ZJNU-2020暑期专题5 二分图&网络流 题解

代码什么的网上都可以找到，这里放题解。

后面一些题目（尤其是网络流），可能有的人不理解这个为什么这样建图就可以，可以自行画个图手模一下，慢慢体会，这种题不理解就永远不会（虽然理解了也不一定想的到，但是想到了！就很爽！）。

A HDU 6582 Path

题意

有向图，n个点，m条边，求去掉最短路的最小代价。去掉每条路代价等于长度。 问最少花费多少，能使得从1到n的最短路长度至少增加1（不连通也算答案）。输出最少花费。

做法

总体思路

由于从1到n的最短路可能存在多条，所以需要先把所有最短路上的边拎出来建一张图G； 然后在图G上跑最小割，用最小的花费把源点1和汇点n割开，割开之后原图中所有原有最短路必然不联通，此时1 ~ n的最短路长度必然增加； 因为“最小割=最大流”定理，所以只需要在图G上跑最大流算法即可。

建图 这相当于要找出从1到n的所有最短路。 如果在dijkstra中用vector记录每个节点的多个前驱节点（或边），有可能会MLE； 怎么办呢？ 观察每一条边，如果其存在于从1到n的最短路上，那么满足式子： dist[u]+cost[i]+dist2[v]==dist[n]， 其中dist[j]是从1到j的最短路，cost[i]是第i条边的权值，dist2[j]是从j到n的最短路； dist2是通过反向建图跑dijkstra得到的。

B HDU 5093 Battle ships

题意

给出一个n行m列的图，*代表海域，o代表冰水，#代表冰山，要想在海域中放置船，保证船与船之间不能相互看到，之间只要有山就不能看到，问最多能放多少船。

做法

二分图和网络流均可，这里是二分图做法

主要是将行与列分开来考虑，每一座冰山隔开的海水，都可以看成一个“块” 在这上面是可以放船的，连续的海水可能中间会有冰，但不影响看成一个块

列也是同样处理的，每一个 列与行的交点的海水，说明这个行与列是矛盾的，只能存在一个 就将它们连边，最后就是去求二分图的最大匹配

和L题的行列建图其实是很像的，但是这里在碰到了“山”之后就等于重新多了一个新的行（因为这一块还能重新匹配），列也是同理。

```
#***
*#**
**#*
ooo#
```

假如一个这样的图，正常情况下我们第一行的行标都是1，第二行的行标都是2...但是由于有山的存在，我们可以把行看成如下

```
#111
2#33
44#5
ooo#
```

同理把列看成如下

```
#245
1#45
13#5
ooo#
```

在有水*的地方，我们把对应的行列连接，跑最大匹配即可。（你品！你细品！）

C HDU 4160

题意

大娃娃可以套在小娃娃外面（各边严格小），问最后最少得到几个娃娃

做法

题目中的娃娃可以看做点，嵌套关系可以看做有向的路径，这样发现这题就是一个裸的最小路径覆盖问题

最小路径覆盖，如果满足条件：

$$w_i \leq w_j, l_i \leq l_j, h_i \leq h_j$$

那么i->j连边，然后就是求最大匹配。我相信你明白的！

D HDU 3605

题意

一个星球上可以住若干个人，给出每个人想要选择的星球，然后给出每个星球最多容纳的人数。

做法

网络流和二分图的多重匹配均可

如果不管数据范围，这道题的网络流建模还是比较好想的。

从源点出发连上所有的人，容量为1，从每一个人连边到其所有能适应的星球，容量为1，再从星球连边到汇点，容量为星球能容纳的人数上限。跑最大流后看看最大流是否与人数相等，如果相等则存在可行解，否则无解。 但要注意到本题的数据范围。人数有100000，这样直接跑最大流是会出问题的。再看一看m的数据范围，诶.....只有10。于是我们可以想到一定有非常多的人，他们对所有星球的适应情况是一样的。所以我们可以用位运算的方式，把每一个人对星球的适应情况压缩在一个整数里面，然后统

计这个数相同的人的人数，相当于把这些人都看作一个人。然后连边的时候就不是连所有的人，而是连不同的类型，而容量就是这种类型的人的个数，其他建图不变。

E HDU 2835

题意

给定一个二分图，N个点对应M个点，两两之间存在一组关系，每组关系一个权值。题目中了给定了一个匹配方案，现在要求满足这组关系中的最大的匹配权值在原方案上增长了多少？并且还要求出在原匹配方案上改**最少**多少条边才能够得到这个最大匹配？

做法

描述起来有点复杂，给链接

链接：https://blog.csdn.net/u013480600/article/details/38736501

F HDU 6437

题意

给出m个视频，有两种类型，每个视频有一个开始时间和结束时间，每部视频最多给一个人看，观看后得到wi的快乐值，一次只能看一部视频。 一个人连续观看两部相同类型的视频会有一个 -W 快乐值得效果。问现在有k个人，能获得的快乐值总和最多是多少。

做法

费用流，将人看成流量，路径模拟观看视频的过程。因为要求最大的快乐值，所以我们要把费用变成负的跑最小费用（这样得到的答案*(-1)就是最大费用了） 每个视频拆点，连一条流量为1，费用为-w的边，表示看了这部电影会有-w的快乐值；超级源点到次源点连一条流量为k，费用为0的边，表示一共有k个人会看视频； 次源点向每个视频连一条流量为1，费用为0的边，表示第一次跑去看电影的情况；视频之间如果不交叉，相同类型连一条流量为1，费用为W的边，不同类型连一条流量为1，费用为0的边。

最大费用最大流就是建立负边

G HDU 6611

题意

给你一个序列，你要在序列中寻找k个非下降子序列，使得所有子序列的和最大。

做法

最大上升子序列一直是可以用最小费用最大流的思路解的，一个点a[i]与它之后所有大于它的点建边，然后把它自己拆点建边，边流量为1，费用为a[i]，然后起点与所有点建边，终点与所有点建边，跑一个固定费用为k的费用流就是答案了。

但是要用Dijkstra优化！！！

H Gym 101512A

题意

简化的说：有n个城市（分别从1-n进行标记），在第i个城市里有g个人，现在发生了一场大地震，n个城市中有m个城市是安全的（接下来会给你m个安全的城市的坐标），这些人要从i城市中向其他城市跑，只有s秒的逃跑时间。接下来会给你r条路，每条路的信息有a b p t 表示从a到b城市有一条单向边，每秒钟可以有p个人进入这条路，通过这条路要花费t秒。问s秒之后最多有多少人成功到达安全地点。

做法

要将城市对时间进行拆点，每个城市的每一秒都是一个不同的点，假设a到b需要t秒通过p个人，那么从第a个城市的第0秒到第b个城市的第t秒可以连接一条流量为p的边，从第a个城市的第1秒到第b个城市的第t+1秒可以连接一条流量为p的边....以此类推。

同时，城市a从第x秒到城市a的第x+1秒可以连一条流量为无穷大的边，表示这些人也可以选择呆在这个城市，然后将第s秒的目标城市与超级汇点相连接，流量为无穷大，跑一遍dinic就可以得出答案来了。

```
#include<bits/stdc++.h>
using namespace std;
const int Ni = 2000005;
const int MAX = 1<<26;
struct Edge{
    int u,v,c;
    int next;
}edge[3*Ni];
int n,m;
int edn;//边数
int p[Ni];//父亲
int d[Ni];
int sp,tp;//原点，汇点

void addedge(int u,int v,int c){
    edge[edn].u=u; edge[edn].v=v; edge[edn].c=c;
    edge[edn].next=p[u]; p[u]=edn++;

    edge[edn].u=v; edge[edn].v=u; edge[edn].c=0;
    edge[edn].next=p[v]; p[v]=edn++;
}
int bfs(){
    queue <int> q;
    memset(d,-1,sizeof(d));
    d[sp]=0;
    q.push(sp);
    while(!q.empty()){
        int cur=q.front();
        q.pop();
        for(int i=p[cur];i!=-1;i=edge[i].next){
            int u=edge[i].v;
            if(d[u]==-1 && edge[i].c>0){
                d[u]=d[cur]+1;
                q.push(u);
            }
        }
    }
    return d[tp] != -1;
}
int dfs(int a,int b){
    int r=0;
    if(a==tp)return b;
    for(int i=p[a];i!=-1 && r<b;i=edge[i].next)
```

```
    {
        int u=edge[i].v;
        if(edge[i].c>0 && d[u]==d[a]+1)
        {
            int x=min(edge[i].c,b-r);
            x=dfs(u,x);
            r+=x;
            edge[i].c-=x;
            edge[i^1].c+=x;
        }
    }
    if(!r)d[a]=-2;
    return r;
}

int dinic(int sp,int tp){
    int total=0,t;
    while(bfs()){
        while(t=dfs(sp,MAX))
        total+=t;
    }
    return total;
}
int allti;
int gainp(int x,int thattime){
    return (x-1)*(allti+1)+thattime+1;
}
int main(){
    int i,u,v,c,x,cas;
    cin>>cas;
    while(cas--){
        edn=0;
        memset(p,-1,sizeof(p));
        int st,allflow;
        scanf("%d%d%d%d",&n,&st,&allflow,&allti);
        sp=0,tp=n*(allti+1)+1;
        addedge(0,gainp(st,0),allflow);
        scanf("%d",&m);
        for(int i=0;i<m;i++){
            scanf("%d",&x);
            addedge(gainp(x,allti),tp,MAX);
        }
        int r,fr,to,peo,ti;
        scanf("%d",&r);
        while(r--){
            scanf("%d%d%d%d",&fr,&to,&peo,&ti);
            for(int i=0;i<=allti-ti;i++){
                addedge(gainp(fr,i),gainp(to,i+ti),peo);
            }
        }
        for(int i=1;i<=n;i++){
            for(int j=0;j<allti;j++){
                addedge(gainp(i,j),gainp(i,j+1),MAX);
            }
        }
        printf("%d\n",dinic(sp,tp));
    }
    return 0;
```

```
}
```

I SPOJ NETADMIN

题意

给出n户人家，m条街道，只有第一户人家能够上网，其他家上网需要从第一家拉一根网线，一根网线只能给目标人家上限，每根网线只有一种颜色， 要求每条街道不同人家的网线颜色不同，求满足指定的k户人家都能够上网的最少颜色数量。

做法

题目比较拗口，有点不好理解。

可以理解为从1号点出发，连带颜色的网线到每个点，同一颜色的线在路径上不能重合，就像是网络流从1号出发，每个点都要和汇点连一个1单位流量的边一样。

下面是另一种理解，可以看成求1到每个点的路径经过边次数的最大值的最小值。

流量的上限其实就是网络流中需要的最大流量，我们可以二分这个答案，对于每次check，建边跑最大流即可。

有向图中是add(u,v,c)

无向图中 假设u到v的流量是x，v到u的流量是y，那就相当于u到v流了x+c-x=c

所以add(u,v,c) add(v,u,c)

```
#include<cstdio>
#include<cstring>
#include<queue>
#include<cmath>
#include<iostream>
using namespace std;
const int Ni = 550;
const int MAX = 1<<26;
struct Edge{
    int u,v,c;
    int next;
}edge[20*Ni];
int n,m,k;
int edn;//边数
int p[Ni];//父亲
int d[Ni];
int sp,tp;//原点，汇点
int in[Ni*Ni],out[Ni*Ni];
int a[Ni];
void addedge(int u,int v,int c)
{
    edge[edn].u=u; edge[edn].v=v; edge[edn].c=c;
    edge[edn].next=p[u]; p[u]=edn++;

    edge[edn].u=v; edge[edn].v=u; edge[edn].c=0;
    edge[edn].next=p[v]; p[v]=edn++;
}
int bfs()
```

```
{
    queue <int> q;
    memset(d,-1,sizeof(d));
    d[sp]=0;
    q.push(sp);
    while(!q.empty())
    {
        int cur=q.front();
        q.pop();
        for(int i=p[cur];i!=-1;i=edge[i].next)
        {
            int u=edge[i].v;
            if(d[u]==-1 && edge[i].c>0)
            {
                d[u]=d[cur]+1;
                q.push(u);
            }
        }
    }
    return d[tp] != -1;
}
int dfs(int a,int b)
{
    int r=0;
    if(a==tp)return b;
    for(int i=p[a];i!=-1 && r<b;i=edge[i].next)
    {
        int u=edge[i].v;
        if(edge[i].c>0 && d[u]==d[a]+1)
        {
            int x=min(edge[i].c,b-r);
            x=dfs(u,x);
            r+=x;
            edge[i].c-=x;
            edge[i^1].c+=x;
        }
    }
    if(!r)d[a]=-2;
    return r;
}

int dinic(int sp,int tp)
{
    int total=0,t;
    while(bfs())
    {
        while(t=dfs(sp,MAX))
        total+=t;
    }
    return total;
}
int ck(int mid){
    edn=0;
    memset(p,-1,sizeof(p));
    for(int i=1;i<=m;i++){
        addedge(out[i],in[i],mid);
        addedge(in[i],out[i],mid);
    }
```

```
    for(int i=1;i<=k;i++){
        addedge(a[i],tp,1);
    }
    int ans=dinic(sp,tp);
    return ans>=k;
}
int main()
{
    int i,u,v,c,t,x;
    cin>>t;

    while(t--){
        scanf("%d%d%d",&n,&m,&k);
        sp=1;tp=501;
        for(int i=1;i<=k;i++){
            scanf("%d",&a[i]);
        }
        for(int i=1;i<=m;i++){
            scanf("%d%d",&out[i],&in[i]);
        }
        int l=0,r=m+1,ans;
        while(l<=r){
            int mid=(l+r)/2;
            if(ck(mid)){
                r=mid-1;
                ans=mid;
            }
            else l=mid+1;
        }
        printf("%d\n",ans);
    }
    return 0;
}
```

J HDU 3468

题意

你和iSea去寻宝，路上只能沿着$A_{Z||a}z$的Rally最短路径走，路上你省下一分钟的时间去挖金子，也就是说只能挖一个金子，挖完后就不能再挖，问最多能挖多少金子。（输出保证是A~Z || a~z，要不没法做）

做法

首先bfs找到每次会合有可能取的金币的位置，每个位置标号设为p1，每次会合的出发点设为p2，源点S，汇点T

可以建如下三类边：<u, v, w>分别表示一条边的起点，终点，边权

1. <S, p2, 1> 表示每一次出发最多拿到1枚金币
2. <p2, p1, 1> 表示每次出发可以拿到最短路径上的任意一个金币
3. <p1, T, 1> 表示从每个有金币的位置最多拿走一枚金币

如此建图，就能够保证流量最大时，iSea拿到的金币最多，当然，符合规则。

这题还有一关键点：如何bfs求得最短路经过的所有点。其实也比较简单，直接bfs，bfs过程中给每个位置赋值最短距离，搜到目标位置，跳出，然后反向dfs，按照距离差值为1递减的顺序，所搜到的所有点，都是最短路径上的点。

这道题也可以用二分图最大匹配来做。相对于上面的建图，去掉源点S，汇点T，即可。求得p2构成的X集合和p1构成的Y集合，题目上，每个出发点最多拿到一枚金币，求取最大金币数，正好符合二分图最大匹配的定义。

K题到N题都是简单题，感觉应该都会的就不放题解了，网上很多的自己看叭。