网络流习题整理

1.最大流(无费用只需要让流量最大)

因为之前写过两个例题就直接放链接了。
Dinic实现的最大流
时间复杂度:$O(n^2m)$
EK实现的最大流 时间复杂度：$O(nm^2)$

#6004.「网络流 24 题」圆桌聚餐

思路：以单位为集合A,桌子为集合B，源点连容量为A人数的边，B与汇点连容量为桌子容量的边。

然后跑最大流，即可，输出方案就顺序从1到m,跑一遍流量为0的路径。

注意的时候：边的空间大小要开N^2+100，然后其他要开$2N$。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=505,M=N*N+10,mod=1e9+7,inf=0x3f3f3f3f;
#define mst(a) memset(a,0,sizeof a)
#define fmst(a) memset(a,-1,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
int n,m,s,t,h[N],cur[N],cnt=1,dep[N];
struct node{
    int to,nt,w;
}e[M];
void add(int u,int v,int w){
    e[++cnt]={v,h[u],w},h[u]=cnt;
    //if(cnt>M) cout<<cnt<<endl;
}
void Add(int u,int v,int w){
    add(u,v,w),add(v,u,0);
}
queue<int>q;
bool bfs(){ //给图分层.
    mst(dep);
    dep[s]=1;q.push(s);
    while(!q.empty()){
        int u=q.front();q.pop();
        cur[u]=h[u];    //每次初始化cur[]
        for(int i=h[u];i;i=e[i].nt){
            int v=e[i].to,w=e[i].w;
            if(w&&!dep[v])
                dep[v]=dep[u]+1,q.push(v);
        }
    }
```

```
    return dep[t];
}
int dfs(int u,int flow){
    if(u==t)return flow;
    int res=flow;
    for(int &i=cur[u];i;i=e[i].nt){ //优化3.
        int v=e[i].to,w=e[i].w;
        if(w&&dep[u]+1==dep[v]){
            int now=dfs(v,min(res,w));  //这里有个小细节要用res而不是flow，因为要取到当前最小,flow可能不是最小的.
            if(!now) dep[v]=1;  //优化1
            else{
                e[i].w-=now;    //建立反边.
                e[i^1].w+=now;
                res-=now;
            }
        }
        if(!res)return flow;    //优化2
    }
    //printf("flow=%d,res=%d\n",flow,res);
    return flow-res;
}
int main(){
    scanf("%d%d",&m,&n);
    s=n+m+1,t=s+1;
    int sum=0,ans=0;
    for(int i=1,x;i<=m;i++){
        scanf("%d",&x),Add(s,i,x);
        sum+=x;
    }
    for(int i=1,x;i<=n;i++){
        scanf("%d",&x);
        for(int j=1;j<=m;j++) Add(j,i+m,1);
        Add(i+m,t,x);
    }
    while(bfs())ans+=dfs(s,inf);
    if(ans!=sum) puts("0"),exit(0);
    puts("1");
    for(int i=1;i<=m;i++){
         for(int j=h[i];j;j=e[j].nt){
              if(!e[j].w&&e[j].to!=s)
                 printf("%d ",e[j].to-m);
          }
          puts("");
       }
    return 0;
}
```

#6006. 「网络流 24 题」试题库

思路：构建令A为题型组成的集合，B为题号组成的题型。

源点与A连容量为所需题型题目个数的边，B与汇点连容量为1(即题目个数贡献为1)的边。

然后跑一遍最大流。

至于输出方案，就是从题型1开始跑到k，遍历一遍流量为0的路径即可，因为走过该路后流量肯定为0。

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=6e3+5,M=1e5+5,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define fmst(a) memset(a,-1,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
int n,k,s,t,h[N],cur[N],cnt=1,dep[N];
struct node{
    int to,nt,w;
}e[N<<1];
void add(int u,int v,int w){
    e[++cnt]={v,h[u],w},h[u]=cnt;
}
void Add(int u,int v,int w){
    add(u,v,w),add(v,u,0);
}
queue<int>q;
bool bfs(){ //给图分层.
    mst(dep);
    dep[s]=1;q.push(s);
    while(!q.empty()){
        int u=q.front();q.pop();
        cur[u]=h[u];    //每次初始化cur[]
        for(int i=h[u];i;i=e[i].nt){
            int v=e[i].to,w=e[i].w;
            if(w&&!dep[v])
                dep[v]=dep[u]+1,q.push(v);
        }
    }
    return dep[t];
}
int dfs(int u,int flow){
    if(u==t)return flow;
    int res=flow;
    for(int &i=cur[u];i;i=e[i].nt){ //优化3.
        int v=e[i].to,w=e[i].w;
        if(w&&dep[u]+1==dep[v]){
            int now=dfs(v,min(res,w));  //这里有个小细节要用res而不是flow，因为要取到当前最小,flow可能不是最小的.
            if(!now) dep[v]=1;  //优化1
            else{
                e[i].w-=now;    //建立反边.
                e[i^1].w+=now;
                res-=now;
            }
        }
        if(!res)return flow;    //优化2
    }
    return flow-res;
}
int main(){
```

```
    scanf("%d%d",&k,&n);
    s=2*(n+k)+1,t=s+1;
    int sum=0,ans=0;
    for(int i=1,x;i<=k;i++){
        scanf("%d",&x),Add(s,i,x);
        sum+=x;
    }
    for(int i=1;i<=n;i++){
        int p,x;
        scanf("%d",&p);
        while(p--){
            scanf("%d",&x);
            Add(x,i+k,1);
        }
        Add(i+k,t,1);
    }
    while(bfs())ans+=dfs(s,0x7fffffff);
    if(ans!=sum) puts("No Solution!"),exit(0);
    for(int i=1;i<=k;i++){
         printf("%d:",i);
         for(int j=h[i];j;j=e[j].nt){
               if(!e[j].w&&e[j].to!=s)
                  printf(" %d",e[j].to-k);
          }
          puts("");
       }
    return 0;
}
```

2.最小费用最大流.

P3381 【模板】最小费用最大流

模板题，用EK的bfs改为跑$spfa$，时间复杂度：$O(nmf)$，f为最大流量。

具体看代码。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=1e5+5,M=2e4+5,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define fmst(a) memset(a,0x7f,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
#define pb push_back
int cnt=1,h[N],flow[N],dis[N],vis[N],mc,mf,n,m,s,t;
queue<int>q;
struct edge{
    int to,nt,f,w;//f:flow ,w:cost
}e[N];
struct Pre{
```

```
    int i,u;
}pre[N];//记录前驱结点和边的信息，便于更新残余网络，建立反边.(反悔和贪心的思想)
void add(int u,int v,int f,int w){
    e[++cnt]={v,h[u],f,w},h[u]=cnt;
}
bool spfa(){// 跑spfa
    fmst(dis),fmst(flow),mst(vis);  //初始化.
    q.push(s),vis[s]=1,dis[s]=0,pre[t].u=-1;//预处理
    while(!q.empty()){
        int u=q.front();q.pop();vis[u]=0;
        for(int i=h[u];i;i=e[i].nt){
            int v=e[i].to,f=e[i].f,w=e[i].w;
            if(f>0&&dis[v]>dis[u]+w){
                dis[v]=dis[u]+w;
                pre[v].u=u,pre[v].i=i;
                flow[v]=min(flow[u],f);
                if(!vis[v]) vis[v]=1,q.push(v);
            }
        }
    }
    return pre[t].u!=-1;
}
void MCMF(){    //MIncost Maxflow
    while(spfa()){
        int u=t,x=flow[t];
        mf+=x;
        mc+=x*dis[u];
        //printf("%d %d\n",flow[t],dis[t]);
        while(u!=s){
            e[pre[u].i].f-=x;
            e[pre[u].i^1].f+=x;
            u=pre[u].u;
        }
    }
}
int main(){
    scanf("%d%d%d%d",&n,&m,&s,&t);
    for(int i=1;i<=m;i++){
        int u,v,f,w;
        scanf("%d%d%d%d",&u,&v,&f,&w);
        add(u,v,f,w),add(v,u,0,-w);
    }
    MCMF();
    printf("%d %d\n",mf,mc);
    return 0;
}
```

LOJ的模板题，这里我优化一下pre可以只用开一个数组就行了。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=405,M=3e4+5,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define fmst(a) memset(a,0x7f,sizeof a)
#define lx x<<1
#define rx x<<1|1
```

```cpp
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
#define pb push_back
int cnt=1,h[N],flow[N],dis[N],vis[N],mc,mf,n,m,s,t;
queue<int>q;
struct edge{
    int to,nt,f,w;//f:flow ,w:cost
}e[M];
int pre[N];
void add(int u,int v,int f,int w){
    e[++cnt]={v,h[u],f,w},h[u]=cnt;
}
bool spfa(){// 跑spfa
    fmst(dis),fmst(flow),mst(vis);  //初始化.
    q.push(s),vis[s]=1,dis[s]=0,pre[t]=-1;//预处理
    while(!q.empty()){
        int u=q.front();q.pop();vis[u]=0;
        for(int i=h[u];i;i=e[i].nt){
            int v=e[i].to,f=e[i].f,w=e[i].w;
            if(f>0&&dis[v]>dis[u]+w){
                dis[v]=dis[u]+w;
                pre[v]=i;
                flow[v]=min(flow[u],f);
                if(!vis[v]) vis[v]=1,q.push(v);
            }
        }
    }
    return pre[t]!=-1;
}
void MCMF(){    //MIncost Maxflow
    while(spfa()){
        int u=t,x=flow[t];
        mf+=x;
        mc+=x*dis[u];
        //printf("%d %d\n",flow[t],dis[t]);
        while(u!=s){
            e[pre[u]].f-=x;
            e[pre[u]^1].f+=x;
            u=e[pre[u]^1].to;
        }
    }
}
int main(){
    scanf("%d%d",&n,&m);
    s=1,t=n;
    for(int i=1;i<=m;i++){
        int u,v,f,w;
        scanf("%d%d%d%d",&u,&v,&f,&w);
        add(u,v,f,w),add(v,u,0,-w);
    }
    MCMF();
    printf("%d %d\n",mf,mc);
    return 0;
}
```

牛客多校1.H.Minimum-cost Flow

前话：因为多组没有初始化cnt边的数量，导致一直超时，懵逼了，对于多组数据重建图时，只需要初始化cnt和$h[]$数组$(head[])$即可。

题意：给定(n,m)无向图，m条带权有向边(费用已知容量未知)，给q次询问，每次询问给出$u,v,(u\le v)$,问在所有边容量为$\frac{u}{v}$下，总流量为1的最小费用。

思路：$MCMF$+贪心。

设$cost(x,y)$为 在所有边容量为x下，总流量为y的最小费用。

考虑先跑一边$MCMF$，求出在所有边容量为1下的所有增广路。

则题意变为求$cost(\frac{u}{v},1)$。

因为所有边容量一样，所以$cost=\sum\limits_{增广路个数\ cnt}^{cnt}flow[end]\times dis[end]=capacity\times\sum\limits_{增广路个数\ cnt}^{cnt}dis[end]$

。

即$cost$与$capacity$成线性关系。

所以有$cost(\frac{u}{v},1)=\frac{cost(u,v)}{v}$。

且设$capacity=1$对应的最大流为$maxflow$，

则$capactiy=u$对应的最大流也是线性关系为$maxflow\times u$。

所以我们先考虑特判不成立的情况。

即$1:v=0$，分母显然不能为0.

$2:maxflow\times u<v$,表示能到达的最大流不能满足总流量为v。

接下我们考虑怎么构造答案。

我们需要构造出一个总流量为v的方案，而每条增广路的流量贡献为u。

所以我们设$v=a\times u+b$。

表示我们需要a条增广路和一条流量只需要b的增广路。

之前我们已经求出了$cost(1,maxflow)$的所有增广路，因为$MCMF$求增广路的顺序就是由最小费用到最大费用的，所以我们直接根据贪心思想取前a条，和第$a+1$条作为b。

这样$cost(u,v)=a\times\sum_{i=1}^{a}aug[i]+b\times aug[a+1]$。

$aug[i]$表示第i条增广路的费用。

则最终答案为：$\frac{cost(u,v)}{v}$，约分一下即可。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=1O5,M=2e4+5,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define fmst(a) memset(a,0x7f,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
```

```
#define fi first
#define se second
#define pb push_back
int cnt=1,h[N],flow[N],dis[N],vis[N],n,m,s,t,ans[N],tot;
queue<int>q;
struct edge{
    int to,nt,f,w;//f:flow ,w:cost
}e[N<<1];
int pre[N];
void add(int u,int v,int f,int w){
    e[++cnt]={v,h[u],f,w},h[u]=cnt;
}
bool spfa(){// 跑spfa
    fmst(dis),fmst(flow),mst(vis);  //初始化.
    q.push(s),dis[s]=0,vis[s]=1,pre[t]=-1;//预处理
    while(!q.empty()){
        int u=q.front();q.pop();vis[u]=0;
        for(int i=h[u];i;i=e[i].nt){
            int v=e[i].to,f=e[i].f,w=e[i].w;
            if(f>0&&dis[v]>dis[u]+w){
                dis[v]=dis[u]+w;
                pre[v]=i;
                flow[v]=min(flow[u],f);
                if(!vis[v]) vis[v]=1,q.push(v);
            }
        }
    }
    return pre[t]!=-1;
}
void MCMF(){    //MIncost Maxflow
    while(spfa()){
        int u=t,x=flow[t];
        ans[++tot]=dis[u];
        while(u!=s){
            e[pre[u]].f-=x;
            e[pre[u]^1].f+=x;
            u=e[pre[u]^1].to;
        }
    }
    for(int i=1;i<tot;i++) ans[i+1]+=ans[i];
}
ll gcd(ll a,ll b){
    return b==0?a:gcd(b,a%b);
}
int main(){
    while(~scanf("%d%d",&n,&m)){
    s=1,t=n,tot=0,cnt=1;
    mst(h);
    for(int i=1;i<=m;i++){
        int u,v,w;
        scanf("%d%d%d",&u,&v,&w);
        add(u,v,1,w),add(v,u,0,-w);
    }
    MCMF();
    int q;
    scanf("%d",&q);
    while(q--){
        int u,v;
```

```
        scanf("%d%d",&u,&v);
        if((!v)||1LL*tot*u<v) puts("NaN");
        else {
            int a=v/u,b=v%u;
            ll fz=1LL*ans[a]*u+1LL*b*(ans[a+1]-ans[a]);
            ll g=gcd(fz,v);
            printf("%lld/%lld\n",fz/g,v/g);
        }
    }
    }
    return 0;
}
```

3.二分图匹配。

#6000. 「网络流 24 题」搭配飞行员

思路: 二分图匹配显然可以匈牙利搞，但是网络流专题，就用网络流，显然方法就是建立一个超级源点连接第一类点，再建立一个超级汇点连接第二类点，然后建图跑EK或者$dinic$就行了。

EK写法：

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=205,M=1e6+5,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
int n,m,st,ed,pre[N],g[N][N];   //pre[i] 记录路径.
int bfs(){  //求增广路的最大流速.
    int flow[N];    //保存流速.
    mst(pre);
    queue<int>q;
    q.push(st);
    flow[st]=inf;
    while(!q.empty()){
        int u=q.front();q.pop();
        for(int i=1;i<=n+2;i++){
            if(i!=st&&g[u][i]&&!pre[i]){
                pre[i]=u;
                q.push(i);
                flow[i]=min(flow[u],g[u][i]);
            }
        }
    }
    return pre[ed]?flow[ed]:-1;
}
int EK_flow(){
    int ans=0;
    while(1){
        int flow=bfs();
        //printf("%d\n",flow);
```

```
        if(flow==-1) break;//没有增广路说明当前已经是最大流.
        int now=ed;
        while(now!=st){
            int fa=pre[now];
            g[fa][now]-=flow;
            g[now][fa]+=flow;    //重点：对残留网络建立反向路径
            now=fa;
        }
        ans+=flow;
    }
    return ans;
}
int main(){
    scanf("%d%d",&n,&m);
    st=n+1,ed=st+1;
    for(int i=1;i<=m;i++) g[st][i]=1;
    for(int i=m+1;i<=n;i++) g[i][ed]=1;
    int u,v;
    while(~scanf("%d%d",&u,&v)){
        g[u][v]=1;
    }
    printf("%d\n",EK_flow());
    return 0;
}
```

*dinic*写法:

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=2000+10,M=1e4+5,mod=1e9+7,inf=0x3f3f3f3f;
#define mst(a) memset(a,0,sizeof a)
#define fmst(a) memset(a,-1,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
int n,m,s=105,t=106,h[N],cur[N],cnt=1,dep[N];
struct node{
    int to,nt,w;
}e[M];
void add(int u,int v,int w){
    e[++cnt]={v,h[u],w},h[u]=cnt;
}
queue<int>q;
bool bfs(){ //给图分层.
    mst(dep);
    dep[s]=1;q.push(s);
    while(!q.empty()){
        int u=q.front();q.pop();
        cur[u]=h[u];    //每次初始化cur[]
        for(int i=h[u];i;i=e[i].nt){
            int v=e[i].to,w=e[i].w;
            if(w&&!dep[v])
                dep[v]=dep[u]+1,q.push(v);
```

```
        }
    }
    return dep[t];
}
int dfs(int u,int flow){
    if(u==t)return flow;
    int res=flow;
    for(int &i=cur[u];i;i=e[i].nt){ //优化3.
        int v=e[i].to,w=e[i].w;
        if(w&&dep[u]+1==dep[v]){
            int now=dfs(v,min(res,w));  //这里有个小细节要用res而不是flow，因为要取到当前最小,flow可能不是最小的.
            if(!now) dep[v]=1;  //优化1
            else{
                e[i].w-=now;    //建立反边.
                e[i^1].w+=now;
                res-=now;
            }
        }
        if(!res)return flow;    //优化2
    }
    return flow-res;
}
int main(){
    scanf("%d%d",&n,&m);
    int u,v;
    for(int i=1;i<=m;i++) add(s,i,1),add(i,s,0);
    for(int i=m+1;i<=n;i++) add(i,t,1),add(t,i,0);
    while(~scanf("%d%d",&u,&v)){
        add(u,v,1),add(v,u,0);
    }
    int ans=0;
    while(bfs()) ans+=dfs(s,inf);
    printf("%d\n",ans);
    return 0;
}
```

4.跟网络流没什么关系的题目

6121.「网络流 24 题」孤岛营救问题

跟网络流没什么关系，就是状压+bfs跑一遍就行了，开一个结构体储存x,y下标，key的状态和时间即可。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=15,M=2e4+5,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
#define pb push_back
```

```
int mp[N][N][N][N];
int key[N][N],vis[N][N][(1<<10)];
int d[4][2]={0,1,0,-1,1,0,-1,0};
struct node{
    int x,y,k,s;
}now;
int n,m,p,k;
void bfs(){
    queue<node>q;
    q.push({1,1,key[1][1],0});
    vis[1][1][key[1][1]]=1;
    while(!q.empty()){
        now=q.front();q.pop();
        //printf("(%d,%d),key=%d,s=%d\n",now.x,now.y,now.k,now.s);
        if(now.x==n&&now.y==m){
            printf("%d\n",now.s);
            return;
        }
        for(int i=0;i<4;i++){
            int x=now.x+d[i][0],y=now.y+d[i][1];
            if(x<1||x>n||y<1||y>m) continue;
            int s=now.s+1,Key=now.k|key[x][y];
            int type=mp[now.x][now.y][x][y];
            if(type>0&&!(now.k&((1<<(type-1))))) continue;
            if(!vis[x][y][Key]&&type) vis[x][y][Key]=1,q.push({x,y,Key,s});
        }
    }
    puts("-1");
    return;
}
int main(){
    memset(mp,-1,sizeof mp);
    scanf("%d%d%d%d",&n,&m,&p,&k);
    for(int i=1,x1,y1,x2,y2,z;i<=k;i++)
        scanf("%d%d%d%d%d",&x1,&y1,&x2,&y2,&z),mp[x1][y1][x2][y2]=mp[x2][y2][x1]
[y1]=z;
    int s;
    scanf("%d",&s);
    while(s--){
        int x,y,q;
        scanf("%d%d%d",&x,&y,&q);
        key[x][y]|=(1<<(q-1));
    }
    bfs();
    return 0;
}
```