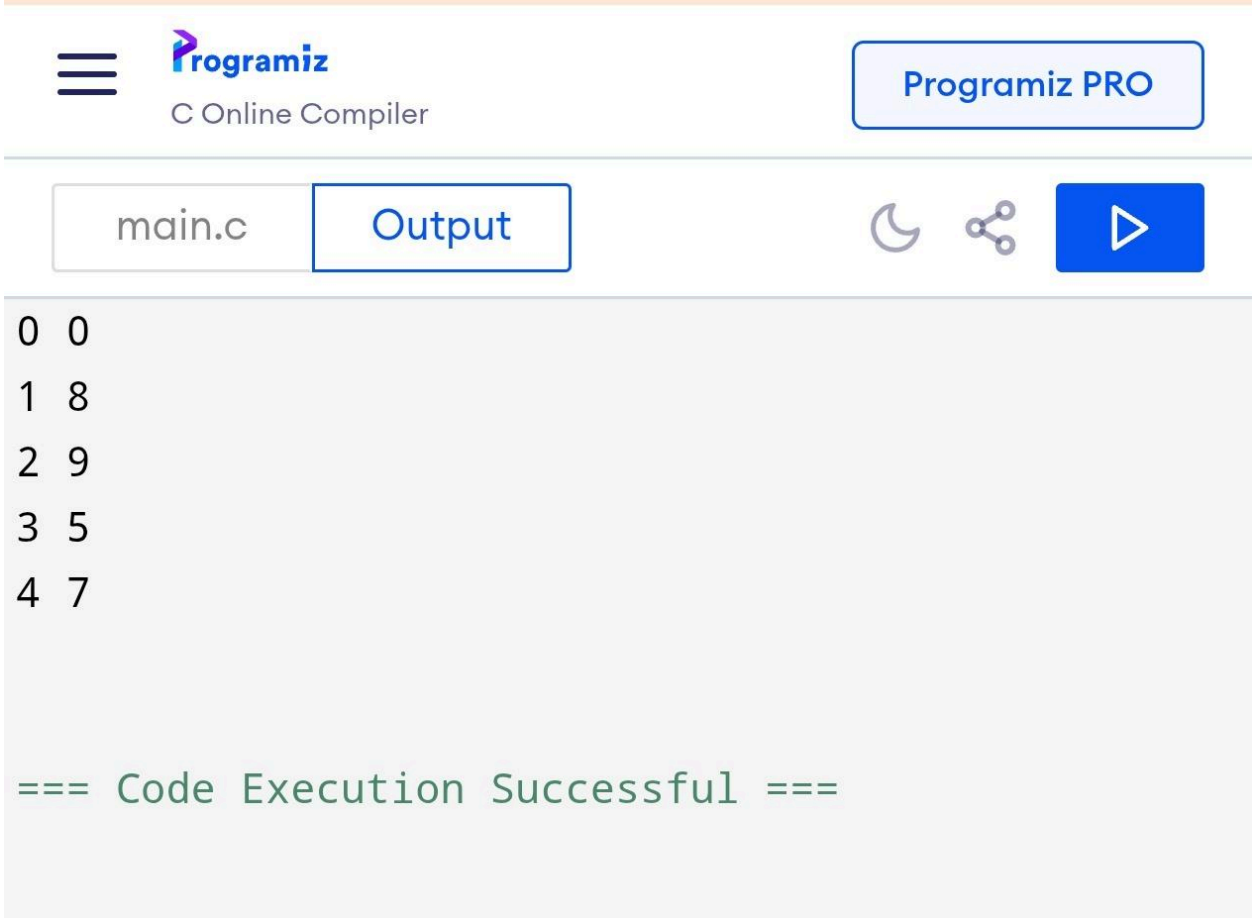
1. **Dijkstra Algorithm Implementation**

```
#include <stdio.h>
#include <limits.h>

#define MAX 100
#define INF INT_MAX

int heap[MAX], pos[MAX], dist[MAX], size;

void swap(int *a, int *b){ int t=*a; *a=*b; *b=t; }

void heapify(int i){
   int s=i, l=2*i+1, r=2*i+2;
   if(l<size && dist[heap[l]]<dist[heap[s]]) s=l;
   if(r<size && dist[heap[r]]<dist[heap[s]]) s=r;
   if(s!=i){
      pos[heap[i]]=s; pos[heap[s]]=i;
      swap(&heap[i],&heap[s]);
      heapify(s);
   }
}

void build_heap(int n){
   size=n;
   for(int i=n/2-1;i>=0;i--) heapify(i);
}

int extract_min(){
   int v=heap[0];
   heap[0]=heap[size-1];
   pos[heap[0]]=0;
   size--;
   heapify(0);
   return v;
}

void decrease_key(int v){
   int i=pos[v];
   while(i>0 && dist[heap[(i-1)/2]]>dist[heap[i]]){
      pos[heap[i]]=(i-1)/2;
      pos[heap[(i-1)/2]]=i;
      swap(&heap[i],&heap[(i-1)/2]);
```

```
        i=(i-1)/2;
    }
}

void dijkstra(int g[MAX][MAX], int n, int s){
    for(int i=0;i<n;i++){ dist[i]=INF; heap[i]=i; pos[i]=i; }
    dist[s]=0;
    build_heap(n);
    while(size>0){
        int u=extract_min();
        for(int v=0;v<n;v++)
            if(g[u][v] && dist[v]>dist[u]+g[u][v]){
                dist[v]=dist[u]+g[u][v];
                decrease_key(v);
            }
    }
}

int main(){
    int n=5;
    int g[MAX][MAX]={
        {0,10,0,5,0},
        {0,0,1,2,0},
        {0,0,0,0,4},
        {0,3,9,0,2},
        {7,0,6,0,0}
    };
    int s=0;
    dijkstra(g,n,s);
    for(int i=0;i<n;i++) printf("%d %d\n",i,dist[i]);
}
```


Programiz
C Online Compiler
Programiz PRO
main.c
Output
0 0
1 8
2 9
3 5
4 7
=== Code Execution Successful ===