2023.8.3 图论题单-下

1. Number of Simple Paths

【问题描述】

给你一张n个节点、n条边的无向连通图，请计算出这张图中长度大于等于1的不同的简单路径的数量。保证图中没有自环和重边。其中，简单路径中的节点必须互不相同，一条路径的长度定义为它所包含的边的数量。

两条路径仅有方向不同时被认为是同一条，例如1→2和2→1。

【链接】https://www.luogu.com.cn/problem/CF1454E

【输入格式】

第一行一个整数t（1 ⩽ t ⩽ 2 × 10^4），表示数据组数。

每组数据中，第一行是一个整数n（3 ⩽ n ⩽ 2 × 10^5），表示图的节点数和边数。接下来n行，每行两个整数u、v，表示一条无向边。

【输出格式】

对每组数据输出一个整数，表示不同的简单路径的数量。

【输入样例】

3

3

1 2

2 3

1 3

4

1 2

2 3

3 4

4 2

5

1 2

2 3

1 3

2 5

4 3

【输出样例】

6

11

18

【数据范围】

数据保证所有的n之和不超过2 × 10^5。

题解

如果将题目中的"基环树"改成一棵树，答案即为 C(n, 2)；而对于这道题，两点间的简单路径（以下简称路径）可以有 2 种：

1. 如果两点属于同一个树状结构，那么两点间有且仅有 1 条路径；
2. 如果两点属于不同的树状结构，存在 2 条路径。

因此，我们可以先计算出每两点之间有 2 条路径的情况，再减去每个树状结构内的 1 条路径，公式如下：

answer = C(n, 2) - Σ(i=1 to k) C(ti, 2)

代码

```cpp
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
using namespace std;
const int maxn=4e5+5;
int size[maxn]={};
int in[maxn]={};
vector<int> edge[maxn]={};
//dfs求子树大小
void dfs(int x,int f){
        size[x]=1;
        for(auto i:edge[x]){
                if(i==f||in[i]>1)//跳过根节点和环上的边
                        continue;
                dfs(i,x);
                size[x]+=size[i];
```

```
        }
}
int main(){
        int t;cin>>t;
        while(t--){
                int n;cin>>n;
                for(int i=1;i<=n;i++){
                        in[i]=0;size[i]=0;edge[i].clear();
                }
                for(int i=1;i<=n;i++){
                        int x, y;cin>>x>>y;
                        edge[x].push_back(y);
                        edge[y].push_back(x);
                        in[x]++;
                        in[y]++;
                }
                queue<int>q;//拓扑排序判环
                for(int i=1;i<=n;i++){
                        if(in[i]==1)q.push(i);
                };
                while(!q.empty()){
                        int x=q.front();
                        q.pop();
                        for(auto y:edge[x]){
                                --in[y];
                                if(in[y]==1){
                                        q.push(y);
                                }
                        }
                }
                //最后如果in数组中入度仍大于1则说明该节点为子树根节点
                        vector<int>a;
                //一棵无向无环树的简单路径数
                        long long ans=(long long)1*n*(n-1)/2;
            //求各个子树大小
                        for(int i=1;i<=n;i++){
                                if(in[i]>1){
                                        dfs(i,-1);
                                        a.push_back(size[i]);
                                }
                        }
                //加上该子树经过环的另一边到其他不同子树所产生的额外的简单路径数
                        for(auto i:a){
                                ans+=(long long)1*i*(n-i);
                                n-=i;
                        }
                        cout<<ans<<endl;
```

```
65                  }
66          return 0;
67          }
```

2. 飞行路线

【问题描述】

Alice 和 Bob 现在要乘飞机旅行，他们选择了一家相对便宜的航空公司。该航空公司一共在n个城市设有业务，设这些城市分别标记为0到n-1，一共有m种航线，每种航线连接两个城市，并且航线有一定的价格。

Alice 和 Bob 现在要从一个城市沿着航线到达另一个城市，途中可以进行转机。航空公司对他们这次旅行也推出优惠，他们可以免费在最多k种航线上搭乘飞机。那么 Alice 和 Bob 这次出行最少花费多少？

【链接】https://www.luogu.com.cn/problem/P4568

【样例说明】

对于30%的数据，2≤n≤50，1≤m≤300，k=0。

对于50%的数据，2≤n≤600，1≤m≤6×10^3，0≤k≤1。

对于100%的数据，2≤n≤10^4，1≤m≤5×10^4，0≤k≤10，0≤s,t,a,b≤n，a!=b，0≤c≤10^3。

【输入格式】

第一行三个整数n,m,k分别表示城市数，航线数和免费乘坐次数。

接下来一行两个整数s,t分别表示他们出行的起点城市编号和终点城市编号。

接下来m行，每行三个整数a,b,c表示存在一种航线，能从城市a到达城市b，或从城市b到达城市a，价格为c。

【输出格式】

输出一行一个整数，为最少花费

【样例输入】

5 6 1

0 4

0 1 5

1 2 5

2 3 5

3 4 5

2 3 3

0 2 100

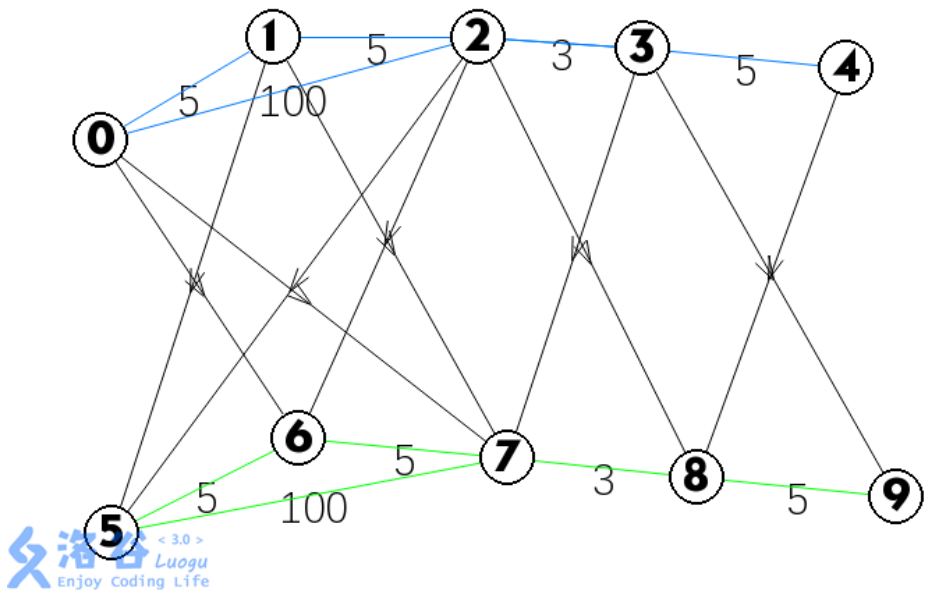
【样例输出】

8

题解

如果没有免费的额度，就直接跑最短路

对于k次免费，可以建立k层的分层图，每条边从上层连到下层，并且费用是0



代码

```
#include <bits/stdc++.h>using namespace std;
const int maxn=1e4*12+1,inf=0x3f3f3f3f;
int n,m,k,s,t,a,b,c,d[maxn],ans;
vector<pair<int,int> >e[maxn];

inline void add_edge(int f,int t,int v){
    e[f].push_back(make_pair(t,v));
    e[t].push_back(make_pair(f,v));
    e[f].push_back(make_pair(t+n,0));
    e[t].push_back(make_pair(f+n,0));
}

inline void dijkstra(int s){
    memset(d,inf,sizeof(d));
    priority_queue<pair<int,int> >q;
    d[s]=0;
    q.push(make_pair(-d[s],s));
```

```
    while(!q.empty())
    {
        int now=q.top().second;
        q.pop();
        for(int i=0;i<e[now].size();i++)
        {
            int v=e[now][i].first;
            if(d[v]>d[now]+e[now][i].second)
            {
                d[v]=d[now]+e[now][i].second;
                q.push(make_pair(-d[v],v));
            }
        }
    }
}

int main(){
    ios::sync_with_stdio(0);
    cin>>n>>m>>k>>s>>t;
    s++;
    t++;
    for(int i=1;i<=m;i++)
    {
        cin>>a>>b>>c;
        a++;
        b++;
        for(int j=1;j<=k;j++)
        {
            add_edge(a+n*(j-1),b+n*(j-1),c);
        }
        add_edge(a+n*k,b+n*k,c);
        add_edge(b+n*k,a+n*k,c);
    }
    dijkstra(s);
    ans=d[t];
    for(int i=1;i<=k;i++)
    {
        ans=min(ans,d[t+n*i]);
    }
    cout<<ans<<endl;
    return 0;
}
```

3. Jzzhu and Cities

【问题描述】

给定n个点，m条带权边的无向图，以及k条特殊边，每条特殊边连接点1和i。要求找出最多可以删除这k条特殊边中的多少条，使得每个点到点1的最短距离不变。

【链接】

https://www.luogu.com.cn/problem/CF449B

【输入格式】

第一行包含三个整数n、m、k，表示点的个数、带权边的数量和特殊边的数量。

接下来m行，每行包含三个整数u、v、x，表示带权边连接u和v，权值为x。

接下来k行，每行包含两个整数s、y，表示特殊边连接1和s，权值为y。

【输出格式】

输出一个整数，表示最多可以删除的特殊边的数量，使得每个点到点1的最短距离不变。

【输入样例】

5 5 3

1 2 1

2 3 2

1 3 3

3 4 4

1 5 5

3 5

4 5

5 5

【输出样例】

2

【数据范围】

约束条件：

1 ⩽ n, m, k ⩽ 10^5

1 ⩽ u, v, s ⩽ n

1 ⩽ x, y ⩽ 10^9

题解

要判断最短路是否唯一，可以按照以下方法进行：

1. 如果特殊边的长度大于 dis[v]，则将该边删除。
2. 如果特殊边的长度等于 dis[v]，同时到达顶点 v 的最短路不止一条，那么删除该特殊边，并将最短路数量减少 1。
3. 在求最短路的过程中，及时更新顶点 v 的最短路数量。

代码

```
#include<iostream>#include<string.h> #include<queue>#define pa pair<int,int>#def

int n,m,k,S=1,m0,ans;
int dis[N],h[N],a[N][3],cnt[N];
bool vis[N];
struct edge{int next,to,w;
}e[M];
priority_queue< pa,vector<pa>,greater<pa> > q;

void ins(int u,int v,int w){
        e[++m0]=(edge){h[u],v,w};h[u]=m0;
}

void dijkstra(int s){
        memset(dis,0x3f,sizeof(dis));
        dis[s]=0,q.push(make_pair(0,s));

        while(!q.empty()){
                int u=q.top().second; q.pop();
                if(vis[u]) continue; vis[u]=1;
                for(int i=h[u];i;i=e[i].next){
                        int v=e[i].to;
                        if(dis[v]==dis[u]+e[i].w) cnt[v]++;
                        if(dis[v]>dis[u]+e[i].w){
                                dis[v]=dis[u]+e[i].w;
                                cnt[v]=1;
                                q.push(make_pair(dis[v],v));
                        }
                }
        }

}

int main(){
        int i,u,v,w;
        cin>>n>>m>>k;
        for(i=1;i<=m;i++) cin>>u>>v>>w,ins(u,v,w),ins(v,u,w);
```

```
38          for(i=1;i<=k;i++){
39                  cin>>a[i][1]>>a[i][2];
40                  v=a[i][1],w=a[i][2];
41                  ins(1,v,w),ins(v,1,w);
42          }
43
44          dijkstra(S);
45          for(i=1;i<=k;i++){
46                  v=a[i][1],w=a[i][2];
47                  if(dis[v]<w) ans++;
48                  if(dis[v]==w)
49                          if(cnt[v]>1) ans++,cnt[v]--;
50          }
51
52          cout<<ans<<endl;
53          return 0;
54  }
```

4. 恢复道路网

【问题陈述】

弗洛伊德算法可以从一张图的基础上计算任意两点之间的最短路。给定一个弗洛伊德求出的最短路矩阵，问是否存在一张图，使得应用弗洛伊德算法后可以得到这样的最短路矩阵。如果不存在输出-1。如果存在，输出所有满足要求的图中总边权最小的一张图的总边权和。

【约束】

1≤图的点数≤300

【输入】

输入从标准输入中按以下格式给出：

N

A1,1 A1,2 … A1,N

A2,1 A2,2 … A2,N

…

AN,1 AN,2 … AN,N

【输出】

如果不存在满足条件的网络，请打印 。如果存在，请打印尽可能短的道路总长度。-1

【链接】https://www.luogu.com.cn/problem/AT_arc083_b

【示例输入1】

3

0 1 3

1 0 2

3 2 0

【示例输出1】

3

【示例输入 2】

3

0 1 3

1 0 1

3 1 0

【示例输出 2 】

-1

【示例输入 3】

5

0 21 18 11 28

21 0 13 10 26

18 13 0 23 13

11 10 23 0 17

28 26 13 17 0

【示例输出 3】

82

【示例输入 4】

3

0 1000000000 1000000000

1000000000 0 1000000000

1000000000 1000000000 0

【示例输出 4】

3000000000

题解

题意：给出通过弗洛伊德得到的两两最短路的矩阵，问是否存在一个图通过floyd可以得到这个矩阵，输出原图最小的边权和。

我们考虑什么时候这个不存在原图，唯一的可能就是这个图自相矛盾，也就是存在点i,j,k,使得dis[i][k]+dis[k][j]<dis[i][j]。这样就无解，输出-1，否则一定有解。

接下来考虑最优策略，如果存在三个点i,j,k，使得dis[i][k]+dis[k][j]=dis[i][j]，那么我们不需要在ij之间连边，也可以得到i，j的最短路。而如果不存在，我们就只能连一条i，j，权值为dis[i][j]的边。

这样最后统计一下必须加入的边的权值和即可。

代码

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<cmath>
#include<set>
#include<vector>
#include<ctime>
#define ll long long
#define pr(x) cerr<<#x<<"="<<x<<endl
using namespace std;
ll n,k,ans,a[500][500],b[500][500],i,j;
int main()
{
 scanf("%lld",&n);
 for (i=1;i<=n;i++)
  for (j=1;j<=n;j++)
                {
   scanf("%lld",&a[i][j]);
                }
 for (k=1;k<=n;k++)
  for(i=1;i<=n;i++)
   for (j=1;j<=n;j++)
   if (k!=i&&j!=k&&i!=j)
                        {
    if (a[i][k]+a[j][k]<a[i][j])
                                {
     printf("-1\n");
     return 0;
                                }
```

```
31      if (a[i][k]+a[j][k]==a[i][j]) b[i][j]=b[j][i]=1;
32                          }
33   for(i=1;i<=n;i++)
34    for (j=1;j<=n;j++)
35                  {
36     if (b[i][j]==0) ans+=a[i][j];
37                  }
38          cout<<ans/2;
39  }
```

5. 杀人游戏

【问题描述】

一位冷血的杀手潜入 Na-wiat，并假装成平民。警察希望能在 N 个人里面，查出谁是杀手。警察能够对每一个人进行查证，假如查证的对象是平民，他会告诉警察，他认识的人， 谁是杀手， 谁是平民。假如查证的对象是杀手， 杀手将会把警察干掉。现在警察掌握了每一个人认识谁。每一个人都有可能是杀手，可看作他们是杀手的概率是相同的。问：警察在最优查证策略下，保证自身安全并知道谁是杀手的概率是多少?

【输入格式】

第一行有两个整数 N,M。

接下来有 M 行，每行两个整数 x,y，表示 x 认识 y（y 不一定认识 x

【输出格式】

仅包含一行一个实数，保留小数点后面 6 位，表示最大概率。

【链接】https://www.luogu.com.cn/problem/P4819

【样例输入】

5 4

1 2

1 3

1 4

1 5

【样例输出】

0.800000

【样例说明】

警察只需要查证 1。假如1是杀手，警察就会被杀。假如 1不是杀手，他会告诉警察 2,3,4,5 谁是杀手。而 1 是杀手的概率是 0.2,所以能知道谁是杀手但没被杀的概率是0.8。对于 100%的数据有

$1 \leqslant N \leqslant 10\,0000, 0 \leqslant M \leqslant 30\,0000$

题解

这道题目的一个难点在于如何计算这个概率。

我们假设，有5个村民谁也不认识谁，其中有一个是凶手，这时警察必须抽中四个人作为平民，也就是获胜概率为0.2

也就是说，失败的概率是我们至少要尝试几次才能找到凶手除以总人数，那么答案就是1减失败的概率。

所以我们要求出至少要问几个人才能知道谁是凶手。

考虑强连通分量，一个强连通分量只要问一个人就可以知道整个连通分量的情况了。

所以求出scc缩点，这样形成了DAG，然后找到DAG上有几个入度为0的点，这些点必须由警察去询问一次。

但是还有一种情况，就是最后还剩下一个入度为0且大小为1的联通块，并且这个联通块在DAG上的所有指向的点的情况都是已知的，那么这个点可以直接判断是不是凶手，注意要特判这种情况。

代码

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<cmath>
#include<set>
#include<vector>
#include<ctime>
#define ll long long
#define pr(x) cerr<<#x<<"="<<x<<endl
using namespace std;
#define N 310000
int size1,size,dfn[N],low[N],in[N],be[N],cnt,num[N],g[N],g1[N];
int st[N],top,scc,du[N],n,m,i,p,o,ans;
struct node
{
 int from,to,next;
}e[N],e1[N];
void add(int o,int p)
{
        e[++size].to=p;
```

```
        e[size].from=o;
        e[size].next=g[o];
        g[o]=size;
}
void add1(int o,int p)
{
        e1[++size1].to=p;
        e1[size1].from=o;
        e1[size1].next=g1[o];
        g1[o]=size1;
}
void tarjan(int x)
{
        in[x]=1;
        st[++top]=x;
        dfn[x]=low[x]=++cnt;
 for (int k=g[x];k;k=e[k].next)
        {
  int y=e[k].to;
  if (!dfn[y])
                {
   tarjan(y);
                        low[x]=min(low[x],low[y]);
                }
  else if (in[y]) low[x]=min(low[x],dfn[y]);
        }
 if (dfn[x]==low[x])
        {
                scc++;
  while (st[top+1]!=x)
                {
   int y=st[top--];
                        be[y]=scc;
                        num[scc]++;
                        in[y]=0;
                }
        }
}
bool dfs(int x)
{
 for (int k=g1[x];k;k=e1[k].next)
        {
  if (du[e1[k].to]==1) return false;
        }
 return true;
}
int main()
```

```
{
 scanf("%d %d",&n,&m);
 for (i=1;i<=m;i++)
        {
  scanf("%d %d",&o,&p);
  add(o,p);
        }
 for (i=1;i<=n;i++)
        {
  if (!dfn[i]) tarjan(i);
        }
 for (i=1;i<=size;i++)
        {
  if (be[e[i].from]!=be[e[i].to])
                du[be[e[i].to]]++,add1(be[e[i].from],be[e[i].to]);
        }
 //for (i=1;i<=n;i++) printf("be[%d]=%d\n",i,be[i]);
 for (i=1;i<=scc;i++)if (du[i]==0) ans++;
 //pr(ans);
 for (i=1;i<=scc;i++)
        {
  if (du[i]==0&&num[i]==1) if (dfs(i))
                {
                        ans--;
   break;
                }
        }
 //pr(ans);
 printf("%.6lf\n",((double)(n-ans))/(double)n);
}
```

6. 小倍数

【问题陈述】

给出一个数k，求一个k的倍数，使得这个数的各位数字之和最小，求这个和。

【链接】https://atcoder.jp/contests/arc084/tasks/arc084_b

【输入】

一个数K，在1e5内

【示例输入 1】

6

【示例输出 1】

3

12=6×2产生最小的总和。

【示例输入 3】

79992

【示例输出 3】

36

题解

既然是k的倍数，那么我们可以在模k的剩余系下考虑。

考虑建图，如果得到了x，那么在x的基础上各位数字之和加1就可以得到x+1,得到x可以用0的代价得到x*10.

但是有特例，比如从9变到10并不是加1而是减8.但是这个我们不需要考虑，因为从1到10有一条代价为0的路径，最短路一定是这一条。

所以对每个数建图，每个数可以向x+1连一条长1的边，向x*10连长为0的边。

这样转移就完整了。跑出最短路即可。但是从0到0的最短路显然是0.

我们要强制0要出去走一圈。而0只有一个出边1，所以从1开始计算最短路，答案再加上1即可。

代码

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<cmath>
#include<set>
#include<vector>
#include<ctime>
#define ll long long
#define pr(x) cerr<<#x<<"="<<x<<endl
#define N 100000
using namespace std;
struct node
{
 int to,next,v;
}e[310000];
int q[310000],in[310000],head,tail,g[310000],size,dis[310000],k,i;
void add(int o,int p,int q)
```

```
{
        e[++size].to=p;
        e[size].next=g[o];
        g[o]=size;
        e[size].v=q;
}
void bfs()
{
        q[++tail]=1;
        in[1]=1;
 while (head!=tail)
        {
  int x=q[++head];
  if (head==N) head=0;
                in[x]=0;
  for (int k=g[x];k;k=e[k].next)
                {
   int y=e[k].to;
   if (dis[y]>dis[x]+e[k].v)
                        {
                                dis[y]=dis[x]+e[k].v;
    if (!in[y])
                                {
                                        q[++tail]=y;
     if (tail==N) tail=0;
                                        in[y]=1;
                                }
                        }
                }
        }
}
int main()
{
 ///freopen("a.in","r",stdin);
 //freopen("a.out","w",stdout);
 scanf("%d",&k);
 memset(dis,0x7f,sizeof(dis));
        dis[1]=0;
 for (i=0;i<=k;i++)
        {
  add(i,(i+1)%k,1);
  if (i>0)add(i,i*10%k,0);
        }
 bfs();
 printf("%d\n",1+dis[0]);
 return 0;
}
```

7. Connected Components

【问题描述】

一个无向图，给出m条边，有k次询问，每次询问将第li到ri条边暂时删去，求这时候有多少个连通块.
N<=500，1<=M,K<=10000

【输入】

第一行输入n，m

接下来m行每行输入一条边

接下来一行表示k

接下来k行每行一个l和r

【输出】

对于每个询问，输出联通块的数量

【链接】https://www.luogu.com.cn/problem/CF292D

【输入】

6 5

1 2

5 4

2 3

3 1

3 6

6

1 3

2 5

1 5

5 5

2 4

3 3

【输出】

4

5

6

3

4

2

题解

可以发现，最终都可以转化为一个前缀和一个后缀的并查集。

一个并查集的大小为500，足够对每个前缀和后缀都开一个并查集。

但是问题在于如何合并两个并查集。

注意并查集保留的是联通性信息，而对于一张图，只要把并查集中所有的边都连上就可以保证连通性被表达完全了。

于是可以将第二个并查集每个点和父亲这条边加入第一个并查集。这样就可以在O（n）的时间完成一次查询。

代码

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<cmath>
#include<map>
#include<vector>
#include<ctime>
#define ll long long
#define pr(a) cerr<<#a<<"="<<a<<endl
#define pri(a,lo) {cerr<<#a<<"={";for (int ol=0;ol<=lo;ol++)cerr<<a[ol]<<",";cer
using namespace std;
int n,i,o,p,k,m;
pair<int,int> e[11000];
struct uf
{
 int fa[510],num;
 void clear(){num=n;for (int i=1;i<=n;i++) fa[i]=i;}
 int find(int x){if (fa[x]==x) return x;fa[x]=find(fa[x]);return fa[x];}
 void merge(int x,int y)
 {
  int l=find(x),r=find(y);
  if (l!=r) num--,fa[l]=r;
```

```
        }
}l[11000],r[11000];
uf add(uf x,uf y)
{
 for (int i=1;i<=n;i++) if (y.fa[i]!=i) x.merge(i,y.find(i));
 return x;
}
int main()
{
 scanf("%d %d",&n,&m);
 for (i=1;i<=m;i++) scanf("%d %d",&o,&p),e[i]=make_pair(o,p);
        l[0].clear(),r[m+1].clear();
 for (i=1;i<=m;i++) l[i]=l[i-1],l[i].merge(e[i].first,e[i].second);
 for (i=m;i>=1;i--) r[i]=r[i+1],r[i].merge(e[i].first,e[i].second);
 scanf("%d",&k);
 while (k--) {scanf("%d %d",&o,&p);printf("%d\n",add(l[o-1],r[p+1]).num);}
}
```