# 最小割

## 概念

### 割

对于一个网络流图 $G=(V,E)$ 其割的定义为一种 **点的划分方式**：将所有的点划分为 $S$ 和 $T=V-S$ 两个集合，其中源点 $s\in S$ 汇点 $t\in T$

### 割的容量

我们定义割 $(S,T)$ 的容量 $c(S,T)$ 表示所有从 $S$ 到 $T$ 的边的容量之和，即 $c(S,T)=\sum_{u\in S,v\in T}c(u,v)$ 当然我们也可以用 $c(s,t)$ 表示 $c(S,T)$

### 最小割

最小割就是求得一个割 $(S,T)$ 使得割的容量 $c(S,T)$ 最小。

## 证明

### 最大流最小割定理

**定理** $f(s,t)_{max}=c(s,t)_{min}$

对于任意一个可行流 $f(s,t)$ 的割 $(S,T)$ 我们可以得到：

$f(s,t)=S_{出边的总流量}-S_{入边的总流量}\le S_{出边的总流量}=c(s,t)$

如果我们求出了最大流 $f$ 那么残余网络中一定不存在 $s$ 到 $t$ 的增广路径，也就是 $S$ 的出边一定是满流 $S$ 的入边一定是零流，于是有：

$f(s,t)=S_{出边的总流量}-S_{入边的总流量}=S_{出边的总流量}=c(s,t)$

结合前面的不等式，我们可以知道此时 $f$ 已经达到最大。

## 代码

### 最小割

通过 **最大流最小割定理**，我们可以直接得到如下代码：

参考代码：

```cpp
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <queue>

const int N = 1e4 + 5, M = 2e5 + 5;
int n, m, s, t, tot = 1, lnk[N], ter[M], nxt[M], val[M], dep[N], cur[N];

void add(int u, int v, int w) {
  ter[++tot] = v, nxt[tot] = lnk[u], lnk[u] = tot, val[tot] = w;
}
void addedge(int u, int v, int w) { add(u, v, w), add(v, u, 0); }
int bfs(int s, int t) {
  memset(dep, 0, sizeof(dep));
  memcpy(cur, lnk, sizeof(lnk));
  std::queue<int> q;
  q.push(s), dep[s] = 1;
  while (!q.empty()) {
    int u = q.front();
    q.pop();
    for (int i = lnk[u]; i; i = nxt[i]) {
      int v = ter[i];
      if (val[i] && !dep[v]) q.push(v), dep[v] = dep[u] + 1;
    }
  }
  return dep[t];
}
int dfs(int u, int t, int flow) {
  if (u == t) return flow;
  int ans = 0;
  for (int &i = cur[u]; i && ans < flow; i = nxt[i]) {
    int v = ter[i];
    if (val[i] && dep[v] == dep[u] + 1) {
      int x = dfs(v, t, std::min(val[i], flow - ans));
      if (x) val[i] -= x, val[i ^ 1] += x, ans += x;
    }
  }
  if (ans < flow) dep[u] = -1;
  return ans;
}
int dinic(int s, int t) {
  int ans = 0;
  while (bfs(s, t)) {
    int x;
    while ((x = dfs(s, t, 1 << 30))) ans += x;
  }
  return ans;
}
int main() {
```

```
  scanf("%d%d%d%d", &n, &m, &s, &t);
  while (m--) {
    int u, v, w;
    scanf("%d%d%d", &u, &v, &w);
    addedge(u, v, w);
  }
  printf("%d\n", dinic(s, t));
  return 0;
}
```