1.堆优化的 Dijkstra

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=510;
const int inf=0x3fffffff;
typedef pair<int,int> P;
struct Node{
    int to,dis;
    Node(int a,int b){
        to=a;dis=b;
    }
    Node(){}
};
vector<Node> G[maxn];
int n,m,st,ed,cost[maxn][maxn];
int d[maxn],c[maxn],pre[maxn];
void Dijkstra(int s){
    fill(d,d+maxn,inf);
    fill(c,c+maxn,inf);         //花费 c 越小越好
//  fill(w,w+maxn,0);     //weight 权值越大越好
//  w[s]=weight[s];
    d[s]=0;
    c[s]=0;
    priority_queue<P,vector<P>,greater<P> >q;
    q.push(P(0,s));
    while(!q.empty()){
        P now=q.top();q.pop();
        int u=now.second;
        if(now.first>d[u]) continue;
        for(auto i:G[u]){
            int v=i.to,dis=i.dis
            if(d[u]+dis<d[v]){
                d[v]=d[u]+dis;
                c[v]=c[i.me]+cost[u][v];
            //  w[v]=w[u]+weight[v];
                q.push(P(d[v],v));          //将原点距离本节点的距离及本节点的
编号入队列
                pre[v]=u;
            }else if(d[u]+dis==d[v]){
                if(c[u]+cost[u][v]<c[v]){
                    c[v]=c[u]+cost[u][v];
                    pre[v]=u;
                //  w[v]=w[u]+weight[v];
```

```
                    }

                }
            }
        }
    }
}
void DFS(int v){
    if(v==st){
        printf("%d ",v);
        return;
    }
    DFS(pre[v]);
    printf("%d ",v);
}
int main() {
    int c1,c2,dis,cos;
    scanf("%d%d%d%d",&n,&m,&st,&ed);
    for(int i=0;i<m;i++){
        scanf("%d%d%d%d",&c1,&c2,&dis,&cos);
        cost[c1][c2]=cost[c2][c1]=cos;
        Node node(c2,dis,c1);
        G[c1].push_back(node);
        node.to=c1;
        node.me=c2;
        G[c2].push_back(node);
    }
    Dijkstra(st);
    DFS(ed);
    printf("%d %d\n",d[ed],c[ed]);
    return 0;
}
```

2.堆优化的 Dijkstra+DFS

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=510;
const int inf=0x3fffffff;
struct Node{
    int to,dis;
    Node(int a,int b){
        to=a;dis=b;
    }
    Node(){};
```

```
};
typedef pair<int,int> P;
int n,m,st,ed,tolvalue=inf;
int d[maxn],cost[maxn][maxn];
vector<Node> G[maxn];
bool vis[maxn]={false};
vector<int> temp,ans,pre[maxn];
int w[maxn],weight[maxn];
void Dijkstra(int s){
	priority_queue<P,vector<P>,greater<P> >q;
	fill(d,d+maxn,inf);
	d[s]=0;
	q.push(P(0,s));
	while(!q.empty()){
		P now=q.top();q.pop();
		int u=now.second;
		if(now.first>d[u]) continue;
		for(auto i:G[u]){
			int v=i.to,dis=i.dis;
				if(d[u]+dis<d[v]){
					d[v]=d[u]+dis;
					q.push(P(d[v],v));
					pre[v].clear();
					pre[v].push_back(u);
				}else if(d[u]+dis==d[v]){
					pre[v].push_back(u);
				}
			}
	}
}
void DFS(int s){
	if(s==st){
		int val=0;
		//int weit=0;
		temp.push_back(st);
		for(int i=temp.size()-1;i>0;i--){
			int id=temp[i],nextid=temp[i-1];
			val+=cost[id][nextid];
		}
	/*	for(int i=temp.size()-1;i>=0;i--){
			weit+=weight[i];
		}*/
		if(val<tolvalue){
			tolvalue=val;
```

```
                ans=temp;
            }
            temp.pop_back();
        }else{
            temp.push_back(s);
            for(auto i:pre[s]){
                DFS(i);
            }
            temp.pop_back();
        }
}

int main() {
    int c1,c2,dis,cos;
    scanf("%d%d%d%d",&n,&m,&st,&ed);
    for(int i=0;i<m;i++){
        scanf("%d%d%d%d",&c1,&c2,&dis,&cos);
        Node node(c2,dis);
        G[c1].push_back(node);
        node.to =c1;
        G[c2].push_back(node);
        cost[c1][c2]=cost[c2][c1]=cos;
    }
    Dijkstra(st);
    DFS(ed);
    reverse(ans.begin(),ans.end());
    for(auto i:ans)
        printf("%d ",i);
    printf("%d %d\n",d[ed],tolvalue);
    return 0;
}
```

3.完全的 SPFA

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=510;
const int inf=0x3fffffff;
typedef pair<int,int> P;
struct Node{
    int to,dis,me;
    Node(int a,int b,int c){
        to=a;dis=b;me=c;
    }
```

```
};
int n,m,st,ed,weight[maxn],vis[maxn];
int d[maxn],w[maxn],num[maxn],numinq[maxn];
vector<Node> G[maxn];
set<int> pre[maxn];
bool inq[maxn];
bool SPFA(int s){
    fill(d,d+maxn,inf);
//  fill(c,c+maxn,inf);
    fill(w,w+maxn,0);
    fill(num,num+maxn,0);
    memset(inq,false,sizeof(inq));
//  c[s]=0;
    d[s]=0;
    w[s]=weight[s];
    num[s]=1;
    priority_queue<P,vector<P>,greater<P> >q;
    q.push(P(0,s));
    inq[s]=true;
    numinq[s]++;
    while(!q.empty()){     //尽量别用 i.to,i.dis 很容易搞晕的，用 u,v,dis
        P now=q.top();q.pop();
        inq[now.second]=false;
        int u=now.second;
        for(auto i:G[u]){
            int v=i.to,dis=i.dis;
            if(d[u]+dis<d[v]){
                d[v]=d[u]+dis;
                w[v]=w[u]+weight[v];
                q.push(P(d[v],v));
                num[v]=num[u];
                pre[v].clear();
                pre[v].insert(u);
                if(!inq[v]){
                    q.push(P(d[v],v));
                    inq[v]=true;
                    numinq[v]++;
                    if(numinq[v]>=n) return false;
                }
            }else if(d[u]+dis==d[v]){
                pre[v].insert(u);
                if(w[u]+weight[v]>w[v]){
                    w[v]=w[u]+weight[v];
                }
```

```
                    num[v]=0;
                    for(auto k:pre[v])
                        num[v]+=num[k];
                }
            }
        }
        return true;
}
int main() {
        int c1,c2,dis;
        scanf("%d%d%d%d",&n,&m,&st,&ed);
        for(int i=0;i<n;i++){
            scanf("%d",&weight[i]);
        }
        for(int i=0;i<m;i++){
            scanf("%d%d%d",&c1,&c2,&dis);
            Node node(c2,dis,c1);
            G[c1].push_back(node);
            node.to =c1;
            node.me =c2;
            G[c2].push_back(node);
        }
        if(SPFA(st))
            printf("%d %d\n",num[ed],w[ed]);
        return 0;
}
```

SPFA+DFS

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=510;
const int inf=0x3fffffff;
struct Node{
        int to,dis;
        Node(int a,int b){
            to=a;dis=b;
        }
};
int n,m,st,ed,num,numinq[maxn];
vector<Node> G[maxn];
vector<int>temp,ans;
set<int> pre[maxn];
int cost[maxn][maxn],d[maxn];
int minvalue=inf;
```

```
bool inq[maxn];
bool SPFA(int s){
    fill(d,d+maxn,inf);
    fill(inq,inq+maxn,false);
    d[s]=0;
    queue<int> q;
    q.push(s);
    numinq[s]++;
    inq[s]=true;
    while(!q.empty()){
        int u=q.front();q.pop();
        inq[s]=false;
        for(auto i:G[u]){
            int v=i.to,dis=i.dis;
            if(d[u]+dis<d[v]){
                d[v]=d[u]+dis;
                pre[v].clear();
                pre[v].insert(u);
                if(!inq[v]){
                    q.push(v);
                    inq[v]=true;
                    numinq[v]++;
                    if(numinq[v]>=n) return false;
                }
            }else if(d[u]+dis==d[v])
                    pre[v].insert(u);
        }
    }
    return true;
}
void DFS(int s){
    if(s==st){
        int val=0;
        temp.push_back(st);
        for(int i=temp.size()-1;i>0;i--){
            int id=temp[i],nextid=temp[i-1];
            val+=cost[id][nextid];
        }
        if(val<minvalue){
            minvalue=val;
            ans=temp;
        }
        temp.pop_back();
    }else{
```

```
            temp.push_back(s);
            for(auto i:pre[s]){
                DFS(i);
            }
            temp.pop_back();
        }
}
int main() {
        int c1,c2,dis,cos;
        scanf("%d%d%d%d",&n,&m,&st,&ed);
        for(int i=0;i<m;i++){
            scanf("%d%d%d%d",&c1,&c2,&dis,&cos);
            cost[c1][c2]=cost[c2][c1]=cos;
            Node node(c2,dis);
            G[c1].push_back(node);
            node.to =c1;
            G[c2].push_back(node);
        }
        SPFA(st);
        DFS(ed);
        reverse(ans.begin(),ans.end());
        for(auto i:ans)
            printf("%d ",i);
        printf("%d %d\n",d[ed],minvalue);
        return 0;
}
```

优先队列 Kruscal

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=50010;
struct Node{
        int u,v,dis;
        Node(int a,int b,int c){
            u=a;v=b;dis=c;
        }
        bool friend operator <(Node a,Node b){
            return a.dis>b.dis;
        }
};
int n,m,root,father[maxn];
int findFather(int a){
```

```
    if(father[a]==a)
        return a;
    int temp=findFather(father[a]);
    father[a]=temp;
    return temp;
}
int main() {
    priority_queue<Node> q;
    int c1,c2,dis,ans,edge_num=0;
    scanf("%d%d%d",&n,&m,&root);
    while(m--){
        scanf("%d%d%d",&c1,&c2,&dis);
        Node node(c1,c2,dis);
        q.push(node);
    }
    iota(father,father+maxn,0);
    while(edge_num!=n-1){
        Node now=q.top();q.pop();
        int fa=findFather(now.u);
        int fb=findFather(now.v);
        if(fa!=fb){
            father[fa]=fb;
            ans=now.dis ;
            edge_num++;
        }
    }
    puts(to_string(ans).c_str());
    return 0;
}
```