最小割

如果沿着剩余容量大于0的边，无法从s到t，那么这个残余网络就被称为割。这些残余网络中边权和最小的割成为最小割。

最大流最小割定理

$f(s,t)_{max} = c(s,t)_{min}$

EK算法和Dinic算法能处理 10^3 到 10^4 的网络规模。

在**有向图**中，通过求出最大流就可以用最大流最小割定理求得最小割。

无向图最小割

Stoer-Wagner算法

1.min=MAXINT，固定一个顶点p

2.从点p开始扩展到最大生成树，记录最后扩展的顶点和最后扩展的边。

3.计算最后扩展到的顶点的切割值（即与此顶点相连的所有边权和），若比min小则更新min。

4.合并最后扩展的那条边及两个顶点为一个点

5.转到2，合并n-1次后结束，答案即为min

在prim中加入堆优化，总的复杂度为O(n^2logn)

最小割树(Gomory-Hu Tree)

最小割树的定义

定义一棵树T为最小割树，如果对于树上的所有边(s,t)，树上去掉(s,t)后产生的两个集合恰好是原图上(s,t)的最小割把原图分成的两个集合，且边(u,v)的权值等于原图上(u,v)的最小割

构造

树上去掉(s,t)后产生的两个集合恰好是原图上(s,t)的最小割把原图分成的两个集合

原理

将网络流的图转化为一棵树，其中原图u到v的最小割转化到树上。

树的一个性质是：删除一条边，树变得不连通。

我们可以任意选两个点s与t，跑最小割（即最大流），然后再连一条从s到t的边

又Dinic算法最后一次bfs相当于求一个最小割，原图就被分为了两部分。

最后分治即可，复杂度为O(mn^3)

u到v的最小割就算树上从u到v的路径。

性质

原图上u,v两点最小割就是最小割树上u到v的路径上权值最小的边。

实现细节

每次求最大流时，都要先恢复开始的网络流（即退流）

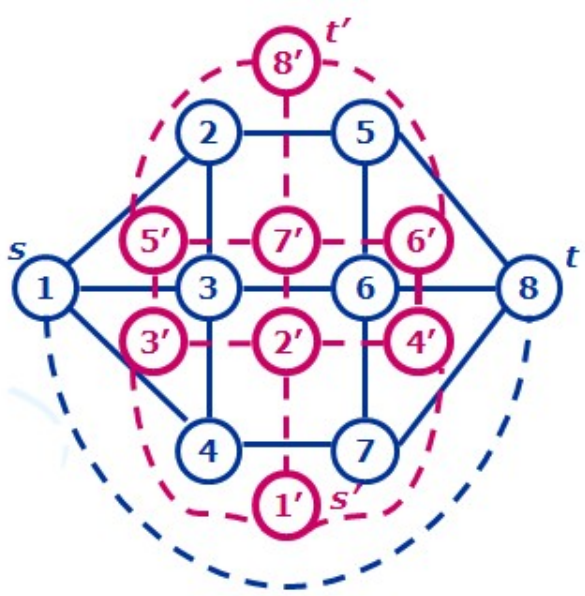
插入网络流的边时，要双向插入（没有指明起点）

下标从0开始到n

每一次最大流时，先保存源点和汇点，防止被覆盖

对偶图与最小割

平面图最小割=平面图最短路。



参考例题**[NOI2010] 海拔。**

例题

[Dahno Dahno]

将图划分为两个集合，得分就是集合内连边权值和。问最大得分。

```
//#pragma GCC optimize("Ofast", "inline", "-ffast-math")
//#pragma GCC target("avx,sse2,sse3,sse4,mmx")
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=507;

/*    Stoer_Wagner  O(n^3)  (POJ 2914)
   求无向图的最小割,把图分为两个子图的最小花费
   邻接矩阵建图,w[][]点从0开始
   算法流程:
     1.min=MAXINT，固定一个顶点P
     2.从点P用“类似”prim的s算法扩展出“最大生成树”，记录最后扩展的顶点和最后扩展的边
     3.计算最后扩展到的顶点的切割值（即与此顶点相连的所有边权和），若比min小更新min
     4.合并最后扩展的那条边的两个端点为一个顶点（当然他们的边也要合并，这个好理解吧？）
     5.转到2，合并N-1次后结束
     6.min即为所求，输出min
   prim本身复杂度是O(n^2)，合并n-1次，算法复杂度即为O(n^3)
   如果在prim中加堆优化，复杂度会降为O((n^2)logn)            */

//int read(){   int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return x*f;}
//void write(int x){if(x>9) write(x/10);putchar(x%10+'0');}

ll w[N][N],v[N],dis[N];
bool vis[N];
int n,m;

ll Stoer_Wagner(int n){
    ll i,j,res=1e18;
    for(int i=0;i<n;++i) v[i]=i;
    while(n>1){
        int k=1,pre=0;
```

```
        for(int i=1;i<n;++i){
            dis[v[i]]=w[v[0]][v[i]];
            if(dis[v[i]]>dis[v[k]]) k=i;
        }
        memset(vis,0,sizeof(vis));
        vis[v[0]]=1;
        for(int i=1;i<n;++i){
            if(i==n-1){
                res=min(res,dis[v[k]]);
                for(int j=0;j<n;++j){
                    w[v[pre]][v[j]]+=w[v[j]][v[k]];
                    w[v[j]][v[pre]]+=w[v[j]][v[k]];
                }
                v[k]=v[--n];
            }
            vis[v[k]]=1;
            pre=k;
            k=-1;
            for(int j=1;j<n;++j){
                if(!vis[v[j]]){
                    dis[v[j]]+=w[v[pre]][v[j]];
                    if(k==-1||dis[v[k]]<dis[v[j]]) k=j;
                }
            }
        }
    }
    return res;
}

void Solve(){
    scanf("%d",&n);
    ll sum=0;
    for(int i=0;i<n;++i){
        for(int j=0;j<n;++j){
            scanf("%lld",&w[i][j]);
            sum+=w[i][j];
        }
    }
    printf("%lld\n",sum-Stoer_Wagner(n)*2);
}

signed main(){
//  ios::sync_with_stdio(0);
//  cin.tie(0);cout.tie(0);
//  freopen("in.cpp","r",stdin);
//  freopen("out.cpp","w",stdout);
    int T=1;
//  cin>>T;
//  clock_t start,finish;
//  start=clock();
    while(T--){
        Solve();
    }
//  finish=clock();
//  cerr<<((double)finish-start)/CLOCKS_PER_SEC<<endl;  return 0;
}
```

P2762 太空飞行计划问题

有n个实验和m个器材，给定实验需要的器材以及它的收益，再给定每个器材的成本，问最大的净利润。
跑最小割相当于选择部分实验和部分器材，剩下的实验和器材都会被割掉，此时再用实验的总价值减去就能得到答案

```
#include<bits/stdc++.h>
using namespace std;
const int N=2007;
const int inf=(1<<30);

inline int min(int x,int y){return x>y?y:x;}
inline int max(int x,int y){return x>y?x:y;}

struct E{
    int v,w,nxt;
}e[N];

int head[N],cnt=0;
int dep[N],cur[N];
int n,m,S,T;
bool flag;

inline int read(){
    int x=0;char ch=getchar();
    while(!isdigit(ch)) ch=getchar();
    while(isdigit(ch)) x=x*10+ch-'0',ch=getchar();
    if(ch=='\n') flag=1;
    return x;
}

inline void addedge(int u,int v,int w){
    e[++cnt]=(E){v,w,head[u]};head[u]=cnt;
    e[++cnt]=(E){u,0,head[v]};head[v]=cnt;
}

inline void init(){
    for(int i=0;i<=n+m+1;++i) head[i]=-1;
    cnt=1;S=0;T=n+m+1;
}

int bfs(){
    for(int i=0;i<=n+m+1;++i) dep[i]=inf;
    queue<int>q;
    q.emplace(S);
    dep[S]=0;
    while(q.size()){
        int fro=q.front();
        q.pop();
        for(int i=head[fro];~i;i=e[i].nxt){
            int to=e[i].v,w=e[i].w;
            if(w&&dep[to]>dep[fro]+1){
                dep[to]=dep[fro]+1;
                q.emplace(to);
            }
        }
    }
    return dep[T]!=inf;
}

int dfs(int s,int mw){
    if(!mw||s==T) return mw;
```

```
    int flow=0;
    for(int &i=cur[s];~i;i=e[i].nxt){
        int to=e[i].v,w=e[i].w;
        if(w<=0||dep[to]!=dep[s]+1) continue;
        int cw=dfs(to,min(w,mw));
        if(cw<=0) continue;
        flow+=cw;mw-=cw;
        e[i].w-=cw;
        e[i^1].w+=cw;
        if(!mw) break;
    }
    return flow;
}

int dinic(){
    int res=0;
    while(bfs()){
        for(int i=0;i<=T;++i) cur[i]=head[i];
        res+=dfs(S,inf);
    }
    return res;
}

signed main(){
    n=read();m=read();
    init();
    int sum=0,x;
    for(int i=1;i<=n;++i){
        x=read();
        sum+=x;
        addedge(S,i,x);
        flag=0;
        while(!flag){
            x=read();
            addedge(i,n+x,inf);
        }
    }
    for(int i=1;i<=m;++i){
        x=read();
        addedge(i+n,T,x);
    }
    int res=dinic();
    for(int i=1;i<=n;++i) if(dep[i]!=inf) cout<<i<<" ";
    cout<<"\n";
    for(int i=1;i<=m;++i) if(dep[i+n]!=inf) cout<<i<<" ";
    cout<<"\n";
    cout<<sum-res<<"\n";
}
```

[ICPC-Beijing 2006] 狼抓兔子

给定$n*m$的网络，每个路口向下、向右和向右下连边，问最小割。$n,m\leq 1000$

直接建图跑Dinic最大流即可，但是数据特殊，把当前弧优化去掉才能过。(500ms)

更优秀的解法：平面图最小割转对偶图最短路。（从左下角跑到右上角，将图分成两份，每个小三角形看成一个点跑最短路）(3.54s)

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
```

```cpp
#define id(i,j) ((i-1)*m+j)
typedef long long ll;
const int N=1e6+1,M=N*6;

int read(){
    int x=0,f=1;char ch=getchar();
    while(ch<'0'||ch>'9') ch=getchar();
    while(ch>='0'&&ch<='9') x=(x<<3)+(x<<1)+ch-'0',ch=getchar();
    return x;
}

struct E{
    int v,w,nxt;
}e[M];
int head[N],cur[N],cnt=1;

inline void addedge(int f,int v,int w){
    e[++cnt]={v,w,head[f]};head[f]=cnt;
    e[++cnt]={f,w,head[v]};head[v]=cnt; //注意这里反向边的容量是w
}

int n,m,s,t;
int lv[N],q[N<<2];
inline int min(int a,int b){return a>b?b:a;}

inline int bfs(){ //BFS分层
    memset(lv,0,sizeof(lv));
    int hd=1,tl=1;
    q[1]=s;lv[s]=1;
    while(hd<=tl){
        int fro=q[hd++];
        for(int i=head[fro];i;i=e[i].nxt){
            int to=e[i].v,dis=e[i].w;
            if(dis>0&&!lv[to]){
                lv[to]=lv[fro]+1;
                q[++tl]=to;
            }
        }
    }
    return lv[t];
}

inline int dfs(int p=s,int flow=inf){
    if(p==t) return flow;
    int add=0; //剩余的流量
    for(int i=head[p];i&&add<flow;i=e[i].nxt){ //从当前弧开始出发
        int to=e[i].v,dis=e[i].w;
        if(dis>0&&lv[to]==lv[p]+1){ //向层数高的地方增广
            int c=dfs(to,min(dis,flow-add)); //尽可能多地传递流量
            if(!c){lv[to]=-1;continue;}
            e[i].w-=c; //更新残差流量
            e[i^1].w+=c;
            add+=c;
        }
    }
    return add; //返回传递出去的流量大小
}

ll Dinic(){
    ll ans=0;
    while(bfs()) ans+=dfs();
    return ans;
```

```
}

signed main(){
    n=read();m=read();
    s=id(1,1),t=id(n,m);
    for(int i=1;i<=n;++i)
        for(int j=1;j<=m-1;++j)
            addedge(id(i,j),id(i,j+1),read());
    for(int i=1;i<=n-1;++i)
        for(int j=1;j<=m;++j)
            addedge(id(i,j),id(i+1,j),read());
    for(int i=1;i<=n-1;++i)
        for(int j=1;j<=m-1;++j)
            addedge(id(i,j),id(i+1,j+1),read());
    printf("%lld\n",Dinic());
    return 0;
}
```

[NOI2010] 海拔

$(n+1)*(n+1)$的网格图上，相邻两点间有一些人流。左上角点的海拔为0，右下角海拔为1，没单位人流从海拔低的地方走到海拔高的地方会消耗对应的体力。问最少消耗多少体力。

```
#include<bits/stdc++.h>
#define pii pair<int,int>
#define mk make_pair
#define F first
#define S second
using namespace std;
const int N=2e6+7;
const int inf=0x3f3f3f3f;

int n,s,t;
struct E{
    int v,w,nxt;
}e[N];
int head[N],cnt=0;
void addedge(int u,int v,int w){
    e[++cnt]=(E){v,w,head[u]};head[u]=cnt;
}
int num(int x,int y){
    return y+(x-1)*n;
}

priority_queue<pii>q;
int dis[N],vis[N];
void dijkstra(){
    for(int i=s;i<=t;++i) dis[i]=inf,vis[i]=0;
    dis[s]=0;
    q.push(mk(0,s));
    while(q.size()){
        int fro=q.top().S;
        q.pop();
        if(vis[fro]) continue;
        vis[fro]=1;
        for(int i=head[fro];i;i=e[i].nxt) {
            int to=e[i].v,w=e[i].w;
            if(dis[to]>dis[fro]+w){
```

```
                dis[to]=dis[fro]+w;
                q.push(mk(-dis[to],to));
            }
        }
    }
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n;
    ++n;t=n*n+1;
    for(int i=1;i<=n;++i){
        for(int j=1,val;j<n;++j){
            cin>>val;
            if(i==1) addedge(s,num(i,j),val);
            else if(i==n) addedge(num(i-1,j),t,val);
            else addedge(num(i-1,j),num(i,j),val);
        }
    }
    for(int i=1;i<n;++i){
        for(int j=1,val;j<=n;++j){
            cin>>val;
            if(j==1) addedge(num(i,j),t,val);
            else if(j==n) addedge(s,num(i,j-1),val);
            else addedge(num(i,j),num(i,j-1),val);
        }
    }
    for(int i=1;i<=n;++i){
        for(int j=1,val;j<n;++j){
            cin>>val;
            if(i==1) addedge(num(i,j),s,val);
            else if(i==n) addedge(t,num(i-1,j),val);
            else addedge(num(i,j),num(i-1,j),val);
        }
    }
    for(int i=1;i<n;++i){
        for(int j=1,val;j<=n;++j){
            cin>>val;
            if(j==1) addedge(t,num(i,j),val);
            else if(j==n) addedge(num(i,j-1),s,val);
            else addedge(num(i,j-1),num(i,j),val);
        }
    }
    dijkstra();
    printf("%d",dis[t]);
    return 0;
}
```

参考资料

OI-wiki

《网络流建模》——周尚彦

https://blog.csdn.net/DDelphine/article/details/77935670

https://oi-wiki.org/graph/stoer-wagner/

https://www.cnblogs.com/zhang-qc/p/6516432.html

https://www.luogu.com.cn/blog/mydcwfy-342891/solution-p4897

https://www.cnblogs.com/fengxunling/p/10248928.html