图论——最短路问题

Dijkstra 算法

一、常见最短路问题

1.**单目标最短路径问题：**找出从每一顶点 v 到某指定顶点 u 的一条最短路径。把图中的每条边反向，我们就可以把这一问题转化为单源最短路径问题。

2.**单对顶点间的最短路径问题：**对于某给定顶点 u 和 v，找出从 u 到 v 的一条最短路径。如果我们解决了源顶点为 u 的单源问题，则这一问题也就获得了解决。一般来讲，目前还未发现比最好的单源算法更快的方法。

3.**每对顶点间的最短路径问题：**对于每对顶点 u 和 v，找出从 u 到 v 的最短路径。

二、最短路算法

常见最短路算法有：

floyd 算法：在边权非负的有向图中计算每对顶点间的最短路径问题。该算法在图的传递闭包的基础上形成；

Dijkstra 算法：在边权非负的有向图中计算单源最短路径问题。该算法建立在松弛技术基础之上；

Bellman-Ford 算法：能在更一般的情况下解决单源点最短路径问题。所谓一般情况，指的是有向图的边权可以为负,但不允许存在负权回路。该算法亦是建立在松弛技术基础之上的；

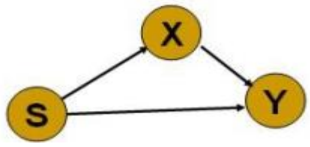
SPFA 算法：是一种很高效的求图的最短路径的算法，正负权都可以，还能够判断负权回路问题。

三、Dijkstra 算法

Dijkstra 算法是解决**正边权单源最短路**问题，他是基于松弛原理。

1、 松弛原理

三角形性质：设源点 **s** 到点 **x**、**y** 的最短路径长度为 **d[x]**、**d[y]**。**x** 与 **y** 之间的距离是 **g[x][y]**，则有下面的“三角形定理”：



d[x]+g[x][y]>=d[y]

松驰：

若在处理过程中，有两点 **x**、**y** 出现不符合“三角形定理”，则可松驰一下：

```
if(d[x]+g[x][y]<d[y])
    d[y]=d[x]+g[x][y];
```

2、 算法过程

贪心思想，从起点 **v0** 每次新扩展一个距离最短的点，再以这个点为中间点，更新起点到其他所有点的距离。由于所有边权都为正，故不会存在一个距离更短的没被扩展过的点，所以这个点的距离永远不会再被改变，因而保证了算法的正确性。

算法实现时，用一维数组 vis**[i]**表示源点到顶点 **i** 的最短距离求出没有。用 **d[i]**记录源点 **v0** 到顶点 **i** 的距离值。

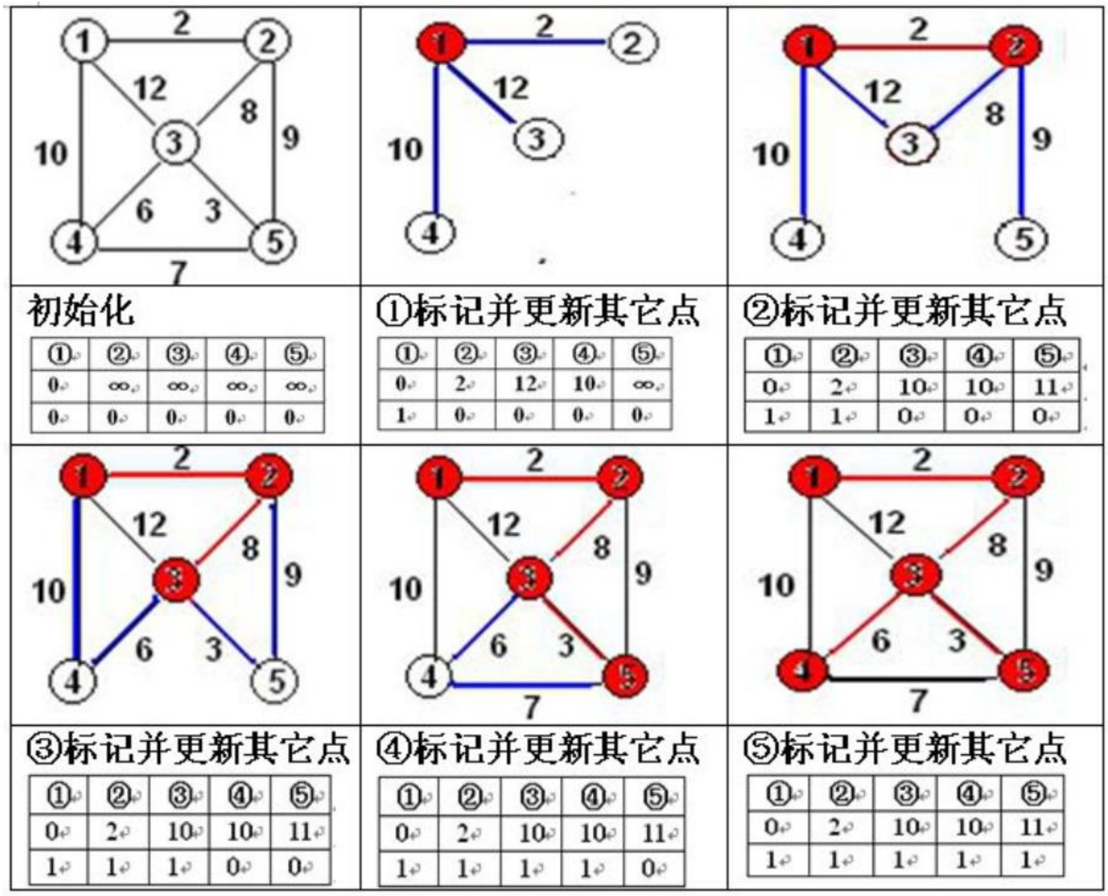
算法步骤：

初始化 d[v0]=0，源点到其他点的距离值 d[i]=∞。②经过 n 次如下步骤操作，最后得到

v0 到 n 个顶点的最短距离：

- A.选择一个未标记的点 k 并且 d[k]的值是最小的；
- B.标记点 k，即 vis[k]=1；
- C.以 k 为中间点，修改源点 v0 到其他未标记点 j 的距离值 d[j]

算法过程图演：



三、例题

例 1：P3015 最短路

平面上有 n 个点（n<=100），每个点的坐标均在-10000~10000 之间。其中的一些点之间有连线。

若有连线，则表示可从一个点到达另一个点，即两点间有通路，通路的距离为两点间的直线距离。现在的任务是找出从一点到另一点之间的最短路径。

【分析】最短路模板题

【参考代码】

```
#include <bits/stdc++.h>
#define INF 0x7fffffff
using namespace std;
double dis[110],g[110][110];
int visit[110];
int x[110],y[110];
int n,m,a,b,s,t;
double dist(int a,int b){
```

```
    return sqrt(double((x[a]-x[b])*(x[a]-x[b]))+ double((y[a]
                      -y[b])*(y[a]-y[b])));
}
void DJ(int start){
    int x;
    dis[start]=0;
    for(int i=1;i<=n;i++){
        int mini=INF;
        for(int j=1;j<=n;j++)
            if(!visit[j]&&dis[j]<mini)
                mini=dis[j],x=j;
        visit[x]=1;
        for(int j=1;j<=n;j++)
            if(!visit[j]&&g[x][j]<INF&&dis[j]>dis[x]+g[x][j])
                dis[j]=dis[x]+g[x][j];
    }
}
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
        dis[i]=INF;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            g[i][j]=INF;
    for(int i=1;i<=n;i++)
        scanf("%d%d",&x[i],&y[i]);
    scanf("%d",&m);
    while(m--){
        scanf("%d%d",&a,&b);
        g[a][b]=g[b][a]=dist(a,b);
    }
    cin>>s>>t;
    DJ(s);
    printf("%.2lf",dis[t]);
    return 0;
}
```

上述算法效率是 N*N.

对于稀疏图效率不高，我们通过分析算法发现，每次被标记的点在减少，而查找最小值有大量重复运算，对于最小值维护我们很容易想到堆，所以可以用**堆**来优化。

例 2：A2050 模板题

题意：n 个节点，m 条带权边构成的无向图，给定起点与终点，求出起点到终点的最短路

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
#define N 10000
struct Node{
   int u,v,w;
}e[N<<1];//边目录
struct T{
   int d,u;
   bool operator<(const T&t)const{
      return d>t.d;
   }
};
int first[N]={0},nxt[N<<1],cnt=0;
int n,m,st,ed;
int vis[N],dis[N];
void add(int u,int v,int w){//前向星加边模板
   e[++cnt].u=u;e[cnt].v=v;e[cnt].w=w;
   nxt[cnt]=first[u];first[u]=cnt;
}
void init(){
   scanf("%d%d%d%d",&n,&m,&st,&ed);
   for(int i=1;i<=m;i++){
      int u,v,w;
      scanf("%d%d%d",&u,&v,&w);
      add(u,v,w);add(v,u,w);
   }
}
void dijkstra(int s){
   memset(vis,0,sizeof(vis));
   memset(dis,0x3f,sizeof(dis));
   dis[s]=0;
   priority_queue< T > q;
   q.push(T{dis[s],s});
   while(!q.empty()){
      T t= q.top();  q.pop();
      int d=t.d,u=t.u;
      if(vis[u])continue;
      vis[u]=1;
      for(int i=first[u];i;i=nxt[i]){
         int v=e[i].v;
         if(d+e[i].w<dis[v]){
            dis[v]=d+e[i].w;
```

```
            q.push(T{dis[v],v});
         }
      }
   }
}
int main(){
   init();
   dijkstra(st);
   cout<<dis[ed];
   return 0;
}
```

例 3：P3013 香甜的黄油

【题意】找出一个点，这个点使得所有奶牛到改点距离和最小

【分析】单汇一般都是转为单源问题。将图反建，由于本题是无向图，所以不需要反向建图。

算法：

枚举每个点作为源点求最短路，然后求最短路总和，打擂台即可。

例 4：P3011 最小花费

在 n 个人中，某些人的银行账号之间可以互相转账。这些人之间转账的手续费各不相同。给定这些人之间转账时需要从转账金额里扣除百分之几的手续费，请问 A 最少需要多少钱使得转账后 B 收到 100 元。

【分析】本题有一点变形，相当于是将边权变为了乘法。

且问的是达到终点是 100 元，起点是多少。我们可以逆向思维，可以假设开始是 100 元，到了终点是多少，然后再做一次运算即可。

代码上将最短路的松弛

If（dist[u]+w < dist[v]）

改为：

If（dist[u]*w < dist[v]）

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
#define maxn 2010
#define maxm 8000010
struct  edge{
      int v,next; double w;
}map[maxm];
int head[maxn],tmpcnt=0;
void add(int s,int v,int z){
   map[++tmpcnt].v=v;
   map[tmpcnt].next=head[s];
   head[s]=tmpcnt;
   map[tmpcnt].w=1.0-z*0.01;
}
int n,m,a,b,tx,ty,tz;
```

```
double dis[maxn],vis[maxn];
struct T{
   double d;int u;
   bool operator<(const T&t)const{
      return d < t.d;
   }
};
void  dijkstra(){
   for(int i=1;i<=n;i++)dis[i]=-1e9;
   dis[a]=1.0;
   priority_queue< T > q;
   q.push((T){dis[a],a});
   while(!q.empty()){
      T t= q.top();  q.pop();
      double d=t.d;
      int u=t.u;
      if(vis[u])continue;
      vis[u]=1;
      for(int i=head[u];i!=-1;i=map[i].next){
         if(dis[map[i].v] < dis[u]*map[i].w){
            dis[map[i].v]=dis[u]*map[i].w;
            q.push((T){dis[map[i].v],map[i].v});
         }
      }
   }

}
int main(){
   scanf("%d%d",&n,&m);
   memset(head,-1,sizeof(head));
   for(int i=1;i<=m;i++)   {
      scanf("%d%d%d",&tx,&ty,&tz);
      add(tx,ty,tz);
      add(ty,tx,tz);
   }
   scanf("%d%d",&a,&b);
   dijkstra();
   printf("%.8lf",100.0/dis[b]);
   return 0;
}
```