**CONTENTS**

1. Vision Mission of the Institute
2. Vision Mission of the Department
3. PEOs
4. POs
5. PSOs
6. COs
7. Content beyond Syllabus. ( if any)
8. Laboratory Regulations and Safety Rules
9. Index
10. Experiments

**Vision of the Institute**

To be the fountainhead of novel ideas & innovations in science & technology & persist to be a foundation of pride for all Indians.

**Mission of the Institute**

* To provide value based broad Engineering, Technology and Science where education in students are urged to develop their professional skills.
* To inculcate dedication, hard work, sincerity, integrity and ethics in building up overall professional personality of our student and faculty.
* To inculcate a spirit of entrepreneurship and innovation in passing out students.
* To instigate sponsored research and provide consultancy services in technical, educational and industrial areas.

**Vision of the Department**

Attaining global recognition in computer science and engineering education, research and training to meet the growing needs of the industry and society.

**Mission of the Department**

Provide quality undergraduate and postgraduate education, in both the theoretical and applied foundations of computer science, and train students to effectively apply this education to solve real-world problems, thus amplifying their potential for lifelong high-quality careers.

**Program Education Objectives (PEOs)**

1. To prepare students for successful careers in software industry that meet the needs of Indian and multinational companies.
2. To develop the skills among students to analyze real world problem & implement with computer engineering solution and in multidisciplinary projects.
3. To provide students with solid foundation in mathematical, scientific and engineering fundamentals to solve engineering problems and required to pursue higher studies.
4. To develop the ability to work with the core competence of computer science & engineering i.e. software engineering, hardware structure & networking concepts so that one can find feasible solution to real world problems.

5. To inseminate in studentsprofessional and ethical attitude, effective communication skills, team work skills, multidisciplinary approach, and an ability to relate engineering issues to broader social context.

6. To motivate students perseverance for lifelong learning and to introduce them to professional ethics and codes of professional practice.

**Program Outcomes (POs)**

PO1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

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

PO3**.Design/development of solutions:** Design solutions for complex engineering problems 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.

PO4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6**. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9**. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in 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 effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. **Project management and finance:** Demonstrate knowledge and understanding of 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.

PO12**. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

**Program Specific Outcomes (PSOs)**

1. Apply Concepts in core areas of computer science -Data Structures, Database Management System, Operating System, Computer Networks and Software Engineering to solve technical issues.
2. Apply Software Engineering principal and practices to provide software solutions.
3. Ability to work in team and apply the knowledge acquired to develop new real life system and able to adapt societal need of future.

**Course Outcomes (Cos)**

CO402.1 Define the concept of algorithm, divide-conquer strategies and compute algorithms

including formulating recurrence relations (BT-1, BT-3)

CO402.2 Apply greedy algorithm for various problems. (BT-3)

CO402.3 Demonstrate dynamic programming for various problems.(BT-3)

CO402.4 Demonstrate Backtracking and Branch-Bound technique for various problems. (BT-3)

CO402.5 Describe various operations on Graph, Tree and Search technique. (BT-1)

**Content Beyond Syllabus (if any)**

**Dynamic Programming Problems:**

* **Optimal Binary Search Trees**
* **Longest Common Subsequence**

**Laboratory Regulations and Safety Rules**

1. Without Prior permission do not enter into the Laboratory.

2. While entering into the LAB students should wear their ID cards.

3. The Students should come with proper uniform.

4. Student should not use mobile phone inside the laboratory.

5. Students should sign in the LOGIN REGISTER before entering into the laboratory.

6. Students should come with observation and record note book to the laboratory.

7. Do not change any computer setting.

8. Students should maintain silence inside the laboratory.

9. After completing the laboratory exercise, make sure to SHUTDOWN the system properly.

**Enrollment No.:-**

**INDEX**

**List of Experiments**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Experiment Name** | **Date** | **Grade** | **Sign** |
| 1. | Writeaprogram forStrassen’s MatrixMultiplication. (CO1) |  |  |  |
| 2. | Writeaprogram for findingthe maximum andminimum valuefrom list. (CO1) |  |  |  |
| 3. | Writeaprogram for knapsack problem. (CO2) |  |  |  |
| 4. | Writeaprogram for Job sequencingwithdeadline.(CO2) |  |  |  |
| 5. | Writeaprogram to implement heap sort. (CO2) |  |  |  |
| 6. | Writeaprogram for minimumspanningtreesusingKruskal’s algorithm. (CO2) |  |  |  |
| 7. | Writeaprogram for minimumspanningtreesusingPrim’s algorithm. (CO2) |  |  |  |
| 8. | Writeaprogram for DFS and BFS Graph Traversal (CO5). |  |  |  |
| 9. | Writeaprogram for Single sourceshortestpath. (CO2) |  |  |  |
| 10. | Writeaprogram for All Pair Shortest Path (CO3) |  |  |  |

**Experiments Beyond Syllabus**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Experiment Name** | **Date** | **Grade** | **Sign** |
| 1. | Write Program to implement coin changeproblem usingdynamicprogramming. |  |  |  |
| 2. | Writeaprogram to find rank of string among allits permutations sorted lexicographically. |  |  |  |

**Experiment No. 1**

**Objective:**Writeaprogram forStrassen’s MatrixMultiplication. (CO1)

**Program:**

#include<stdio.h>

#include<conio.h>

int main(){

int a[2][2],b[2][2],c[2][2],i,j;

int p,q,r,s,t,u,v;

clrscr();

printf("Enter the elements of first matrix: \n");

for(i=0;i<2;i++)

for(j=0;j<2;j++)

scanf("%d",&a[i][j]);

printf("Enter the elements of second matrix: \n");

for(i=0;i<2;i++)

for(j=0;j<2;j++)

scanf("%d",&b[i][j]);

printf("\nThe first matrix is\n");

for(i=0;i<2;i++){

printf("\n");

for(j=0;j<2;j++)

printf("%d\t",a[i][j]);

}

printf("\nThe second matrix is\n");

for(i=0;i<2;i++){

printf("\n");

for(j=0;j<2;j++)

printf("%d\t",b[i][j]);

}

p= (a[0][0] + a[1][1])\*(b[0][0]+b[1][1]);

q= (a[1][0]+a[1][1])\*b[0][0];

r= a[0][0]\*(b[0][1]-b[1][1]);

s= a[1][1]\*(b[1][0]-b[0][0]);

t= (a[0][0]+a[0][1])\*b[1][1];

u= (a[1][0]-a[0][0])\*(b[0][0]+b[0][1]);

v= (a[0][1]-a[1][1])\*(b[1][0]+b[1][1]);

c[0][0]=p+s-t+v;

c[0][1]=r+t;

c[1][0]=q+s;

c[1][1]=p-q+r+u;

printf("\nAfter multiplication using Strassen's Matrix Multiplication\n");

for(i=0;i<2;i++){

printf("\n");

for(j=0;j<2;j++)

printf("%d\t",c[i][j]);

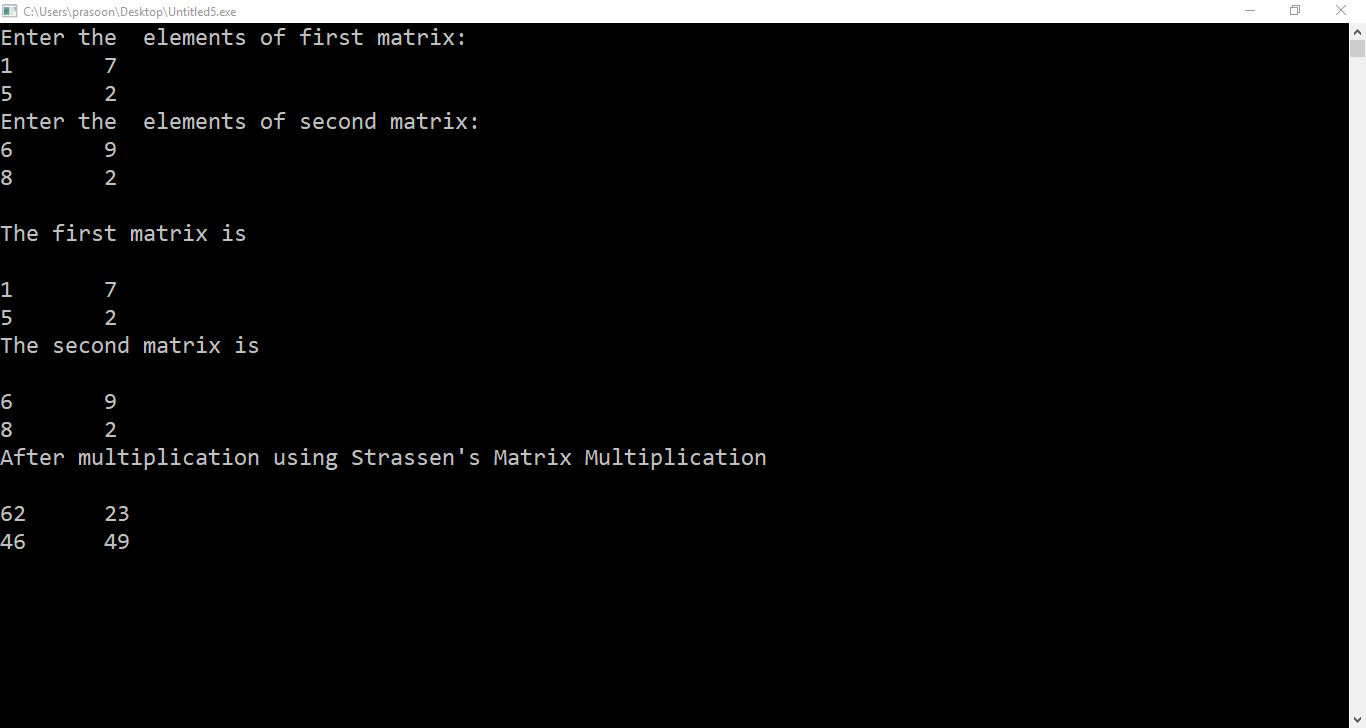
}

getch();

return 0;

}

**Output**



**Viva Questions**

**1. What is thetime complexity ofthe Strassen’sMatrixMultiplication?(BT-1)**

**Ans**. O (N2.81)

**2. Can you give recurrencerelationequation oftheStrassen’sMatrixMultiplication?**

**(BT-2)**

**Ans**.T(N)=7T(N/2)+O(N2)

**Experiment No. 2**

**Objective:**Writeaprogram forfindingthe maximum andminimum valuefrom list. (CO1)

**Program:**

#include<stdio**.**h>

#include<conio**.**h>

int maxmin**(**int,int,int**\***,int **\*)**;

int num**[**20**]**;

main**()**

{

int max,min;

int n,i;

**//**clrscr**()**;

printf**("**enter number of elements in the array\n**")**;

scanf**("%**d**"**,&n**)**;

printf**("**enter the numbers\n**")**;

for**(**i**=**0;i<n;i**++)**

{

scanf**("%**d**"**,&num**[**i**])**;

}

maxmin**(**0,n**-**1,&max,&min**)**;

printf**("**the maximum element is**- %**d\n**"**,max**)**;

printf**("**the minimum element is**- %**d\n**"**,min**)**;

getch**()**;

}

int maxmin**(**int i,intj,int**\***max,int**\***min**)**

{ int max1,min1;

if**(**i**==**j**)**

**\***max**=\***min**=**num**[**i**]**;

else if**(**i**==(**j**-**1**))**

{

if**(**num**[**i**]**<num**[**j**])**

{

**\***max**=**num**[**j**]**;

**\***min**=**num**[**i**]**;

}

else if**(**num**[**i**]**>num**[**j**])**

{

**\***max**=**num**[**i**]**;

**\***min**=**num**[**j**]**;

}

}

else

{

int mid**=(**i**+**j**)/**2;

maxmin**(**i,mid,max,min**)**;

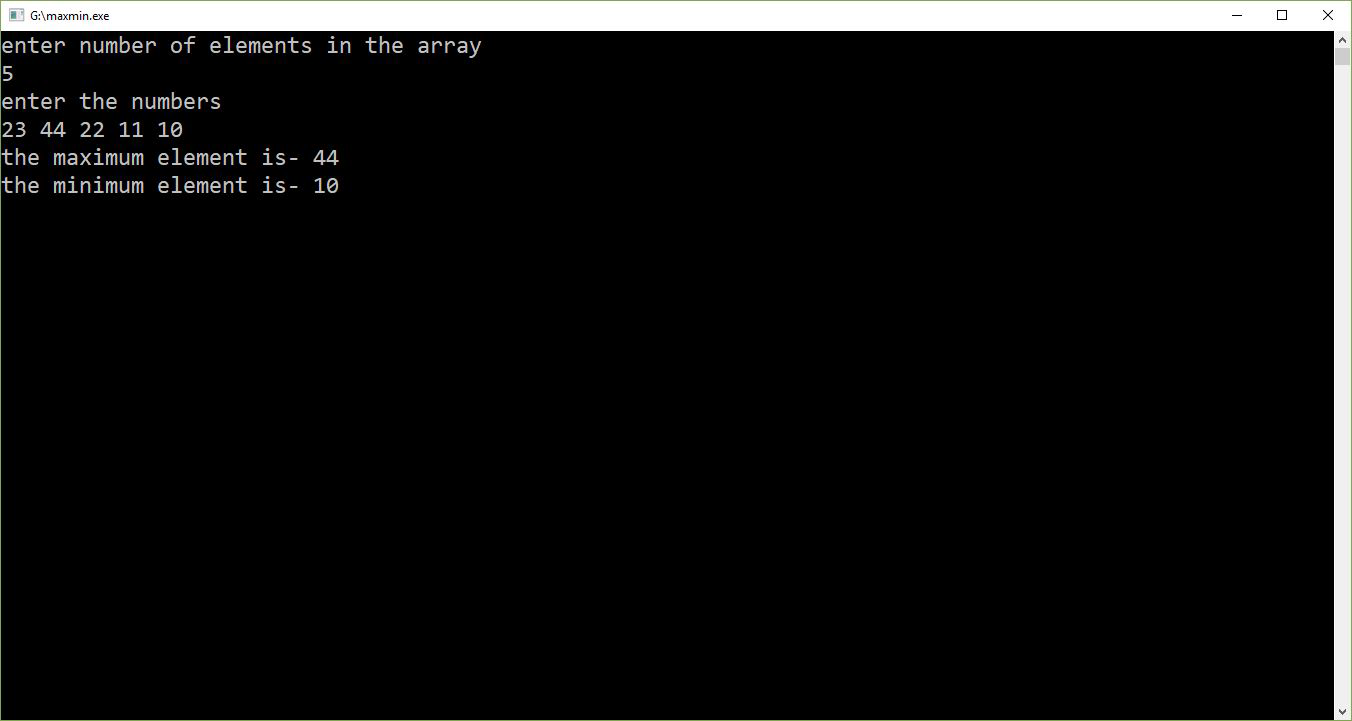
maxmin**(**mid**+**1,j,&max1,&min1**)**;

if**(\***max<max1**) \***max**=**max1;

if**(\***min>min1**) \***min**=**min1;

}}

**Output**



**Viva Questions**

**1. What is the total no. of comparisons done?(BT-1)**

**Ans**. When n is even it does n/2 + 3n/2 + 2 comparisons, when n is odd (n-1)/2 + 3(n-1)/2 +2.

**2. Which algorithm technique is followed by the given problem?(BT-2)**

**Ans**. Divide and conquer.

**Experiment No. 3**

**Objective:**Writeaprogram forknapsack problem(CO2).

**Program:**

#include<stdio.h>

#include<conio.h>

Void main()

{

clrscr();

int n,m,i,u,j;

int p[20],w[20],s[20];

float x[20],pw[20];

float optimal=0.0;

for(i=1;i<=n;i++)

s[i]=i;

printf("Enter number of objects:");

scanf("%d",&n);

printf("Enter capacity of KnapSack:");

scanf("%d",&m);

printf("Enter profits of items:");

for(i=1;i<=n;i++)

scanf("%d",&p[i]);

printf("Enter Weights of items:");

for(i=1;i<=n;i++)

scanf("%d",&w[i]);

for(i=1;i<=n;i++)

pw[i]=(float)p[i]/w[i];

for(i=n;i>1;i--)

{

for(j=1;j<=i-1;j++)

{

if(pw[j]<pw[j+1])

{

float value=pw[j+1];int value1=p[j+1]; int value2=w[j+1];

pw[j+1]=pw[j]; p[j+1]=p[j]; w[j+1]=w[j];

pw[j]=value; p[j]=value1; w[j]=value2;

int value3=s[j+1];

s[j+1]=s[j];

s[j]=value3;

}

}

}

for(i=1;i<=n;i++)

x[i]=0.0;

u=m;

for(i=1;i<=m;i++)

{

if(w[i]>u)

break;

else

x[i]=1.0;

u=u-w[i];

}

if(i<=n)

x[i]=(float)u/w[i];

for(i=1;i<=n;i++)

optimal=optimal+p[i]\*x[i];

printf("\nOptimal Solution is %f\n",optimal);

for(i=n;i>1;i--)

{

for(j=1;j<=i-1;j++)

{

if(s[j]>s[j+1])

{

int value=s[j+1]; float value1=x[j+1];

s[j+1]=s[j]; x[j+1]=x[j];

s[j]=value; x[j]=value1;

}

}

}

printf("The x values are\n");

for(i=1;i<=n;i++)

{

printf("x%d:\t",i);

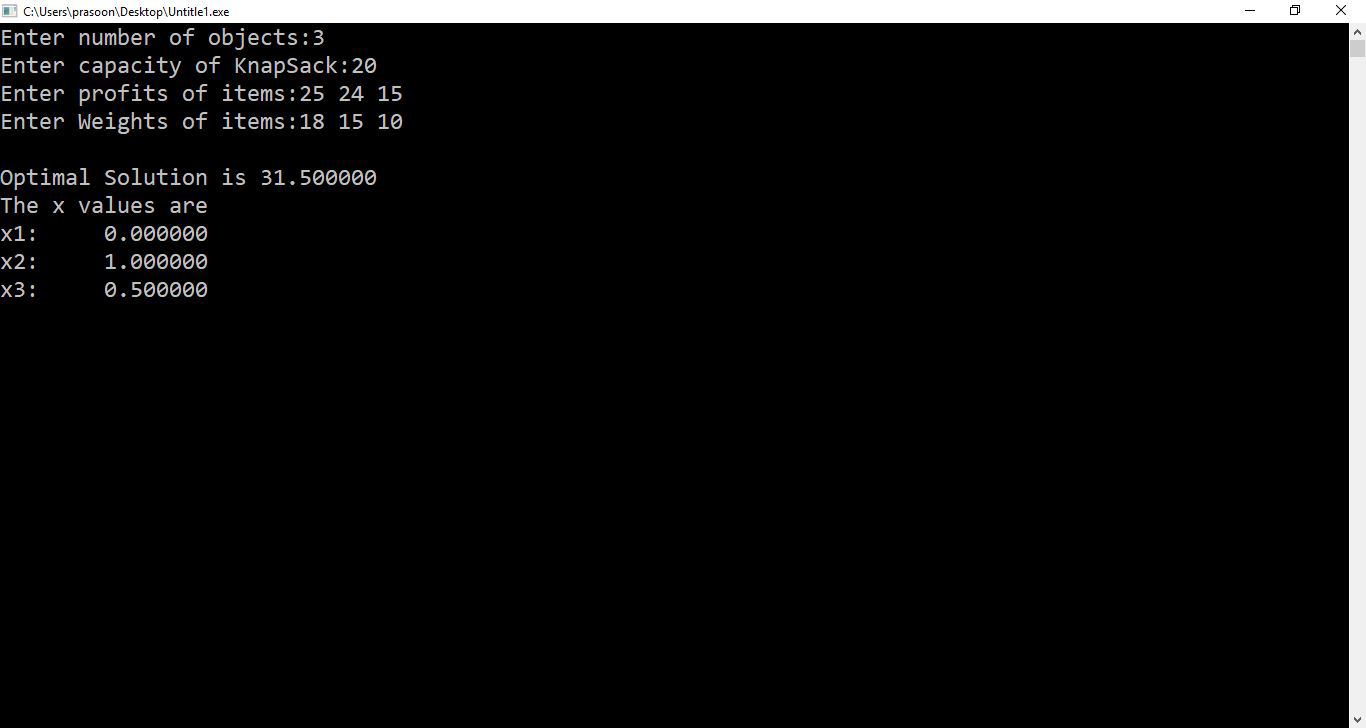
printf("%f\t\n",x[i]);

}

getch();

}

**Output**



**Viva Questions**

**1. What is thedifference between fractional and 0-1 knapsack problem?(BT-1)**

**Ans**. In, fractional knapsack we can break items for maximizing the total value of knapsack, whereas in 0-1 knapsack problem, we are not allowed to break items. We either take the whole item or don’t take it.

**2. What is the time complexity of knapsack problem?(BT-1)**

**Ans**. O(nlogn).

**Experiment No. 4**

**Objective:**Writeaprogram forJob sequencingwithdeadline(CO2).

**Program:**

#include<stdio.h>

#include<conio.h>

int main()

{

int d[20],p[20],n,s[20];

int i,q,j;

clrscr();

for(i=0;i<n;i++)

s[i]=i;

printf("Enter the no. of jobs:");

scanf("%d",&n);

printf("Enter the profits of jobs:");

for(i=0;i<n;i++)

scanf("%d",&p[i]);

printf("Enter the deadline of jobs:");

for(i=0;i<n;i++)

scanf("%d",&d[i]);

for(i=n;i>1;i--)

{

for(j=0;j<i-1;j++)

{

if(p[j]<p[j+1])

{

int value=p[j+1];

p[j+1]=p[j];

p[j]=value;

int value1=d[j+1];

d[j+1]=d[j];

d[j]=value1;

int value2=s[j+1];

s[j+1]=s[j];

s[j]=value2;

}

}

}

int max=d[0];

for(i=1;i<n;i++)

{

if(d[i]>max)

max=d[i];

}

job(p,d,n,max,s);

getch();

return 0;

}

void job(int profit[],int dead[],int n,int maxdead,int s[])

{ int result[20],total=0,x=0,i,j,id[20],k=0;

for( i=0;i<maxdead;i++)

{result[i]=0;

}

for( i=0;i<n;i++)

{ if(result[dead[i]-1]==0)

{result[dead[i]-1]=profit[i];

total=total+profit[i];

id[dead[i]-1]=s[i];

}

else

{ for(j=dead[i]-1;j>=0;j--)

{if(result[j]==0)

{result[j]=profit[i];

total=total+profit[i];

id[j]=s[i];

}

}

}

}

printf("Total profit is %d",total);

printf("\nJob sequence is : \n");

printf("Job\tprofit\n");

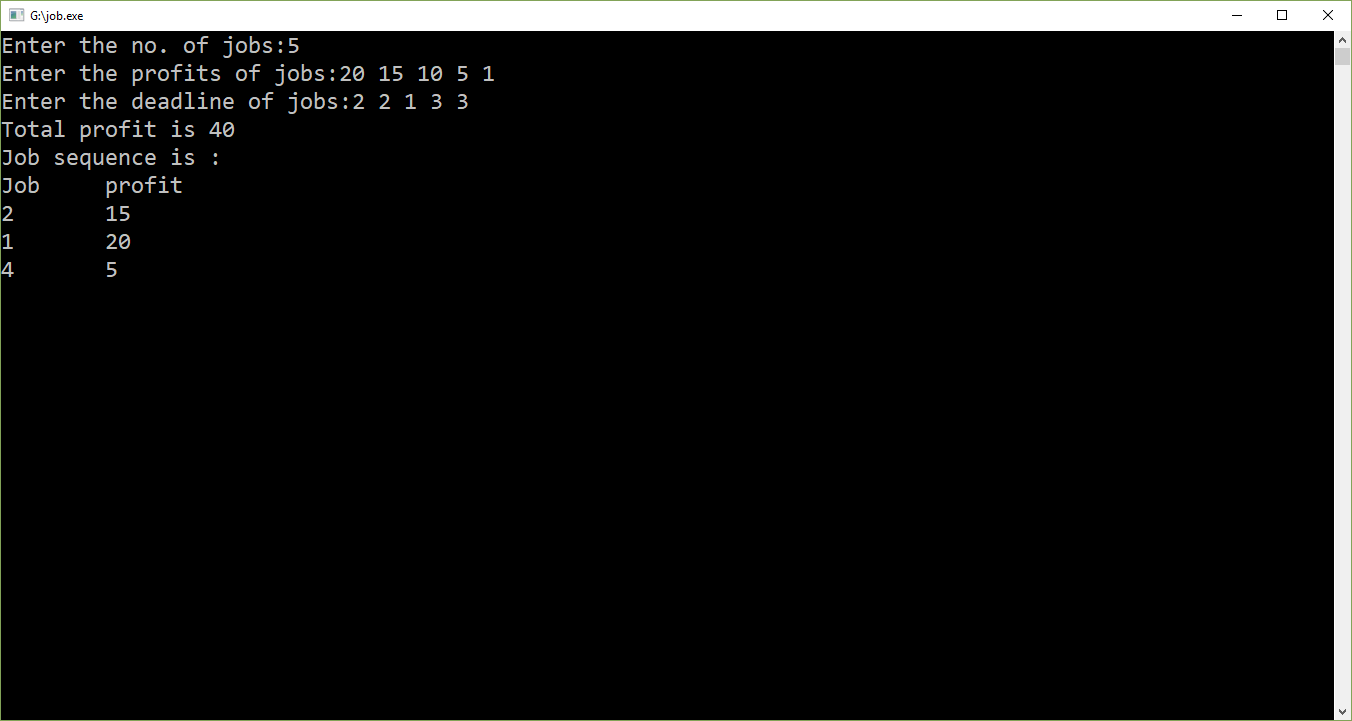
for(i=0;i<maxdead;i++)

{printf("%d\t%d\n",id[i]+1,result[i]);

}

}

**Output**



**Viva Questions**

**1. Will you state job scheduling problem. (BT-2)**

Ans:- Job scheduling is the process of allocating system resources to many different tasks by an operating system (OS).

**2. What can you say about the complexity of job sequence with dead line.(BT-2)**

Ans:- Worst case computing time of job sequence problem is O(N2).

**Experiment No. 5**

**Objective:**Writeaprogram to implement heap sort(CO1).

**Program:**

#include <stdio.h>

#include <conio.h>

void max\_heapify(int \*a, int i, int n)

{

int j, temp;

temp = a[i];

j = 2\*i;

while (j <= n)

{

if (j < n && a[j+1] > a[j])

j = j+1;

if (temp > a[j])

break;

else if (temp <= a[j])

{

a[j/2] = a[j];

j = 2\*j;

}

}

a[j/2] = temp;

return;

}

void heapsort(int \*a, int n)

{

int i, temp;

for (i= n; i>= 2; i--)

{

temp = a[i];

a[i] = a[1];

a[1] = temp;

max\_heapify(a, 1, i- 1);

}

}

void build\_maxheap(int \*a, int n)

{

int i;

for(i= n/2; i>= 1; i--)

{

max\_heapify(a, i, n);

}

}

void main()

{

clrscr();

int n, i, x,a[20];

printf("enter no of elements of array\n");

scanf("%d",&n);

for (i= 1; i<= n; i++)

{

printf("enter element %d\n",i);

scanf("%d",&a[i]);

}

build\_maxheap(a,n);

heapsort(a, n);

printf("sorted output\n");

for (i = 1; i<= n; i++)

{

printf("%d\n",a[i]);

}

getch();

}

**Output**



**Viva Questions**

**1. Whatisheapsortindatastructure?(BT-1)**

**Ans.**Thebinary**heapdata structure is**anarraythatcanbe viewedas acompletebinary tree.Eachnodeof thebinarytreecorrespondstoanelement of thearray. Thearrayis completelyfilledonall levelsexceptpossibly lowest.

**2. Canheapsortin place?(BT-3)**

**Ans.**Youcanbuild the**heap**ontheverysamearray youwanna**sort**, and afterthatyouapply the**heapsort**algorithm, so it**sorts**inplace. Incomputerscience,anin-**place**algorithm (or in Latin insitu)isanalgorithmwhich transforms inputusingadata structurewith asmall,constant amount ofextrastoragespace.



**Experiment No. 6**

**Objective:**Writeaprogram forminimumspanningtreesusingKruskal’s algorithm(CO2).

**Program:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

int i,j,k,a,b,u,v,n,ne=1;

int min,mincost=0,cost[9][9],parent[9];

int find(int);

int uni(int,int);

void main()

{

clrscr();

printf("\n\tImplementation of Kruskal's algorithm\n");

printf("\nEnter the no. of vertices:");

scanf("%d",&n);

printf("\nEnter the cost adjacency matrix:\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;

}

}

printf("The edges of Minimum Cost Spanning Tree are\n");

while(ne < n)

{

for(i=1,min=999;i<=n;i++)

{

for(j=1;j <= n;j++)

{

if(cost[i][j] < min)

{

min=cost[i][j];

a=u=i;

b=v=j;

}

}

}

u=find(u);

v=find(v);

if(uni(u,v))

{

printf("%d edge (%d,%d) =%d\n",ne++,a,b,min);

mincost +=min;

}

cost[a][b]=cost[b][a]=999;

}

printf("\n\tMinimum cost = %d\n",mincost);

getch();

}

int find(int i)

{

while(parent[i])

i=parent[i];

return i;

}

int uni(int i,int j)

{

if(i!=j)

{

parent[j]=i;

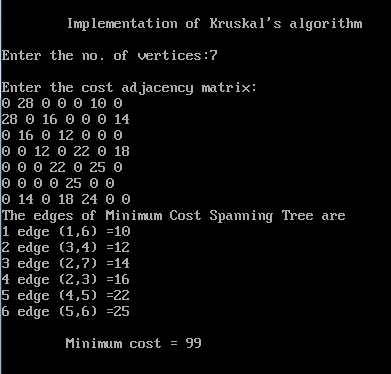
return 1;

}

return 0;

}

**Output**

**Viva Questions**

1. **Can you explain Kruskal algorithm can be solve on the basis of . (BT-2)**

Ans. Cost.

1. **What is time complexity of Kruskal Algorithm? (BT-2)**

Ans. the total running time of Kruskal’s algorithm is O (ElogV)

**Experiment No. 7**

**Objective:** Writeaprogram forminimumspanningtreesusingPrim’s algorithm(CO2).

**Program:**

#include<stdio.h>

#include<conio.h>

int a,b,u,v,n,i,j,ne=1;

int visited[10]={0},min,mincost=0,cost[10][10];

void main()

{

clrscr();

printf("\nEnter the number of nodes:");

scanf("%d",&n);

printf("\nEnter the adjacency matrix:\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("\n Edge %d:(%d %d) cost:%d",ne++,a,b,min);

mincost+=min;

visited[b]=1;

}

cost[a][b]=cost[b][a]=999;

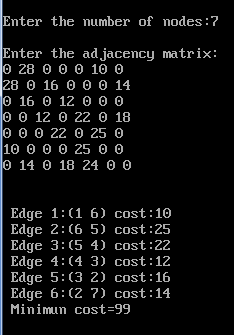
}

printf("\n Minimun cost=%d",mincost);

getch();

}

**Output**



**Viva Questions**

* 1. **Can you explain Prims algorithm can be solve on the basis of . (BT-2)**

Ans. Cost.

* 1. **What is time complexity ofPrims Algorithm?(BT-2)**

Ans. the total running time of Prims algorithm is O (ElogV)

**Experiment No. 8**

**Objective:** Writeaprogram forDFS and BFS Graph Traversal(CO5).

**Program: (BFS)**

#include<stdio.h>

#include<conio.h>

#define max 10

int n,adj[max][max],visited[max];

void bfs();

void readmatrix();

main()

{

int source;

//clrscr();

printf("Enter the source node:");

scanf("%d",&source);

readmatrix();

bfs(source);

getch();

}

void readmatrix()

{

int i,j;

printf("Enter number of nodes:");

scanf("%d",&n);

printf("Enter adjacency matrix\n");

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

scanf("%d",&adj[i][j]);

for(i=1;i<=n;i++)

visited[i]=0;

}

void bfs(int source)

{

int queue[max];

int i,front,rear,root;

printf("The Order is\n");

front=rear=0;

visited[source]=1;

queue[rear++]=source;

printf("%d ",source);

while(front!=rear)

{

root=queue[front];

for(i=1;i<=n;i++)

if(adj[root][i] &&! visited[i])

{

visited[i]=1;

queue[rear++]=i;

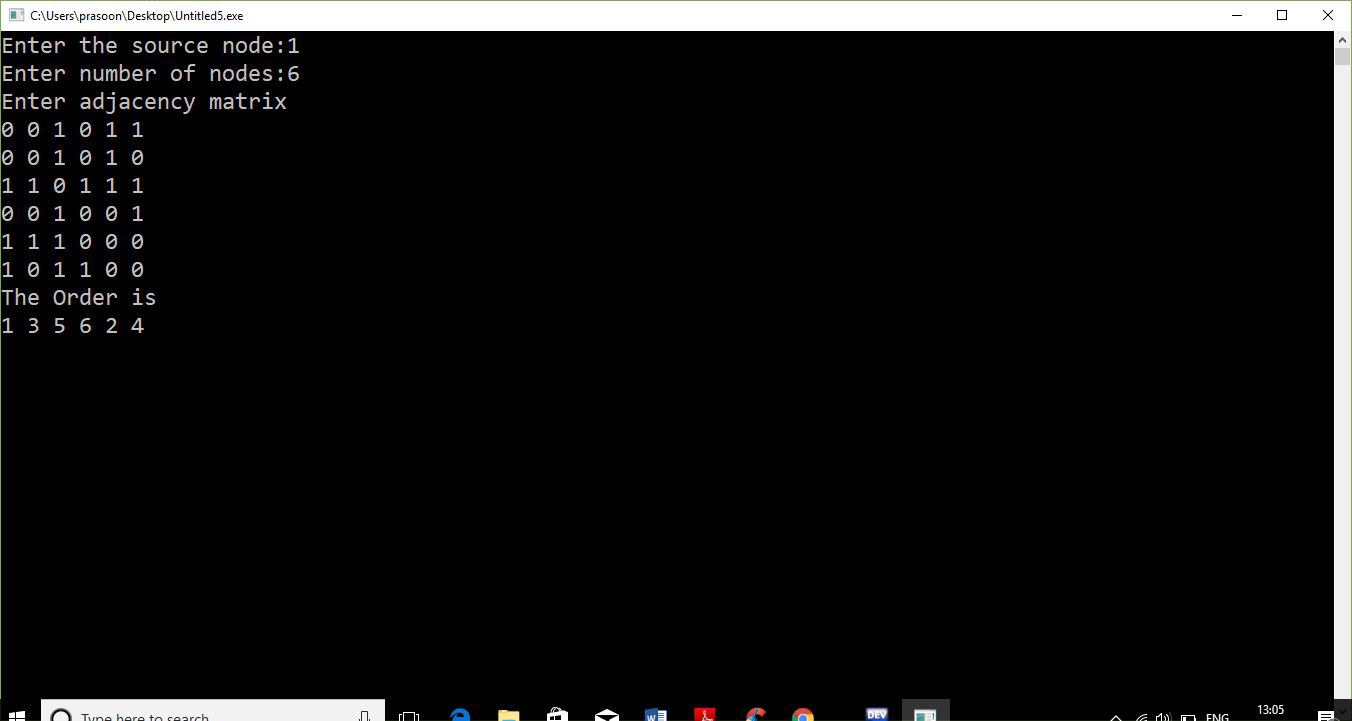
printf("%d ",i);

}

front++;

}}

**Output**



**Program: (DFS)**

#include<stdio.h>

#include<conio.h>

int i,j,k,n;

int graph[20][20];

int visited[20];

void dfs(int);

void main()

{

clrscr();

printf("Enter number of nodes:");

scanf("%d",&n);

printf("Enter the adjacency matrix\n");

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

scanf("%d",&graph[i][j]);

}

visited[i]=0;

}

for(i=1;i<=n;i++)

{

if(visited[i]==0)

{

dfs(i);

}

}

getch();

}

void dfs(int l)

{

visited[l]=1;

printf("Node visited is %d\n",l);

for(k=1;k<=n;k++)

{

if(visited[k]==0 && graph[l][k]==1)

{

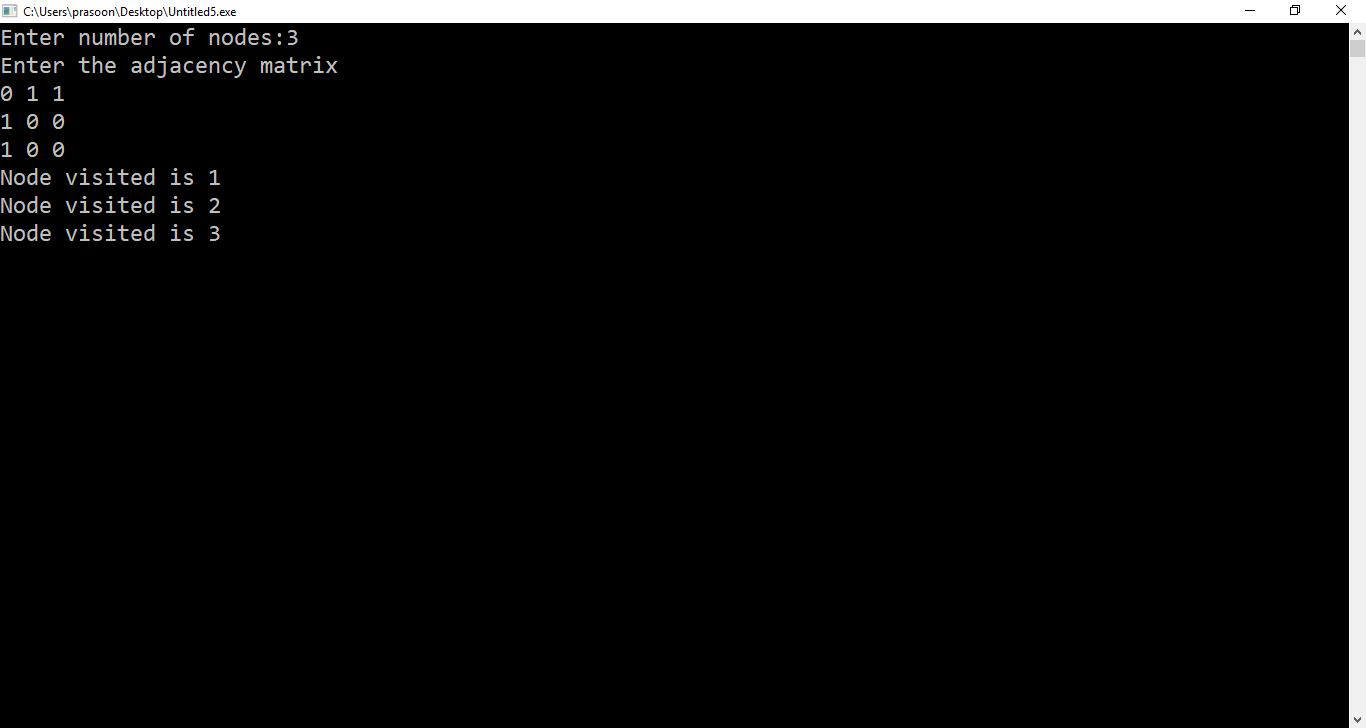
dfs(k);

}

}

}

**Output**



**Viva Questions**

* 1. **DFS is equivalent to which of the traversal traversal in the binary trees?(BT-1)**

**Ans**.It is equivalent to the pre-order traversal of a binary tree.

**2. In DFS, how many times a node is visited?(BT-2)**

**Ans**.Equivalent to the no. of indegree of the order.

**Experiment No. 9**

**Objective:** Writeaprogram forSingle sourceshortestpath. (CO2)

**Program:**

#include<stdio.h>

#include<conio.h>

#define infinity 999

void dij(int n,intv,int cost[10][10],int dist[100])

{

int i,u,count,w,flag[10],min;

for(i=1;i<=n;i++)

flag[i]=0,dist[i]=cost[v][i];

count=2;

while(count<=n)

{

min=99;

for(w=1;w<=n;w++)

if(dist[w]<min &&!flag[w])

min=dist[w],u=w;

flag[u]=1;

count++;

for(w=1;w<=n;w++)

if((dist[u]+cost[u][w]<dist[w]) &&!flag[w])

dist[w]=dist[u]+cost[u][w];

}

}

void main()

{

int n,v,i,j,cost[10][10],dist[10];

clrscr();

printf("\n Enter the number of nodes:");

scanf("%d",&n);

printf("\n Enter the cost matrix:\n(for infinity use 999)");

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]=infinity;

}

printf("\n Enter the source matrix:");

scanf("%d",&v);

dij(n,v,cost,dist);

printf("\n Shortest path:\n");

for(i=1;i<=n;i++)

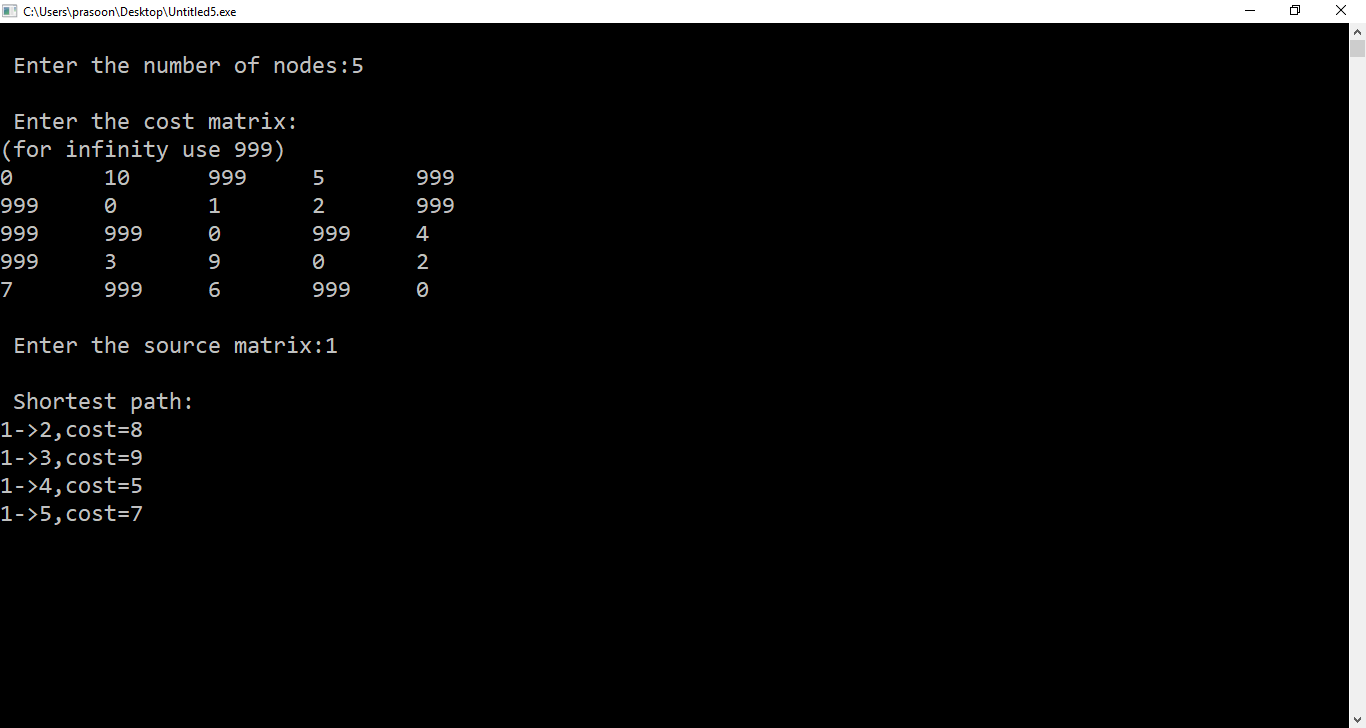
if(i!=v)

printf("%d->%d,cost=%d\n",v,i,dist[i]);

getch();

}

**Output**



**Viva Questions**

**1. How would you implement Dijkstra’s shortest path algorithm on unweighted graphs so that it runs in linear time, which data structure is used.(BT-2)**

Ans. A Queue because we can find single source shortest path in unweighted graph by using Breadth first search (BFS) algorithm which using "Queue" data structure , which time O(m+n) (i.e. linear with respect to the number of vertices and edges.

**2.  How would you explain In an undirected graph with positive edge weights. Dijkstra's shortest path algorithm can be implemented using the binary heap data structure with time complexity?(BT-2)**

Ans. O ((| E | + | V |) log | V |)

**Experiment No. 10**

**Objective:**Writeaprogram forAll Pair Shortest Path(CO3).

**Program:**

#include<stdlib.h>

#include<stdio.h>

#include<conio.h>

int c[100][100], p[100][100]; //c-cost matrix, p-path matrix(to store the path)

int inf=1000, v; //Assume Infinity as 1000

//int min(int x, int y);

void show();

void path(int, int);

int main()

{

int i, j, k, x;

clrscr();

printf("Enter the number of vertices in the graph: ");

scanf("%d", &v);

for(i=1;i<=v;i++)

for(j=1;j<=v;j++)

if(i==j)

c[i][j]=0;

else

{

printf("Is %d connected to %d?",i,j);

printf("If yes, enter weight, else enter %d: ",inf);

scanf("%d", &c[i][j]);

}

printf("\nEnter adjacency matrix:\n");

printf("(Enter 1000 if there is no path)\n");

for(i=1;i<=v;i++)

for(j=1;j<=v;j++)

{

scanf("%d", &c[i][j]);

p[i][j]=-1;

}

printf("\n");

for(k=1;k<=v;k++)

{

for(i=1;i<=v;i++)

{

for(j=1;j<=v;j++)

{

if(i==j)

c[i][j]=0;

else

{

x=c[i][k]+c[k][j];

if(c[i][j]>x)

{

c[i][j]=x;

p[i][j]=k;

}

}

}

}

show();

printf("\n");

}

printf("From\tTo\tPath\t\tTotal Min. Cost\n");

for(i=1;i<=v;i++)

{

for(j=1;j<=v;j++)

{

if(i!=j)

{

printf("%d\t", i);

printf("%d\t", j);

// printf("Path from %d to %d is: ",i,j);

printf("%d", i);

path(i,j);

printf("%d", j);

printf("\t\t%d", c[i][j]);

printf("\n");

}

}

}

getch();

return 0;

}

void show()

{

int i,j;

for(i=1;i<=v;i++)

{

for(j=1;j<=v;j++)

if(c[i][j]==1000)

printf("INF\t");

else

printf("%d\t", c[i][j]);

printf("\n");

}

}

void path(int i, int j)

{

int k;

k=p[i][j];

if(k==-1)

{

printf("->");

return;

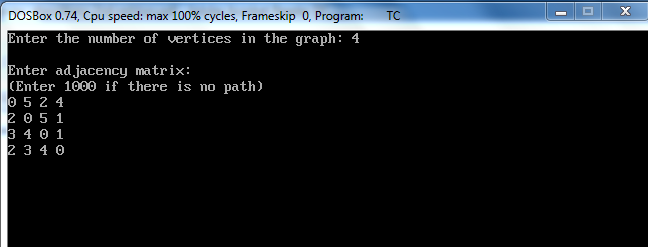
} path(i, k);

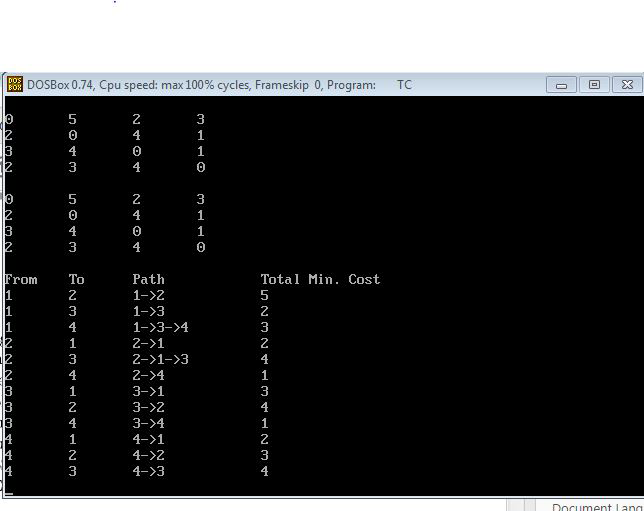
printf("%d",k);

path(k,j);

}

**Output**

****

****

**Viva Questions**

**1. How would you explain Elements of dynamic programming.(BT-2)**

Ans.the two key ingredients that an optimization problem must have in order for dynamic programming to apply: optimal substructure and overlapping subproblems.

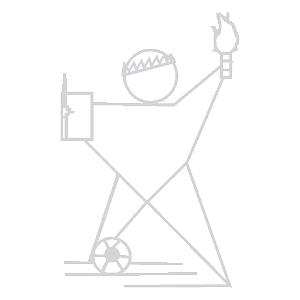
**2.  How would you explain time complexity of All pair shortest path?(BT-2)**

Ans. O ( V3)

**Experiment Beyond Experiment**

**Programtoimplementcoinchangeproblemusingdynamic programming**

#include<stdio.h>



#include<stdio.h>

int count( int S[], int m, int n )

{

int i, j, x, y;

int table[10][10];

for (i=0; i<m; i++)

table[0][i] = 1;

for (i = 1; i< n+1; i++)

{

for (j = 0; j < m; j++)

{

x = (i-S[j] >= 0)? table[i - S[j]][j]: 0;

y = (j >= 1)? table[i][j-1]: 0;

table[i][j] = x + y;

}

}

return table[n][m-1];

}

void main()

{

int arr[] = {1, 2, 3};

int m = sizeof(arr)/sizeof(arr[0]);

int n = 4;

clrscr();

printf(" \n\t total number of ways we can make change is %d ", count(arr, m, n)); getch();

}

**OUTPUT:**

Total number ofways wecan makechangeis 4

**VIVAQUESTIONS**

**1. What is coinchangeproblem?**

Ans.GivenavalueN,ifwewanttomakechangeforNcents,andwehaveinfinitesupplyof eachofS={S1,S2,..,Sm}valuedcoins,howmany wayscanwemakethechange?Theorder of coins doesn’t matter.

Forexample,forN=4andS={1,2,3},therearefoursolutions:{1,1,1,1},{1,1,2},{2,2},{1,3}. So outputshould be4.

**2. What is the complexityofcoinchangeproblem?**

Ans. Time Complexityof coin changeproblem isO(mn)

**Experiment Beyond Experiment**

**Writeaprogramtofindrankofstringamongallitspermutations sortedlexicographically**

#include <stdio.h>

#include <string.h>

int fact(int n)

{

return (n <= 1)? 1 :n \* fact(n-1);

}

int findSmallerInRight(char\* str, int low, int high)

{

int countRight = 0, i;

for (i = low+1; i<= high; ++i)

if (str[i] < str[low])

++countRight;

return countRight;

}

int findRank (char\* str)

{

int len = strlen(str);

int mul = fact(len);

int rank = 1;

int countRight;

int i;

for (i = 0; i<len; ++i)

{

mul /= len - i;

countRight = findSmallerInRight(str, i, len-1);

rank += countRight \* mul ;

}

return rank;

}

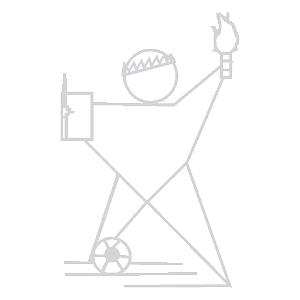
void main()

{

char str[] = "acb";

clrscr();

printf ("\n\t rank of string is %d", findRank(str)); getch();



}

**OUTPUT:**

rank ofstringis 2

**VIVAQUESTIONS**

**1. What is Lexicographic rankofa string?**

Ans) Lexicographicrank of a string is a rank of string among all its permutations sorted lexicographically.Forexample, rank of“abc”is 1, rank of“acb”is 2, and rank of “cba”is 6.

**2. What is the complexityof algorithmforfindinglexicographic rankofa string?**

Ans)Time Complexityis O(n^2)