**Prim’s Algorithm**

**Program:**

#include<stdio.h>

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

int input(int);

int display(int);

int prims(int);

int input(int num)

{

int i, j;

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

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

{

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

{

printf("value of cost[%d][%d] : ",i,j);

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

}

}

return 0;

}

int display(int num)

{

int i, j;

printf("\nThe cost of adjacency matrix\n\n");

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

{

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

{

printf("%d", cost[i][j]);

printf("\t");

}

printf("\n");

}

return 0;

}

int prims(int num)

{

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

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

{

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

{

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

cost[i][j]=999;

}

}

visited[1]=1;

while(ne < num)

{

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

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

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

if(visited[i]!=0)

{

min=cost[i][j];

a=u=i;

b=v=j;

}

printf("\n Edge %d:(%d - %d) cost:%d",ne++,a,b,min);

mincost=mincost+min;

visited[b]=1;

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

}

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

}

int main()

{

int num;

printf("\n\t\t\tPrim's Algorithm");

printf("\n\nEnter the number of nodes= ");

scanf("%d", &num);

input(num);

display(num);

prims(num);

return 0;

}

**Program: Floyd’s**

**Program:**

#include<stdio.h>

#define INF 999

int min(int a,int b)

{ return(a<b)?a:b;

}

void floyd(int p[][10],int n)

{ int i,j,k;

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

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

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

p[i][j]=min(p[i][j],p[i][k]+p[k][j]);

}

void main()

{ int a[10][10],n,i,j;

printf("\nEnter the n value:");

scanf("%d",&n);

printf("\nEnter the graph data:\n");

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

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

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

floyd(a,n);

printf("\nShortest path matrix\n");

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

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

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

printf("\n");

}

}

**Program: n-Queens using Backtracking**

#include<stdio.h>

#include<stdlib.h>

#define MAX 50

int can\_place(int c[],int r)

{ int i;

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

if(c[i]==c[r] || (abs(c[i]-c[r])==abs(i-r)))

return 0;

return 1;

}

void display(int c[],int n)

{ int i,j;

char cb[10][10];

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

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

cb[i][j]='-';

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

cb[i][c[i]]='Q';

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

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

printf("%c",cb[i][j]);

printf("\n");

}

}

void n\_queens(int n)

{

int r;

int c[MAX];

c[0]=-1;

r=0;

while(r>=0)

{ c[r]++;

while(c[r]<n && !can\_place(c,r))

c[r]++;

if(c[r]<n)

{ if(r==n-1)

{ display(c,n);

printf("\n\n");

}

else

{ r++;

c[r]=-1;

}

}

else

r--;

}

}

void main()

{ int n;

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

scanf("%d",&n);

n\_queens(n);

}

**Program: Heapsort using Transform and Conquer**

#include <stdio.h>

void heapify(int arr[], int n, int i)

{

int temp, maximum, left\_index, right\_index;

maximum = i;

right\_index = 2 \* i + 2;

left\_index = 2 \* i + 1;

if (left\_index < n && arr[left\_index] > arr[maximum])

maximum = left\_index;

if (right\_index < n && arr[right\_index] > arr[maximum])

maximum = right\_index;

if (maximum != i) {

temp = arr[i];

arr[i] = arr[maximum];

arr[maximum] = temp;

heapify(arr, n, maximum);

}

}

void heapsort(int arr[], int n)

{

int i, temp;

for (i = n / 2 - 1; i >= 0; i--) {

heapify(arr, n, i);

}

for (i = n - 1; i > 0; i--) {

temp = arr[0];

arr[0] = arr[i];

arr[i] = temp;

heapify(arr, i, 0);

}

}

int main()

{

int arr[] = { 20, 18, 5, 15, 3, 2 };

int n = 6;

printf("Original Array : ");

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

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

}

printf("\n");

heapsort(arr, n);

printf("Array after performing heap sort: ");

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

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

}

return 0;

}