算法设计题-图1

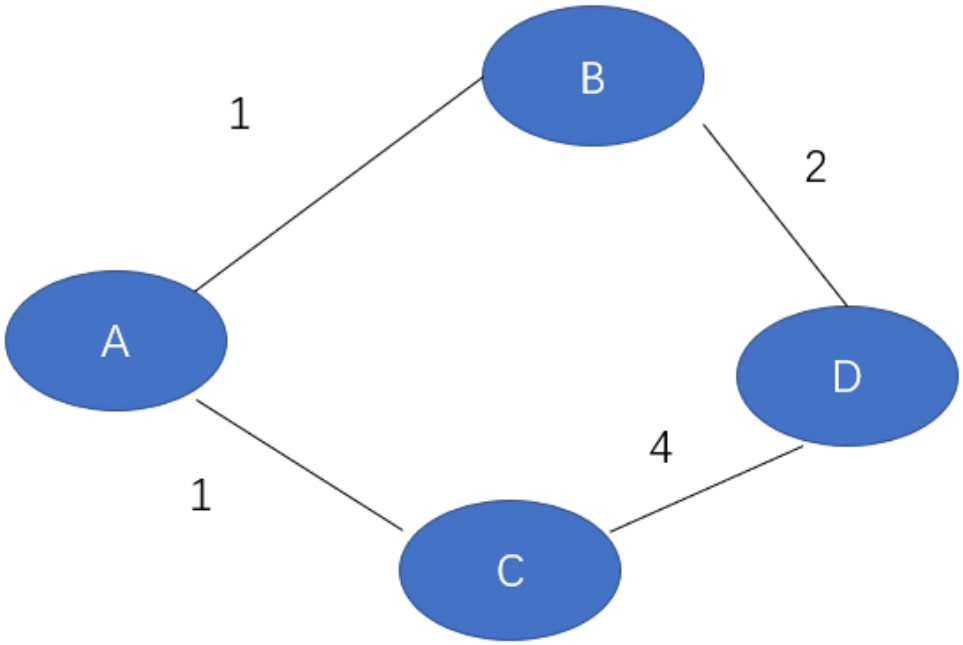
题目描述

A市有n个交通枢纽，其中1号和n号非常重要，为了加强运输能力，A市决定在1号到n号枢纽间修建一条地铁。 地铁由很多段隧道组成，每段隧道连接两个交通枢纽。经过勘探，有m段隧道作为候选，两个交通枢纽之间最多只有一条候选的隧道，没有隧道两端连接着同一个交通枢纽。 现在有n家隧道施工的公司，每段候选的隧道只能由一个公司施工，每家公司施工需要的天数一致。而每家公司最多只能修建一条候选隧道。所有公司同时开始施工。 作为项目负责人，你获得了候选隧道的信息，现在你可以按自己的想法选择一部分隧道进行施工，请设计一个算法，输出修建整条地铁最少需要的天数。

题目分析

对于此题，我们首先了解到此题需要借用图论的ADT实现，并且它的出发点是 最短路径（从1到n的最短路径），但是这个题，我们假设a到b已经连通，那么a到b的最短时间是 从a到b的这条路径的最长的一段子值。

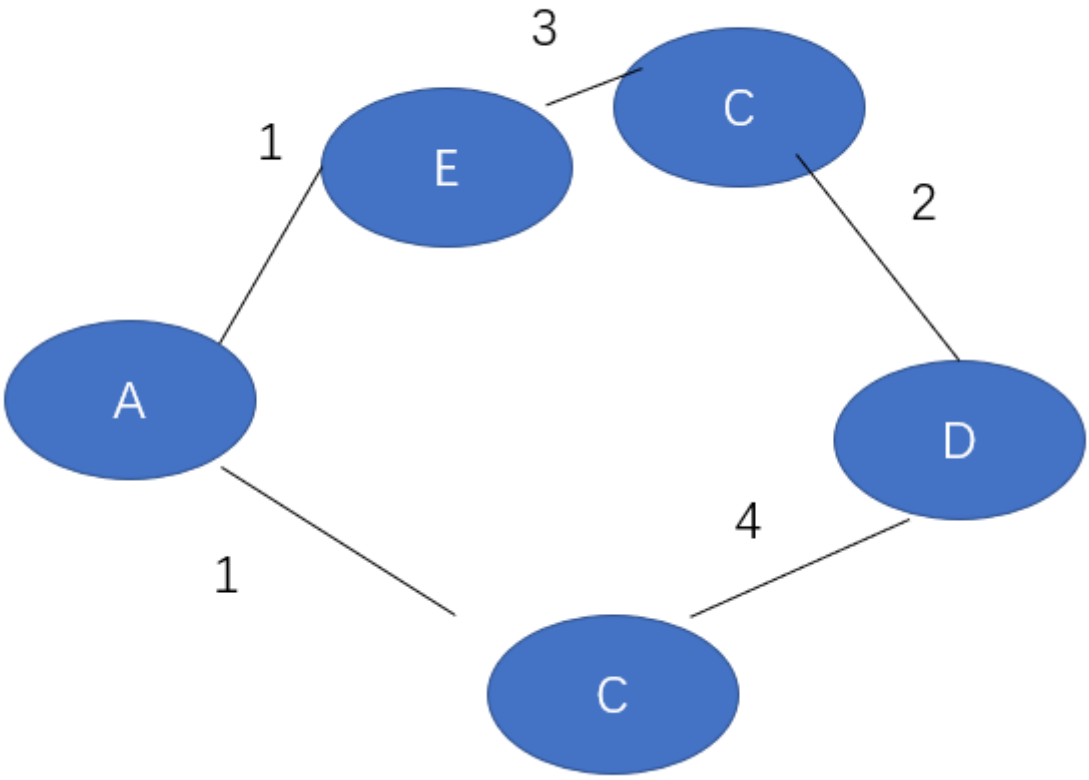
举个例子



我们在求解从a到d的路径，只需要求 a d的所有连通路径中，最大值最小的一个

比如 ABD （最大值为2） ACD （最大值为4），我们取ABD

例子再极化一下，更容易考虑



仍然是从A到D，两条路线，AECD（最大值3） ACD 最大值（4），我们仍然选择AECD（即使AECD总长大于ACD）

我们从 最短路径的 Dijkstra 算法改进一下，把判断条件从 **总长最短** 改为 ** 连通的最大值最小**

()

```
struct node{
    int v;
    int c;
    node(int _v, int _c):v(_v),c(_c){}
    bool operator <(const node&a)const{
        return c > a.c;
    }
};

void dijstra(){
    priority_queue<node> pq;
    pq.push(node(1,0));
    dist[1] =0;
    while(!pq.empty()){
        node tmp = pq.top();
        pq.pop();
        int u = tmp.v;
        if(vis[u]) continue;
        vis[u] = 1;
        for(int i=0; i<g[u].size(); ++i){
            int v = g[u][i].v;
            int cost = g[u][i].c;
```

```
            if(max(cost, dist[u]) < dist[v]){
                dist[v] = max(cost, dist[u]);
                pq.push(node(v,dist[v]));
            }
        }
    }
}
```

（由于我对Dij算法了解不是很透彻，所以附上（考试时）的弗洛伊德算法实现（更改判断条件））

```
for (int k=0; k<n; k++) {
        for (int i=0; i<n; i++) {
            for (int j=0; j<n; j++) {
                if (i == j) {
                    continue;
                }
                if (x.matrix[i][k]!=-1 && x.matrix[k][j]!=-1) {
                    int max=max(x.matrix[i][j], x.matrix[i][k],x.matrix[k][j]);
                    x.setEdge(i,j,max);
                }
            }
        }
    }
```

另外附上 最小生成树 （应该是此题的基本算法的最优解）的算法思想

（百度学习）

> 看到题目的最后一句就知道是克鲁斯卡尔算法，克鲁斯卡尔是持续往生成树中添加最小边，则生成树中同时包含1和n时，则连接1和n的路径中边的最大值一定是最小的。

```
#include<stdio.h>
#include<stdlib.h>
#define Maxsize 200005

typedef struct{
    int a,b;
    int time;
}ArcNode;

int n,m;
int V[Maxsize];
ArcNode List[Maxsize];

void CreateList()
{
    scanf("%d%d",&n,&m);
    for(int i=0;i<m;++i)
        scanf("%d%d%d",&List[i].a,&List[i].b,&List[i].time);
}

int cmp(const void *a,const void *b)
{
```

```
    ArcNode *A,*B;
    A=(ArcNode*)a;
    B=(ArcNode*)b;
    return A->time-B->time;
}

int GetRoot(int v)
{
    int start,root,post;
    start=v;
    while(V[v]!=v) v=V[v];
    root=v;
    while(V[start]!=root){
        post=V[start];
        V[start]=root;
        start=post;
    }
    return root;
}

void Kruskal()
{
    qsort(List,m,sizeof(ArcNode),cmp);
    for(int i=1;i<=n;++i) V[i]=i;
    for(int i=0;i<m;++i){
        int a=GetRoot(List[i].a);
        int b=GetRoot(List[i].b);
        if(a!=b) V[b]=a;
        if(GetRoot(1)==GetRoot(n)){
            printf("%d",List[i].time);
            break;
        }
    }
}

int main()
{
    CreateList();
    Kruskal();

    return 0;
}
```