ShramSadhanaBombayTrust's
COLLEGEOFENGINEERINGANDTECHNOLOGY,
BAMBHORIPOSTBOXNO.94,JALGAON– 425001. (M.S.)
Includedundersection2(f)&12(B)oftheUGCAct,1956
ISO 9001:2008 certified
PhoneNo.(0257)2258393,FaxNo.(0257)2258392
Website-www.sscoetjalgaon.ac.inEmail:
sscoetjal@gmail.com

ISO9001:2008

DEPARTMENT OF COMPUTER

LaboratoryManuals

Class: T.E. I.T/Comp (Term-II)

Subject:DesignandAnalysisofAlgorithmLab Academic

Year: 2023-24

Semester: VI

Name of the Faculty :Mayuri Chandratre

VisionoftheDepartment

Todeveloptechnocratsasperindustrytechnicalneedsandsocialvalues.

MissionoftheDepartment

Toprovideconduciveenvironmenttoimparttechnicalknowledge throughteaching,self-learningandskilldevelopmentprogramsto stand out in competitive world.

ProgramEducationalObjectives(PEOs)

1. **CoreKnowledge**
   To provide students with Core Knowledge in mathematical, scientific and basic engineeringfundamentalsnecessarytoformulate,analyzeandsolveengineering problems and also to pursue advanced study or research.

2. **Employment/ContinuingEducation**
   TotrainstudentswithgoodbreadthofknowledgeincoreareasofInformation Technology and related engineering so as to comprehend engineering trade-offs, analyze, design, and synthesize data and technical concepts to create novel products and solutions for the real life problems.

3. **ProfessionalCompetency**
   To inculcate in students to maintain high professionalism and ethical standards, effective oral and written communication skills, to work as part of teamsonmultidisciplinaryprojectsanddiverseprofessionalenvironments,and relate engineering issues to the society, global economyand to emerging technologies.

ProgramOutcomes(POs)

- **Engineering knowledge: Apply the knowledge of mathematics, science,** engineeringfundamentals,andanengineeringspecializationtothesolutionof complex engineering problems.
- **Problem analysis:** Identify, formulate, review research literature, and analyze complexengineeringproblemsreachingsubstantiatedconclusionsusingfirstprinciples of mathematics, natural sciences, and engineering sciences.
- **Design/developmentofsolutions:**Designsolutionsforcomplexengineeringproblems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- **Conductinvestigationsofcomplexproblems:**Useresearch-basedknowledge andresearchmethodsincludingdesignofexperiments,analysisandinterpretationof data, and synthesis of the information to provide valid conclusions.
- **Moderntoolusage:**Create,select,andapplyappropriatetechniques,resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- **Theengineerandsociety:**Applyreasoninginformedbythecontextualknowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- **Environment and sustainability:** Understand the impact of theprofessional engineeringsolutionsinsocietalandenvironmentalcontexts,anddemonstrate the knowledge of, and need for sustainable development.
- **Ethics:**Applyethicalprinciplesandcommittoprofessionalethicsandresponsibilities and norms of the engineering practice.
- **Individualandteamwork:**Functioneffectivelyasanindividual,andasamember or leader in diverse teams, and in multidisciplinary settings.
- **Communication:**Communicateeffectivelyoncomplexengineeringactivitieswith theengineeringcommunityandwithsocietyatlarge,suchas,beingabletocomprehend and write effective reports and design documentation, make effective presentations,and give and receive clear instructions.
- **Projectmanagementandfinance:**Demonstrateknowledgeandunderstandingof the engineering and management principles and apply these to one's own work,as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **Life-longlearning:**Recognizetheneedfor,andhavethepreparationandabilityto engage in independent and life-long learning in the broadest context of technological change.

ProgramSpecificOutcomes(PSO)

ITGraduateswillbeableto:

1. SoftwareSystem:Toapplysoftwareengineeringprinciplestostudy, Analyze,design, implement, test and maintain software system.
2. **Open Source Software:** Demonstrate familiarity and practical competencewithabroadrangeofprogramminglanguagesandopen source platforms.
3. **ComputerProficiency:** Exhibitproficiency throughlatest technologies indemonstratingtheabilityforworkefficacytotheindustry&society.

CourseOutcomes(COs)

SubjectName:DesignandAnalysisofAlgorithmsLab

UponsuccessfulcompletionoflabCourse,studentwillbeableto:

CO1:AnalyzeandImplementdivideandconquerapproach. CO2: Implement dynamic programming approach

CO3:ImplementBranchandboundingapproach CO4: Implement backtracking approach.

CO5:Implementgreedyalgorithmapproach

CO-PO-PSOMappingforDesignandAnalysisof Algorithms Lab

CO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	3	3	2	2	2	1			1	2	1	2	3	2	1
CO2	3	3	2	2	2	1	1	1	1	2	1	2	3	2	1
CO3	3	3	2	2	2	1	1	1	1	2	1	2	3	2	1
CO4	3	3	2	2	2	1	1	1	1	2	1	2	3	2	1
CO5	3	3	2	2	2	1	1	1	1	2	1	2	3	2	1
Average	3	3	2	2	2	1	1	1	1	2	1	2	3	2	1

SSBT'sCollegeofEngineering&Technology,Bambhori,Jalgaon

DepartmentofComputer/ITEngineering

T.E.ITSemVI2

022-23

DesignandAnalysisof Algorithms

Lab

ListofExperiment

Sr.No.	Title
1	Analyze&ImplementInsertionsortalgorithm
2	Analyze&ImplementQuicksortalgorithmusingDivideand Conquer
3	Implement0/1KnapsackusingDynamicProgramming
4	ImplementTravelSalesmanproblemusingBranchandBounding
5	ImplementgraphcoloringProblemusingbacktracking
6	ImplementjobsequencingAlgorithmusingGreedyAlgorithm

SSBT'sCollegeofEngineering&Technology,Bambhori,Jalgaon
DEPARTMENTOFINFORMATIONTECHNOLOGY

Name	**Class:T.E.**
Subject: Design and analysis of algorithm	**Roll**
Date of	**Date of**

EXPERIMENT NO:

TITLE:Programforinsertionsort

AIM: Writeaprogramforsortingofnumbersusinginsertionsort algorithm.

Hardware/SoftwareRequirement:PC,WindowsXP/7**,**Ccompiler,Editor.

Theory

Insertionsortisasimplesortingalgorithmthatbuildsthefinalsortedarray(orlist)oneitem atatime.Itismuchlessefficientonlargeliststhanmoreadvancedalgorithmssuchasquicksort,heap sort, or merge sort. But it is very easy to implement and efficient for small data sets.

Best,worst,andaveragecases:

Thebestcaseinputisanarraythatisalreadysorted.Inthiscaseinsertionsorthasalinearrunningtime (i.e., O(*n*)).During each iteration, the first remaining element of the input is only compared with the right-most element of the sorted subsection of the array.

Thesimplestworstcaseinputisanarraysortedinreverseorder.Thesetofallworstcaseinputsconsists of all arrays where each element is the smallest or second-smallest of the elements before it. In these cases everyiteration oftheinnerloop will scan and shift theentiresorted subsection ofthe array before inserting the next element. This gives insertion sort a quadratic running time (i.e., O (n^2)).

Theaveragecaseisalsoquadratic,whichmakesinsertionsortimpracticalforsortinglargearrays. However, insertion sort is one of the fastest algorithms for sorting very small arrays.

HowtoimplementInsertionSort?

1. Considerthefirst elementtobesortedand therest tobe unsorted.
2. Comparewiththesecond element:Ifthesecondelement<thefirstelement,inserttheelementin the correct position of the sorted portion. Else, leave it as it is.
3. Repeat1and2untilallelements aresorted.

Result/Output:-

_____

_____

_____

ReferenceBooks:-

1. Aho,“Design&AnalysisofComputerAlgorithms”,PearsonLPE.
2. RussMiller,“Algorithms:SequentialandParallel”,DreamtechPress.
3. Goodrich,“AlgorithmDesign:FoundationandAnalysis”,WileyIndia.
4. Grama,“AnIntrotoParallelComputing:Design&AnalysisofAlgorithms”,Second Edition, Pearson LPE.
5. Baase,“ComputerAlgorithms:IntrotoDesign&Analysis”,ThirdEdition,PearsonLPE

QuestionsforViva:-

1. Explaininsertionsortwithexample.
2. Writeanpseudocodeforinsertionsort.
3. Explainbestcase,worstcaseandaverage caseof insertionsort.

NameofTeacher **Sign**

Miss. Mayuri Chandtratre

```
//Cprogramforinsertionsort
#include<math.h>#include
<stdio.h>
/*Functiontosortanarrayusinginsertionsort*/
void insertionSort(int arr[], int n)
{
        inti,key,j;
        for(i= 1;i <n;i++){
                key=arr[i];
                j = i - 1;
                /*Moveelementsofarr[0..i-1],thatare
                greater than key, toone position ahead
                of their current position */
                while(j>=0&&arr[j]>key){
                        arr[j + 1] = arr[j];
                        j=j-1;
                }
                arr[j+1]=key;
        }
}
//Autilityfunctiontoprintanarrayofsizen void
printArray(int arr[], int n)
{
        inti;
        for(i=0;i <n;i++)
```

```
                printf("%d",arr[i]);
        printf("\n");
}

/*Driverprogramtotestinsertionsort*/ int
main()
{
        intarr[]={12,11,13,5,6 };
        intn=sizeof(arr)/ sizeof(arr[0]);

        insertionSort(arr,n);
        printArray(arr, n);

        return0;
}
```

SSBT'sCollegeofEngineering&Technology,Bambhori,Jalgaon
DEPARTMENTOFINFORMATIONTECHNOLOGY

Name	**Class:T.E.**
Subject: Design and analysis of algorithm	**Roll**
Date of	**Dateof**

EXPERIMENT NO:

TITLE:ProgramforQuicksort.

AIM: Writeaprogramforquicksort ofnumbersusingdivideandconquerapproach.

Hardware/SoftwareRequirement:PC,WindowsXP/7**,**Ccompiler,Editor.

Theory

QuickSortisalsobasedontheconceptof**DivideandConquer**,justlikemergesort.Butinquick sort all the heavy lifting(major work)is done while **dividing**the array into sub arrays, while in case of mergesort,alltherealworkhappensduring**merging**thesubarrays. Incaseofquicksort,thecombine step does absolutely nothing.

Itisalso called**partition-exchangesort**.Thisalgorithmdividesthelistintothreemainparts:

1. Elementsless thanthe**Pivot**element
2. Pivotelement(Centralelement)
3. Elementsgreaterthanthepivotelement

Pivotelementcanbeanyelementfromthearray,itcanbethefirstelement, thelastelementorany random element. In this tutorial, we will take the rightmost element or the last element as **pivot**.

HowtoimplementQuickSort?

Followingarethestepsinvolvedinquicksortalgorithm:

1. Afterselectinganelementas**pivot**,whichisthelastindexofthearrayinourcase,wedivide thearrayforthefirsttime.
2. In quick sort, we call this **partitioning**. It is not simple breaking down of array into 2 sub arrays,butincaseofpartitioning,thearrayelementsaresopositionedthatalltheelements smallerthanthe **pivot**will beontheleft sideofthepivot andall theelements greaterthan thepivotwillbeontherightsideofit.
3. Andthe**pivot**elementwillbeatitsfinal**sorted**position.

4. Theelementstotheleftandright,maynotbesorted.
5. Thenwepicksubarrays,elementsontheleftof**pivot**andelementsontherightof**pivot**, and weperform**partitioning** onthembychoosinga**pivot** inthesubarrays.

ComplexityAnalysisofQuickSort

Foranarray,inwhich**partitioning**leadstounbalancedsubarrays,toanextentwhereontheleftside there are no elements, with all the elements greater than the **pivot**, hence on the right side. And if keep on getting unbalanced sub arrays, then the running time is the worst case, which is $O(n^2)$ Where as if **partitioning** leads to almost equal sub arrays, then the running timeis thebest, with time complexity as **O(n*log n)**.

WorstCaseTimeComplexity[Big-O]: **$O(n^2)$**
BestCaseTimeComplexity[Big-omega]:**O(n*logn)**
Average Time Complexity [Big-theta]: **O (n*log n)**
Space Complexity: **O (n*log n)**

Result/Output:-

_____

_____

_____

ReferenceBooks:-

1. Aho,"Design&AnalysisofComputerAlgorithms",PearsonLPE.
2. RussMiller,"Algorithms:SequentialandParallel",DreamtechPress.
3. Goodrich,"AlgorithmDesign:FoundationandAnalysis",WileyIndia.
4. Grama,"AnIntrotoParallelComputing:Design&AnalysisofAlgorithms",Second Edition, Pearson LPE.
5. Baase,"ComputerAlgorithms:IntrotoDesign&Analysis",ThirdEdition,PearsonLPE

QuestionsforViva:-

1. Explainquick sortwithexample.
2. Writeanpseudocodeforquicksort.
3. Explainbestcase,worstcaseandaveragecaseofquicksort.

NameofTeacher **Sign**

Miss. Mayuri Chandratre

ImplementingQuick SortAlgorithm

BelowwehaveasimpleCprogramimplementingtheQuicksort algorithm:

```
//simpleCprogramforQuickSort #
include <stdio.h>

// to swap two numbers
voidswap(int*a,int* b)
{
   int t =*a;
   *a=*b;
   *b =t;
}

/*
   a[]isthearray,pisstartingindex,thatis0, and r is
   the last index of array.
*/
voidquicksort(inta[], intp,intr)
{
   if(p<r)
   {
      int q;
      q=partition(a,p,r);
      quicksort(a, p, q);
      quicksort(a,q+1,r);
   }
}

intpartition(inta[],int low,int high)
{
   intpivot=arr[high];//selectinglastelementaspivot int i =
   (low - 1);// index of smaller element

   for(int j =low;j <=high-1;j++)
   {
      //Ifcurrentelementissmallerthanorequaltopivot if
      (arr[j] <= pivot)
      {
         i++;    //incrementindexofsmallerelement
         swap(&arr[i], &arr[j]);
      }
   }
   swap(&arr[i+1],&arr[high]);
```

```
    return(i+1);
}

// function to print the array
voidprintArray(inta[],intsize)
{
    inti;
    for(i=0;i <size; i++)
    {
        printf("%d", a[i]);
    }
    printf("\n");
}

intmain()
{
    int arr[] ={9, 7,5, 11, 12, 2,14, 3, 10, 6};
    int n =sizeof(arr)/sizeof(arr[0]);

    //callquickSortfunction
    quickSort(arr, 0, n-1);

    printf("Sortedarray:n");
    printArray(arr, n);return
    0;

}
```

SSBT'sCollegeofEngineering&Technology,Bambhori,Jalgaon
DEPARTMENTOFINFORMATIONTECHNOLOGY

Name **Class:T.E.**

Subject: Design and analysis of algorithm **Roll**

Date of **Date of**

EXPERIMENTNO:

TITLE:0/1Knapsackproblem

AIM: Writeaprogramtosolve0/1knapsackproblemusingdynamicprogramming approach.

Hardware/SoftwareRequirement:PC,WindowsXP/7**,**Ccompiler,Editor.

Theory

Definition of 0/1 knapsack problem is"Given a set of items, each with a weight and a value,determinethenumberofeachitemtoincludeinacollectionsothat thetotalweightis less than or equal to a given limit and the total value is as large as possible."

Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack. In other words, given two integer arrays val[0..n-1] and wt[0..n-1] which represent values and weights associated with n items respectively. Also givenanintegerWwhichrepresentsknapsackcapacity,findoutthemaximumvaluesubsetof val[] such that sum of the weights of this subset is smaller than or equal to W. You cannot break an item, either pick the complete item, or don't pick it (0-1 property).

A simple solution is to consider all subsets of items and calculate the total weight and valueofallsubsets.Considertheonlysubsetswhosetotalweightissmallerthan W.Fromall such subsets, pick the maximum value subset.

1) Optimal Substructure:
Toconsiderallsubsetsof items,therecanbetwocasesforeveryitem:(1)theitemisincluded in the optimal subset, (2) not included in the optimal set.
Therefore,themaximumvaluethatcanbeobtainedfromnitemsismaxoffollowingtwo values.
1) Maximumvalueobtained byn-1items andWweight (excludingnth item).
2) Valueofnthitemplusmaximumvalueobtainedbyn-1itemsandWminusweightofthe nth item (including nth item).

IfweightofnthitemisgreaterthanW,thenthenthitemcannotbeincludedandcase1isthe only possibility.

2) OverlappingSub problems
Problemsareevaluatedagainandagain,thisproblemiscalledOverlappingSubproblem'sproperty.

TimeComplexity:

O(nW) wheren is thenumberofitems and Wis thecapacityofknapsack.

Result/Output:-

_____

_____

_____

ReferenceBooks:-

1. Aho,"Design&AnalysisofComputerAlgorithms",PearsonLPE.
2. RussMiller,"Algorithms:SequentialandParallel",DreamtechPress.
3. Goodrich,"AlgorithmDesign:FoundationandAnalysis",WileyIndia.
4. Grama,"AnIntrotoParallelComputing:Design&AnalysisofAlgorithms",Second Edition, Pearson LPE.
5. Baase,"ComputerAlgorithms:IntrotoDesign&Analysis",ThirdEdition,PearsonLPE

QuestionsforViva:-

1. whatis0/1knapsackproblem?.
2. Explain0/1knapsackproblemwithexample.

NameofTeacher **Sign**

Miss. Mayuri Chandratre

```
//ADynamicProgrammingbasedsolutionfor0-1Knapsackproblem
#include<stdio.h>

//Autilityfunctionthatreturnsmaximumoftwointegers int
max(int a, int b) { return (a > b)? a : b; }

//Returnsthemaximumvaluethatcanbeputinaknapsackofcapacity W int
knapSack(int W, int wt[], int val[], int n)
{
int i, w;
intK[n+1][W+1];

//BuildtableK[][]inbottomupmanner for (i
= 0; i <= n; i++)
{
        for(w=0; w<=W;w++)
        {
                if(i==0 ||w==0)
                        K[i][w]=0;
                elseif(wt[i-1]<=w)
                                K[i][w]=max(val[i-1]+K[i-1][w-wt[i-1]],K[i-1][w]);
                else
                                K[i][w]=K[i-1][w];
```

```
        }
}

returnK[n][W];
}

intmain()
{
        int val[]={60, 100, 120};
        int wt[]={10, 20, 30};
        int W=50;
        int n = sizeof(val)/sizeof(val[0]);
        printf("%d",knapSack(W,wt,val,n));
        return 0;
}
```

SSBT'sCollegeofEngineering&Technology,Bambhori,Jalgaon
DEPARTMENTOFINFORMATIONTECHNOLOGY

Name	**Class:T.E.**
Subject: Design and analysis of algorithm	**Roll**
Date of	**Date of**

EXPERIMENT NO:

TITLE:Programfortravellingsalesmanproblem.

AIM: Writeaprogramfortravellingsalesmanproblemusingbranchandboundtechnique.

Hardware/SoftwareRequirement:PC,WindowsXP/7**,**Ccompiler,Editor**.**

Theory

Travellingsalesmanproblemconsistof"Givenasetofcitiesanddistancebetweeneverypair ofcities,the problem is to findthe shortest possible tourthat visits everycityexactlyonce and returns to the starting point."

Branch and bound is an algorithm design paradigm which is generally used for solving combinatorial optimization problems. These problems are typically exponential in terms of time complexityandmayrequireexploringallpossiblepermutationsinworstcase.TheBranchandBound Algorithm technique solves these problems relatively quickly.

HowtoimplementTravellingsalesman problem?

FollowingarethestepsinvolvedinTSP:

1. Createstatespacetreestartingfromrootnode.

2. Calculatecostofeachnodeandchild node.

3. Selectmostpromisingchildforfurtherexpansion(givesoptimalsolution)killremainingothernode of that expansion.

4. Repeatstep3tillyoureachedtoagainstarting node.

Result/Output:-

_____

_____

_____

ReferenceBooks:-

1. Aho,"Design&AnalysisofComputerAlgorithms",PearsonLPE.
2. RussMiller,"Algorithms:SequentialandParallel",DreamtechPress.
3. Goodrich,"AlgorithmDesign:FoundationandAnalysis",WileyIndia.
4. Grama,"AnIntrotoParallelComputing:Design&AnalysisofAlgorithms",Second Edition, Pearson LPE.
5. Baase,"ComputerAlgorithms:IntrotoDesign&Analysis",ThirdEdition,PearsonLPE

QuestionsforViva:-

1. ExplainTravellingsalesmanproblemwithexample.
2. Differentiatebetweendynamicandbranchandboundtechnique.

NameofTeacher **Sign**

Miss. Mayuri Chandratre

TRAVELINGSALESMANUSINGBRANCHANDBOUNDTECHNIQUE

```
#include<stdio.h>#include<co
nio.h>

inta[10][10],visited[10],n,cost=0;

voidget()
{
int i,j;

printf("EnterNo.ofCities:");
scanf("%d",&n);
printf("\nEnterCostMatrix:\n");
for( i=0;i<n;i++)
{
printf("\nEnterElementsofRow#:%d\n",i+1); for(
j=0;j<n;j++)
scanf("%d",&a[i][j]);
visited[i]=0;
}
printf("\n\nThecostlistis:\n\n");
for( i=0;i<n;i++)
{
printf("\n\n");
for(j=0;j<n;j++)
printf("\t%d",a[i][j]);
}
}
```

```
voidmincost(intcity)
{
int i,ncity;
visited[city]=1;
printf("%d–>",city+1);
ncity=least(city);

if(ncity==999)
{
ncity=0;
printf("%d",ncity+1);
cost+=a[city][ncity];
return;
}

mincost(ncity);
}

intleast(intc)
{
inti,nc=999;
intmin=999,kmin;
for(i=0;i<n;i++)
{
if((a[c][i]!=0)&&(visited[i]==0))
if(a[c][i]<min)
{
min=a[i][0]+a[c][i];
kmin=a[c][i];
nc=i;
}
}
if(min!=999)
cost+=kmin;
return nc;
}
```

```
voidput()
{
printf("\n\nMinimumcost:");
printf("%d",cost);
}

voidmain()
{
clrscr();
get();
printf("\n\nThePathis:\n\n");
mincost(0);
put();
getch();
}
```

Output
Input Sample:

No.ofNodes:6

Cost Matrix:

99	10	15	20	99	8
5	99	9	10	8	99
6	13	99	12	99	5
8	8	9	99	6	99
99	10	99	6	99	99
10	99	5	99	99	99

SSBT'sCollegeofEngineering&Technology,Bambhori,Jalgaon
DEPARTMENTOFINFORMATIONTECHNOLOGY

Name	**Class:T.E.**
Subject: Design and analysis of algorithm	**Roll**
Date of	**Date of**

EXPERIMENT NO:

TITLE:Programforgraphcoloringproblem.

AIM: Writeaprogramforgraphcoloringproblemusingbacktrackingtechnique.

Hardware/SoftwareRequirement:PC,WindowsXP/7**,**Ccompiler,Editor**.**

Theory

Givenanundirectedgraphand anumberm,determineifthegraph canbecoloredwithatmost mcolorssuch thatno twoadjacentverticesofthe grapharecolored withthesamecolor. Herecoloring of a graph means the assignment of colors to all vertices.

Input:
1) A 2D array graph[V][V] where V is the numberof vertices in graph and graph[V][V] is adjacency matrixrepresentationofthegraph.Avaluegraph[i][j]is1ifthereisadirectedgefromitoj,otherwise graph[i][j] is 0.
2) Anintegerm whichisthemaximum numberofcolorsthat canbeused.

Output:
Anarraycolor[V]thatshouldhavenumbersfrom1tom.color[i]shouldrepresentthecolorassignedto the ith vertex. The code should also return false if the graph cannot be colored with m colors.

BactrackingAlgorithm

Backtracking algorithmmakestheprocesstosolvetheproblemmoreefficientbyavoidingmuchbad decision that needed to be made in the naive approach. We start by coloring a single vertex**,** then we move to its adjacent vertex**.** We color it with that color which has not been used to color any of its connected vertices**.** After coloring it we then move to its adjacent vertex which is uncolored. We repeat the process until all vertices of the given graph are colored.

In case we find for a vertex that all its adjacent (connected) are colored with different colors andno color is left to make it color different from them, then it means the given number of colors i.e „m", is insufficient to color the graph. Hence we require more colors i.e bigger chromatic number.

Howtoimplementgraph coloringproblem?

FollowingarethestepsinvolvedinTSP:

1. Confirmwhetheritisvalidtocolorthevertexwithcurrentcolor?bycheckingwhetheranyofits adjacent vertices are colored with the same color?
2. Ifyesthencolorit orelsetrywithanother color.
3. Ifnoothercolorisavailablethenbacktracki.eun-colorlastcoloredvertexandreturn false.
4. Aftereachcoloringcheckifallverticesarecoloredornot.Ifyesthenendtheprogramby returning true, else continue.
5. Nowselectanyoneoftheuncoloredadjacentverticesofthecurrentcoloredvertexandrepeat the whole process

   Herebacktrackingmeanstostopfurtherrecursivecallsonadjacentverticesbyreturningfalse.In this algorithm Step-2 (Continue) and Step-4 (backtracking) is causing the program to try different color option.

Continue–tryadifferentcolorforcurrent vertex.
Backtrack–tryadifferentcolorforlastcoloredvertex.

Result/Output:-

_____

_____

_____

ReferenceBooks:-

1. Aho,"Design&AnalysisofComputerAlgorithms",PearsonLPE.
2. RussMiller,"Algorithms:SequentialandParallel",DreamtechPress.
3. Goodrich,"AlgorithmDesign:FoundationandAnalysis",WileyIndia.
4. Grama,"AnIntrotoParallelComputing:Design&AnalysisofAlgorithms",Second Edition, Pearson LPE.
5. Baase,"ComputerAlgorithms:IntrotoDesign&Analysis",ThirdEdition,PearsonLPE

QuestionsforViva:-

1. Explaingraphcoloringproblemwithexample.
2. Whatisbacktracking?

NameofTeacher Sign

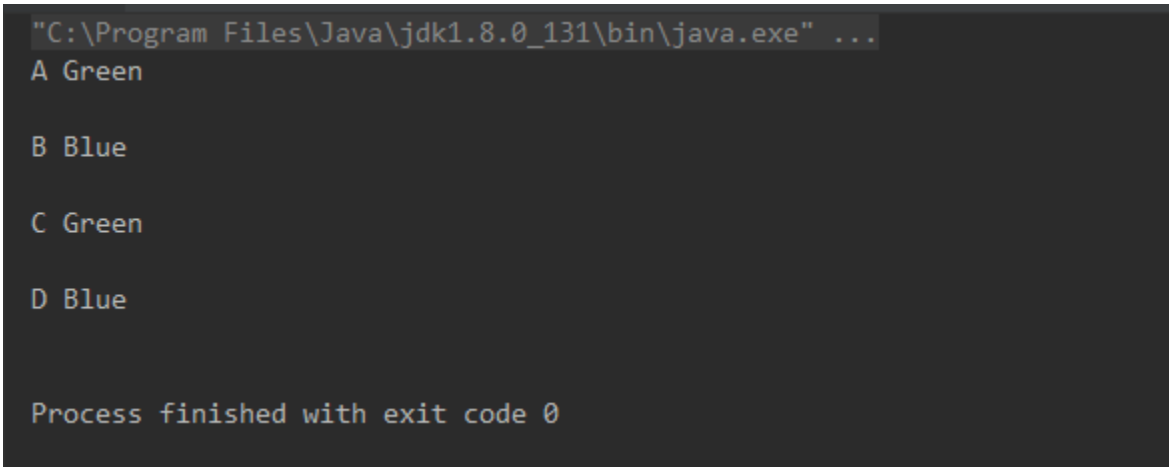
Miss. Mayuri Chandratre

Graphcoloring usingbacktracking

```
#include <stdio.h>
#include <stdlib.h>
#include<stdbool.h>

//Number of vertices
#definenumOfVertices4

//functionPrototypes
boolcanColorWith(int,int );

//0 -Green
//1 -Blue

charcolors[][30]={"Green","Blue"}; int
color_used = 2;
intcolorCount;

//Graphconnections
intgraph[numOfVertices][numOfVertices] ={{0,1,0,1},
                                        {1, 0, 1, 0},
                                        {0, 1, 0, 1},
                                        {1, 0, 1, 0}};

typedef struct{
   char name;
   boolcolored;
   int color;

} Vertex;

//VertexList
Vertex*vertexArray[numOfVertices];

int hasUncoloredNeighbours(int idx){
   int i;
   for(i=0;i<numOfVertices;i++){
       if(graph[idx][i]==1&&vertexArray[i]->colored==false) return i;
   }
   return-1;
}

boolsetColors(int idx){
```

```
    intcolorIndex, unColoredIdx;

        for(colorIndex=0;colorIndex<color_used;colorIndex++){

            //Step-1:checkingvalidity
            if(!canColorWith(colorIndex,idx))continue;//Step-2:Continue

            //Step-2: coloring
            vertexArray[idx]->color=colorIndex;
            vertexArray[idx]->colored = true;
            colorCount++;

            //Step-4 : Whether all vertices colored?
            if(colorCount==numOfVertices)//BaseCase
               returntrue;

            //Step-5:Nextuncoloredvertex
            while((unColoredIdx=hasUncoloredNeighbours(idx))!=-1){
                  if(setColors(unColoredIdx))
                     returntrue;
            }

        }

        // Step-3 : Backtracking
        vertexArray[idx]->color = -1;
        vertexArray[idx]->colored =false;
        return false;
}

//Functiontocheckwhetheritisvalidtocolorwithcolor[colorIndex] bool
canColorWith(int colorIndex, int vertex) {
    Vertex*neighborVertex;
    int i;

    for(i=0;i<numOfVertices; i++){

        //skippingifvertexarenotconnected
        if(graph[vertex][i] == 0) continue;

        neighborVertex=vertexArray[i];
        if(neighborVertex->colored&&neighborVertex->color==colorIndex)
            returnfalse;
    }

    returntrue;
```

```
}


intmain()
{
   inti;

   //CreatingVertex
   VertexvertexA,vertexB,vertexC, vertexD;

   vertexA.name='A';
   vertexB.name='B';
   vertexC.name='C';
   vertexD.name='D';

   vertexArray[0]=&vertexA;
   vertexArray[1]=&vertexB;
   vertexArray[2]=&vertexC;
   vertexArray[3]=&vertexD;

   for(i=0; i<numOfVertices;i++){
      vertexArray[i]->colored =false;
      vertexArray[i]->color = -1;
   }

   boolhasSolution=setColors(0); if

   (!hasSolution)
      printf("NoSolution");
   else {
      for(i=0;i<numOfVertices;i++){
         printf("%c%s\n",vertexArray[i]->name,colors[vertexArray[i]->color]);
      }
   }

   return0;
}
```


"C:\Program Files\Java\jdk1.8.0_131\bin\java.exe" ...
A Green
B Blue
C Green
D Blue
Process finished with exit code 0

SSBT'sCollegeofEngineering&Technology,Bambhori,Jalgaon
DEPARTMENTOFINFORMATIONTECHNOLOGY

Name	**Class:T.E.**
Subject: Design and analysis of algorithm	**Roll**
Date of	**Date of**

EXPERIMENT NO:

TITLE:Programforjob sequencing.
AIM: Writeaprogramforjobsequencingusinggreedyapproach.

Hardware/SoftwareRequirement:PC,WindowsXP/7**,**Ccompiler,Editor**.**

Theory

Givenanarrayofjobswhereeveryjobhasadeadlineandassociatedprofitifthejobisfinished before the deadline. It is also given that every job takes single unit of time, so the minimum possible deadline for any job is 1.How to maximize total profit if only one job can be scheduled at a time. A SimpleSolutionistogenerate allsubsetsofgiven setofjobsandcheckindividualsubsetforfeasibility ofjobsinthatsubset.Keeptrackofmaximumprofitamongallfeasiblesubsets.Thetimecomplexityof this solution is exponential.

GreedyAlgorithm

Amongall the algorithmicapproaches, thesimplest andstraightforwardapproach isthe Greedy method. Inthisapproach, thedecisionistakenonthebasisofcurrentavailableinformationwithout worrying about the effect of the current decision in future.

Greedy algorithms build a solution part by part, choosing the next part in such a way, that it gives an immediate benefit. This approach never reconsiders the choices taken previously. This approach is mainlyusedtosolveoptimizationproblems.Greedymethodiseasytoimplementandquiteefficientin most of the cases. Hence, we can say that Greedy algorithm is an algorithmic paradigm based on heuristicthatfollowslocaloptimalchoiceateachstepwiththehopeoffindingglobaloptimalsolution.

Inmanyproblems,itdoesnotproduceanoptimalsolutionthoughitgivesanapproximate(nearoptimal) solution in a reasonable time.

Howtoimplementjobsequencing problem?

Followingstepsareinvolvedinjobsequencingalgorithm,

1) Sortalljobsindecreasingorderofprofit.
2) Initializetheresult sequenceas firstjobinsortedjobs.
3) Dofollowingforremainingn-1 jobs
   a) Ifthecurrentjobcanfitinthecurrentresultsequencewithoutmissingthedeadline, add currentjob to the result. Else ignore the current job.

Result/Output:-

_____

_____

_____

ReferenceBooks:-

1. Aho,"Design&Analysis ofComputerAlgorithms",PearsonLPE.
2. RussMiller,"Algorithms:SequentialandParallel",DreamtechPress.
3. Goodrich,"AlgorithmDesign:FoundationandAnalysis",WileyIndia.
4. Grama,"AnIntrotoParallelComputing:Design&AnalysisofAlgorithms",Second Edition, Pearson LPE.
5. Baase,"ComputerAlgorithms:IntrotoDesign&Analysis",ThirdEdition,PearsonLPE

QuestionsforViva:-

1. Explainjobsequencingwith example.
2. Whatisgreedyalgorithm?

NameofTeacher **Sign**

Miss. Mayuri Chandratre

```
//Programtofindthemaximumprofitjobsequencefromagivenarrayofjobswithdeadlinesand profits
#include<stdio.h>#def

ine MAX 100

typedefstructJob{ char
  id[5];
  intdeadline;
  int profit;
}Job;

voidjobSequencingWithDeadline(Jobjobs[],intn);

int minValue(int x, int y) {
  if(x<y)returnx;
  returny;
}

intmain(void){
  //variables
  int i, j;

  //jobswithdeadlineandprofit
  Job jobs[5] = {
    {"j1",2,60},
    {"j2",1, 100},
    {"j3",3,20},
    {"j4",2,40},
    {"j5",1,20},
  };

  //temp
  Jobtemp;

  //numberofjobs int
  n = 5;

  //sortthejobsprofitwiseindescendingorder for(i
  = 1; i < n; i++) {
    for(j = 0; j < n - i; j++) {
      if(jobs[j+1].profit >jobs[j].profit) {
      temp = jobs[j+1];
        jobs[j+1]=jobs[j];
        jobs[j] = temp;
      }
    }
  }

  printf("%10s%10s%10s\n","Job","Deadline","Profit");
  for(i = 0; i < n; i++) {
    printf("%10s%10i%10i\n",jobs[i].id,jobs[i].deadline,jobs[i].profit);
  }

  jobSequencingWithDeadline(jobs,n);

  return 0;
}
voidjobSequencingWithDeadline(Jobjobs[],intn) {
  //variables
  int i, j, k,maxprofit;

  //freetimeslots
  inttimeslot[MAX];

  //filledtime slots
  intfilledTimeSlot=0;

  //findmaxdeadlinevalue
  int dmax = 0;
  for(i = 0; i < n; i++) {
    if(jobs[i].deadline>dmax){
    dmax = jobs[i].deadline;
    }
  }
```

```
//freetimeslotsinitiallysetto-1[-1denotesEMPTY] for(i
  = 1; i <= dmax; i++) {
    timeslot[i]=-1;
  }

  printf("dmax:%d\n",dmax);

  for(i = 1; i <= n; i++) {
    k=minValue(dmax,jobs[i-1].deadline); while(k
    >= 1) {
      if(timeslot[k]==-1){
        timeslot[k] = i-1;
        filledTimeSlot++;
        break;
      }
      k--;
    }

    //ifalltimeslotsarefilledthenstop
    if(filledTimeSlot == dmax) {
      break;
    }
  }

  //required jobs
  printf("\nRequiredJobs:"); for(i
  = 1; i <= dmax; i++) {
    printf("%s",jobs[timeslot[i]].id);

    if(i < dmax) {
      printf("-->");
    }
  }

  //requiredprofit
  maxprofit = 0;
  for(i = 1; i <= dmax; i++)
    {maxprofit+=jobs[timeslot[i]].profit;
  }
  printf("\nMaxProfit:%d\n",maxprofit);
}
```

Output

```
 JobDeadline Profit j2
                  1       100
        j1        2        60
        j4        2        40
        j3        3        20
        j5        1        20
 dmax:3

 RequiredJobs:j2--> j1 ---- >j3
MaxProfit: 180
```