SSSP

● Graded

Student

SRUTHI SUBRAMANIAN

Total Points

50 / 50 pts

Autograder Score

50.0 / 50.0

Passed Tests

Test 1 (10/10)
Test 2 (20/20)
Test 3 (20/20)

Autograder Results

Test 1 (10/10)

Test 2 (20/20)

Test 3 (20/20)

Submitted Files

hw.c

Download

```c
//Final Code for SSSP
#include <stdio.h>

int main() {
    int n;
    scanf("%d",&n);
    int adjmat[n+1][n+1];
    int a=0;
    int b;
    int inf=5;
    int size=n;
    for (int i=0;i<n;i++){
        for (int j=0;j<n;j++){
            adjmat[i][j]=0;
        }
    }
    for (int i=0;i<n;i++){
        scanf("%d", &a);
        while (a!=-1){
            scanf("%d",&b);
            adjmat[i][a]=b;
            inf=inf+b;
            scanf("%d", &a);
        }
    }
    int source;
    scanf("%d", &source);
    int Heap[n+1];
    int d[n+1];
    for (int i=0;i<n;i++){
        d[i]=inf;
        Heap[i]=i;
    }

    int* MinHeapify(int* heap, int ind) {
            //1 indexing!!
            int minind = ind;
            if (ind * 2 <= size && d[heap[2 * ind - 1]] < d[heap[ind - 1]]) {
                minind = 2 * ind;
            }
            if (ind * 2 + 1 <= size && d[heap[2 * ind]] < d[heap[minind - 1]]) {
                minind = 2 * ind + 1;
            }
            if (minind != ind) {
                //swap minind term and X and then call MinHeapify on minind
                int X = ind;
                int t = heap[X - 1];
                heap[X - 1] = heap[minind - 1];
                heap[minind - 1] = t;
                return MinHeapify(heap, minind);
            }
            else {
                return heap;
            }
        }
    int ExtractMin(int* heap){
            int temp;
            temp = heap[0];
            heap[0]=heap[size-1];
            size=size-1;
            MinHeapify(heap,1);
            return temp;
        }
    int find(int* heap, int ind){
        for (int k=0;k<size;k++){
            if (heap[k]==ind){
                return k+1;
            }
        }
    }
    int DecreaseKey(int* heap, int ind, int key){
        d[ind-1]=key;
        int X=find(heap,ind-1);
        int parent;
        int temp;
        while(1){
            if(X==1){
                return 0;
            }
            parent =X/2;
            if (d[heap[X-1]]>d[heap[parent-1]]){
                return 0;
            }
            else{
                temp=heap[parent-1];
                heap[parent-1]=heap[X-1];
                heap[X-1]=temp;
                X=parent;
            }
        }
    }
    int final[n+1];
    DecreaseKey(Heap,source+1,0);
    int u = ExtractMin(Heap);
    int m=0;
    final[m]=u;
    m=m+1;
    for (int j=0;j<n;j++){
        if (adjmat[u][j]!=0){
            if (d[j]>d[u]+adjmat[u][j]){
                DecreaseKey(Heap,j+1,d[u]+adjmat[u][j]);
            }
        }
    }
    while (size>0){
        u=ExtractMin(Heap);
        final[m]=u;
        m=m+1;
        for (int j=0;j<n;j++){
            if (adjmat[u][j]!=0){
                if (d[j]>d[u]+adjmat[u][j]){
                    DecreaseKey(Heap,j+1,d[u]+adjmat[u][j]);
                }
            }
        }
    }
    for(int i=0;i<n;i++){
        printf("%d %d ",final[i],d[final[i]]);
    }
    return 0;
}
```