

图论 1

最短路算法

非负权图单源最短路

`\text{dijkstra}` 算法板子

时间复杂度 $O(m \log m)$

```
template <typename T>
struct dijkstra{
    T dis[MAXN];
    bool vis[MAXN];
    priority_queue<pair<T,int>,vector<pair<T,int>>,greater<pair<T,int>>>q;
    void solve(int src,int n){
        mem(vis,0);
        _rep(i,1,n)
        dis[i]=Inf;
        dis[src]=0;
        q.push(make_pair(dis[src],src));
        while(!q.empty()){
            pair<T,int> temp=q.top();q.pop();
            int u=temp.second;
            if(vis[u])
                continue;
            vis[u]=true;
            for(int i=head[u];i;i=edge[i].next){
                int v=edge[i].to;
                if(dis[v]>edge[i].w+dis[u]){
                    dis[v]=edge[i].w+dis[u];
                    q.push(make_pair(dis[v],v));
                }
            }
        }
    }
};
```

例题

[洛谷 p1462](#)

题意

给定 n 个城市 m 条边以及起点、终点。

要求选择一条路径，满足路径边权和不超过给定值，且路径上的最大点权最小。

题解

二分点权上界，跑 `dijkstra` 时跳过点权大于该上界的点，计算起点到终点的边权和最短路，如果不超过给定值则更新答案。

时间复杂度 $O(n \log m \log v)$

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline char get_char(){
    char c=getchar();
    while(c==' '||c=='\n'||c=='\r')c=getchar();
    return c;
}
```

```
}

inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}

inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXN=1e4+5,MAXM=5e4+5,Inf=2e9;
struct Edge{
    int to,w,next;
}edge[MAXM<<1];
int head[MAXN],edge_cnt,val[MAXN],limit;
void Insert(int u,int v,int w){
    edge[++edge_cnt].next=head[u];
    edge[edge_cnt].to=v;edge[edge_cnt].w=w;
    head[u]=edge_cnt;
}
template <typename T>
struct dijkstra{
    T dis[MAXN];
    bool vis[MAXN];
    priority_queue<pair<T,int>,vector<pair<T,int> >,greater<pair<T,int> >
>q;
    void solve(int src,int n){
        mem(vis,0);
        _rep(i,1,n)
        dis[i]=Inf;
        dis[src]=0;
        q.push(make_pair(dis[src],src));
        while(!q.empty()){
            pair<T,int> temp=q.top();q.pop();
            int u=temp.second;
            if(vis[u])
                continue;
            vis[u]=true;
            for(int i=head[u];i;i=edge[i].next){
                int v=edge[i].to;
                if(val[v]>limit)
                    continue;
                if(dis[v]>edge[i].w+dis[u]){
                    dis[v]=edge[i].w+dis[u];
                    q.push(make_pair(dis[v],v));
                }
            }
        }
    }
};

dijkstra<LL> dj;
```

```
int main()
{
    int n=read_int(), m=read_int(), b=read_int(), u, v, w, lef=Inf, rig=0;
    _rep(i, 1, n){
        val[i]=read_int();
        lef=min(val[i], lef);
        rig=max(val[i], rig);
    }
    while(m--){
        u=read_int(), v=read_int(), w=read_int();
        Insert(u, v, w);
        Insert(v, u, w);
    }
    int mid, ans=-1;
    while(lef<=rig){
        mid=lef+rig>>1;
        limit=mid;
        dj.solve(1, n);
        if(dj.dis[n]<=b){
            ans=mid;
            rig=mid-1;
        }
        else
            lef=mid+1;
    }
    if(ans>=0)
        enter(ans);
    else
        puts("AFK");
    return 0;
}
```

无负环图多源最短路

\$\text{Floyd}\$ 算法板子

时间复杂度 \$O(n^3)\$

```
int n, dis[MAXN][MAXN];
void Floyd(){
    _for(i, 0, n)
        _for(j, 0, n)
            dis[i][j]=Inf;
    _for(i, 0, n)
        dis[i][i]=0;
    _for(k, 0, n)
        _for(i, 0, n)
            _for(j, 0, n)
```

```

    dis[i][j]=min(dis[i][j],dis[i][k]+dis[k][j]);
}

```

例题

[洛谷p1119](#)

题意

给定 n 个城市 m 条边，每个城市在第 t_i 天起才加入点集(保证 t_i 升序)。

q 个询问，每次询问第 t 天的 $dis(i,j)$ 保证询问的 t 升序。

题解

考虑 Floyd 算法本质其实是 dp

$dis[k][i][j]$ 表示只使用前 k 个点作为中转点时 i 和 j 间的最短路，可以用滚动数组省去一维。

所以本题只需要按时间更新即可。

```

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}

```

```
}

inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}

inline char get_char(){
    char c=getchar();
    while(c==' '||c=='\n'||c=='\r')c=getchar();
    return c;
}

inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}

inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXN=200+5,Inf=1e9;
int dis[MAXN][MAXN],t[MAXN];
int main()
{
    int n=read_int(),m=read_int(),u,v,w;
    _for(i,0,n)
    t[i]=read_int();
    _for(i,0,n)
    _for(j,0,n)
    dis[i][j]=Inf;
    while(m--){
        u=read_int(),v=read_int(),w=read_int();
        dis[u][v]=dis[v][u]=w;
    }
    _for(i,0,n)
    dis[i][i]=0;
    int q=read_int(),pos=0,temp;
    while(q--){
        u=read_int(),v=read_int(),temp=read_int();
        while(t[pos]<=temp&&pos<n){
            _for(i,0,n)
            _for(j,0,n)
            dis[i][j]=min(dis[i][j],dis[i][pos]+dis[pos][j]);
            pos++;
        }
        if(t[u]>temp||t[v]>temp||dis[u][v]==Inf)
        enter(-1);
        else
        enter(dis[u][v]);
    }
}
```

```

    }
    return 0;
}

```

有负环图单源最短路

$\text{\texttt{SPFA}}$ 算法板子

平均时间复杂度 $O(Km)$ 最坏时间复杂度 $O(nm)$

```

template <typename T>
struct SPFA{
    T dis[MAXN];
    int len[MAXN];
    bool inque[MAXN];
    bool solve(int src,int n){
        queue<int>q;
        mem(inqe,0);mem(len,0);
        _rep(i,1,n)
        dis[i]=Inf;
        dis[src]=0;len[src]=1;
        q.push(src);
        inque[src]=true;
        while(!q.empty()){
            int u=q.front();q.pop();
            inque[u]=false;
            for(int i=head[u];i;i=edge[i].next){
                int v=edge[i].to;
                if(dis[v]>dis[u]+edge[i].w){
                    dis[v]=dis[u]+edge[i].w;
                    len[v]=len[u]+1;
                    if(len[v]>n)
                        return false;
                    if(!inqe[v]){
                        q.push(v);
                        inque[v]=true;
                    }
                }
            }
        }
        return true;
    }
};

```

例题

[洛谷 p5960](#)

题意

解方程 $\left\{ \begin{array}{l} x_{a1}-x_{b1} \leq y_1 \\ x_{a2}-x_{b2} \leq y_2 \\ \vdots \\ x_{am}-x_{bm} \leq y_m \end{array} \right.$

题解

将 $x_j - x_i \leq y$ 移项，得 $x_j \leq x_i + y$ 。发现该式与单源最短路的三角不等式 $\text{dist}_j \leq \text{dist}_i + \text{dist}_{i \rightarrow j}$ 相似。

考虑添加超级源点 x_0 向所有其他点连一条权为 0 的边（事实上边权数值无特殊要求，边权相当于为所有解加上一个初始值）的单向边。

然后跑最短路算法即可，解得 $x_i = \text{dist}_i + k$ 为方程的一组可行解。

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
```

```

inline char get_char(){
    char c=getchar();
    while(c==' ' || c=='\n' || c=='\r')c=getchar();
    return c;
}
inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXN=1e4+5,MAXM=5e5+5,Inf=1e9;
struct Edge{
    int to,w,next;
}edge[MAXM<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v,int w){
    edge[++edge_cnt].next=head[u];
    edge[edge_cnt].to=v;edge[edge_cnt].w=w;
    head[u]=edge_cnt;
}
template <typename T>
struct SPFA{
    T dis[MAXN];
    int len[MAXN];
    bool inque[MAXN];
    bool solve(int src,int n){
        queue<int>q;
        mem(inqe,0);mem(len,0);
        _rep(i,1,n)
        dis[i]=Inf;
        dis[src]=0;len[src]=1;
        q.push(src);
        inque[src]=true;
        while(!q.empty()){
            int u=q.front();q.pop();
            inque[u]=false;
            for(int i=head[u];i;i=edge[i].next){
                int v=edge[i].to;
                if(dis[v]>dis[u]+edge[i].w){
                    dis[v]=dis[u]+edge[i].w;
                    len[v]=len[u]+1;
                    if(len[v]>n)
                        return false;
                    if(!inqe[v]){
                        q.push(v);
                        inque[v]=true;
                    }
                }
            }
        }
    }
}

```

```
        }
    }
    return true;
}
};

SPFA<int> spfa;
int main()
{
    int n=read_int(),m=read_int(),u,v,w;
    _rep(i,1,n)Insert(n+1,i,0);
    while(m--){
        u=read_int(),v=read_int(),w=read_int();
        Insert(v,u,w);
    }
    if(spfa.solve(n+1,n+1)){
        _rep(i,1,n){
            if(i>1)putchar(' ');
            write(spfa.dis[i]);
        }
    }
    else
        puts("NO");
    return 0;
}
```

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E5%9B%BE%E8%AE%BA_1&rev=1594607462

Last update: 2020/07/13 10:31

