ACA
Algorithmic Graph Theory
Assignment-2

Name : V Nikhil
Roll NO. : 221159
Contact no. : 9398977278
Email : vnikhil22@iitk.ac.in

Problem 1:
Shortest Routes I

Below is the C++ code.

```
#include<bits/stdc++.h>
using namespace std;

vector<long long int> dijkstra(int n,int m,vector<pair<int,long long int>>adj[],int S){
    priority_queue<pair<int,long long int>,vector<pair<int,long long
int>>,greater<pair<int,long long int>>> pq;
    vector<long long int> distance(n);
    distance[S]=0;
    for(int i=0;i<n;i++){
        if(i!= S) distance[i]=1e15;
    }
    pq.push(make_pair(S,0));
    while(!pq.empty()){
        int node=pq.top().first;
        long long int dis=pq.top().second;
        pq.pop();
        if(dis>distance[node]) continue;
    for(auto i:adj[node]){
        if(dis+i.second<distance[i.first]) {
            distance[i.first]=dis+i.second;
            pq.push(make_pair(i.first,distance[i.first]));
        }
    }
    }
    return distance;
}


int main(){
    int n,m;
    cin>>n>>m;
    vector<pair<int,long long int>> adj[n];
    for(int i=0;i<m;i++){
        int a,b;
        long long int c;
        cin>>a>>b>>c;
        adj[a-1].push_back(make_pair(b-1,c));
    }
   for(auto i:dijkstra(n,m,adj,0)) cout<<i<<" ";
}
```

Problem 2:
Shortest Route II

Below is the C++ code.

```
    #include <bits/stdc++.h>
    using namespace std;

    int main(){
        int n,m,q;
        cin>>n>>m>>q;
        long long int cost[n][n];
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                if(i==j) cost[i][j]=0;
                else cost[i][j]=1e14;
            }
        }
        for(int i=0;i<m;i++){
            int a,b;
```

```
          long long int c;
          cin>>a>>b>>c;
        if(cost[a-1][b-1]>c)  cost[a-1][b-1]=c;
             cost[b-1][a-1]=cost[a-1][b-1];
             }

      for(int k=0;k<n;k++){
          for(int i=0;i<n;i++){
              for(int j=0;j<n;j++){
                  if(cost[i][j]>cost[i][k]+cost[k][j]){
                      cost[i][j]=cost[i][k]+cost[k][j];
                      }

              }
          }
      }

  long long int answer[q];
  for(int i=0;i<q;i++){
      int e,r;
      cin>>e>>r;
     if(cost[e-1][r-1]!=1e14) answer[i]=cost[e-1][r-1];
     else answer[i]=-1;
  }
  for(int i=0;i<q;i++){
      cout<<answer[i]<<endl;
  }
  }
```

Problem 2:
Flight Discount

Below is the C++ code.

```
#include<bits/stdc++.h>
using namespace std;

long long int dijkstra(long long int n,long long int m,vector<pair<long long int, long long
int>>adj[],long long int S){
    priority_queue<vector<long long int>,vector<vector<long long
int>>,greater<vector<long long int>>> pq;
    vector< long long int> discount(n);
    vector< long long int> disused(n);
    discount[S]=0;
    disused[S]=0;
    for(long long int i=0;i<n;i++){
        if(i!= S) {discount[i]=1e15;disused[i]=1e15;}
    }
    vector<long long int> land1={0,S,0};
    pq.push(land1);
    while(!pq.empty()){
        long long int dis=pq.top()[0];
        long long int node=pq.top()[1];
        long long int use=pq.top()[2];
        pq.pop();
        if(use==1) if(disused[node]<dis) continue;
        if(use==0) if(discount[node]<dis) continue;
    for(auto i:adj[node]){
        if(dis+i.second<discount[i.first]) {
           if(use==0){if(discount[i.first]>dis+i.second){ discount[i.first]=dis+i.second;
            vector<long long int> land2={discount[i.first],i.first,0};
            pq.push(land2);}
         if(disused[i.first]>dis+(i.second/2)) {  disused[i.first]=dis+(i.second/2);
              vector<long long int> land3={disused[i.first],i.first,1};pq.push(land3);}
            }
            if(use==1) {if(disused[i.first]>dis+(i.second))
            {disused[i.first]=dis+i.second;
            vector<long long int> land4={disused[i.first],i.first,1};
            pq.push(land4);}}
        }
    }
    }
    return disused[n-1];
}


int main(){
    long long int n,m ;
    cin>>n>>m;
    vector<pair<long long int, long long int>> adj[n];
    for(long long int i=0;i<m;i++){
        long long int a,b;
```

```
         long long int c;
        cin>>a>>b>>c;
        adj[a-1].push_back(make_pair(b-1,c));
    }
   cout<<dijkstra(n,m,adj,0);
}
```

Problem 4:
Path Sum : Four Ways

Below is the C++ code.

```
#include<bits/stdc++.h>
using namespace std;
int dijkstra(int matrix[80][80]){
    priority_queue< pair<int,pair<int,int>>,
vector<pair<int,pair<int,int>>>,greater<pair<int,pair<int,int>>>> pq;
    vector<vector<int>> ans;
    for(int i=0;i<80;i++){
        vector<int> pusher(80);
        for(int j=0;j<80;j++){
            pusher[j]=1e7;
        }
        ans.push_back(pusher);}
        ans[0][0]=0;
    pq.push(make_pair(0,make_pair(0,0)));
    while(!pq.empty()){
        pair<int,int>node=pq.top().second;
        int dis=pq.top().first;
        pq.pop();
        if(dis>ans[node.first][node.second]) continue;
        if(node.first<80-1 ) {
            if(dis+matrix[node.first][node.second]<ans[node.first+1][node.second]){
                ans[node.first+1][node.second] = dis+matrix[node.first][node.second] ;

pq.push(make_pair(ans[node.first+1][node.second],make_pair(node.first+1,node.secon
d))); }}

        if(node.first>0  ) {
            if(dis+matrix[node.first][node.second]<ans[node.first-1][node.second]){
                ans[node.first-1][node.second] = dis+matrix[node.first][node.second] ;
pq.push(make_pair(ans[node.first-1][node.second],make_pair(node.first-1,node.second)
)); }}

        if(node.second<80-1 ) {
            if(dis+matrix[node.first][node.second]<ans[node.first][node.second+1]){
                ans[node.first][node.second+1] = dis+matrix[node.first][node.second] ;

pq.push(make_pair(ans[node.first][node.second+1],make_pair(node.first,node.second+
1))); }}

        if(node.second>0  ) {
            if(dis+matrix[node.first][node.second]<ans[node.first][node.second-1]){
                ans[node.first][node.second-1] = dis+matrix[node.first][node.second] ;

pq.push(make_pair(ans[node.first][node.second-1],make_pair(node.first,node.second-1)
));}}
    }
    return ans[80-1][80-1]+matrix[80-1][80-1];

}

int main(){

    int matrix[80][80];
    for(int i=0;i<80;i++){
        for(int j=0;j<80;j++){
            cin>>matrix[i][j];
        }
    }
    cout<<dijkstra(matrix);
}
```