# Q1. Implement Binary Search using Divide and Conquer approach.

**Ans.**

#include<stdio.h>

#include<conio.h>

int bin\_src(int num,int left,int right); int a[100],n;

void main()

{

int i,num,ans;

printf("Enter number of elements in array\n\n"); scanf("%d",&n);

printf("Enter %d numbers\n\n",n); for(i=0;i<n;i++) scanf("%d",&a[i]);

printf("Enter number to search = "); scanf("%d",&num); ans=bin\_src(num,0,n-1);

if(ans)

printf("%d is at position = %d",num,ans); else

printf("%d is not present in array",num,ans); getch();

}

int bin\_src(int num,int left,int right)

{

int mid; while(left<=right)

{

mid=(left+right)/2;

if(a[mid]==num) return mid+1; if(num<a[mid]) right=mid-1; else left=mid+1;

}

return 0;

}

# Output:-

Enter number of elements in array 6

Enter 6 numbers

9 12 24 31 49 57

Enter number to search = 49

49 is at position = 5

**Q2. Implement Merge Sort using Divide and Conquer approach.**

**Ans.**

#include<stdio.h>

#include<conio.h>

void split(int start,int end);

void merge(int start,int mid,int end);

int a[100],b[100];

void main()

{

int i,n;

printf("Enter number of elements in array\n\n"); scanf("%d",&n);

printf("Enter %d numbers\n\n",n); for(i=0;i<n;i++) scanf("%d",&a[i]);

split(0,n-1);

printf("\n\n\nArray after merge sort\n\n\t"); for(i=0;i<n;i++)

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

}

void split(int start,int end)

{

int mid=(start+end)/2; if(start<end)

{

split(start,mid); split(mid+1,end); merge(start,mid,end);

}

else

return;

}

void merge(int start,int mid,int end)

{

int i,j,k; for(i=start,j=mid+1,k=start;i<=mid&&j<=end;k++)

{

if(a[i]<a[j])

b[k]=a[i++]; else

b[k]=a[j++];

}

while(i<=mid)

b[k++]=a[i++];

while(j<=end)

b[k++]=a[j++];

for(i=start;i<=end;i++) a[i]=b[i];

}

# Output:-

Enter number of elements in array 6

Enter 6 numbers

15 7 22 28 2 13

Array after merge sort

2 7 13 15 22 28

**Q2. Implement Quick Sort using Divide and Conquer approach.**

**Ans.**

#include<stdio.h>

#include<conio.h>

void qsort(int left,int right);

int partition(int left,int right,int pivot);

int a[100];

void main()

{

int i,n;

printf("Enter number of elements in array\n\n"); scanf("%d",&n);

printf("Enter %d numbers\n\n",n); for(i=0;i<n;i++) scanf("%d",&a[i]);

qsort(0,n-1);

printf("\n\n\nArray after quick sort\n\n\t"); for(i=0;i<n;i++)

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

}

void qsort(int left,int right)

{

int pivot=a[right],new\_pivot; if(left<right)

{

new\_pivot=partition(left,right,pivot); qsort(left,new\_pivot-1); qsort(new\_pivot+1,right);

}

else

return;

}

int partition(int left,int right,int pivot)

{

int left\_ptr=left,right\_ptr=right-1,temp; while(1)

{

while(a[left\_ptr]<pivot)

left\_ptr++; while(a[right\_ptr]>pivot) right\_ptr--; if(left\_ptr<right\_ptr)

{

temp=a[left\_ptr]; a[left\_ptr]=a[right\_ptr]; a[right\_ptr]=temp;

}

else break;

}

temp=a[left\_ptr]; a[left\_ptr]=a[right]; a[right]=temp;

return left\_ptr;

}

# Output:-

Enter number of elements in array 5

Enter 5 numbers

36 75 13 9 96

Array after quick sort

9 13 36 75 96

**Q4. Find minimum and maximum in an Array using Divide and Conquer approach.**

**Ans.**

#include<stdio.h>

#include<conio.h>

#include<limits.h>

void min\_max(int start,int end);

int a[100],min,max; void main()

{

int i,n;

printf("Enter number of elements in array\n\n"); scanf("%d",&n);

printf("Enter %d numbers\n\n",n); for(i=0;i<n;i++) scanf("%d",&a[i]);

min\_max(0,n-1);

printf("\n\nMaximum element = %d\nMinimum element = %d",max,min); getch();

}

void min\_max(int start,int end)

{

if(start==end)

{

min=a[start]; max=a[start];

}

else if(start==end-1)

{

if(a[start]<a[end])

{

min=a[start]; max=a[end];

}

else

{

min=a[end]; max=a[start];

}

}

else

{

int mid=(start+end)/2; min\_max(start,mid); int temp\_max=max;

int temp\_min=min; min\_max(mid+1,end); if(temp\_max>max) max=temp\_max; if(temp\_min<min) min=temp\_min;

}

}

# Output:-

Enter number of elements in array 6

Enter 6 numbers

21 14 77 46 1 12

Maximum element = 77 Minimum element = 1

**Q5. Print Fibonacci Series using Dynamic Programming approach.**

**Ans.**

#include<stdio.h>

#include<conio.h>

void fibo(int \*a,int n); void main()

{

int n,arr[100],i;

printf("Enter the term number = "); scanf("%d",&n);

fibo(arr,n);

printf("\nFibonacci series upto %d term\n\n",n); for(i=0;i<n;i++)

printf("%d ",arr[i]); getch();

}

void fibo(int \*a,int n)

{

int i; a[0]=0;

a[1]=1;

for(i=2;i<n;i++) a[i]=a[i-1]+a[i-2];

}

# Output:-

Enter the term number = 10 Fibonacci series upto 10 term

0 1 1 2 3 5 8 13 21 34

|  |  |  |
| --- | --- | --- |
| **Q6. Find minimum number of** | **scalar** | **multiplication** |
| **needed for chain of matrix** | **(Using** | **Dynamic** |
| **Programming approach)** |  | |
| **Ans.** |
| #include<stdio.h>  #include<conio.h>  #include<limits.h>  int chain\_matrix(int n,int arr[]); |

void main()

{

int n,i,ans;

printf("Enter number of matrix\n\n"); scanf("%d",&n);

int arr[n+1];

printf("Enter the orders of %d matrix (%d numbers)\n\n",n,n+1); for(i=0;i<=n;i++)

scanf("%d",&arr[i]); ans=chain\_matrix(n,arr); printf("\nMinimum cost = %d",ans); getch();

}

int chain\_matrix(int n,int arr[])

{

int i,j,k,m[n+1][n+1],temp,x; for(i=1;i<=n;i++)

m[i][i]=0; for(x=2;x<=n;x++)

{

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

{

j=i+x-1; m[i][j]=INT\_MAX;

for(k=i;k<j;k++)

{

temp=m[i][k]+m[k+1][j]+arr[i-1]\*arr[j]\*arr[k]; if(temp<m[i][j])

m[i][j]=temp;

}

}

}

return m[1][n];

}

# Output:-

Enter number of matrix 4

Enter the orders of 4 matrix (5 numbers) 30 35 15 5 10

Minimum cost = 9375

**Q7. Implment Knapsack Problem using Dynamic Programming.**

**Ans.**

#include<stdio.h>

#include<conio.h>

int knapsack(int w[],int v[],int n,int x); void main()

{

int x,n,i,ans;

printf("Enter number of weights = "); scanf("%d",&n);

printf("\nEnter bag capacity = "); scanf("%d",&x);

int w[n+1],v[n+1];

printf("\nEnter %d weights\n\n",n); for(i=1;i<=n;i++) scanf("%d",&w[i]);

printf("\nEnter values of %d weights\n\n",n); for(i=1;i<=n;i++)

scanf("%d",&v[i]); ans=knapsack(w,v,n,x); printf("\nMaximum value = %d",ans); getch();

}

int knapsack(int w[],int v[],int n,int x)

{

int m[n+1][x+1],i,j; for(i=0;i<=n;i++) m[i][0]=0;

for(i=0;i<=x;i++) m[0][i]=0;

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

{

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

if(w[i]>j || m[i-1][j]>m[i-1][j-w[i]]+v[i]) m[i][j]=m[i-1][j];

else

m[i][j]=m[i-1][j-w[i]]+v[i];

}

return m[n][x];

}

# Output:-

Enter number of weights = 4 Enter bag capacity = 5 Enter 4 weights

2 1 3 2

Enter values of 4 weights 12 10 20 15

Maximum value = 37

**Q8. Implement All Pair Shortest Path for a graph (Floyd – Warshall Algorithm) using Dynamic Programming approach.**

**Ans.**

#include<stdio.h>

#include<conio.h>

#include<limits.h> void apsp();

int min(int a,int b);

int graph[100][100],n,e,d[100][100][100];

void main()

{

int x,y,i,j,start,w;

printf("Enter number of vertices and edges = "); scanf("%d %d",&n,&e);

int visited[100]={0},r[n+1]; for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++) graph[i][j]=(i==j?0:INT\_MAX);

}

printf("\nEnter start and end vertices and weight of %d edges---

>\n\n",e);

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

{

scanf("%d %d %d",&x,&y,&w);

graph[x][y]=graph[y][x]=w;

}

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

{

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

{

printf("\nShortest path between %d and %d vertices =

%d",i,j,d[i][j][n]);

}

}

getch();

}

void apsp()

{

int i,j,k; for(i=1;i<=n;i++)

{

for(j=i;j<=n;j++) d[i][j][0]=d[j][i][0]=graph[i][j];

}

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

{

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

{

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

{

if(d[i][k][k-1]==INT\_MAX || d[j][k][k-1]==INT\_MAX)

d[i][j][k]=d[i][j][k-1]; else

d[i][j][k]=min(d[i][j][k-1],d[i][k][k-1]+d[k][j][k-1]);

}

}

}

}

int min(int a,int b)

{

return (a<b?a:b);

}

# Output:-

Enter number of vertices and edges = 4 6

Enter start and end vertices and weight of 6 edges--->

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 2 |
| 1 | 3 | 1 |
| 1 | 4 | 4 |
| 2 | 3 | 9 |
| 2 | 4 | 3 |
| 3 | 4 | 2 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Shortest | path | between | 1 | and | 1 | vertices | = | 0 |
| Shortest | path | between | 1 | and | 2 | vertices | = | 2 |
| Shortest | path | between | 1 | and | 3 | vertices | = | 1 |
| Shortest | path | between | 1 | and | 4 | vertices | = | 3 |
| Shortest | path | between | 2 | and | 1 | vertices | = | 2 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Shortest | path | between | 2 | and | 2 | vertices | = | 0 |
| Shortest | path | between | 2 | and | 3 | vertices | = | 3 |
| Shortest | path | between | 2 | and | 4 | vertices | = | 3 |
| Shortest | path | between | 3 | and | 1 | vertices | = | 1 |
| Shortest | path | between | 3 | and | 2 | vertices | = | 3 |
| Shortest | path | between | 3 | and | 3 | vertices | = | 0 |
| Shortest | path | between | 3 | and | 4 | vertices | = | 2 |
| Shortest | path | between | 4 | and | 1 | vertices | = | 3 |
| Shortest | path | between | 4 | and | 2 | vertices | = | 3 |
| Shortest | path | between | 4 | and | 3 | vertices | = | 2 |
| Shortest | path | between | 4 | and | 4 | vertices | = | 0 |

**Q9. Implement Travelling Salesman Problem using Dynamic Programming approach.**

**Ans.**

#include<stdio.h>

#include<conio.h>

#include<limits.h> int least(int c);

void mincost(int city);

int a[100][100],visited[100],n,cost=0;

void main()

{

int i,j;

printf("Enter No. of Cities: "); scanf("%d",&n);

printf("\nEnter Cost Matrix\n"); for(i=0;i < n;i++)

{

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

scanf("%d",&a[i][j]); visited[i]=0;

}

printf("\n\nThe Path is:\n\n"); mincost(0);

printf("\n\nMinimum cost = %d",cost); getch();

}

void mincost(int city)

{

int i,ncity; visited[city]=1; printf("%d -->",city+1); ncity=least(city); if(ncity==INT\_MAX)

{

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

}

mincost(ncity);

}

int least(int c)

{

int i,nc=INT\_MAX;

int min=INT\_MAX,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!=INT\_MAX)

cost+=kmin; return nc;

}

# Output:-

Enter No. of Cities: 4 Enter Cost Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 5 | 4 | 2 |
| 2 | 1 | 5 | 4 |
| 9 | 6 | 2 | 4 |
| 7 | 5 | 3 | 4 |

The Path is:

1 -->4 -->3 -->2 -->1

Minimum cost = 13

**Q10. Implement Single Source Shortest Path (Bellman – Ford Algorithm) using Dynamic Programming approach.**

**Ans.**

#include<stdio.h>

#include<conio.h>

#include<limits.h>

int shortest\_path(int c[100][100],int n,int s,int m); int v[100][100],d[100][100];

void main()

{

int n,i,j,v1,v2,w,p,c[100][100]={0},s,k,vtx,prev,path[100][100]; printf("Enter number of edge and vertex = ");

scanf("%d %d",&n,&vtx); printf("\nEnter source\n\n"); scanf("%d",&s);

printf("\nEnter vertex of each edge and their weight---\n\n"); for(i=1;i<=n;i++)

{

scanf("%d %d %d",&v1,&v2,&w); c[v1][v2]=c[v2][v1]=w;

}

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

{

for(j=i;j<=vtx;j++)

{

if(!c[i][j])

c[i][j]=c[j][i]=INT\_MAX;

}

}

p=shortest\_path(c,n,s,vtx); for(j=1;j<=vtx;j++)

{

if(j!=s)

{

%d",s,j,v[p][j]);

printf("\n\n\nLength of shortest path between %d and %d =

prev=j; for(i=p;i>=1;i--)

{

d[i][prev]!=s)

if(d[i][prev]!=prev && d[i][prev]!=INT\_MAX &&

{

path[j][i]=d[i][prev]; prev=d[i][prev];

}

else

{

}

}

i--;

break;

path[j][i+1]=s; printf("\nShortest path is = "); for(k=i+1;k<=p;k++)

printf("%d ---> ",path[j][k]); printf("%d",j);

}

}

getch();

}

int shortest\_path(int c[100][100],int n,int s,int m)

{

int t,i,j,flag=1,temp; for(i=1;i<=m;i++)

{

if(i==s)

{

v[0][i]=0;

d[0][i]=s;

v[i][i]=0;

}

else

{

}

v[0][i]=INT\_MAX;

d[0][i]=INT\_MAX;

v[1][i]=c[s][i]; if(c[s][i]==INT\_MAX) d[1][i]=INT\_MAX;

else d[1][i]=s;

v[i][s]=0;

d[i][s]=s;

}

for(t=2;t<=n && flag;t++)

{

flag=0; for(i=1;i<=m;i++)

{

if(i==s) continue; v[t][i]=INT\_MAX;

d[t][i]=d[t-1][i]; for(j=1;j<=m;j++)

{

if(j==i || c[j][i]==INT\_MAX || v[t-1][j]==INT\_MAX) continue;

temp=v[t-1][j]+c[j][i]; if(temp<v[t][i])

{

v[t][i]=temp;

d[t][i]=j;

}

}

if(v[t][i]!=v[t-1][i]) flag=1;

}

}

return t-2;

}

# Output:-

Enter number of edge and vertex = 8 5 Enter source

1

Enter vertex of each edge and their weight---

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 2 |
| 1 | 5 | 4 |
| 2 | 5 | 3 |
| 2 | 4 | 9 |
| 5 | 4 | 2 |
| 2 | 3 | 7 |
| 3 | 4 | 3 |
| 1 | 4 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Length of shortest | path between 1 and | 2 | = | 2 |
| Shortest path is = | 1 ---> 2 |  |  |  |
| Length of shortest | path between 1 and | 3 | = | 4 |
| Shortest path is = | 1 ---> 4 ---> 3 |  |  |  |
| Length of shortest | path between 1 and | 4 | = | 1 |
| Shortest path is = | 1 ---> 4 |  |  |  |
| Length of shortest | path between 1 and | 5 | = | 3 |
| Shortest path is = | 1 ---> 4 ---> 5 |  |  |  |

**Q11. Implement 15 puzzle problem using brunch and bound.**

**Ans.**

#include<stdio.h>

#include<conio.h>

int m=0,n=4;

int cal(int temp[10][10],int t[10][10])

{

int i,j,m=0; for(i=0;i < n;i++)

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

{

}

return m;

}

if(temp[i][j]!=t[i][j]) m++;

int check(int a[10][10],int t[10][10])

{

int i,j,f=1; for(i=0;i < n;i++)

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

if(a[i][j]!=t[i][j])

f=0;

return f;

}

void main()

{

int p,i,j,n=4,a[10][10],t[10][10],temp[10][10],r[10][10];

int m=0,x=0,y=0,d=1000,dmin=0,l=0;

printf("\nEnter the matrix to be solved,space with zero :\n"); for(i=0;i < n;i++)

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

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

printf("\nEnter the target matrix,space with zero :\n"); for(i=0;i < n;i++)

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

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

while(!(check(a,t)))

{

l++; d=1000;

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

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

{

if(a[i][j]==0)

{

x=i; y=j;

}

}

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

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

temp[i][j]=a[i][j];

if(x!=0)

{

p=temp[x][y]; temp[x][y]=temp[x-1][y]; temp[x-1][y]=p;

}

m=cal(temp,t); dmin=l+m; if(dmin < d)

{

d=dmin;

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

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

r[i][j]=temp[i][j];

}

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

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

temp[i][j]=a[i][j];

if(x!=n-1)

{

p=temp[x][y]; temp[x][y]=temp[x+1][y]; temp[x+1][y]=p;

}

m=cal(temp,t); dmin=l+m; if(dmin < d)

{

d=dmin;

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

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

r[i][j]=temp[i][j];

}

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

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

temp[i][j]=a[i][j];

if(y!=n-1)

{

p=temp[x][y]; temp[x][y]=temp[x][y+1]; temp[x][y+1]=p;

}

m=cal(temp,t); dmin=l+m; if(dmin < d)

{

d=dmin;

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

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

r[i][j]=temp[i][j];

}

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

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

temp[i][j]=a[i][j];

if(y!=0)

{

p=temp[x][y]; temp[x][y]=temp[x][y-1]; temp[x][y-1]=p;

}

m=cal(temp,t); dmin=l+m; if(dmin < d)

{

d=dmin;

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

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

r[i][j]=temp[i][j];

}

printf("\nCalculated Intermediate Matrix Value :\n"); for(i=0;i < n;i++)

{

for(j=0;j < n;j++) printf("%d\t",r[i][j]);

printf("\n");

}

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

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

{

}

getch();

}

a[i][j]=r[i][j]; temp[i][j]=0;

}

# Output:-

Enter the matrix to be solved,space with zero : 1 2 3 4

5 6 0 8

9 10 7 11

|  |  |
| --- | --- |
| 13 14  Enter | 15 12  the target matrix,space with zero : |
| 1 2 3 | 4 |
| 5 6 7 | 8 |

9 10 11 12

13 14 15 0

Calculated Intermediate Matrix Value :

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 0 | 11 |
| 13 | 14 | 15 | 12 |

Calculated Intermediate Matrix Value :

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 0 |
| 13 | 14 | 15 | 12 |

Calculated Intermediate Matrix Value :

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 0 |

**Q12. Implement 8 Queens problem using Backtracking.**

**Ans.**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<windows.h>

void eight\_queens(int q[],int r); void show(int q[]);

void main()

{

int q[9]; eight\_queens(q,1); getch();

}

void eight\_queens(int q[],int r)

{

if(r==9) show(q); else

{

int i,j,legal; for(j=1;j<=8;j++)

{

legal=1; for(i=1;i<=r-1;i++)

{

if(q[i]==j || fabs(q[i]-j)==fabs(r-i)) legal=0;

}

if(legal)

{

q[r]=j; eight\_queens(q,r+1);

}

}

}

}

void show(int q[])

{

int i,j;

printf("\nQueens position---->\n\n"); for(i=1;i<=8;i++)

printf("%d ",q[i]); printf("\nSolution is ---->\n\n"); for(i=1;i<=8;i++)

{

for(j=1;j<=8;j++)

{

if(q[j]==i)

printf("Q "); else

printf("- ");

}

printf("\n\n");

}

exit(1);

}

# Output:-

Queens position---->

1 5 8 6 3 7 2 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Solution  Q - | is  - | ---->  - | - | - | - | - |
| - - | - | - | - | - | Q | - |
| - - | - | - | Q | - | - | - |
| - - | - | - | - | - | - | Q |
| - Q | - | - | - | - | - | - |
| - - | - | Q | - | - | - | - |
| - - | - | - | - | Q | - | - |
| - - | Q | - | - | - | - | - |

**Q13. Implement Graph Colouring problem using backtracking.**

**Ans.**

#include<stdio.h>

#include<malloc.h> int V;

void printSolution(int color[]);

int isSafe (int v, int graph[V][V], int color[], int c)

{

int i;

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

if (graph[v][i] && c == color[i]) return 0;

return 1;

}

int graphColoringUtil(int graph[V][V], int m, int color[], int v)

{

int c;

if (v == V) return 1;

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

{

if (isSafe(v, graph, color, c))

{

color[v] = c;

if (graphColoringUtil (graph, m, color, v+1) == 1) return 1;

color[v] = 0;

}

}

return 0;

}

int graphColoring(int graph[V][V], int m)

{

int i,\*color = (int \*)malloc(sizeof(int)\*V); for (i = 0; i < V; i++)

color[i] = 0;

if (graphColoringUtil(graph, m, color, 0) == 0)

{

printf("Solution does not exist"); return 0;

}

printSolution(color); return 1;

}

void printSolution(int color[])

{

int i;

printf("Solution Exists:"

" Following are the assigned colors \n"); for (i = 0; i < V; i++)

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

}

int main()

{

int i,j,m;

printf("Enter number of vertices = "); scanf("%d",&V);

int graph[V][V];

printf("Enter the adjacency matrix of the graph\n\n"); for(i=0;i<V;i++)

{

for(j=0;j<V;j++) scanf("%d",&graph[i][j]);

}

printf("Enter number of colour = "); scanf("%d",&m);

graphColoring (graph, m); return 0;

}

# Output:-

Enter number of vertices = 4

Enter the adjacency matrix of the graph

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |

Enter number of colour = 3

Solution Exists: Following are the assigned colors 1 2 3 2

**Q14. Implement Knapsack Problem using Greedy method.**

**Ans.**

#include<stdio.h>

#include<conio.h>

float knapsack(int w[],int v[],int n,int max\_limit); void sort(float \*p,int \*v,int \*w,int n);

void swap(float \*a,float \*b);

int min(int a,int b); float f[100]={0}; void main()

{

int x,n,i; float ans;

printf("Enter number of weights = "); scanf("%d",&n);

printf("\nEnter bag capacity = "); scanf("%d",&x);

int w[n+1],v[n+1];

printf("\nEnter %d weights\n\n",n); for(i=1;i<=n;i++) scanf("%d",&w[i]);

printf("\nEnter values of %d weights\n\n",n); for(i=1;i<=n;i++)

scanf("%d",&v[i]); ans=knapsack(w,v,n,x); printf("\nMaximum value = %f",ans);

printf("\nArray of weight fraction\n\n\t"); for(i=1;i<=n;i++)

printf("%0.2f ",f[i]); getch();

}

float knapsack(int w[],int v[],int n,int max\_limit)

{

int i,current=0,a; float p[n],val=0; for(i=1;i<=n;i++) p[i]=(float)v[i]/w[i]; sort(p,v,w,n);

for(i=1;i<=n&&current<max\_limit;i++)

{

a=min(w[i],max\_limit-current); f[i]=(a==w[i]?1:(float)(max\_limit-current)/w[i]); current+=a;

val+=f[i]\*v[i];

}

return val;

}

void sort(float \*p,int \*v,int \*w,int n)

{

int i,j; for(i=1;i<n;i++)

{

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

{

if(p[j]>p[i])

{

swap(&p[i],&p[j]);

swap(&v[i],&v[j]);

swap(&w[i],&w[j]);

}

}

}

}

void swap(float \*a,float \*b)

{

float temp=\*a;

\*a=\*b;

\*b=temp;

}

int min(int a,int b)

{

return (a<b?a:b);

}

# Output:-

Enter number of weights = 7 Enter bag capacity = 15 Enter 7 weights

2 3 5 7 1 4 1

Enter values of 7 weights 10 5 15 7 6 18 3

Maximum value = 55.333332 Array of weight fraction

1.00 1.00 1.00 1.00 1.00 0.67 0.00

**Q16. Implement DFS and BFS (Using Graph Traversal Algorithm).**

**Ans.**

**DFS :-**

#include<stdio.h>

#include<conio.h>

#include<limits.h>

void dfs(int start,int \*visited,int \*r,int n); int graph[100][100]={0},k=0;

void main()

{

int n,x,y,i,j,e,start;

printf("Enter number of vertices and edges = "); scanf("%d %d",&n,&e);

int visited[100]={0},r[n+1];

printf("\nEnter start and end vertices of %d edges--->\n\n",e); for(i=1;i<=e;i++)

{

scanf("%d %d",&x,&y);

graph[x][y]=graph[y][x]=1;

}

printf("\nEnter the start vertex = "); scanf("%d",&start); dfs(start,visited,r,n);

printf("\nDFS traversal sequence --->\n\n"); for(i=1;i<=n;i++)

printf("%d ",r[i]); getch();

}

void dfs(int start,int \*visited,int \*r,int n)

{

r[++k]=start; visited[start]=1; int i; for(i=1;i<=n;i++)

{

if(graph[i][start] && !visited[i]) dfs(i,visited,r,n);

}

}

# Output:-

Enter number of vertices and edges = 8 10

Enter start and end vertices of 10 edges--->

1 2

1 6

2 6

2 7

2 3

6 5

3 4

3 5

5 4

3 8

Enter the start vertex = 1 DFS traversal sequence --->

1 2 3 4 5 6 8 7

**BFS :-**

#include<stdio.h>

#include<conio.h>

#include<limits.h>

void bfs(int start,int \*visited,int \*r,int n); void enqueue(int u);

int queue\_is\_empty(); int dequeue();

int graph[100][100]={0},k=0,queue[100],front=-1,rear=-1; void main()

{

int n,x,y,i,j,e,start;

printf("Enter number of vertices and edges = "); scanf("%d %d",&n,&e);

int visited[100]={0},r[n+1];

printf("\nEnter start and end vertices of %d edges--->\n\n",e);

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

{

scanf("%d %d",&x,&y);

graph[x][y]=graph[y][x]=1;

}

printf("\nEnter the start vertex = "); scanf("%d",&start); bfs(start,visited,r,n);

printf("\nBFS traversal sequence --->\n\n"); for(i=1;i<=n;i++)

printf("%d ",r[i]); getch();

}

void bfs(int start,int \*visited,int \*r,int n)

{

enqueue(start); visited[start]=1; int i,item;

while(!queue\_is\_empty())

{

item=dequeue(); r[++k]=item; for(i=1;i<=n;i++)

{

if(!visited[i] && graph[item][i])

{

enqueue(i); visited[i]=1;

}

}

}

}

void enqueue(int u)

{

if(front==-1 && rear==-1) front=0;

queue[++rear]=u;

}

int queue\_is\_empty()

{

return (front==-1 && rear==-1)?1:0;

}

int dequeue()

{

int item=queue[front]; if(front==rear) front=rear=-1;

else front++; return item;

}

# Output:-

Enter number of vertices and edges = 9 10

Enter start and end vertices of 10 edges--->

1 2

2 3

1 4

1 5

2 5

3 6

4 7

5 7

7 8

8 9

Enter the start vertex = 1 BFS traversal sequence ---> 1 2 4 5 3 7 6 8 9