*4n*1
```

Thebasicoperationsaredenotedby*.Wecansimplysaythatitisof $O(n^2)$.

4.0 CONCLUSION

Aclearerviewofanestimationofthenontimegrowthofanalgorithm has beenpresented.

5.0 SUMMARY

Basicoperationshavebeenclearlyidentified.Estimationofefficiency of algorithmsispracticallydiscussed.

6.0 TUTOR-MARKEDASSIGNMENT

Writethepseudocodesof thealgorithm toconvertanybinarynumber to decimal andestimatetheworse-caseruntimeefficiency.

7.0 REFERENCES/FURTHERREADINGS

Holmes, B.J.(1997).*BASICProgramming–ACompleteCourseText.* Gp Publications.

Levitin, A.(2003). *Introduction to the Design and Analysis of Algorithms.* PublishedbyAddison- Wesley.

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

Tucker, A.B and Noonan, R.(2006). Programming Languages – Principles and Paradigms. (2nd ed). McGraw – Hill College.

www.personal.kent.edu/wmuhama/algorithms.

www.eslearning.algorithm.com

MODULE3 SORTING AND SOME SPECIAL PROBLEMS

UNIT1 INTRODUCTIONTOSORTINGANDDIVIDE-AND-CONQUERALGORITHMS

CONTENTS

1.0 INTRODUCTION

Thisunitdiscussesthemeaningofsorting.Italsodescribestheconcept of divideandconquer algorithmsasproblem-solvingtechniques.

2.0 OBJECTIVES

Youareexpectedtobeabletoexplainthefollowingattheendofthis unit:

- meaningandsignificanceofsorting
- meaningandpracticalunderstandingofdivideand conqueralgorithms.

3.0 MAINCONTENT

3.1 Introduction to Sorting

Sortingistheprocessofarrangingasetofitemsorobjectsinincreasing or decreasingorder.

TheSignificanceof Sorting

- For orderlyanalysis andpresentationof items
- For locatinganitemor items withinaset
- For findingduplicate valuesor nearestpair withina set
- For findingtheintersectionor unionoftwoor more sets
- Sortingisalsousedasapartofmanygeometricalgorithms(eg convexhull, nearestpair of points inthe plane).

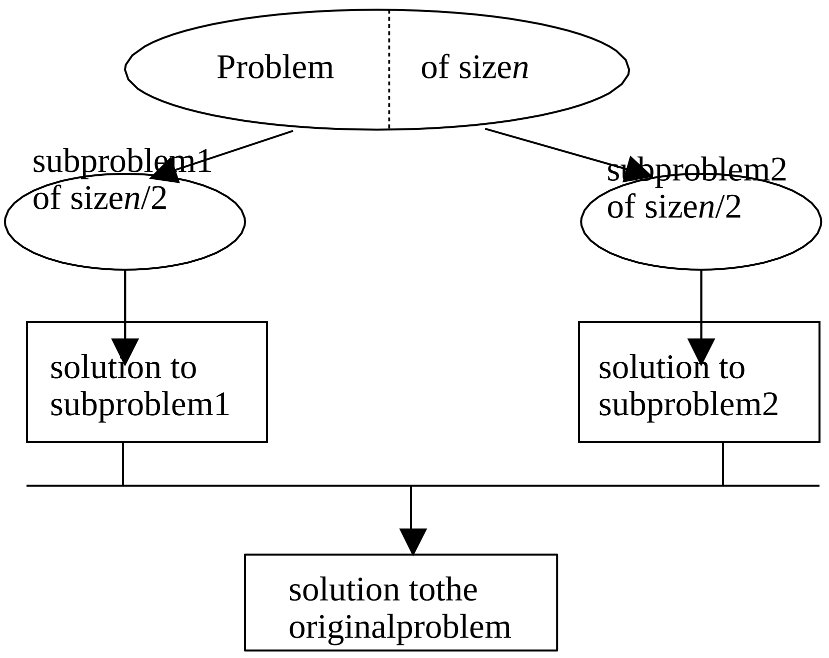
3.1 FundamentalsofDivide-and-Conquer Algorithms

Thisisamethodofdesigningalgorithmsthat(informally)proceedas follows:

Givenaninstanceof theproblemtobe solved,

- split thisinto several, smaller,sub-instances(*ofthesame problem*)
- independently solveeachof thesub-instances
- andthencombinethesub-instancesolutionssoastoyieldasolution for theoriginalinstance.

Thediagrambelow representsthedivide-and-conquer technique:



Examplesofalgorithmsthatadoptthestrategyofdivide-and-conquer algorithms aremergesort, quicksortandbinary search.

3.3 Practical Proof ofDivide-and-Conquer Algorithms

Consideran algorithm,*alpha* say,which isknown to solve all problem instancesof size*n*in atmost*cn*^2steps(where*c*is some constant).We thendiscoveranalgorithm,*beta*say, whichsolves thesameproblemby:

1.Dividinganinstanceinto 3sub-instancesofsize*n/2*.
2.Solving these 3sub-instances.
3.Combiningthethreesub-solutionstaking*dn*steps todothis.

Suppose our original algorithm, *alpha*, is usedtocarryoutstep2.

Let

T(alpha)(n) = Runningtimeofalpha
T(beta)(n) = Runningtime ofbeta
Then,
T(alpha)(n) = cn^2(by definition ofalpha)

But

T(beta)(n)=3 T(alpha)(n/2) + dn

=(3/4)(cn^2) + dn
Soif
dn<(cn^2)/4(i.e. *d<cn/4*)

then

*beta*is*faster thanalpha*

Inparticularforalllargeenough*n*,(*n>4d/c=Constant*),*beta*isfaster than*alpha*.

Thisrealisationof*beta*improvesupon*alpha*byjustaconstantfactor. Butiftheproblemsize, *n*, is largeenoughthen

n >4d/c
n/2 >4d/c
...
n/2^i>4d/c

whichsuggests thatusing*beta*insteadof *alpha*for the"*solves these"* stage*repeatedlyuntilthesub-sub-sub..sub-instancesareofsizen0<= (4d/c)* will yielda still faster algorithm.

So consider the followingnewalgorithm for instancesofsize*n*

```
procedure gamma(n:problem size) is
  begin
    if n<= n^-0then
      SolveproblemusingAlgorithm alpha;
     else
      Split into 3sub-instancesofsizen/2; Use
      gammatosolveeachsub-instance;
      Combinethe3sub-solutions;
     end if;
  end gamma;
```

Let*T(gamma)(n)* denotethe runningtimeofthisalgorithm.

$$T(gamma)(n) = \begin{cases} cn^2 & \text{if } n <= n0 \\ 3T(gamma)(n/2)+dn & \text{otherwise} \end{cases}$$

4.0CONCLUSION

Youshouldhaveunderstoodsorting andwhatthedivideandconquer techniqueis allabout.

5.0SUMMARY

Sortingisthearrangementof itemsina predeterminedorder. Divide-and–Conquer algorithmsrequire dividing problems into sub-instances,solvingthesesub-instances andcombining the solutionstothesub-instancesto formtheoriginalsolution.

6.0 TUTOR-MARKEDASSIGNMENT

1. Writea pseudocode for a divide-and-conquer algorithm for finding a position ofthelargestelementin anarrayof *n*numbers.
2. Findoutaboutthebrute-forcealgorithm andcompareitwith the divide-and-conqueralgorithm.

7.0 REFERENCES/FURTHERREADINGS

Holmes, B.J.(1997).*BASICProgramming–ACompleteCourseText.* Gp Publications.

Levitin, A.(2003). *Introduction to the Design and Analysis of Algorithms.* PublishedbyAddison- Wesley.

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

Tucker, A.B and Noonan, R.(2006). Programming Languages – Principles and Paradigms. (2nd ed). McGraw – Hill College.

www.personal.kent.edu/wmuhama/algorithms.

www.eslearning.algorithm.com

UNIT2 MERGESORT

CONTENTS

1.0 INTRODUCTION

Thisunit describesmergesort sorting technique.Itexplains merge sort as anexampleof divide-and conquer algorithm.

2.0 OBJECTIVES

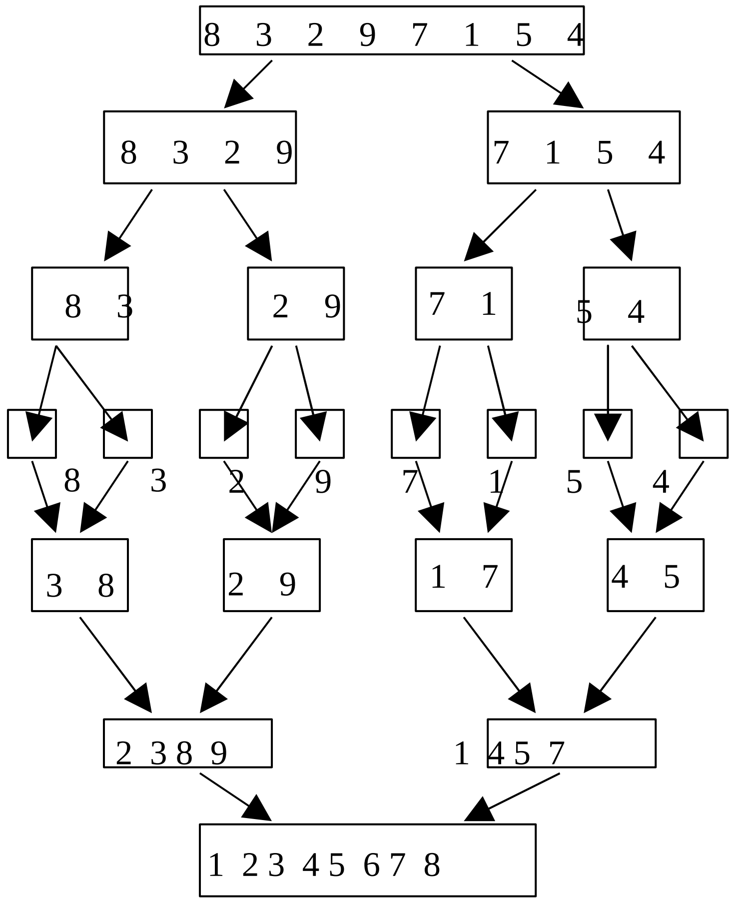
Attheendof thisunit, youshouldbe ableto:

- givethepracticalmeaningof merge sort
- presentthemerge sort algorithm
- statetheperformancesofthemerger sortalgorithm.

3.0 MAINCONTENT

3.1 Understanding Mergesort

Mergesortisanexampleof anapplication that adoptsthe strategyofthe divide-and-conquer technique.It sortsagivenarrayE[0..n –1]by dividingit intotwohalvesE[0..[n/2] –1] andA[[n/2]..n–1],sorting eachof themrecursively, andthenmergingthetwo smaller sorted arrays into a singlesortedone.This figurerepresents themergesorttechnique.



3.2 TheMergeSort Algorithm

ALGORITHMMergesort(E[0..*n*–1])

//SortsarrayE[0..*n*–1] byrecursivemerge sort

//Input: An arrayE[0..*n*–1] oforderableelements

//Output: ArrayE[0..*n*–1] sortedinnondecreasingorder if

n> 1

copyE[0..[*n*/2] –1]to B[0..[*n*/2] –1]

copyE[[*n*/2]..n–1]to C[0..[*n*/2] –1]

Mergesort (B(0..[*n*/2–1])

Mergesort (C[0..[*n*/2] –1])

Merge(B, C, E)

Themerging oftwo sorted arrayscan bedone asfollows: Two pointers (arrayindices)areinitialisedtopoint tothefirst elementsof the arrays being merged. Then the elements pointed to are compared and the smallerof themisaddedtoanew arraybeingconstructed; afterthat,the indexof thatsmallerelementisincrementedtopointtoitsimmediate successor inthe arrayit wascopiedfrom.Thisoperationiscontinued untiloneofthe twogivenarraysis exhausted,andthen the remaining elementsoftheother array arecopiedtotheendof thenewarray.

```
ALGORITHMMerge(B[0..p–1],C[0..q–1],E[0..p+ q–1])
    //Mergestwosortedarrays into one sortedarray
    //Input: Arrays B[0..p–1] and C[0..q–1] bothsorted
    //Output: Sorted arrayA[0..p+ 1–1] oftheelementsof B andC
    i←0';j ←0;k ← 0
    whilei< pandj< qdo if
    B[I]  C[j]
        E[k]←B[I];i← i+ 1
    elseA[k] ←C[j];j←j+ 1
    k←k+ 1
    if i= p
    copy C[j..q–1]toE[k..p+ q–1]
    elsecopy B[i..p–1] to E[k..p+q–1]
```

3.3 TheEfficiencyof the MergeSort Algorithm

Theefficiencyofmergesortis
$C_{worst}(n)$ (nlogn)

Thedetailsofhowthiswasarrivedatwill be studiedina futurestudyin Algorithm Design.

4.0CONCLUSION

Themeaning andalgorithm of mergesorthasbeenpresented.The worse-case efficiencywasalsodiscussed.

5.0SUMMARY

Mergesortadopts thestrategyof divide-and- conquer.
Itrequiresdividingthe arrayintotwohalves and sortingthem recursively,thenmerging thetwo smaller sorted arraysintoa single one.
Merge sort algorithm is divided into two- the merge sort(for splittingandsorting halves)andthemerger(formerging twohalves together).

6.0TUTOR-MARKEDASSIGNMENT

1. Applymergesortto sortthelistE, X, A, M, P. L.Ein alphabeticalorder.
2. Howstableis thedivide-and-conquer algorithm?

7.0 REFERENCES/FURTHERREADINGS

Holmes,B. J . (1997).*BASICProgramming–A CompleteCourse Text*. Gp Publications.

www.eslearning.algorithm.com

Levitin,A.(2003). *Introduction totheDesignandAnalysisof Algorithms.* AddisonWesley.

www.personal.kent.edu/wmuhama/algorithms.

Tucker, A.BandNoonan,R . (2006). *Programming Languages – Principlesand Paradigms*. (2nd ed). McGraw–Hill College.

http://mathworld.wolfram.com/QueensProblem.html.

http://mathworld.wolfram.com/QueensProblem.html..

 CIT237

UNIT3 QUICKSORT

CONTENTS

1.0 INTRODUCTION

Inthisunit,youwillbeintroduced toanewsortingtechnique calledthe Quicksort whichcouldbeusedin rearrangingelementsofanygiven array.

2.0 OBJECTIVES

Bytheendof thisunit, youshouldbeableto:

evaluatetheprocedures involvedinaquicksort algorithm
transversearrayusingthequicksort algorithm.

3.0 MAINCONTENT

3.1 Introduction to Quick Sort

Quick sortisanotherimportantsorting algorithmthatisbasedonthe divide-and-conquer approach.Unlikemerge sort,whichdividesits input'selements accordingtotheirposition inthe array,quick sort divides them according to their value. Specifically, it rearranges elementsof agiven arrayA[0..n – 1] to achieveits partitionasituation where all theelements beforesomepositionsare smallerthanorequalto A[s] and alltheelementsafterpositionsare greaterthanorequalto a[s]:

A[0]… a[s –1]   a[s] a[s + 1]…a[n–1]
All are A[s]   all are A[s]s

Obviously,afterapartitionhasbeenachieved,A[s]swillbeinitsfinal positioninthesortedarray,andwecancontinuesortingthetwosub

 96

arrays of the elements preceding and following A[s]s independently (e.g.bythesamemethod).

3.2 TheAlgorithm

ALGORITHMquicksort(A[l..r])
//Sortsa subarraybyquicksort
//Input: Asub arrayA[l..r] of A[0..n–1],definedbyitsleft and right indices
// l andr
//Output: Thesub arrayA[l..r] sortedina nondecreasingorder if
i< r
s ←Partition([Al..r])//sisa split position
Quicksort(A[l..s–1])
Quicksort(A[s+ 1..r])

ApartitionofA[0..n–1]and,moregenerally,ofitssubarraya[l..r](0 l<r n–1)canbeachievedbythefollowingalgorithm.First,we selectanelementwithrespecttowhosevaluewearegoingtodividethe subarray.Becauseofitsguidingrole,wecallthiselementthepivot. Thereareseveraldifferentstrategiesforselectingapivot;wewillreturn tothisissuewhenweanalysethealgorithm'sefficiency.Fornow,we usethesimpleststrategyofselectingthesubarray'sfirstelement:p= A[l].

Threesituationsmayarise,dependingonwhetherornotthescanning indiceshavecrossed.IfscanningindicesIandjhavenotcrossed,i.e.,I <j, we simply exchange A[i] and A[j] and resume the scans by incrementingI anddecrementing jrespectively:

i j←
P all are p p . .. p all are

If thescanningindiceshave crossedover,i.e.I >I,wehavepartitioned thearray afterexchangingthepivot withA[j]:

P all are p p all are p

Finally,if thescanningindicesstopwhilepointingtothesame element, i.e.,I=j,thevaluetheyarepointingtomustbeequalto p(why?).Thus, we have partitionedthearray:

P all are p = p all are p

Wecancombinedthelastcasewiththecaseofcrossed-overindices(I> j)byexchangingthepivot with A[j] wheneverI j.

Here is apseudocode implementing thispartitioningprocedure.

```
ALGORITHM          Partition (A[l..r])
    //Partitionsa subarraybyusingitsfirstelementas a pivot
        //Input: Asub arrayA[l..r] of A[0..n–1],definedbyits left
    and right
    //   indiceslandr(l<r)
    //Output: a partition of A[l..], with thesplitposition returnedas
    //   thisfunction'svalue
    p← a[l]
    i←l; j←r + 1
    repeat
        repeatI ←I + 1until A[I]   p
        repeatj←j–1until A[j]  p
        swap(A[I],A[j])
        until I   j
        swap(A[I],A[j])  //uno lastswapwhenI   j
        swap(A[l],A[j]
        returnj
```

3.3 TheEfficiencyofQuick SortAlgorithm

TheefficiencyofQuicksortis

$C_{worst}(n)$ (n^2)

$C_{best}(n)$ $(n\log_2 n)$

$C_{avg}(n) =0$

Thedetailsofhowthiswasarrivedatwill be studiedinthestudyin Algorithm Design.

4.0 CONCLUSION

Thisunithaspresentedanothertypeofsortingtechniquecalledthe quick sort technique which dividesits important elements according to their position or valuetothe array.

5.0 SUMMARY

Quicksortis another important sortingalgorithm thatisbasedon thedivide-and-conquer approach.

Itdivides theinput elements accordingtotheir respectivevalues.

6.0 TUTOR-MARKEDASSIGNMENT

a. Design analgorithm torearrange elementsof a given arrayof n realnumberssothat allitsnegativeelementsprecedeallits positiveelements. Your algorithm shouldbebothtime andspace-efficient.

b. Applyquicksortto sorttheE, X, A, M,P,L, E

7.0 REFERENCES/FURTHERREADINGS

Cormen, T.H., Leiserson,C.F. Rivest, R.L.,Stein,C. (2009). Introduction toAlgorithms (3rd ed). M/T Press, Cambridge.

Goodrich, M.T., and Tamassia, R. (2002). *AlgorithmDesign. Foundations, Analysis,and Internet Examples*. NewYork:John Wiley&Sons.

Tucker, A.BandNoonan,R. (2006). *Programming Languages – Principlesand Paradigms*. (2nd ed). McGraw–Hill College.

http://mathworld.wolfram.com/QueensProblem.html.

http://mathworld.wolfram.com/QueensProblem.html.

UNIT4 BINARYSEARCH

CONTENTS

1.0 INTRODUCTION

Thisunitintroducesthebinarysearch algorithmaswell asits efficiency and theproceduresinvolvedin traversing theelementsofanygiven array. Thebinary searchisan effectivewayof traversing arrays.

2.0 OBJECTIVES

Attheendof thisunit, youshouldbe ableto:

- explainhowthebinary searchoperates
- explain how the binary search is used to traverse an array of elements
- explainthealgorithm of thebinarysearch
- explaintheefficiencyof thebinarysearch.

3.0 MAINCONTENT

3.1 Understanding the BinarySearch

TheBinarySearch

Thebinary searchisaremarkably efficient algorithm forsearchingina sorted array.Itworksbycomparinga searchkeyK withthearray’s middle element A[m].Iftheymatch,the algorithmstops; otherwise,the sameoperationisrepeatedrecursivelyforthefirsthalfofthearrayifK < A[m] and for thesecondhalfif K>[m]:

K

A[0]. . . A[m–1] A[m] A[m+ 1] . .. .A[n–1]

Searchhere if K < A[m]   searchhereif K > A[m]

As anexample,letus applythebinary searchtosearching for K= 70in thearray

314 27 3139 4255 7074 81 8593 98

Theiterationsof thealgorithm aregiveninthefollowingtable:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Value	3	14	27	31	39	42	55	70	74	81	85	93	98
Interation 1	l						m					r	
Interation 2	l						m					r	
Interation 3	l,						m					r	

3.2 TheAlgorithm

Although thebinarysearch isclearlybased on arecursiveidea,it can be easily implemented asanonrecursive algorithm,too.Hereisapseudo code for thisnonrecursiveversion:

```
ALGORITHM            BinarySearch(A[0..n–1],K)
    //Implements nonrecursivebinarysearch
    //Input: An arrayA[0..n–1] sortedinascendingorder and
    //          a searchkeyK
    //Output: Anindexof thearray'selementthatis equaltoK
    //     or –1if there is nosuchelement l
    ←0;r ←n–1
    whilel   r do
        m← [(l+ r)/2]
        if K=A[m]returnm
        elseif K< A[m] r ←m–1
        elsel←m+ 1
        return–1
```

3.3 TheEfficienciesofthe BinarySearch

Theefficiencies of thebinarysearch are

$C_{worst}(n)$ (logn)
$C_{best}(n) = 1$
$C_{avg}(n) =(\log_2 n)$

Thedetailsofhowthiswasarrivedatwill be studiedininAlgorithm Design.

4.0 CONCLUSION

Thisunithaspresentedanother effectivewayoftransversing arrays through the binary search algorithm which works by comparing a searchkeywiththe arrays' middleelement A[m]**.**

5.0 SUMMARY

Thisunit has discussedthebinarysearch
Thebinary searchisbasedonrecursivedatawhilethe implementationisbasedonnon-recursivealgorithm.

6.0 TUTOR-MARKEDASSIGNMENT

Writea programin anyprogramming languagetoimplementthebinary searchonany setof integers.

7.0 REFERENCES/FURTHERREADINGS

Cormen, T.H., Leiserson,C.F. Rivest, R.L.,Stein,C. (2009) .*Introduction toAlgorithms*(3rd) CambridgeM/T Press

Goodrich, M.T., and Tamassia, R. (2002). *AlgorithmDesign*. *Foundations*, *Analysis,and Internet Examples*. NewYork: John Wiley&Sons.

Tucker, A.BandNoonan,R .(2006). *ProgrammingLanguages– Principlesand Paradigms*. (2nded). McGraw–Hill College.

http://mathworld.wolfram.com/QueensProblem.html.http://mathwo

rld.wolfram.com/QueensProblem.html.

UNIT5 SELECTIONSORT

CONTENTS

1.0 INTRODUCTION

Thisunitintroducesyou to the selectionsort algorithmused in scanning the all thegiven elementsof an arraytofindits smallest element and substituteit withinthefirstelement.

2.0 OBJECTIVES

Attheendof thisunit, youshouldbe ableto:

- describehowselectionsortoperates
- explainhowselection sortisusedtotransverse elementsinanarray describean algorithm oftheselection sort.

3.0 MAINCONTENT

3.1 Selection Sort

We start selection sort by scanning the entire given list to find its smallestelement andexchangeit withthefirstelement,puttingthe smallest elementinitsfinalposition inthesortedlist.Thenwescanthe list,startingwiththesecondelement,tofindthesmallestamongthelast n–1 elementsandexchangeitwiththesecond element,putting the second smallest elementinitsfinalposition.Generally, ontheith pass through the list, which we number from 0 to n – 2, the algorithm searchesforthe smallest item among the last n – Ielements and swapsit with Ai:

$$A_0 \quad A_1 \quad \ldots \quad A_{i-1}\ A_i, \ldots, A_{min}, \ldots, A_{n-1}$$

Intheir finalpositions thelastn-1elements.

After n–1passes,thelistis sorted.

3.2 TheAlgorithm

Here is apseudocodeofthisalgorithm,which,for simplicity, assumes thatthelistis implementedas anarray.

```
ALGORITHMSelection Sort(A[0..n–1])
    //Thealgorithm sortsa givenarrayby selectionsort
    //Input: An arrayA[0..n–1] of orderableelements
    //Output: ArrayA[0..n–1] sortedinascendingorder for
    i←0ton–2do
    min ←i
    for j←i+ 1ton–1do
    if A[j]< A[min] min ←j
    swapA[i] andA[min]
```

As anexample,theaction of thealgorithm onthelist89,45,68,90,29, 34, 17isillustratedinfigure

89	45	68	90	29	34	17
17	45	68	90	29	34	89
17	29	68	90	45	34	89
17	29	34	90	45	68	89
17	29	34	45	90	68	89
17	29	34	45	68	90	89
17	29	34	45	68	89	90

Figure:Selection sort'soperationonthelist89,45,69,90,29,34,17. Eachlinecorrespondstooneiterationof thealgorithm,i.e.,,apass throughthelist'stailtotherightof theverticalbar;an element inbold indicatesthesmallest elementfound.Elementstotheleftof thevertical bararein theirfinalpositions andarenot consideredin thisand subsequentiterations.

3.3 TheEfficiencies

Thus,selectionsortisa (n^2)algorithmonallinputs.Note,however, thatthenumber of keyswapsisonly (n)or,moreprecisely,n–1(one for eachrepetitionofthe Iloop).Thispropertydistinguishesselection sortpositivelyfrommanyother sorting algorithms.

4.0 CONCLUSION

Bynow, you shouldbe abletosort anarrayofelementsinadesired order, usingtheselectionsortalgorithm.

5.0 SUMMARY

Thisunit further explains selectionsort, itsalgorithm, and efficiencies.
Thenumberof key swapsisonly (n) or, moreprecisely,n–1 whichdistinguishesit fromother sortingalgorithms.

6.0 TUTOR-MARKEDASSIGNMENT

a. Illustrate selection sort using any set of sample data
b. Writea programin anyprogramming languagetoimplementa selection sortonany setof integers.

7.0 REFERENCES/FURTHERREADINGS

Cormen,T.H.,Leiserson,C.F.Rivest,R.L.,Stein,C.(2009)*Introduction toAlgorithms* (3rd ed). Cambridge:M/T Press.

Goodrich,M.T.,andTamassia,R.(2002).*AlgorithmDesign.Foundations, Analysis,and InternetExamples*. NewYork: JohnWiley&Sons.

Tucker A.B and Noonan R. (2006). *Programming Languages – Principlesand Paradigms*. (2nd ed). McGraw–Hill College.

http://mathworld.wolfram.com/QueensProblem.html.http://mathwo

rld.wolfram.com/QueensProblem.html.

UNIT6 BUBBLESORT

CONTENTS

1.0 INTRODUCTION

Thebubble sortisalsoanothereffectivewayinthevarietyof sort techniquesavailableinprogramming.Itwill beintroducedinthisunit as well as itsalgorithm.

2.0 OBJECTIVES

Attheendof thisunit youshouldbeableto:

- explainhowbubblesortoperates
- explainhowbubblesortisusedtotransversearrayofhomogenous elements
- describethealgorithm ofbubblesort
- explaintheefficiencyof thebubblesort algorithm.

3.0 MAINCONTENT

3.1 Understanding Bubble Sort

Another brute-forceapplicationtothesortingproblemisto compare adjacentelementsofthelist and exchangethemiftheyareoutof order. By doing it repeatedly,we end up “bubblingup”thelargest element to thelast positionon thelist.Thenext passbubblesupthe secondlargest element,andsoonuntil, after n–1passes,thelistissorted.PassI (0 I n–2) of bubblesortcanbe representedbythefollowingdiagram: A0,

$\ldots, Aj \leftrightarrow Aj+1, .., A_{n-i}-1A_{n-i} \quad \ldots \quad A_{n-1}$

Theaction of thealgorithm onthelist89,45,68,90,29,34,17is illustratedas anexample.

Bubble Sort

89 ↔	45	68	90	29	34	17
45	89 ↔	68	90	29	34	17
45	68	89 ↔	90 ↔	29	34	17
45	68	89	29	90 ↔	34	17
45	68	89	29	34	90 ↔	17
45	68	89	29	34	17	|90

45 ↔	68 ↔	89 ↔	29	34	17	|90
45	68	29	89 ↔	34	17	|90
45	68	29	34	89 ↔	17	|90
45	68	29	34	17	89	90 etc

Thefirsttwopassesof bubblesortonthelist89,45,68,90,29,34,17. Notethatanew line isshownaftera swapoftwo elementsisdone.The elementstotherightoftheverticalbarareintheirfinalpositionsand are not consideredin subsequentiterationsof thealgorithm.

3.2 TheAlgorithm

Here is apseudocode of thisalgorithm.

ALGORITHMbubblesort(A[0..n–1])

```
//Thealgorithm sortsarrayA(0..n–1] bybubblesort
//Input: An array a[0..n–1] of orderableelements
//Output: ArrayA[0..n–1] sortedinascendingorder for
I ←0ton–2do
for j←0ton–2–I do
if A[j+ 1]< A[j]swapA[j] andA[j+ 1]
```

3.3 TheEfficiency

Itis also in Θ (n^2) in the worst and average cases.In fact, even among elementarysorting methods,bubble sortisaninferiorchoice,and,ifit were notfor itscatchyname, youwouldprobablynever hear of it.

4.0 CONCLUSION

Bynowyoushouldbeabletouse thebubblesort, havingdiscovered thatitis afastandeasywaytotransverse anarrayofanygivensetof elements.

5.0 SUMMARY

Intheunit youhave justconcluded,thefollowingwereexamined:

Thebubblesortalgorithm.
Itswayof sorting elements of anygivenarray.
Bubblesort effectiveness.

6.0 TUTOR-MARKEDASSIGNMENT

a. Writea programin anyprogramming languagetoimplementthebubble sortalgorithm on any setof integers.
b. Determine the better one between Quick sort and Bubble sort

7.0 REFERENCES/FURTHERREADINGS

Cormen, T.H., Leiserson,C.F. Rivest, R.L.,Stein,C. (2009). *Introduction to Algorithms (3rd ed). Cambridge:M/TPress.*

Goodrich,M.T.,andTamassia,R. (2002).*AlgorithmDesign. Foundations, Analysis, and Internet Examples*. NewYork: John Wiley&Sons.

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

Tucker, A. B andNoonan, R . (2006). *Programming Languages – Principlesand Paradigms*. (2nd ed). McGraw–Hill College.

http://mathworld.wolfram.com/QueensProblem.html.

http://mathworld.wolfram.com/QueensProblem.html.

UNIT7 SPECIAL PROBLEMSAND ALGORITHMS

CONTENTS

1.0 INTRODUCTION

Thisunitshowssomespecialproblemsandalgorithms.Hillclimbing can beused tosolve problemsthathavemanysolutions,butwhere some solutionsare better thanothers.

2.0 OBJECTIVES

Attheendof thisunit youshouldbeableto:

- operatethehill climbing technique
- showhowhillclimbing is usedtosolveproblems resolvetheKnight'stour problem
- describeandresolveann*ntour problem
- knight-tour Problem
- understand N-Queen Problems
- describe Game Trees
- understand Subset Sum

3.0 MAINCONTENT

3.1 Hill ClimbingTechnique

Hillclimbingisanoptimisationtechniquewhichbelongstothefamily of local search. Itisarelativelysimpletechniquetoimplement,making it apopular first choice.Although moreadvancedalgorithmsmaygive better results,thereare situationswherehillclimbing works well.

Hillclimbingcanbeusedtosolveproblemsthathavemanysolutions butwheresomesolutions arebetter thanothers.Thealgorithm isstarted witha (bad)solutiontotheproblem,and sequentiallymakes small changestothesolution,eachtimeimprovingita littlebit.At somepoint thealgorithm arrivesata pointwhereitcannotseeanyimprovement anymore,atwhichpointthealgorithmterminates.Ideally,atthatpointa solution isfoundthatisclosetooptimal,butitisnotguaranteedthathill climbing willever comeclosetotheoptimalsolution.

Anexampleofaproblemthat canbesolvedwithhill climbingisthe Travellingsalesmanproblem.Itiseasytofindasolutionthatwillvisit allthe cities,butthissolution will probablybeverybadcomparedtothe optimalsolution.The algorithm startswith such a solutionandmakes smallimprovementstoit,suchasswitchingtheorderinwhichtwo citiesare visited.Eventually,a muchbetter routeisobtained.

Hillclimbingisusedwidelyin artificial intelligencefields,forreaching agoalstatefromastartingnode.Thechoiceof nextnodeand starting node canbe variedtogivealistof relatedalgorithms.

Hillclimbingterminateswhentherearenosuccessorsofthecurrent state which are better than the current state itself. This is often a problem. For example, consider the followingroutemap:

Theseproblemsareessentially theresult of*localmaxima*inthesearch space-pointswhicharebetterthananysurroundingstate,butwhich arenot the solution.There are some waysin which we can get round this(to some extent) bytweakingorextendingthealgorithmabit.We coulduse a limited amountof backtracking, sothatwerecord alternative reasonablelookingpathswhichweren'ttakenandgobacktothem.Or we couldweaken therestrictionthatthenext statehasto bebetterby lookingaheadabitinthesearch-maybethenextbutonestateshould bebetterthanthe currentone.Noneof thesesolutionsisperfect,andin generalhill climbing isonlygoodfor alimited classofproblemswhere wehave an evaluationfunctionthatfairlyaccuratelypredictsthe actual distancetoa solution.

Thiscanbedescribedas follows:

1. Startwith *current-state*= initial-state.
2. Until*current-state*=goal-stateORthereisnochangein *current-state*do:

   a. Getthesuccessorsofthecurrentstateandusethe`evaluation function toassigna scoretoeachsuccessor.
   b. Ifoneofthesuccessorshasabetterscorethanthecurrent-state thensetthenewcurrent-statetobethesuccessorwiththebest score.

Ifoneofthe successorshasabetterscorethan the current statethen set thenewcurrent statetobe thesuccessor with thebestscore.

Note that the algorithmdoesnot attempt to exhaustivelytryeverynode andpath,sononodelistoragendaismaintained-justthecurrentstate. Ifthere areloopsin thesearchspacethen using hill climbingyou shouldn't encounter them- youcan'tkeepgoingup andstillgetbackto where youwerebefore.

3.2 TheKnight's–Tour Problem

Aknight'stourofa chessboard (or any other grid) is a sequence of moves by a knight chess piece (which may only make moves which simultaneously shift one square along one axis and two along the other) such that each square of the board is visited exactly once. It is therefore a Hamiltonian path on the graphs consisting of vertices corresponding to the chessboard squares and edges corresponding to legal knight moves. If the final position of such a tour is a knight's move away from the initial position of the knight, the tour is called a re-entrant or closed, and is therefore a Hamiltonian circuit. The figures below show six knight's tours on an 8×8 chessboard, all but the first of which are re-entrant. The final tour has the additional property that it is a semi magic square with row and column sums of 260 and main diagonal sums of 348 and 168.

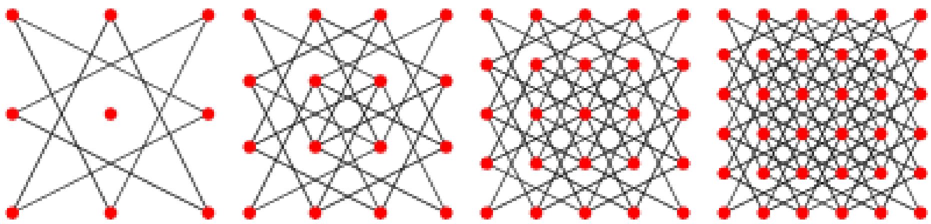


Figure6:SixKnight'sTouronan 8×8 ***chessboard***

The $m \times n$ knight'stourgraphisagraphon mn verticesinwhicheach vertex represents a square in an $m \times n$ chessboard,and each edge correspondstoalegalmovebyaknight.The $m \times n$ knight'stourgraphis implementedas Knight'sTourGraph[*m*, *n*] inthe Mathematica package.

Thenumberofedgesinthe $n \times n$ knight'stourgraphis $4(n-2)(n-1)$(8 timesthetriangular numbers),sofor $n=1$,2,...,thefirstfewvaluesare 0, 0, 8, 24, 48, 80, 120, ...

The numbers of (undirected) closed knight's tours on a $(2n) \times (2n)$ chessboardfor $n=1$,2,...are0,0,9862,13267364410532,....Thereare no closed tours for $m \times m$ boards with m odd. The number of cycles covering the directed knight's graph for an 8×8 chessboard was computedbyLöbbingandWegener(1996)as8121130233753702400. They also computed the number of undirected tours, obtaining an incorrectanswerof33439123484294(whichisnotdivisibleby4asit must be), andsoareredoingthecalculation.Theapparentlycorrect valueof13267364410532 appearsin Wegener'ssubsequentbook (Wegener2000),and also agreeswith unpublished calculationsofB.D. McKay.

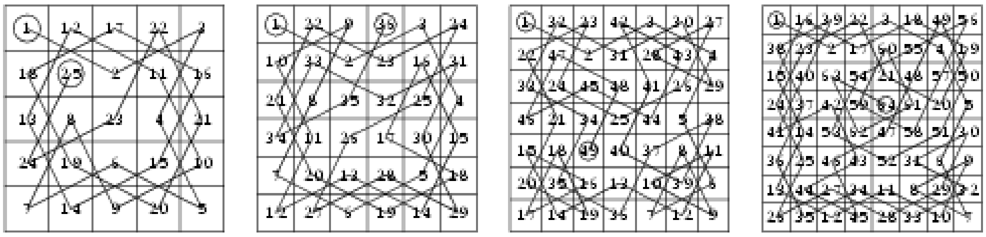
Thenumberofpossibletoursona $4\times k$boardfor $k=3$,4,...are8,0,82, 744, 6378, 31088, 189688, 1213112, ...

The following additional results are given by Kraitchik (1942, pp. 264-265).Thereare14toursonthe 3×7rectangle,twoofwhichare symmetrical.Thereare376toursonthe 3×8rectangle,noneofwhichis closed.Thereare16symmetrictoursonthe 3×9rectangleand8closed toursonthe 3×10rectangle.Thereare58symmetrictoursonthe 3×11 rectangle and 28 closed tours on the 3×12rectangle. There are five doublysymmetrictoursonthe 6×6square.Thereare1728toursonthe 5×5 square,8ofwhicharesymmetric.Thelongest"uncrossed"knight's toursonan $n\times n$boardfor $n=3$, 4, ... are 2, 5, 10, 17, 24, 35, ...

Backtrackingalgorithms (inwhich theknightis allowed tomoveasfar aspossibleuntilitcomestoablindalley,atwhichpointitbacksup somenumberof stepsandthentriesadifferentpath)canbeusedtofind knight'stours,butsuchmethods canbe veryslow.A backtracking algorithm tries to construct a solution to a computational problem incrementally, one small piece at a time. Whenever the algorithm needs to decide between multiple alternatives to the next component of the solution, it recursively evaluates every alternative and then chooses the best one.

Warnsdorff(1823)proposed an algorithm thatfindsapathwithout any backtracking by computing ratings for "successor" steps at each position.Here,successorsofapositionarethosesquaresthathavenot yetbeen visited andcan bereachedbyasinglemovefromthe given position. Therating ishighest forthesuccessorwhosenumber of successorsisleast.In thisway, squarestendingto beisolated arevisited first and therefore prevented from being isolated (Roth). The time neededforthis algorithmgrowsroughlylinearlywiththe numberof squaresof thechessboard,butunfortunatelycomputer implementation showsthatthis algorithm runsintoblind alleys forchessboardsbigger than 76×76, despitethefactthatit works wellonsmaller boards(Roth).

 CIT237



Recently,Conrad*etal.*(1994)discovered another lineartime algorithm and provedthat itsolvestheHamiltonianpathproblemforall$n\geq5$.The Conrad*etal.*algorithm works bya decomposition ofthechessboardinto smaller chessboards(notnecessarilysquare)forwhich explicitsolutions areknown.Thisalgorithmisrathercomplicatedbecauseithastodeal withmanyspecialcases,buthasbeenimplementedin Mathematicaby A.Roth.Exampletoursareillustratedabovefor$n \times n$boardswith $n = 5$to8.

3.3 *n*n* Queen'sProblem

Inchess,a queencanmoveasfar as shepleases,horizontally,vertically, ordiagonally.A chessboard has8 rows and 8 columns.Thestandard 8 by8 Queen'sproblemdescribeshow toplace8queensonanordinary chessboardsothat none ofthem can hitanyotherin one move.Thisis an amusing puzzleand chessplayersand researchershavebeen finding bestsolutionstothisproblem.

The problem is to place n queens on an n * n chessboard, so that no two queens are attacking each other. For readers not familiar with the rules of chess, this means that no two queens are in the same row, the same column, or the same diagonal.

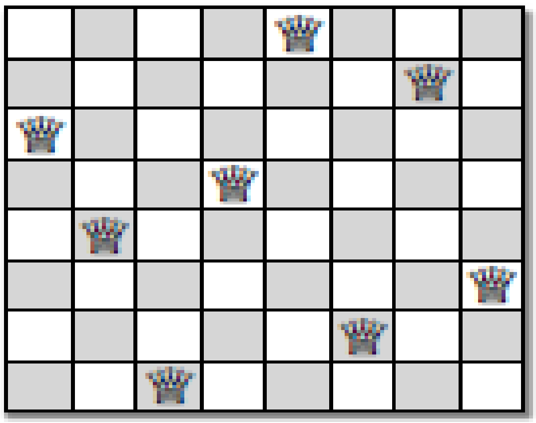


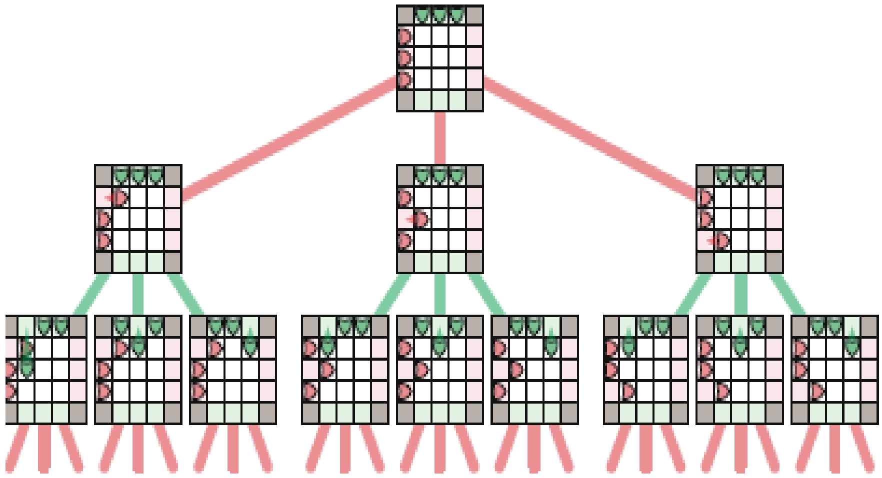
Figure 3 Gauss's first solution to the 8 queens problem, represented by the array [5, 7, 1, 4, 2, 8, 6,3]

Anobviousmodificationof the8by8problemistoconsider an*N*by*N* "chessboard"and askifonecanplace*N*queensonsuch aboard.It is easy to see that this isimpossible if*N*is2 or3, and it'sreasonably straightforwardto findsolutions when*N*is4,5,6,or7.Theproblem beginstobecomedifficultformanualsolutionpreciselywhen*N*is8. Thefactthat thisnumber coincidentallyequalsthe dimensionsofan ordinarychessboardhasprobablycontributedtothepopularityof the problem.

3.4 Game Trees

Consider the following simple two-player game1 played on an *n* **n* square grid with a border of squares. Each player has *n* tokens that they move across the board from one side to the other.

A ***state*** of the game consists of the locations of all the pieces and the identity of the current player. These states can be connected into a *game tree*, which has an edge from state *x* to state *y* if and only if the current player in state *x* can legally move to state *y*. The root of the game tree is the initial position of the game, and every path from the root to a leaf is a complete game.



To navigate through this game tree, we recursively define a game state to be ***good*** or ***bad*** as follows:

• A game state is *good* if either the current player has already won, or if the current player can move to a bad state for the opposing player.
• A game state is *bad* if either the current player has already lost, or if every available move leads to a good state for the opposing player.

Equivalently, a non-leaf node in the game tree is good if it has at least one bad child, and a non-leaf node is bad if all its children are good. By induction, any player that finds the game in a good state on their turn can win the game, even if their opponent plays perfectly; on the other hand, starting from a bad state, a player can win only if their opponent makes a mistake.This recursive definition immediately suggests the following recursive backtracking algorithm to determine whether a given game state is good or bad. At its core, this algorithm is just a depth-first search of the game tree; equivalently, the game tree is the recursion tree of the algorithm! A simple modification of this backtracking algorithm finds a good move (or even all possible good moves) if the input is a good game state.

All game-playing programs are ultimately based on this simple backtracking strategy. However, since most games have an enormous number of states, it is not possible to traverse the entire game tree in practice. Instead, game programs employ other heuristics5 to *prune* the game tree, by ignoring states that are obviously (or "obviously") good or bad, or at least better or worse than other states, and/or by cutting off the tree at a certain depth (or *ply*) and using a more efficient heuristic to evaluate the leaves.

```
PlayAnyGame(X, player):
  if player has already won in state X
      return Good
  if player has already lost in state X
      return Bad
  for all legal moves X ⇝ Y
      if PlayAnyGame(Y, ¬player) = Bad
          return Good        ⟨⟨X ⇝ Y is a good move⟩⟩
  return Bad                 ⟨⟨There are no good moves⟩⟩
```

3.5 Subset Sum

Let's consider a more complicated problem, called SubsetSum: Given a set *X* of positive integers and *target* integer *T*, is there a subset of elements in *X* thatadd up to *T*? Notice that there can be more than one such subset. For example, if *X* = (8, 6, 7, 5, 3, 10, 9) and *T* = 15, the answer is True, because the subsets (8, 7) and (7, 5, 3) and (6, 9) and (5, 10) all sum to 15. On the other hand, if *X* = (11, 6, 5, 1, 7, 13, 12) and *T* = 15, the answer is False.

There are two trivial cases. If the target value *T* is zero, then we can immediately return True, because the empty set is a subset of *every* set *X*, and the elements of the empty set add up to zero.6 On the other hand, if *T* <0, or if *T* 6= 0 but the set *X* is empty, then we can immediately return False. For the general case, consider an arbitrary element *x* ∈*X*. (We've already handled the case where *X* is empty.) There is a subset of *X* that sums to *T* if and only if one of the following statements is true:

• There is a subset of *X* that *includes* *x* and whose sum is *T*.
• There is a subset of *X* that *excludes* *x* and whose sum is *T*.

In the first case, there must be a subset of$X \setminus \{x\}$that sums to $T - x$; in thesecond case, there must be a subset of *X* n fxg that sums to *T*. So we can solveSubsetSum(*X*, *T*) by reducing it to two simpler instances:

SubsetSum($X \setminus \{x\}$, $T - x$) and SubsetSum($X \setminus \{x\}, T$). The resulting recursive algorithm is shownbelow.

```
⟨⟨Does any subset of X sum to T?⟩⟩
SubsetSum(X, T):
  if T = 0
      return True
  else if T < 0 or X = ∅
      return False
  else
      x ← any element of X
      with ← SubsetSum(X \ {x}, T − x)    ⟨⟨Recurse!⟩⟩
      wout ← SubsetSum(X \ {x}, T)        ⟨⟨Recurse!⟩⟩
      return (with ∨ wout)
```

3.6 Text Segmentation –Longest Increasing Subsequence

Suppose you are given a string of letters representing text in some foreignlanguage, but without any spaces or punctuation, and you want to break this string into its individual constituent words.

For any sequence *S*, a ***subsequence*** of *S* is another sequence obtained from *S* by deleting zero or more elements, without changing the order of the remaining elements; the elements of the subsequence need not be contiguous in *S*. For example, when you drive down a major street in any city, you drive througha *sequence* of intersections with traffic lights, but you only have to stop at a *subsequence* of those intersections, where the traffic lights are red. If you're very lucky, you never stop at all: the empty sequence is a subsequence of *S*. On the other hand, if you're very unlucky, you may have to stop at every intersection: *S* is a subsequence of itself.

As another example, the strings BENT, ACKACK, SQUARING, and SUBSEQUENT are all subsequences of the string SUBSEQUENCEBACKTRACKING, as are the empty string and the entire string SUBSEQUENCEBACKTRACKING, but the strings QUEUE and EQUUS and TALLYHO are not. A subsequence whose elements are contiguous in the original sequence is called a ***substring***; for example, MASHER and LAUGHTER are both subsequences of MANSLAUGHTER, but only LAUGHTER is a substring.

Now suppose we are given a sequence of *integers*, and we need to find the longest subsequence whose elements are in increasing order. More concretely, the input is an integer array *A*[1 .. *n*], and we need to compute the longest possible sequence of indices 1 _ *i*1 < *i*2 <_ _ _ < *i*`

 CIT237

_ n such that $A[ik] < A[ik+1]$for all k.One natural approach to building this ***longest increasing subsequence*** is to*decide*, for each index j in order from 1 to n, whether or not to include $A[\, j]$ inthe subsequence

```
LISBIGGER(prev,A[1..n]):
  if n = 0
      return 0
  else if A[1] ≤ prev
      return LISBIGGER(prev,A[2..n])
  else
      skip ← LISBIGGER(prev,A[2..n])
      take← LISBIGGER(A[1],A[2..n]) + 1
      return max{skip, take}
```

4.0 CONCLUSION

Inthis unit,youwillobservethatthehillclimbing canbeusedto solve problemsthathavemanysolutionsbut where some solutions arebetter than others.An exampleof aproblem that can besolved withhill climbingisthe travellingsalesman problem.It isalso used widelyin artificialintelligencefieldsforreachingagoalstatefrom astarting node.

5.0 SUMMARY

Thisunit has addressedthe following:

- Hill climbingtechniqueandalgorithm
- Howhillclimbing is usedinsolving problems
- TheKnight’s tour problems.
- TheQueen’s problem
- Games trees

6.0TUTOR-MARKEDASSIGNMENT

a. Brieflydescribetheclimbing techniqueandalgorithm.
b. Brieflyexplainhowtheclimbingtechniqueisusedinsolving problems.
c. ComparetheKnight’stourproblemandtheQueen’sproblem youhavelearnt aboutinthisunit.
d. State two application areas of game trees algorithm

7.0 REFERENCES/FURTHERREADINGS

http://mathworld.wolfram.com/QueensProblem.html.

http://mathworld.wolfram.com/QueensProblem.html.
Ahrens, W.(1910).*Mathematische UnterhaltungenundSpiele.* Leipzig, Germany: Teubner, p. 381.

Ball,W.W. R. andCoxeter,H. S.M.(1987). *MathematicalRecreationsandEssays(13thed).*NewYork:Dover, pp. 175-186,.

Chartrand,G."TheKnight's Tour."§6.2in *IntroductoryGraphTheory* (1985).NewYork:Dover, pp. 133-135.

Conrad,A.;Hindrichs,T.; Morsy, H.;andWegener,I.(1994). "Solution of theKnight'sHamiltonianPathProblemonChessboards." *Discr. Appl.Math.***50**, 125-134.

de Polignac. *ComtesRendusAcad. Sci.Paris*, Apr. 1861.

de Polignac. *Bull.Soc.Math.de France***9**, 17-24, 1881.

Dudeney, H. E.(1970). *AmusementsinMathematics.*NewYork:Dover, pp. 96and102-103.

Elkies, N. D. and Stanley, R.P. "TheMathematicalKnight." *Math. Intell.* **25**, No. 1, 22-34, Winter2003.

Euler,L. "Solution d'unequestion curieuse quineparoit soumise a aucuneanalyse."*Mémoires de l'Académie Royaledes Scienceset Belles LettresdeBerlin,Année 1759***15**, 310-337, 1766.

Euler,L. *CommentationesArithmeticae Collectae, Vol.1.*(1849) Leningrad, pp. 337-355,.

Friedel, F."The Knight'sTour."
http://www.chessbase.com/columns/column.asp?pid=163.

Gardner, M. (1978)."Knightsof theSquare Table."Ch.14in *MathematicalMagicShow:MorePuzzles,Games,Diversions,IllusionsandOtherMathematicalSleight-of-MindfromScientificAmerican.*NewYork:Vintage,pp.188-202,.

Gardner, M. (1984). *TheSixthBookofMathematicalGamesfromScientificAmerican.*Chicago,IL:Universityof ChicagoPress, pp. 98-100.

Guy, R. K.(199)."The QueensProblem."§C18in *UnsolvedProblemsinNumberTheory,2nded.*NewYork:Springer-Verlag,pp. 133-135.

Jelliss,G."Knight'sTour Notes."

http://www.ktn.freeuk.com/.Jelliss,G."ChronologyofKnight'sTours."
http://www.ktn.freeuk.com/cc.htm.

Kraitchik, M.(1942)."The Problemof theKnights." Ch.11in *MathematicalRecreations.*NewYork:W.W. Norton,pp. 257-266.

Kyek,O.;Parberry, I.;andWegener,I."BoundsontheNumber of Knight'sTours."*Discr. Appl.Math.***74**, 171-181, 1997.

Lacquière.*Bull. Soc. Math. deFrance* **8**, 82-102and132-158, 1880.

Madachy, J. S.(1970). *Madachy'sMathematicalRecreations.*New York:Dover,pp.87-89.

Murray, H. J.R. (1902)."TheKnight'sTour, AncientandOriental." *BritishChessMagazine*, pp. 1-7.

Pegg, E.Jr."Leapers (ChessKnightsandthe Like)" http://www.mathpuzzle.com/leapers.htm.

Roget,P. M. (1840).*Philos.Mag.* **16**, 305-309.

Rose, C. "TheDistribution of theKnight." http://www.tri.org.au/knightframe.html.

Roth,A."The Problemof theKnight: A FastandSimpleAlgorithm." http://library.wolfram.com/infocenter/MathSource/909/.

Rubin, F. (1974)."ASearch Procedure for Hamilton PathsandCircuits." *J. ACM* **21**, 576-580.

Ruskey, F."Information onthe Knight'sTour Problem." http://www.theory.csc.uvic.ca/~cos/inf/misc/Knight.html.

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

Skiena, S. (1990). *ImplementingDiscreteMathematics:CombinatoricsandGraphTheorywithMathematica.*Reading,MA:Addison- Wesley, p.166.

Sloane, N. J.A. Sequences A001230,A003192/M1369, A006075/M3224, A033996,and A079137in"TheOn-Line Encyclopediaof Integer Sequences."

Steinhaus,H.(1999). *MathematicalSnapshots,3rded.*NewYork: Dover, p. 30.

Thomasson,D."The Knight'sTour." http://www.borderschess.org/KnightTour.htm.

vander Linde, A.(1874). *GeschichteundLiteraturdesSchachspiels,Vol.2.*Berlin: Springer-Verlag,pp.101-111.

Vandermonde,A.-T."Remarquessur les Problèmes de Situation." *L'Histoirede l'AcadémiedesSciencesavec lesMémoires,Année 1771*. Paris:Mémoirs, pp. 566-574andPlate I,1774.

Velucchi, M. "Knight'sTour:The UltimateKnight's Tour Pageof Links." http://www.velucchi.it/mathchess/knight.htm.

Volpicelli,P. (1872)."Soluzionecompletae generale, mediantela geometriadi situazione,delproblema relativo allecorse del cavallo sopraqualunquescacchiere."*Atti della RealeAccad. dei Lincei***25**, 87-162.

Warnsdorff,H. C. (1823). von*Des Rösselsprungs einfachsteund allgemeinsteLösung.*Schmalkalden.

Watkins, J. (2004). *AcrosstheBoard:TheMathematicsofChessboard Problems.*Princeton,NJ:PrincetonUniversity Press.

Algorithms (2019). Jeff Erickson. 1st Edition, available in Amazon

UNIT8 PRACTICALEXERCISE III

CONTENTS

1.0 INTRODUCTION

Inthisunit,youwillbeexposedtothepracticalapplicationsofallthe sorting algorithmsyoulearntinthepreviousunitsof thiscourse.

2.0 OBJECTIVES

Attheendof thisunit, youshouldbe ableto:

- explainhowbubblesortisimplementedinc programming language
- showhowquick sortisimplementedinc programming language.

3.0 MAINCONTENT

3.1 ProblemI

WritetheC codes for implementing bubblesort

Solution

```
void bubblesort(int numbers[],int array_size)
{
 int i, j,temp;

 for(i=(array_size - 1);i>= 0;i--)
 {
  for(j=1;j<= i;j++)
  {
   if (numbers[j-1]> numbers[j])
   {
    temp=numbers[j-1];
    numbers[j-1] = numbers[j];
```

```
        numbers[j] = temp;
      }
    }
  }
}
```

3.2 Problem II

WritetheC codes for implementing quicksort

Solution

```
void quicksort(int numbers[],int array_size)
{
  q_sort(numbers, 0, array_size - 1);
}

void q_sort(int numbers[], int left, int right)
{
  int pivot, l_hold,r_hold;

  l_hold = left;
  r_hold= right;
  pivot = numbers[left];
  while(left<right)
  {
    while((numbers[right] >= pivot) &&(left<right))
      right--;
    if (left!= right)
    {
      numbers[left] = numbers[right];
      left++;
    }
    while((numbers[left] <= pivot) &&(left<right))
      left++;
    if (left!= right)
    {
      numbers[right] = numbers[left];
      right--;
    }
  }
  numbers[left] = pivot;
  pivot = left;
  left= l_hold;
  right = r_hold;
  if (left< pivot)
```

```
   q_sort(numbers, left, pivot-1);
  if (right > pivot)
   q_sort(numbers, pivot+1,right);
}
```

4.0 CONCLUSION

Thisunit has showedyouhowtoimplementbubblesortusingtheC codes. Also,itshowedyouhowtoexecutethequicksortusingtheC codesas well.

5.0 SUMMARY

Bynowyoushouldhave learnthow:

Toimplementthebubbleandthequicksortusingthe Ccodes.
Towriteprograms usingtheCprogramming language.

6.0 TUTOR-MARKEDASSIGNMENT

1. Studythecodesaboveandrunthemwith sampledataona C compiler.
2. Writethecodes for implementing anysorting algorithm in anyprogramming languagethatyouhavelearnt.
3. Apply game trees algorithm to solve a practical problem

7.0 REFERENCES/FURTHERREADINGS

Cormen, T.H., Leiserson, C.F. Rivest, R.L., Stein, C. (2001). *Introduction to Algorithms* (2nd ed). Cambridge:M/T Press.

Goodrich, M.T., and Tamassia,R. (2002). *Algorithm Design. Foundations,Analysis,andInternetExamples*.NewYork:John Wiley&Sons.

Scott, M. L. (2015). Programming Language Pragmatics 4th Edition

Tucker, A.B and Noonan, R. (2006). *Programming Languages – Principlesand Paradigms*. (2nd ed). McGraw–Hill College.

http://mathworld.wolfram.com/QueensProblem.html.http://mathwo

rld.wolfram.com/QueensProblem.html.