# 网络流_10 最小割定理与应用

首先我们来回忆一下最小割的相关概念：

> ## 割
>
> 对于一个流网络`G=(V,E)`,可将其点集`V`分成两个不重不漏的集合`S`，`T`，有
>
> $$S \cup T = V$$
> $$S \cap T = \varnothing$$
>
> 其中有以下限制：
>
> $$\text{源点} s \in S, \text{汇点} t \in T$$
>
> **割的容量 c(S,T)**
>
> 所有从S指向T的有向的容量之和
>
> $$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$$
>
> **割的流量**
>
> 所有从S到T的的流量与从T到S的流量之差：
>
> $$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in T} \sum_{v \in S} f(u,v)$$
>
> **性质1:**
>
> 设f为流网络G的一个流，该流网络的源节点为s,汇点为t,设(S,T)为流网络G的任意切割，则横跨切割`(S,T)`的**净流量**：
>
> $$f(S,T) = |f|$$
>
> 即**对于每一个割的流量，都能对应一个流网络中的流量**
>
> **性质2:**
>
> $$\forall [S,T] \ \forall f \text{ 有} \quad f(S,T) \le c(S,T)$$
>
> 换句话来说，对于流网络任意流，都小于任意割的容积，因此就有**最大流小于等于最小割。**其中注意，最大流指流网络的最大流量，最小割指的是最小割的容量
>
> $$\text{最大流} |f| \le \text{最小割} c(S,T)$$
>
> ## 最大流最小割定理
>
> 以下三个定理相互等价
>
> - `f`是`G`中的一个最大流
> - 残留网络`Gf`不包括任何增广路径
> - `|f|=c(S,T)`,其中`(S,T)`是流网络G的某个切割

**因此，对于求最小割，就是求最大流**

### 更加形象的理解最小割：

最小割就是割去图中的一些边使得整张图无法从源点到汇点连通。

## 网络战争

> 一个带权无向图`G=(V,E)`,每条边e有一个权`we`求将点S和点T分开的一个边集C是的该割集的平均边权最小，即最小化$\frac{\sum_{e\in C}w_e}{|C|}$
>
> 注意，此处的边权指的是将某些边删去后S和T将不再连通

- 分析：

这是一道01分数规划的题目，老规矩设$\frac{\sum_{e\in C}w_e}{|C|}=\lambda$然后通过不断二分λ来求得最小值

既然删去某些边之后将不会再连通，因此对于两集合之间的割是必选的。其次选上这些割之后，我们仍然可以在两集合中的边进行选择。化简式子$\frac{\sum_{e\in C}w_e}{|C|}<\lambda$ then $\sum_{e\in C}w_e<\lambda\cdot|C|$ then $\sum_{e\in C}w_e-\lambda\cdot|C|<0$ then |C|表示边数，将其带入得到最终式子：$\sum_{e\in C}(w_e-\lambda)<0$

因此对于不是割边的边，如果小于λ，我们选上能够使得答案最小化。

- 有技巧

对于无向图，统一的建边方式：

```
void add(int a,int b,int c){
    e[idx] = b, w[idx] = c, ne[idx] = h[a], h[a] = idx ++;
    e[idx] = a, w[idx] = c, ne[idx] = h[b], h[b] = idx ++;
}
```

然后对于每次二分通过对`w[i]`进行转化即可

Code：

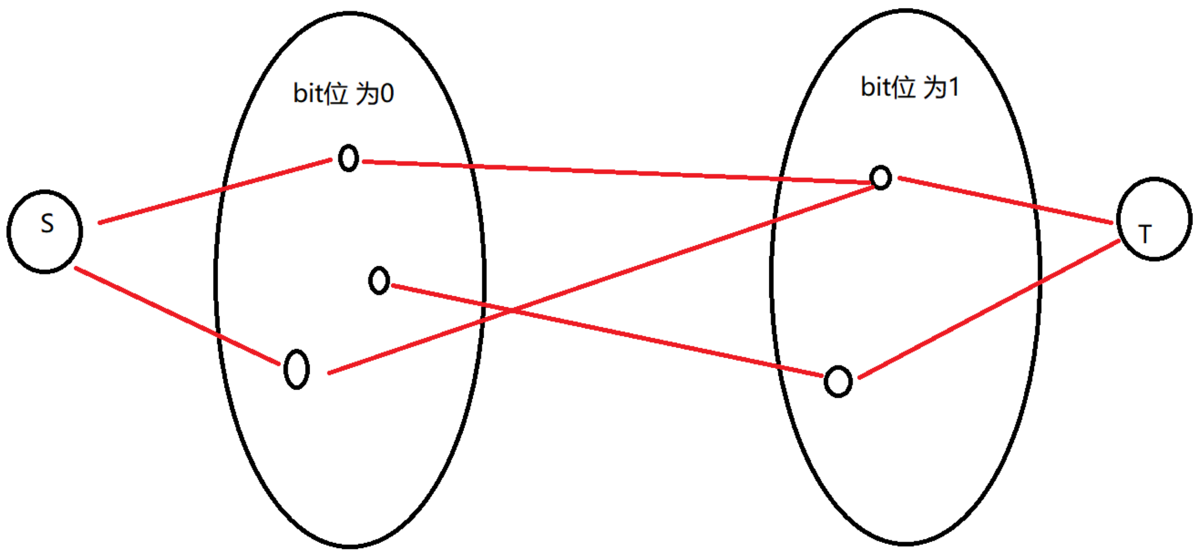
```
bool check(double x){
    double ans = 0;
    for(int i=0;i<idx;i+=2)
        if(w[i]<=x) {
            ans += w[i] - x;
            f[i]=f[i^1] = 0;
        }
        else f[i] = f[i^1] = w[i] -  x;
    return ans + dinic() < 0;
}
```

## 最优标号

> 给定一个无向图 G=(V,E)G=(V,E)，每个顶点都有一个标号，它是一个$[0,2^{31}-1]$内的整数。
>
> 不同的顶点可能会有相同的标号。对每条边 (u,v)，我们定义其费用 cost(u,v) 为 u 的标号与 v的标号的异或值。现在我们知道一些顶点的标号。你需要确定余下顶点的标号使得所有边的费用和尽可能小。

思路：位运算经典思考方式，按位考虑。下面对于每一位：

我们可以抽象图为：




对于有边的两点我们连双向边，对于已经确定的且第bit位的点如果是0则从源点连单向，如果为1则向汇点连单向即可。最终求最小割就是当前位对应的最小值。

```
void build(int bit){
    memset(h,-1,sizeof h); idx =0 ;
    rep(i,1,m) {
        int u=edge[i].x,v=edge[i].y;
        add(u,v,1,1);
    }
    rep(i,1,n)
        if(p[i]>=0){
            if(p[i]>>bit&1) add(i,T,INF,0);
            else add(S,i,INF,0);
        }
}

LL solve(int bit){
    build(bit);
    return dinic();
}
```

## 方格取数问题

> 给定一些方格，请设计一个算法取数要求取出的所有数中没有公共边

- 思路

对于方格问题，首先因该想到的是进行二染色，然后将`i+j&1==1`分成左集合，另外为右集合。这样左右集合对应的数都不能同时取到。从源点向左集合连容量为取值的边，从右集合向汇点连容量为取值的边。用总点数减去最小割就能保证割去的边使得左右集合不连通的同时，割去的边权值最小。

### 技巧一：

如果有些边不想让其成为割边的话，则人为得将这些边设置为容量为正无穷的边。

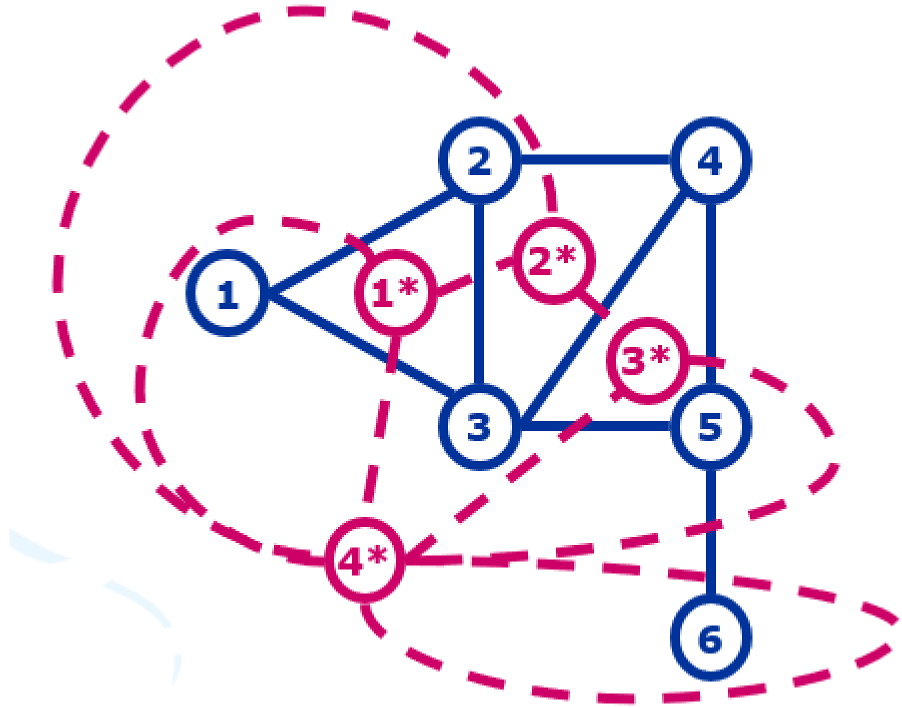
## 应用2： 平面图上最小割=对偶图最短路

**平面图：**

- (欧拉公式)如果一个联通的平面图有n个点，m条边和f个面，则`f=m-n+2`
- 每个平面图G都有一个与其对偶的平面图G'
    - G‘中的每个点对应G中的一个面

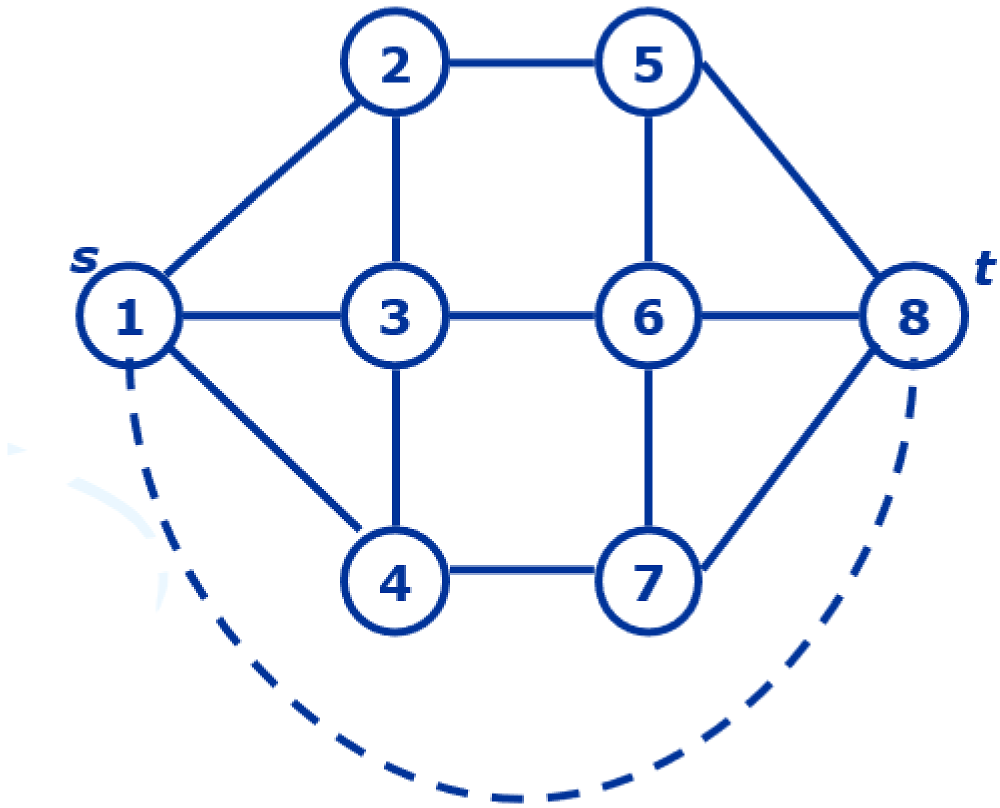

性质

- 对偶图中的每条边e有当$e \in 面\{f1, f2\}$则加入边$(f_1^*, f_2^*)$
    - 即有公共边的面转化成点之后互相之间连一条线
- n$e$只属于一个面$f$，加入回边$(f_1^*，f_2^*)$

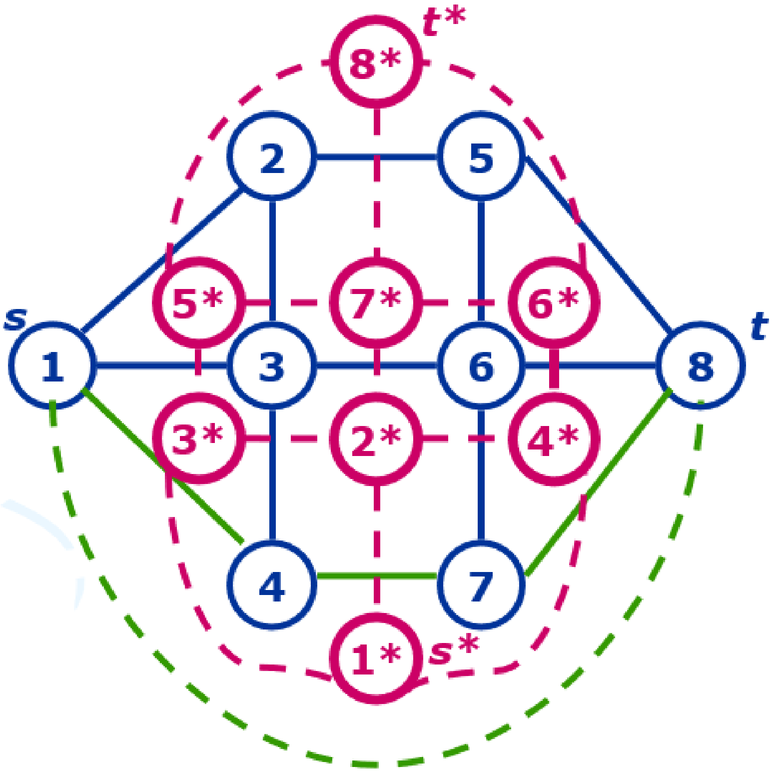



**利用最短路求最小割**

对于一个s-t图进行改造，连接s和t，得到一个附加面：




然后建立起该图的对偶图，令附加面为$s^*$,不封闭面为$t^*$,得到新图G'




(记住要删去或者不要添加$s^*$和$t^*$之间的边)

在新图中我们可以发现从$s$到$t$的一条路径对应$s^*$到$t^*$的一个割，因此最小割的容量就等于最短路径长度！

### 应用3：最小割树