A T M E
College of Engineering

COLLEGEOFENGINEERING
13thKMStone,BannurRoad,Mysore-560028

DEPARTMENTOFCOMPUTERSCIENCEANDENGINEERING–AI&ML

(ACADEMIC YEAR 2024-25)

LABORATORYMANUAL

SUBJECT:ANALYSIS&DESIGNOFALGORITHMS

SUB CODE: BCSL404

SEMESTER:IVSCHEME:2022

PreparedBy	VerifiedBy	Approvedby
Ms.GEETHA.B	Mrs. VANITHA G NAIK	Dr.MSSUNITHAPATEL
INSTRUCTOR	FACULTIESCO_ORDINATORS	HOD,CSE-AI &ML

INSTITUTIONALMISSIONANDVISION

Objectives

- Toprovidequalityeducationandgroomtop-notchprofessionals,entrepreneurs and leadersfordifferent fields ofengineering, technologyand management.
- ToopenaTraining-R&D-Design-Consultancycellineachdepartment,graduallyintroduce doctoral and postdoctoral programs, encourage basic & applied research in areas of socialrelevance,and develop theinstituteas a centerofexcellence.
- Todevelopacademic,professionalandfinancialallianceswiththeindustryas well as the academia atnationaland transnational levels.
- Tocultivatestrongcommunityrelationshipsandinvolvethestudentsandthestaff in Local community service.
- Toconstantlyenhancethevalueoftheeducationalinputswiththeparticipationof students,faculty, parents andindustry.

Vision

- Developmentofacademicallyexcellent,culturallyvibrant,socially responsible and globally competent humanresources.

Mission

- Tokeeppacewithadvancementsinknowledgeandmakethestudentscompetitiveand capable at the global level.
- To create an environment for the studentsto acquire the right physical, intellectual, emotionalandmoralfoundationsand shineastorchbearersoftomorrow'ssociety.
- Tostrivetoattainever-higherbenchmarksofeducationalexcellence.

DepartmentofComputerScience&Engineering

VisionoftheDepartment

- To develop highly talented individuals in Computer Science and Engineering to deal with realworld challenges in industry, education, researchand society.

MissionoftheDepartment

- Toinculcate professionalbehavior,strong ethicalvalues,innovativeresearchcapabilities and leadership abilities in the young minds & to provide a teaching environment that emphasizes depth, originality and critical thinking.
- Motivatestudents toputtheir thoughtsandideas adoptableby industryor topursue higher studies leadingto research.

Programoutcomes(POs)

EngineeringGraduateswillbeableto:

PO1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals,andanengineeringspecializationtothesolutionofcomplexengineeringproblems.

PO2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, naturalsciences,and engineering sciences.

PO3.**Design/developmentofsolutions**:Designsolutionsforcomplexengineeringproblemsand designsystemcomponentsorprocessesthatmeetthespecifiedneedswithappropriateconsideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4.**Conductinvestigationsofcomplexproblems**:Useresearch-basedknowledgeandresearch methods including design of experiments, analysis and interpretation of data, and synthesis ofthe information to providevalid conclusions.

PO5.**Moderntoolusage**:Create,select,andapplyappropriatetechniques,resources,andmodern engineeringand ITtools includingpredictionandmodelingtocomplexengineering activities with an understandingof thelimitations.

PO6.**Theengineerandsociety**:Applyreasoning informedbythecontextualknowledgetoassess societal,health,safety, legalandculturalissuesandtheconsequentresponsibilitiesrelevant tothe professional engineeringpractice.

PO7. **Environment and sustainability**: Understand the impact of the professional engineering solutions insocietalandenvironmentalcontexts,anddemonstratetheknowledgeof,andneed for sustainable development.

PO8.**Ethics**:Applyethicalprinciplesandcommittoprofessionalethicsandresponsibilities and norms ofthe engineering practice.

PO9.**Individualandteamwork**:Functioneffectivelyasanindividual,andasamemberorleader in diverse teams,andin multidisciplinary settings.

PO10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effectivereportsanddesigndocumentation,makeeffectivepresentations,andgiveandreceiveclear instructions.

PO11. **Project management and finance**: Demonstrate knowledge and understanding of the engineeringandmanagementprinciplesandapplythesetoone'sownwork,asamemberandleader ina team,to manage projects andin multidisciplinary environments.

PO12.Life-longlearning:Recognizetheneedfor,andhavethepreparationandabilitytoengage in independentandlife-longlearninginthe broadestcontextoftechnologicalchange.

ProgramSpecificOutcomes

PSO1: Abilitytoapplyskillsinthefieldofalgorithms, databasedesign,webdesign,cloud computingand data analytics.

PSO2:Applyknowledgeinthefieldofcomputernetworksforbuildingnetworkandinternetbased applications

ProgramEducationalObjectives(PEOs):

1. Empowerstudentswithastrongbasisinthemathematical,scientificandengineering fundamentalsto solvecomputationalproblemsandtopreparethemfor employment, higher learning and R&D.
2. Gaintechnical knowledge, skills andawarenessof current technologiesof computer science engineering and to develop an abilityto design and provide novel engineering solutions forsoftware/hardwareproblemsthroughentrepreneurial skills.
3. Exposure to emerging technologies and work in teams on interdisciplinary projects witheffective communication skills andleadershipqualities.
4. Ability to function ethically and responsibly in a rapidly changing environment by applyinginnovativeideasinthelatesttechnology,tobecomeeffectiveprofessionalsin ComputerScience to beara life-long career inrelate dare as.

Analysis&DesignofAlgorithmsLab		Semester	4
CourseCode	**BCSL404**	CIEMarks	50
TeachingHours/Week(L:T:P:S)	0:0:2:0	SEEMarks	50
Credits	01	ExamHours	2
Examinationtype(SEE)	Practical		

Sl.No	Experiments
1	DesignandimplementC/C++ProgramtofindMinimumCostSpanningTreeofagivenconnected undirectedgraphusingKruskal'salgorithm.
2	DesignandimplementC/C++ProgramtofindMinimumCostSpanningTreeofagivenconnected undirectedgraphusingPrim'salgorithm.
3	a. DesignandimplementC/C++ProgramtosolveAll-PairsShortestPathsproblemusingFloyd's algorithm. b. DesignandimplementC/C++ProgramtofindthetransitiveclosureusingWarshal's algorithm.
4	DesignandimplementC/C++Programtofindshortest pathsfromagivenvertexinaweighted connectedgraphtootherverticesusingDijkstra'salgorithm.
5	Designand implementC/C++ProgramtoobtaintheTopologicalorderingofverticesinagiven digraph.
6	DesignandimplementC/C++Programtosolve0/1KnapsackproblemusingDynamic Programmingmethod.
7	DesignandimplementC/C++ProgramtosolvediscreteKnapsackandcontinuousKnapsack problemsusing greedyapproximationmethod.
8	DesignandimplementC/C++ProgramtofindasubsetofagivensetS={sl,s2,. ..... ,sn}ofn positiveintegerswhosesumisequaltoagivenpositiveintegerd.
9	Design and implement C/C++ Program to sort a given set of n integer elements using Selection Sortmethod andcomputeits timecomplexity. Runthe programforvaried values of n>5000 and record the time taken to sort. Plot a graph of the time taken versus n. The elementscan be read fromafileorcanbegeneratedusingtherandomnumbergenerator.
10	Design and implement C/C++ Program to sort a given set of n integer elements using Quick Sort methodandcomputeitstimecomplexity.Runtheprogramforvariedvaluesofn>5000andrecord the time taken to sort. Plota graph of the time taken versus n. The elementscanbe read fromafileorcanbegeneratedusingtherandomnumbergenerator.
11	Design and implement C/C++ Program to sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n> 5000, and record the time taken to sort. Plota graph of the time taken versus n. The elementscan be read fromafileorcanbegeneratedusingtherandomnumbergenerator.
12	DesignandimplementC/C++ProgramforNQueen'sproblemusingBacktracking.

Courseoutcomes(CourseSkillSet):

Attheendofthecoursethestudentwillbeableto:

1. Developprogramstosolvecomputationalproblemsusingsuitablealgorithmdesignstrategy.
2. Comparealgorithmdesignstrategiesbydevelopingequivalentprogramsandobservingrunning times for analysis (Empirical).
3. Makeuseofsuitableintegrateddevelopmenttoolstodevelopprograms
4. Chooseappropriatealgorithmdesigntechniquestodevelopsolutiontothecomputationaland complex problems.
5. Demonstrateandpresentthedevelopmentofprogram,itsexecutionandrunningtime(s)and record the results/inferences.

AssessmentDetails(bothCIEandSEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together

ContinuousInternalEvaluation(CIE):

CIEmarksforthepracticalcourseare**50Marks**.

Thesplit-upofCIEmarksforrecord/journalandtestareintheratio**60:40**.

- Eachexperimentistobeevaluatedforconductionwithanobservationsheetandrecordwrite-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments are designedbythefacultywhoishandlingthelaboratorysessionandaremadeknowntostudentsat the beginning of the practical session.
- Recordshouldcontainallthespecifiedexperimentsinthesyllabusandeachexperimentwrite-up will be evaluated for 10 marks.
- Totalmarksscoredbythestudentsarescaleddownto**30marks**(60% ofmaximum marks).
- Weightagetobegivenforneatnessandsubmissionofrecord/write-upon time.
- Departmentshallconductatestof100marksafterthecompletionofalltheexperimentslistedin thesyllabus.
- Inatest,testwrite-up,conductionofexperiment,acceptableresult,andproceduralknowledgewill carry a weightage of 60% and the rest 40% for viva-voce.
- Thesuitablerubricscanbedesignedtoevaluateeachstudent'sperformanceandlearningability.
- Themarksscoredshallbescaleddownto**20marks**(40% of themaximum marks).

TheSumofscaled-downmarksscoredinthereportwrite-up/journalandmarksofatestisthetotalCIE marks scored by the student.

SemesterEndEvaluation(SEE):

- SEEmarksforthepracticalcourseare50Marks.

- SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the Head of the Institute.
- Theexaminationscheduleandnamesofexaminersareinformedtotheuniversitybeforethe conduction of the examination. These practical examinations aretobe conducted between theschedule mentioned in the academic calendar of the University.
- Alllaboratoryexperimentsaretobeincludedforpracticalexamination.
- (Rubrics)Breakupofmarksandtheinstructionsprintedonthecoverpageoftheanswerscript to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.
- Studentscanpickonequestion(experiment)fromthequestionslotpreparedbytheexaminers jointly.
- Evaluation of test write-up/ conductionprocedureand result/vivawillbeconducted jointly byexaminers.

GeneralrubricssuggestedforSEE arementionedhere,writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marksand scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall bedecided by the examiners)

Changeofexperimentisallowedonlyonceand15%ofMarksallottedtotheprocedureparttobe made zero.

CONTENTS

INTRODUCTION

Needforstudyingalgorithms Theoretical importance

- Thestudyofalgorithmsistheconceptsofcomputerscience.
- Itisastandardsetofimportantalgorithms,theyfurtherouranalyticalskills& help us indevelopingnew algorithmsfor required applications

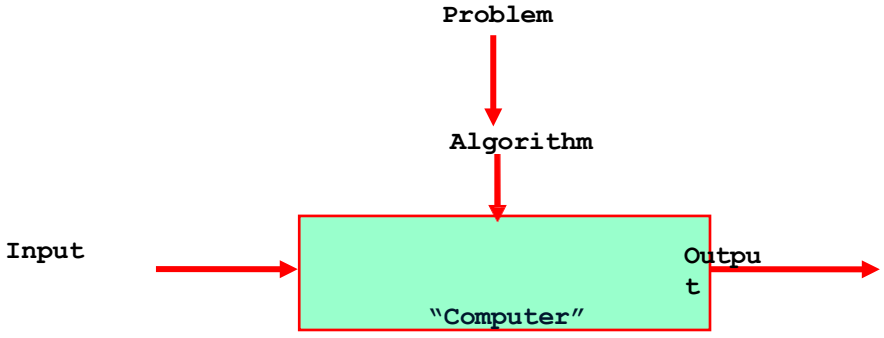
Algorithm

Analgorithmisasequenceofunambiguousinstructionsforsolvingaproblem,i.e.,forobtaining a required output for anylegitimate input ina finite amount oftime.

Inaddition,allalgorithmsmustsatisfythefollowingcriteria:

1. **Input**: Eachalgorithmshouldhavezero or more inputs. The range of inputforwhichalgorithmsworksshouldbespecifiedcarefully.
2. **Output**: Thealgorithmshouldproducecorrectresults.Atleastonequantity has to beproduced.
3. **Definiteness**:Eachinstructionshouldbeclearandunambiguous.
4. **Effectiveness:Every**instructionshouldbesimpleandshouldtransformthegivenInput tothe desired output. sothatit canbe carriedout,in
5. **Finiteness**: Ifwetraceouttheinstructionofanalgorithm,thenforallcases,the algorithm must terminate after a finite numb2erofsteps.

Notionofalgorithm



FundamentalsofAlgorithmicproblemsolving

- Understandingtheproblem
- Ascertainthecapabilitiesofthecomputationaldevice
- Exact/approximatesolution.
- Decideontheappropriatedatastructure
- Algorithmdesigntechniques
- Methodsofspecifyinganalgorithm
- Provinganalgorithmscorrectness
- Analyzinganalgorithm

ImportantProblemTypesofdifferentcategories

- **Sorting**

Itreferstotheproblemofre-arrangingtheitemsofagivenlistinascendingordescendingorder. The various algorithms that can be used for sorting are bubble sort, selection sort, insertion sort, quick sort, merge sort,heap sort etc.

- **Searching**

∗ Thisproblemdealswithfindingavalue,calledasearchkey,inagivenset.

∗ Thevariousalgorithmsthatcanbeusedforsearchingarebinary search,linear search,hashing,interpolation search etc.

- **Stringprocessing**

∗ Thisproblemdealswithmanipulationofcharactersorstrings,stringmatching, searchand replace, deleting a stringinatext etc.

∗ Stringprocessingalgorithmsuchasstringmatchingalgorithmsisusedinthedesign of assemblersand compliers.

- **Graphproblems**

∗Thegraphis acollectionofverticesandedges.
∗Ex:graphtraversalproblems,topologicalsortingetc

- **Combinatorialproblems**

Theseproblemsareusedtofindcombinatorialobjectsuchaspermutationandcombinations.

Ex:travelling salesmanproblem, graphcoloring problem etc

- **Geometricproblems**

Thisproblemdealwithgeometricobjectssuchaspoints,curves, lines,polygonetc.

Ex:closest-pair problem, convexhull problem

- **Numericalproblems**

Theseproblemsinvolvingmathematicalmanipulationssolvingequations, computing differentiationsandintegrations,evaluatingvarioustypesoffunctionsetc.

Ex:Gausseliminationmethod,Newton-Rapsonmethod

FundamentalsofdataStructures

Sincemostofthealgorithmsoperateonthedata,particularways ofarranging thedataplaya criticalroleinthe design& analysisofalgorithms.

Adatastructurecanbedefinedasaparticular wayofarrangementofdata.

Thecommonlyuseddatastructuresare:

1. Lineardatastructures
2. Graphs
3. Trees.
4. Setsanddictionaries

1. Lineardatastructures

Themostcommonlineardatastructuresarearraysandlinkedlists.

Arrays: Isacollectionofhomogeneousitems.Anitem'splacewithinthecollectionis called an index.

Linkedlist:finitesequenceofdataitemsi.e.itisacollectionofdataitems inacertainorder.

2. Graphs

Adatastructurethatconsistsofasetofnodesandasetofedgesthatrelatethe nodestoeach other is called agraph.

Undirectedgraph:Agraphinwhichtheedgeshavenodirection

- **Directedgraph(Digraph**):Agraphinwhicheachedge isdirectedfromone vertex to another (or thesame)vertex.

3. Tree

Atreeisaconnectedacyclicgraphthathasnocycle.

4. Setsanddictionaries

* Asetisdefinedasanunorderedcollectionofdistinctitemscalledanelementoftheset
* Dictionaryisadatastructurethatimplementssearching,addingofobjects

Analysisofalgorithms

Analysisofalgorithmsmeanstoinvestigateanalgorithm'sefficiencywithrespectto resources:

Runningtime(timeefficiency)

Itindicateshowfastanalgorithminquestionruns

- **Memoryspace(spaceefficiency)**

Itdealswiththeextra spacethe algorithmrequires

Theoreticalanalysisoftimeefficiency

Algorithmefficiencydependsonthe**inputsizen.**Andforsomealgorithmsefficiency dependson**type of input**.

Wehavebest,worst&averagecaseefficiencies

Worst-caseefficiency:

Efficiency(numberoftimesthebasicoperationwillbeexecuted)**fortheworstcase**

inputofsize n.*i.e.*Thealgorithmrunsthelongestamongallpossibleinputsofsizen.

Best-caseefficiency:

Efficiency(numberoftimesthebasicoperationwillbeexecuted**forthebestcase input ofsizen.** ***i.e.*** Thealgorithmrunsthe fastestamong allpossibleinputs ofsize n

Average-caseefficiency:

Averagetimetaken(numberoftimesthebasicoperationwillbeexecuted)**tosolveallthe possible instances (random) of the input.**

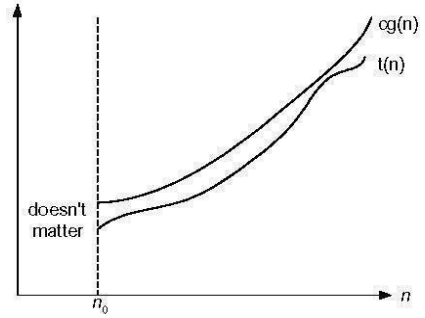
AsymptoticNotations

- Asymptoticnotationisawayofcomparingfunctionsthatignoresconstant factors and small input sizes.
- Threenotationsusedtocompareordersofgrowthofanalgorithm'sbasicoperationare:

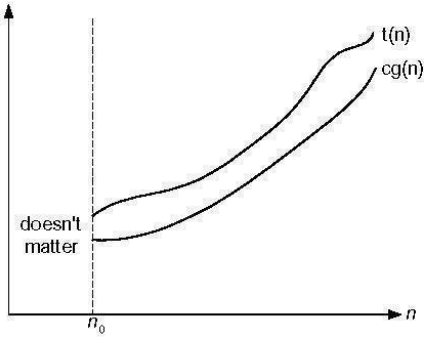
O,Ω,θnotations.

Big-oh notation:

Afunctiont(n)issaidtobeinO(g(n)),denotedt(n)€O(g(n)),ift(n)isbounded abovebysome constant multiple ofg(n) for all large ni.e.,ifthereexist some positive constant c and some nonnegativeintegern◌0suchthat**t(n)<=cg(n)foralln>=n**0



Big-ohnotationt(n)€O(g(n))



Big-omeganotationt(n)€Ω,(g(n))

Big-omeganotation:

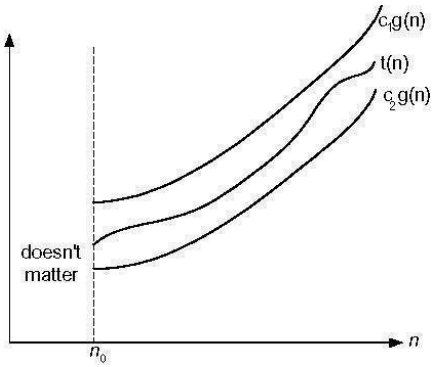
Afunctiont(n)issaidtobeinΩ(g(n)),denotedt(n)€Ω(g(n)),ift(n)isboundedbelowby some constant multiple ofg(n) for all large ni.e., ifthere exist some positive constant c and Somenonnegativeintegern◌0suchthat

t(n)>=cg(n)foralln>=n_0Big-

theta notation:

Afunctiont(n)issaidtobeinθ(g(n)),denotedt(n)€θ(g(n)),ift(n)isboundedboth aboveandbelowbysomepositiveconstantmultipleofg(n)foralllargeni.e.,ifthereexist some positive constant c

c2g(n)<=t(n)<=c1g(n)foralln>= n_0



Big-thetanotationt(n)€θ(g(n))

BasicAsymptoticEfficiencyclasses:

1	**constant**
log*n*	**logarithmic**
N	**linear**
n*log*n	***n*log*n***
n^2	**quadratic**
n^3	**cubic**
2^n	**exponential**
n!	**factorial**

1. DesignandimplementC/C++Programtofind MinimumCostSpanningTreeofa given connected undirected graph using Kruskal's algorithm.

```
#include<stdio.h>int

e=1,min_cost=0;

voidmain()
{
intn,i,j,min,a,u,b,v,cost[20][20],parent[20];
printf("Enter the no. of vertices:");
scanf("%d",&n);
printf("\nEnterthecostmatrix:\n");
for(i=1;i<=n;i++)for(j=1;j<=n;j++)
scanf("%d",&cost[i][j]);
for(i=1;i<=n;i++)
parent[i]=0;
printf("\nTheedgesofspanningtreeare\n"); while(ne<n)
{
min=999;
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
if(cost[i][j]<min)
{
min=cost[i][j];
 a=u=i;
b=v=j;
}}}
while(parent[u])
u=parent[u];
while(parent[v])
v=parent[v];
if(u!=v)
{
 printf("Edge%d\t(%d->%d)=%d\n",ne++,a,b,min);
 min_cost=min_cost+min;
 parent[v]=u;
 }
 cost[a][b]=cost[a][b]=999;
 }
 printf("\nMinimumcost=%d\n",min_cost);
 }
```

Output:

Entertheno.ofvertices: 6

Enterthecostmatrix:
9993 999 999 65
3 999 1 999 9994
9991 999 6 9994
999999 6 999 8 5
6 9999998 999 2
5 4 4 5 2 999

Theedgesofspanningtreeare
Edge 1 (2->3)=1
Edge2 (5->6)=2
Edge3 (1->2)=3
Edge4 (2->6)=4
Edge5 (4->6)=5

Minimumcost=1

2. DesignandimplementC/C++ProgramtofindMinimumCostSpanningTreeofa given connected undirected graph using Prim's algorithm.

```
#include<stdio.h>
inta,b,u,v,n,i,j,ne=1;
intvisited[10]={0},min,mincost=0,cost[10][10];
void main()
{
printf("\nEnterthenumberofnodes:");
scanf("%d",&n);
printf("\nEntertheadjacencymatrix:\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
{
scanf("%d",&cost[i][j]);
if(cost[i][j]==0)
cost[i][j]=999;
}
visited[1]=1;
printf("\n");
while(ne<n)
{
for(i=1,min=999;i<=n;i++)
for(j=1;j<=n;j++)
if(cost[i][j]<min)
if(visited[i]!=0)
{
min=cost[i][j];
 a=u=i;
b=v=j;
}
if(visited[u]==0|| visited[v]==0)
{
printf("\nEdge%d:(%d%d)cost:%d",ne++,a,b,min);
mincost+=min;
visited[b]=1;
}
cost[a][b]=cost[b][a]=999;
}
printf("\nMinimuncost=%d",mincost);
}
```

Output

```
Enterthenumberofnodes:4

 Entertheadjacencymatrix:
999    1    5    2
1    999999999
59999993
29993        999


 Edge1:(12)cost:1
 Edge2:(14)cost:2
 Edge3:(43)cost:3
 Minimun cost=6
```

3a)DesignandimplementC/C++ProgramtosolveAll-PairsShortestPathsproblem using Floyd's algorithm.

```
 #include
 <stdio.h>
 #include<limits.h>
 #define V 4
     voidfloydWarshall(intgraph[V][V])
{
 intdist[V][V];
 for(inti=0;i<V;i++)for(int j=0; j<V;j++) dist[i][j]
= graph[i][j];
 for(int k=0;k<V;k++) for (inti=0;i<V; i++)
 for(intj=0;j<V; j++)
 if(dist[i][k]!=INT_MAX&&dist[k][j]!=INT_MAX&& dist[i][k]+dist[k][j]
 <dist[i][j])dist[i][j]=dist[i][k]+ dist[k][j]
 ("Shortestdistancesbetweeneverypairofvertices:\n"); for
 (int i = 0; i < V; i++)
 {
 for(intj=0; j<V; j++)
 {
 if(dist[i][j]==INT_MAX)
 printf("INF\t");
  else
 printf("%d\t",dist[i][j]);
 }
  printf("\n");
 }
 }

intmain(){
int graph[V][V]={{0,INT_MAX,3,INT_MAX},
{2,0,INT_MAX,INT_MAX},
 {INT_MAX,7,0,1},{6,INT_MAX,INT_MAX,0}};
floydWarshall(graph);
return 0;
}
```

Output:

Shortestdistancesbetweeneverypairofvertices:

```
0    10   3    4
2    0    5    6
7    7    0    1
6    16   9    0
```

3BDesignandimplementC/C++ProgramtofindthetransitiveclosureusingWarshal's algorithm.

```
#include<stdio.h>
intn,a[10][10],p[10][10];
voidpath()
{
int i,j,k;
for(i=0;i<n;i++)
for(j=0;j<n;j++)
p[i][j]=a[i][j];
for(k=0;k<n;k+
+)
for(i=0;i<n;i++)
for(j=0;j<n;j++)
if(p[i][k]==1&&p[k][j]==1)
p[i][j]=1;
}
voidmain()
{
inti,j;
printf("Enterthenumberofnodes:");
scanf("%d",&n);printf("\nEntertheadjacencymatrix:\n");for(i=0;i<n;i++)
for(j=0;j<n;j++)
scanf("%d",&a[i][j]);path();
printf("\nThepathmatrixisshownbelow\n");
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
printf("%d",p[i][j]);
printf("\n");
}
}
```

Output:

Enterthenumberofnodes:4

Enter the adjacency matrix:

0100
0010
0001
1000

Thepathmatrixisshownbelow1111

11 11
11 11
11 11

4. Design and implement C/C++ Program to find shortest paths from a given vertexinaweightedconnectedgraphtootherverticesusingDijkstra'salgorithm.

```
#include<stdio.h>
voiddij(int,int[20][20],int[20],int[20],int); void
main()
 {
inti,j,n,visited[20],source,cost[20][20],d[20]; printf("Enter
no. of vertices: ");
scanf("%d",&n);
printf("Enterthecostadjacencymatrix\n");
 for (i= 1; i <= n; i++)
{
for(j=1;j<=n;j++)
{
 scanf("%d",&cost[i][j]);
}
}
printf("\nEnterthesourcenode:");
scanf("%d", &source);
dij(source,cost,visited,d,n); for
(i = 1; i <= n; i++)
{
if(i!=source)
printf("\nShortestpathfrom%dto%dis%d",source,i,d[i]);
}
}
voiddij(intsource,intcost[20][20],intvisited[20],intd[20],int n)
{
inti,j,min,u,w;
for(i=1; i<=n;i++)
 {
visited[i]=0;
d[i]=cost[source][i];
}
visited[source]=1;
d[source]=0;
for(j=2;j<=n;j++)
{
min =999;
for(i=1;i<=n;i++)
{
if(!visited[i]) {if(d[i]<min)
{
 min=d[i];u = i;
 }
```

```
}
}
visited[u]= 1;
for(w=1;w<=n;w++)
{
if(cost[u][w]!=999&&visited[w] ==0)
{
 if(d[w] >cost[u][w]+d[u])
d[w] =cost[u][w]+d[u];
}
}
}
}
```

Output:

```
Enterno.ofvertices:6
Enterthecostadjacencymatrix
9993     999   9996     5
3   999    1   9999994
9991    999   6   999   4
9999996     999   8    5
6   999999     8   9992
5   4    4   5    2   999

Enter the source node: 1

Shortestpathfrom1to2is3
Shortestpathfrom1to3is4
Shortestpathfrom1to4is10
Shortestpathfrom1to5is6
Shortestpathfrom1to6is5
```

5. DesignandimplementC/C++ProgramtoobtaintheTopologicalorderingof vertices in a given digraph.

```
#include<stdio.h>
voidfindindegree(int[10][10],int[10],int);
void topological(int,int [10][10]);
voidmain()
{
inta[10][10],i,j,n;
printf("Enterthenumberofnodes:");scanf("%d",&n);
printf("\nEnter the adjacency matrix\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&a[i][j]);
printf("\nTheadjacencymatirxis:\n");for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf("%d\t",a[i][j]);
}
printf("\n");
}
topological(n,a);
}
voidfindindegree(inta[10][10],intindegree[10],intn)
{
inti,j,sum;for(j=1;j<=n;j++)
{
sum=0;for(i=1;i<=n;i++)
{
sum=sum+a[i][j];
}
indegree[j]=sum;
}
}
voidtopological(intn,inta[10][10])
{
intk,top,t[100],i,stack[20],u,v,indegree[20];k=1;
top=-1; findindegree(a,indegree,n);
for(i=1;i<=n;i++)
{
if(indegree[i]==0)
{
stack[++top]=i;
}
}
while(top!=-1)
```

```
{
u=stack[top--];t[k++]=u;
for(v=1;v<=n;v++)
{
if(a[u][v]==1)
{
indegree[v]--;
if(indegree[v]==0)
{
stack[++top]=v;
}
}
}
}
printf("\nTopologicalsequenceis\n");for(i=1;i<=n;i++)
printf("%d\t",t[i]);
}
```

Output:

Enterthenumberofnodes:5 Enter

the adjacency matrix
0 0 10 0
0 0 10 0
0 0 01 1
0 0 00 1
0 0 0 0 0

Theadjacencymatirxis:
0 0 1 0 0
0 0 1 0 0
0 0 0 1 1
0 0 0 0 1
0 0 0 0 0

Topologicalsequenceis
2 1 3 4 5

6. DesignandimplementC/C++Programtosolve0/1Knapsackproblemusing Dynamic Programming method.

```
#include<stdio.h>
#define MAX 50
intp[MAX],w[MAX],n;
int knapsack(int,int);
intmax(int,int);
 void main()
{
intm,i,optsoln;

printf("Enterno.ofobjects:");scanf("%d",&n);printf("\nEntertheweights:\n");for(i=1;i<=n;i++)
scanf("%d",&w[i]); printf("\nEnter the profits:\n"); for(i=1;i<=n;i++) scanf("%d",&p[i]);
printf("\nEnter the knapsack capacity:");
scanf("%d",&m); optsoln=knapsack(1,m);
printf("\nTheoptimalsoluntionis:%d",optsoln);
}
intknapsack(inti,int m)
{
if(i==n)
return(w[n]>m)?0:p[n];
if(w[i]>m)
returnknapsack(i+1,m);
returnmax(knapsack(i+1,m),knapsack(i+1,m-w[i])+p[i]);
}
intmax(inta,intb)
{
if(a>b)returna; elsereturn b;
}
```

Output:

Enterno.ofobjects:3

Enter the weights:
100 1014

Entertheprofits:
20 18 15

Entertheknapsackcapacity:116

The optimal solution is:38

7. DesignandimplementC/C++ProgramtosolvediscreteKnapsackand continuous Knapsack problems using greedy approximation method.

```
#include<iostream>
#include
<vector>#include<al
gorithm>
usingnamespacestd;
//Structuretorepresentan
item struct Item
{
intweight;intvalue;
};
//Functiontosolvediscreteknapsackusinggreedyapproach int
discreteKnapsack(vector<Item>& items, int capacity)
{
//Sortitemsbasedontheirvalueperunitweightsort(items.begin(), items.end(),
[](const Item& a, const Item& b)
{
return(double)a.value/a.weight >(double)b.value/b.weight;
});
inttotalValue=0;
intcurrentWeight=0;
//Filltheknapsackwithitemsfor
(const Item& item : items) {
if(currentWeight+item.weight<=capacity)
{currentWeight+=item.weight;
totalValue += item.value;
}
}
returntotalValue;
}
// Function to solve continuous knapsack using greedy approach
doublecontinuousKnapsack(vector<Item>&items,intcapacity){
// Sort items based on their value per unit weight
sort(items.begin(),items.end(),[](constItem&a,constItem&b)
 {
return(double)a.value/a.weight >(double)b.value/b.weight;
});
doubletotalValue=0.0;
int currentWeight = 0;
//Filltheknapsackwithitemsfractionallyfor
(constItem&item:items)
 {
if(currentWeight+item.weight<=capacity)
{currentWeight+=item.weight;
totalValue += item.value;
```

```
    }
    else
    {
    intremainingCapacity=capacity-currentWeight;
    totalValue+=(double)item.value/item.weight*remainingCapacity;
    break;

    }
    }
    returntotalValue;
    }
   intmain()
 {vector<Item>items;intn,capacity;
    //Inputnumber ofitemsandcapacityofknapsackcout
    <<"Enterthenumberofitems:";cin >>n;
    cout<<"Enterthecapacityofknapsack:";cin>>capacity;
    //Inputtheweightandvalueofeach item
    cout<<"Entertheweightandvalueofeachitem:"<<endl;for(int i =0;i <n; i++)
     {
    Itemitem;
    cout<<"Item"<<i+1<<":";cin>>
    item.weight >> item.value;
    items.push_back(item);
    }
    //Solvediscreteknapsackproblem
    intdiscreteResult=discreteKnapsack(items,capacity);
    cout<<"Maximumvalue fordiscreteknapsack:"<<discreteResult<<endl;
    //Solvecontinuousknapsackproblem
    doublecontinuousResult=continuousKnapsack(items,capacity);
    cout<<"Maximumvalueforcontinuousknapsack:"<<continuousResult<<endl;return0;
    }
```

Output:

Enterthe number of items:4
Enterthecapacityofknapsack:10
Entertheweightandvalueofeachitem:
Item1:2 10
Item2:3 5
Item3:5 15
Item 4:7 7
Maximum value for discrete knapsack :30
Maximumvalueforcontinuousknapsack:30

8. DesignandimplementC/C++Programtofindasubsetofagivenset S={sl,s2,.....,sn}ofn positiveintegerswhosesumisequaltoagivenpositiveintegerd.

```
#include<stdio.h>void
subset(int,int,int);
intx[10],w[10],d,count=0;
void main()
{
inti,n,sum=0;

printf("Entertheno.ofelements:");
scanf("%d",&n);
printf("\nEntertheelementsinascendingorder:\n");
for(i=0;i<n;i++)
scanf("%d",&w[i]);
printf("\nEnterthesum:");
scanf("%d",&d);
for(i=0;i<n;i++)
sum=sum+w[i];
if(sum<d)
{
printf("Nosolution\n");
return;
}
subset(0,0,sum);
if(count==0)
{
printf("Nosolution\n");
return;
}
}
voidsubset(intcs,int k,int r)
{
int i; x[k]=1;
if(cs+w[k]==d)
{
printf("\n\nSubset%d\n",++count);
for(i=0;i<=k;i++)
if(x[i]==1)
 printf("%d\t",w[i]);
}
elseif(cs+w[k]+w[k+1]<=d)
subset(cs+w[k],k+1,r-w[k]);
if(cs+r-w[k]>=d&&cs+w[k]<=d)
{
x[k]=0;
```

```
subset(cs,k+1,r-w[k]);
}
}
```

Output:

Entertheno.ofelements:5

Entertheelementsinascendingorder: 1
2
5
6
8

Enterthesum:9

Subset1
1 2 6

Subset2
1 8

9. Design and implement C/C++ Program to sort a given set of n integer elements using Selection Sort method and compute its time complexity. Run the program for varied values of n>5000andrecordthetimetakentosort.Plotagraphofthetimetakenversus n. The elements can be read from a file or can be generated using the random number generator.

```
#include  <stdio.h>
#include<stdlib.h>
#include <time.h>
//Functiontoperformselectionsort
void selectionSort(int arr[], int n)
{
inti,j,minIndex,temp;for
(i= 0; i< n - 1; i++) {
minIndex=i;
for(j=i+1; j<n;j++) {if(arr[j]
<arr[minIndex])
{minIndex=j;
}
}
//Swapthefoundminimumelementwiththefirstelementtemp= arr[minIndex];
arr[minIndex]=arr[i];arr[i]=
temp;
}
}
//Functiontogeneraterandomnumbersbetween0and999 int
generateRandomNumber() {
returnrand()%1000;
}
intmain(){
//Setnvalueint n
= 6000;
//Allocatememoryforthe array
int*arr=(int*)malloc(n*sizeof(int));
//Generaterandomelementsforthearray
srand(time(NULL));
printf("Randomnumbersforn=%d:\n",n); for
 (int i = 0; i < n; i++)
{
arr[i]=generateRandomNumber();
printf("%d ", arr[i]);
}
printf("\n");
//Recordthestarttimeclock_t
start = clock();
//Performselectionsort
```

```
selectionSort(arr,n);
//Recordtheendtime
clock_tend=clock();
//Calculatethetimetakenfor sorting
doubletime_taken=((double)(end -start))/CLOCKS_PER_SEC;
//Outputthetime takentosortforthecurrentvalueofn
printf("\nTimetakentosort forn=%d:%lfseconds\n\n",n,time_taken);
//Displaysortednumbers
printf("Sortednumbersforn=%d:\n",n); for
(int i = 0; i < n; i++)
{
printf("%d",arr[i]);
}
printf("\n\n");
//Freethedynamicallyallocatedmemoryfree(arr);return 0;
}
```

Output:

Randomnumbersforn=6000:
2431125996779124137215476408 22…(morenumbers).....394.....

Timetakentosprtforn=6000:1.058000seconds

Sorted numbers for n=6000:
001223334 5…..(morenumbers)…..995996 996 997998 999999

10. Design and implement C/C++ Program tosort a givenset of n integerelements using Quick Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph ofthetimetakenversusn. The elements can be read from a file or can be generated using the random number generator.

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
//Functiontoswaptwoelementsvoid
swap(int* a, int* b)
{
inttemp =*a;
*a=*b;
*b=temp;
}
//Functiontopartitionthearrayandreturnthepivotindexint
partition(int arr[], int low, int high) {
intpivot=arr[high];int i
=(low-1);
for(intj=low; j<=high-1;j++)
{
if(arr[j]<pivot){i++;
swap(&arr[i],&arr[j]);
}
}
swap(&arr[i+1],&arr[high]);
return (i + 1);
}

//FunctiontoperformQuick Sort
voidquickSort(intarr[],intlow,inthigh)
{
if(low<high){
intpi=partition(arr,low,high);

quickSort(arr, low, pi - 1);

 quickSort(arr,pi+1,high);
 }
 }
//Functionto generaterandomnumbersbetween0and999int generateRandomNumber()
{
returnrand()%1000;
}
intmain(){
//Setnvalueint n=6000;
//Allocatememoryforthearray
int*arr=(int*)malloc(n*sizeof(int));
```

```
//Generaterandomelementsforthearray srand(time(NULL));
printf("Randomnumbersforn=%d:\n",n);for(int i =
0; i < n; i++)
 {
arr[i]=generateRandomNumber();printf("%d",arr[i]);
}
printf("\n");
//Recordthestarttimeclock_t
start = clock();
// Perform quick sort
quickSort(arr,0,n-1);
//Recordtheendtimeclock_t end
= clock();
//Calculatethetimetakenfor
//sorting
double time_taken =
((double)(end-start))/
CLOCKS_PER_SEC;
//Outputthetimetakentosort forthecurrentvalueofn
```

```
printf("\nTimetakentosort forn=%d:%lfseconds\n\n",n,time_taken);
//Displaysortednumbers
printf("Sortednumbersforn=%d:\n",n); for
(int i = 0; i < n; i++)
 {
printf("%d",arr[i]);
}
printf("\n\n");
//Freethedynamicallyallocatedmemory free(arr);
return 0;
}
```

Output:

Randomnumbersforn=6000:
243112 599677912 413 721 547 640 822…(morenumbers).....394.....

Timetakentosprtforn=6000:1.058000seconds

Sorted numbers for n=6000:
00 12 23 3 3 4 5 …..(morenumbers)…..995 996 996 997998 999 999

11. DesignandimplementC/C++Programtosortagivensetofnintegerelementsusing Merge Sort method and compute its time complexity. Run the program for varied values of n> 5000, and record the time taken to sort. Plot a graph of thetimetakenversusn.The elements can be read from a file or can be generated using the random number generator.

```
#include  <stdio.h>
#include<stdlib.h>
#include <time.h>
//Mergetwo subarrays ofarr[]
//First subarrayisarr[l..m]
//Secondsubarrayisarr[m+1..r]
 voidmerge(intarr[],intl,intm,intr)
 {
 inti, j, k;
 intn1=m-l+1; intn2=r-m;
 //CreatetemporaryarraysintL[n1],R[n2];
 // Copy data to temporary arrays
 L[]andR[]for(i=0;i<n1;i++)
 L[i]=arr[l+i];for(j=0;j<n2;j++)R[j]=arr[m+1+j];// Merge the
 temporaryarrays back into arr[l..r]
 i=0;//Initialindexoffirstsubarray
 j=0;//Initialindexofsecondsubarrayk=l;
 //Initialindexofmergedsubarray
 while (i < n1 && j < n2)
 {
 if(L[i]<=R[j])
  {
 arr[k]=L[i];i++;
 }
  else
 {
 arr[k]=R[j];j++;
 }
  k++;
 }
 //CopytheremainingelementsofL[],iftherearеany while (i
 < n1)
 {
 arr[k]=L[i];i++;
 k++;
 }
//CopytheremainingelementsofR[],iftherearеany while(j<n2)
{
arr[k]=R[j];j++; k++;
}
```

```
}
//Mergesortfunction
voidmergeSort(intarr[],intl,int r)
{
if(l<r)
 {
//Sameas(l+r)/2,butavoidsoverflowforlargelandr int m =
l + (r - l) / 2;
//Sortfirstandsecondhalves mergeSort(arr,
l,m);mergeSort(arr,m+1,r);
//Mergethesortedhalves
merge(arr, l, m, r);
}
}
 //Functiontogeneraterandomnumbersbetween0and999 int
 generateRandomNumber()
 {
    returnrand()%1000;
 }
 intmain()
 {
//Setnvalueint n
= 6000;
//Allocatememoryforthe array
int*arr=(int*)malloc(n*sizeof(int));
//Generaterandomelementsforthearray
srand(time(NULL));
printf("Randomnumbersforn=%d:\n",n); for
 (int i = 0; i < n; i++)
{
arr[i]=generateRandomNumber();printf("%d",arr[i]);
}
printf("\n");//Recordthestarttimeclock_tstart=clock();
// Perform merge sort
mergeSort(arr,0,n-1);
//Recordtheendtimeclock_tend=clock();
//Calculatethetimetakenfor sorting
doubletime_taken=((double)(end -start))/CLOCKS_PER_SEC;
//Outputthetimetakentosort forthecurrentvalueofn
printf("\nTimetakentosort forn=%d:%lfseconds\n\n",n,time_taken);
//Displaysortednumbers
printf("Sortednumbersforn=%d:\n",n); for
(int i = 0; i < n; i++)
 {
printf("%d",arr[i]);
}
```

```
printf("\n\n");
//Freethedynamicallyallocatedmemory
free(arr);
return0;
}
```

Output:

Randomnumbersforn=6000:
243112 599677912 413 721 547 640 822…(morenumbers).....394.....

Timetakentosprtforn=6000:1.058000seconds

Sorted numbers for n=6000:
00 12 23 3 3 4 5 …..(morenumbers)…..995 996 996 997998 999 999

12. DesignandimplementC/C++ProgramforNQueen'sproblemusingBacktracking.

```
#include <iostream>
#include<vector>using
namespace std;
//Functionto printthesolution
voidprintSolution(constvector<vector<char>>&board)
{
for(const auto&row : board)
 {
for(charcell: row)
cout<<""<<cell<<"";cout<<endl;
}
}
//Functiontocheckifaqueencanbeplacedonboard[row][col]bool isSafe(const
vector<vector<char>>& board, int row, int col)
 {
 inti, j;
 intn=board.size();
 //Checktherowontheleftside for
  (i = 0; i < col; i++)
  if(board[row][i]=='Q')returnfalse;
 //Checkupperdiagonalontheleftside
    for(i=row,j=col;i>=0&&j>=0;i--,j--) if
  (board[i][j] == 'Q')
  returnfalse;
 //Checklowerdiagonalontheleftside
    for(i=row,j=col;j>=0&&i<n;i++,j--)if (board[i][j] ==
'Q')
 returnfalse;return
   true;
}
//RecursivefunctiontosolveNQueensproblem
boolsolveNQUtil(vector<vector<char>>&board,intcol)
 {
  intn=board.size();
//Ifallqueensareplaced,returntrueif(col>=n)
return true;
//Considerthiscolumnandtryplacingthisqueeninallrowsonebyone for (int i
  = 0; i < n; i++)
 {
//Checkifthequeencanbeplacedonboard[i][col] if
(isSafe(board, i, col))
{
//Placethisqueeninboard[i][col]board[i][col]='Q';
//Recurtoplacerestofthequeens
```

```
 if(solveNQUtil(board,col+1))
returntrue;
//If placingqueeninboard[i][col]doesn'tleadtoasolution,then removequeenfrom board[i][col]
board[i][col]='-';
}
}
//Ifthequeencannotbeplacedinanyrowinthiscolumn,thenreturnfalsereturn false;
}
//FunctiontosolveNQueensproblemfor4queensvoid
solve4Queens()
 {
intn=4;
vector<vector<char>>board(n,vector<char>(n,'-'));if
(solveNQUtil(board, 0) == false)
 {
cout<<"Solutiondoesnotexist"<<endl;return;
}
printSolution(board);
}
//Driverfunction
int main() {
cout<<"Solutionfor4Queensproblem:"<<endl;solve4Queens();return 0;
}
```

Output:

```
Solutionfor4Queensproblem:
-    -Q   -
-   Q  -  -  -
-    -  -  Q
-   Q-    -
```

VIVAQUESTIONS

1. WhatisanAlgorithm?

AlgorithmisaStepbystepproceduretoSolveagivenproblemforafinitenumberofinput producing finitenumber of output with desired output.

2. WhatisaFlowChart?

Flowchart isaGraphicalRepresentationofasolutiontotheProblem.

3. WhatisthedifferencebetweenAlgorithm,FlowChart,Program?

AlgorithmspecifiesthedifferentthingstobefollowedforsolvingaProblem.

FlowChartisaGraphicalRepresentationofaSolutiontotheProblem.Both Algorithm and Flow Chartare Machine Independent.

ProgramisaSetofInstructionswhichisusedasatooltocommunicatetothemachine to get our workdone,Program is Machine Dependent for particular Machine.

4. WhatistheAimofDAAlaborwhyweneedtostudyDAALab

DAA is a discipline, where we are dealing withdesigningorwritingthealgorithm keeping in ConsiderationofSpaceandTimeComplexity,SuchthatOurAlgorithmshouldexecute in a very minimum amount of time by Minimum Space or RAM.

5. DefineSpaceandTimeComplexity?Among thiswhichoneismoreprioritized?SpaceComplexeityisameasureofAmount ofSpacetakenbyaProgramtofinishitsExecution. Time Complexeityisameasureofamountoftimetakenbyaprogramtocomplteits Execution.DependingUponApplicationitisconsidered,EX:ForMobileorHandheldDevices,We give Prefernce for both Spaceand time.

ForaHugeandInteractiveSystemslikeWebApplicationswegivemorePreferencestotime Complexeity.

6. WhatisDesignandwhatisAnalysisofaProgram?

DesignisaProcessofWritinganalgorithmtoagivenProblemsothatitshouldacceptfinitenumber ofinput and finite number ofoutput with adefinite output and Should Exit appropriately.

Analysis:AnalysisisanextPhaseofWritinganAlgorithm,inthisphasewecalculatetheEfficiencyof anAlgorithm i.e time and space needed by an algorithm.

7. Writethegeneralplanforanalyzingtherecursivealgorithms.

- Identifytheinputs.
- Identifytheoutputisdependedonlyonnumberofinputs.
- IdentifytheBasicOperationinAlgorithm.
- FormorwritetheRecursiveRelationtotheAlgorithm.

8. Whatarethevariousnotationsusedtowriteanalgorithm?

(i)Pseudocode(ii)NaturalLanguageandetc..

9. WhatisaPseudocode?

It'sanotationwhichishavingthecombinationofProgrammingConstructsandEnglishlike Statements.

10. WhatistheTimeComplexeityofBubbleSort,SelectionSort,MergeSort,Quick Sort?(L3)

BubbleSort-n^2,SelectionSort-n^2MergeSort-nlog.nQuickSort -nLogn,WorstcaseforQuickSort-n2

11. WhichsortingagorithmismoreEfficientandwhy?

QuickSortingisMoreEfficient,becausethisalgorithmisinstablealgorithm ndinplace.

12. WhatdoyoumeanbythetermInstableAlgorithms?

TheInstableAlgorithmsareone,whichdividesthearrayascertainlydependinguponpivotor keyelementand hence i index precedes index j

13. Whichalgorithmsarefaster?

InstableAlgorithmsaremuchFastercomparedtoStableAlgorithms.

14. ForwhattypeofinstanceMergesortdobetterthanQuick Sort?

ForaLargerinputandasortedinputvalues.

15. ForwhattypeofinstanceQuicksortdobetterthanMergeSort?

ForSmallerSetofinputnumbers.

16. WhatareInplaceAlgorithms?

InplaceAlgorithmsaretheonewhichdoesn't occupiesExtraSpace.

17. Writetheorderofgrowthtermsasperthetime ExecutioninAscendingOrder.

logn,n,nlogn,n^2,n^3, ... n^n,2^n,n!

18. WhatisBruteForceTechnique?WhenWeShouldUse?

BruteForceisastraightForwardTechniquetosolveaproblem,Weusedtosolvea Problemthroughthisapproachwhenwedon'thavesufficientdatatosolveaproblemin Efficient Way.

19. Whatisthedifferencebetween DivideandConquer,DecreaseandConquer?

DivideandConquercanbesolvedtosolveaproblemwithalargerdatasetandwhen there is nodependency between any of the data sets.

- DivideandSolve asSmallasSmallsets.

ConquerorMergeitgetonefinalresultantdataset.

Decrease and Conquer is almost similar to Divide and Conquer but we are finding a solutionsto theproblem in a different variations,EX:Decrease by Constant (Usually by One),Decrease by Constant factorwhich is almost similar to Divide and Conquer Technique(Usually by two),Decrease by Variable(The Dividing Criteria changes for each iteration depends upon the data set.

20. DefineGreedyTechnique.

GreedyTechniqueisalwaysappliedfortheproblemofthetypeoptimizationtype, which reduces loss and increases profit.

21. DefineOptimal andFeasibleSolution.

OptimalSolutionisasolutionwhichisbestamongNFeasibleSolution.FeasibleSolutionisa solution which Satisfies a Problem Constraints/conditions.

22. CanAProblemsolvedbyallthealgorithmic Techniques.

Yes,butsomeproblemswillgivebetterresultswithsomeAlgorithmicTechniqueandit may give worstresultwhen itis applied with other technique.

23. StateandExplainKnapsackProblem.

FillingtheMaximumnumberofitemstotheKnapsack(Container)WhichIncreases the profit anddecreases the Loss.

24. WhichoneisMostAdmiredalgorithmicTechnique?

DynamicProgramming.

25. WhatisSpanningtreeandMinimumSpanningtree?

AtreeWithoutCyclesarecalledasSpanningtree.AMinimumSpanningTreeisa spanning tree whichyeilds the very less Costwhen all the edges costsummed up.

26. HowManySpanningTreecanaTreecanhave?

Atreecanhave1tomanynumberofPossiblewaysofSpanning Tree.

27. DifferentiatebetweenPrimsandKruskalsAlgorithmfor findingMST.

InPrims Weconsider anyonevertexinthegraphasSourceandWecomputethe distance from that sourceto othervertices,aftercomputingtheverticeswhichhas minimumvalueamong (n-1)verticesisaddedtotreeverticesandthatrespectiveedgesaddedtotreeEdgesSet.Theabove mentionedProcesscontinuestillwereach(n-1) vertices.

In Kruskals we first arrange the edges in Ascending Order and then we start to form the tree which wont formcycles,if adding that edges forms cycles then that edges is dropped from adding to tree edges.The above saidprocess is continues till we reach the count of (n-1) Vertices.

28. WhatistheApplicationofPrimsand KruskalsAlgorithm?

InNetworkstoremovetheCyclicityoftheNetwork.

29. ExplainJobSequencingWithDeadlines?

PlacingorschedulingthemaximumnumberofJobstoamachinewithoutviolatingthedeadlines constraintof any of the Jobsin Sequence.

30. WhytheNameBubbleSortnamed?

BecauseinfirstPassthefirsthighestdatawillbubblesup,sosincethelargest elementbubblesupinthefirstandsecond largest element bubbles up in the Secondpass and so on, so hence the name bubble sort.

31. WhytheNameSelectionSort?(L3)

The Selection sort is named because we initially first select an arrays first element as minimumandwillcomparewithotherelements,soinpassonefirstleastelementgoestothe first position and so on so forth for 2nd,3rd and so on. Selecting

32. WhatisthedifferencebetweenBruteforcestringsmatchingtoHorspoolString Matching Method? (L2)

InbruteForcewecompareeachandeveryelementofthetexttothepatternbyshiftingthetext positionbyoneand in Horspoolmethod weshiftit by numberofshift positions recordedin theshifttable.

33. ExplainMergeSort?

InMerge Sort willdivide the entire input set by2 untilwe reach low<highand later will findasolutiontoeachitembycomparinghalfofthearraydatasettotheotherhalfarraydataset and finally we mergeit to form a sinle array(conquer)

34. WhatistheBasicOperationsinMergesortandQuicksort?

InMergeSorttheBasicOperationsisComparisionsandinQuicksortbasicOperationsis Partitioning andhence also known as partitioning sort.

35. WhytheInsertionSort?

WeareInsertinganelementtoitssuitableplacebycomparingnelementsforeachpass.

36. WhatistheUseofDFSand BFS?

DFSandBFSbothusedtocheckttheConnectivityofagraph,Cyclicityinagraph,Spanning tree of agraph.

37. DifferentiatebetweenDFSandBFS.

DFSandBFSareboththeGraphTraversingTechnique,inwhichDFSTraversetheGraphin a depthwise(Vertical) and BFS Traversethe Graph fromleft toright(Horizontal)

38. WhichDatastructuresusedinBFSandDFS.

BFSUSesQueueasitsdatastructureandDFS usesasstackitsDatastructure.

39. WhatarebackedgesinDFSandCrossEdgesinBFS.

BackEdgesandCrossedgesaretheEdgeswhich alreadyvisitedbyaancestornode.

40. WhatisTopologicalSorting?

TopologicalSortingisaSortingTechniqueusedforsortingVerticesintheGraph.

41. WhatistheConditionsnecessaryforaTopologicalSorting?

ForaTopologicalSortingtheGraphShouldbeDAG(DirectedAcyclicGraph)

42. WhataretheDifferentmethodsusedtosolveatopologicalSorting?

1. SourceRemovalMethod
2. UsingDFSbasedScheme.

43. WhatistheUseofTopologicalSorting?

UseofTopologicalOrderingisinthefieldofOperatingSystemforSchedulingandin Networks,Automation and Robotics.

44. WhatisDijikstra'sAlgorithm?

Dijikstra'sAlgorithmisUsedtofindtheSingleshortestPathfromsourcetotheothervertex.

45. Whatisagraph?

Graphisacomponentwhichishaving asetofEdgesandverticesG={V,E}

46. Whatarethedifferentwaysthatcanberepresentsagraph?

AdjacenyMatrixandAdjacencyList.

47. WhatisAdjacencyMatrix?

IsaMatrixwhichillustratestheGraphintheformofMatrix,ifitisaweightsGraphthenwe initialize the value of the cost in that position(i,j) or else simply we write 1 to mention ther exist an edgebetween (i,j)ORelseweuse0 or 9999 tomention non connectivity ofa graph.

48. WhatisthelimitationsofAlgorithms?

Algorithmcan't findthebetterthesoltionswhenwecomeacrossthetightlowerbound,So wecan findthebettersolutinswhichisnotpossiblewithAlgorithmicway.TofindtheTightlowerboundwe use DecisionTrees.

49. WhatisTightlowerBound?

It is a Lower bound which is a best lowerboundforanproblem,beyondthatno algorithm will produce better results.

50. WhatareDecision Trees?

Decision trees are also known as Comparision Trees used to find the tight lower bound for a particular Problem EX:Tight Lower Bound For Sorting is n.logn
andtight lowerboundforSearchingislognwhichisnotpossibletogetthebetterresult.

51. Whatisapolynomialproblem(P-type)

P-typeproblemaredecisionproblemsinwhichwecanfindthesolutionsinapolynomial time and
is oftype deterministic.

52. WhatisNP-problem?

NP-ProblembelongstodecisionproblemandtheseproblemsareNonDeterministicPolynomiali.e forwhichthe problemdoesn't havedeterministic solutionsand Canbe solved in Polynomial time There are 2 phases in solving a problem.

(i) Guessing(Non-Deterministicstage) ProducingN numberofCandidateOutputs.

(ii) Verification(DeterministicStage)VerifyingThecorrectnessofNNumberof
Candidate Outputs.

53. WhatisNP-CompleteProblems?

Np_Complete Problems belongs to Decision problems and NP type Problems . These problems canbefind the solutionsbyconverting orreducing to theproblemwhichweknow theSolutions.

54. Whatisatriviallowerbound?

Trivialboundcanbe derived byformulating thenumberofinputsthathastobe givenand number

ofoutputsthat has to be generated.

55. ExplainBactrackingW.r.t

(I)SubsetProblem(ii)N-QueensProblem

56. ExplainSubsetProblem.

In a givenSet S ,find the Subset,in which the sum ofall subsetelements is equal tothe sum d which is predefined in a proble m.

57. ExplainN-QueensProblem.

N-Queens Problem is of Placing a N-Queensin a N*N Chess board such thatNo 2- Queens Should be placed in the same Row,Column and same diagnol(N=Should consider both principal diagonal elements)

58. WhatisHamiltonianCircuit?(L2)

Hamiltoniancircuitisaprobleminwhichthatcircuitstartsfromasourcevertexandhasother vertex in any order without repeating and Should end with the Source vertex only i.e source and Destination vertexshould be same.

59. ExplaintheProblemof TSP.(L2)

TravellingSalesPersonProblemisaprobleminwhichheshouldVisitNnumberofcitieswith aminimumnumber of Cost by visiting every city Exactly one and the city what he is started should end with same city.

60. WhatistheConceptofDynamicProgramming?(L3)

Derivinga Solutiontothebasic ConditionandExtractingthesolutionsfortherestoftheother data sets by Previously drawnd Solution.Computer to other algorithmic Technique Dynamic Programmingbecauseitavoidslotofreworkingonthe sameSolutionwhatwehave solvedin the earlierphasesofderivingthesolutiontothe

problem.

61. WhatisthegoalofWarshallsAlgorithm?(L3)

Warshall's algorithm is tofindthe shortestdistance between a node toall the other nodes in the graph.It Uses the Property of Transitive Closurei.e if there exist a path between (i,k) and (k,j) then there surely exista path between (i,j)

(i,k)&(k,j)---□(I,J)WhatistheuseofFloydsalgorithm?(L3)Itisuseto find

the All pairs shortest Path of an Graph.