

# 最大流, 最小割问题及算法实现



Andrew\_liu (/u/4ee453b72aff) [+ 关注](#)  
2015.05.10 15:04\* 字数 954 阅读 23686 评论 13 喜欢 24  
(/u/4ee453b72aff)

本博客采用创作共用版权协议, 要求署名、非商业用途和保持一致. 转载本博客文章也必须遵循署名-非商业用途-保持一致 (<https://link.jianshu.com?t=http://creativecommons.org/licenses/by-nc-sa/3.0/deed.zh>)的创作共用协议.

由于博文中包含一些LaTex格式数学公式, 在简书中显示不好, 所以推荐阅读博客中的版本最大流, 最小割基本问题及算法实现 (<https://link.jianshu.com?t=http://andrewliu.tk/2015/05/10/%E6%9C%80%E5%A4%A7%E6%B5%81-%E6%9C%80%E5%B0%8F%E5%89%B2%E5%9F%BA%E6%9C%AC%E6%A6%82%E5%BF%B5%E5%8F%8A%E7%AE%97%E6%B3%95%E5%AE%9E%E7%8E%B0/>)

对于最大流最小割问题的总结, 如有错误, 欢迎指出.

## 最大流(MaxFlow)问题

给定指定的一个有向图, 其中有两个特殊的点源S(Sources)和汇T(Sinks), 每条边有指定的容量(Capacity), 求满足条件的从S到T的最大流(MaxFlow).

想象一条多条不同水流量的水管组成的网络, s为供水厂, t为水用户, 最大流问题就是找到能够在s到t流通的最大水流量

一个流是最大流当且仅当其残存网络不包含任何增广路径(里面的名称在后面有详细解释)

## 流(Flow)的基本性质

设 $C_{uv}$ 代表边u到v最大允许流量(Capacity),  $f_{uv}$ 代表u到v的当前流量, 那么有以下两个性质:

- $(u, v)$ 为有向图边,  $0 \leq f_{uv} \leq C_{uv}$ , 即对于所有的边, 当前流量不允许超过其Capacity
- 除了s, t之外, 对所有节点有  $\sum_{(v, u)} f_{vu} = \sum_{(u, v)} f_{uv}$ , 即对于任何一点, 流入该点的流量等于流出该点的流量, 流量守恒原则(类似与能量守恒的概念).

非负数值 $f(u, v)$ 为从节点u到节点v的流. 一个流 $|f|$ 的定义:  
 $|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$

最大流问题即要找到一个最大的流f

## Ford-Fulkerson方法

之所以称之为方法, 而不是算法, 因为FF(Ford-Fulkerson简称)包含不同运行时间的几种实现, 是一种迭代的方法.

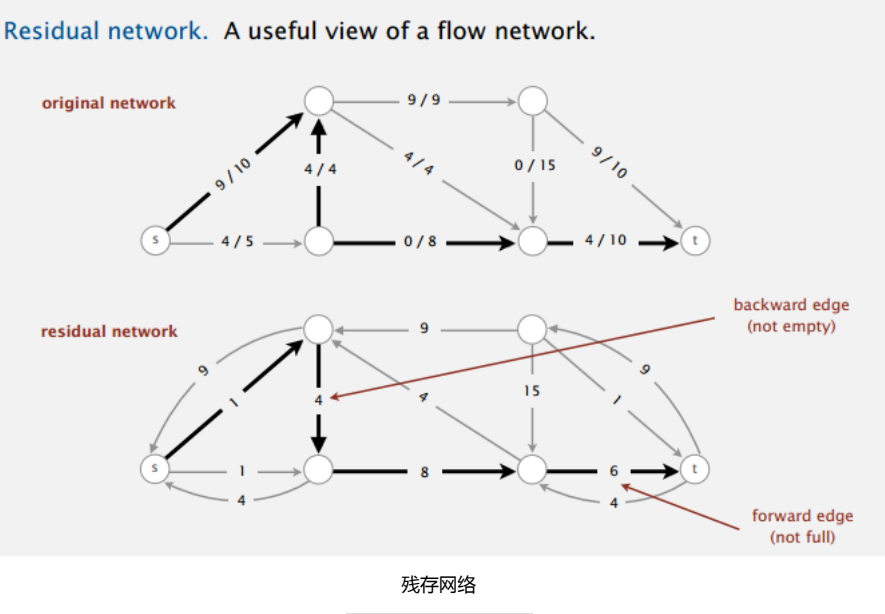
该方法主要依赖于 残存网络, 增广路径和割

```
//伪代码
初始化:所有流 f = 0
while 在残存网络中存在增广路径p
    增加流f的值
return f
```

残存网络

给定网络G和流量f, 残存网络\$G\_f\$由那些仍有空间对流量进行调整的边构成.

残留网络 = 容量网络capacity - 流量网络flow



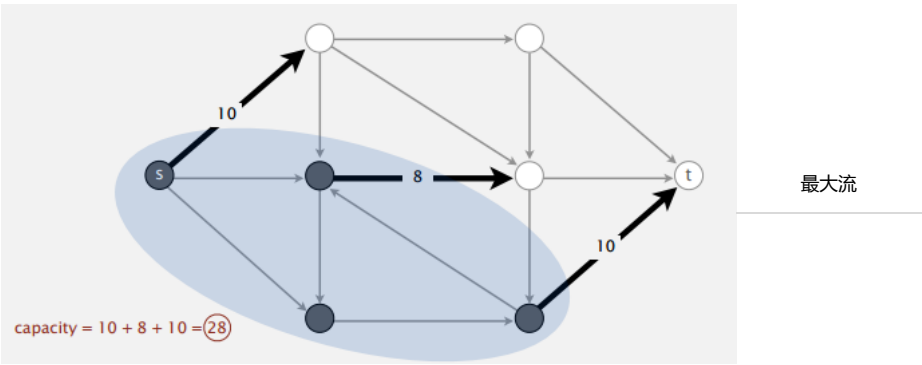
增广路径

增广路径p是残存网络中一条从源节点s到汇点t的简单路径,在一条增广路径p上能够为每条边增加的流量的最大值为路径p的残存容量 $c_f(p) = \min \{c_f(u,v):(u,v) \in p\}$

在一条增广路径p上, 要增加整条增广路径的水流量, 则必须看最小能承受水流量的管道, 不然水管会爆掉, 这最小承受水流量就是 残存容量

割

在有向图网络G中, 割(S, T)将V划分为S和T = V - S, 使得s属于S集合, t属于T集合. 割(S, T)的容量是指从集合S到集合T所有边的容量之和.



(/a  
utn

## 最大流最小割理论

设 $f$ 为流网络 $G = (V, E)$ 中的一个流, 该流网络的源节点为 $s$ , 汇点为 $t$ , 则下面的条件是等价的:

- $f$ 是 $G$ 的一个最大流
- 残存网络 $G_f$ 不包含任何增广路径
- $|f| = c(S, T)$ , 其中 $(S, T)$ 是流网络 $G$ 的某个割

## Ford-Fulkerson算法Java实现

### 伪代码

```
for each edge(u, v)属于G.E(图G的边)
    (u, v).f = 0 //所有边的流为0
//循环终止条件为残存网络中不存在增广路径
while s到t的残存网络中存在增广路径p:
    c(p) = 最小残存容量
    for 增广路径的每条边
        if 这条边属于E集合
            (u, v).f = (u, v).f + c(p) //意思是在原有的流增加最小残存容量.
        else
            (u, v).f = (u, v).f - c(p)
```

^

🔗

```
//边的定义
public class FlowEdge {
    private final int v, w; //边的起点和终点
    private final double capacity; //流量
    private double flow; //流
    public FlowEdge(int v, int w, double capacity) {
        this.v = v;
        this.w = w;
        this.capacity = capacity;
    }
    public int from() {
        return v;
    }
    public int to() {
        return w;
    }
    public double capacity() {
        return capacity;
    }
    public double flow() {
        return flow;
    }
    public int other(int vertex) {
        if (vertex == v) {
            return w;
        } else if (vertex == w) {
            return v;
        } else {
            throw new RuntimeException("Inconsistent edge");
        }
    }
}

//v中残留流量
public double residualCapacityTo(int vertex) {
    if (vertex == v) { //反向边
        return flow;
    } else if (vertex == w) { //正向边
        return capacity - flow;
    } else {
        throw new IllegalArgumentException();
    }
}

//向v中增加delta
public void addResidualFlowTo(int vertex, double delta) {
    if (vertex == v) {
        flow -= delta;
    } else if (vertex == w) {
        flow += delta;
    } else {
        throw new IllegalArgumentException();
    }
}
}
```

(/a  
utn

```
//流图的定义
public class FlowNetwork {
    private final int V; //顶点个数
    private Bag<FlowEdge>[] adj;
    public FlowNetwork(int V) {
        this.V = V;
        adj = (Bag<FlowEdge>[]) new Bag[V];
        for (int v = 0; v < V; v++) {
            adj[v] = new Bag<>();
        }
    }
}

//想流图中增加边
public void addEdge(FlowEdge e) {
    int v = e.from();
    int w = e.to();
    adj[v].add(e); //正向边
    adj[w].add(e); //反向边
}

public int V() {
    return V;
}

public Iterable<FlowEdge> adj(int v) { //返回邻接边
    return adj[v];
}
}
```



```
//FordFulkerson方法的实现
public class FordFulkerson {
    private boolean[] marked; //如果残留网络中有s->v路径, 则为true
    private FlowEdge[] edgeTo; //s->v路径的最后的边
    private double value; //流

    public FordFulkerson(FlowNetwork G, int s, int t) {
        value = 0.0;
        //当找不到增广路径时终止
        while (hasAugmentingPaht(G, s, t)) { //判断是否还有增广路径
            double bottle = Double.POSITIVE_INFINITY;
            for (int v = t; v != s; v = edgeTo[v].other(v)) { //计算最大流量
                bottle = Math.min(bottle, edgeTo[v].residualCapacityTo(v));
            }
            for (int v = t; v != s; v = edgeTo[v].other(v)) {
                edgeTo[v].addResidualFlowTo(v, bottle);
            }
            value += bottle;
        }
    }

    private boolean hasAugmentingPaht(FlowNetwork G, int s, int t) {
        edgeTo = new FlowEdge[G.V()];
        marked = new boolean[G.V()];

        Queue<Integer> q = new Queue<>();
        q.enqueue(s);
        marked[s] = true;
        while (!q.isEmpty()) {
            int v = q.dequeue();
            for (FlowEdge e : G.adj(v)) {
                int w = e.other(v);
                if (e.residualCapacityTo(w) > 0 && !marked[w]) {
                    edgeTo[w] = e;
                    marked[w] = true;
                    q.enqueue(w);
                }
            }
        }
        return marked[t];
    }

    public double value() {
        return value;
    }

    public boolean intCut(int v) { //在残留网络中v->s是否可达
        return marked[v];
    }
}
```

(/a  
utn

## 参考链接

- <算法导论>
- <算法>(普林斯顿Algorithm II)
- 网络流：最大流，最小割 基本概念及算法 ([https://link.jianshu.com?t=http://blog.csdn.net/xzz\\_hust/article/details/22041173](https://link.jianshu.com?t=http://blog.csdn.net/xzz_hust/article/details/22041173))
- 最大流问题-Ford-Fulkerson算法 (<https://link.jianshu.com?t=http://www.acmerblog.com/ford-fulkerson-6135.html>)

小礼物走一走，来简书关注我

赞赏支持

📖 算法之美 (/nb/917681)

举报文章 © 著作权归作者所有



Andrew\_liu (/u/4ee453b72aff)

写了 34109 字，被 2793 人关注，获得了 1962 个喜欢  
(/u/4ee453b72aff)

+ 关注

重度强迫症患者 软件更新狂人 编程爱好者 C/C++/Golang/Python Developer 个人博客: <http://andrewliu.in>



(http://cwb.assets.jianshu.io/notes/images/135618C



(/apps/download?utm\_source=nbc)



登录 (/sign\_in?utm\_source=desktop&utm\_medium=not-signed-in-comment-form)

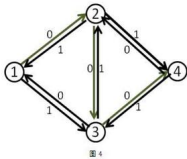
评论

智慧如你，不想发表一点想法 (/sign\_in?utm\_source=desktop&utm\_medium=not-signed-in-nocomments-text)咩~

被以下专题收入，发现更多相似内容

- 代码改变世界 (/c/0f5e015fc36c?utm\_source=desktop&utm\_medium=notes-included-collection)
- 程序员 (/c/NEt52a?utm\_source=desktop&utm\_medium=notes-included-collection)
- 首页投稿 (/c/bDHhpK?utm\_source=desktop&utm\_medium=notes-included-collection)
- 我是程序员；您... (/c/abe194e18e78?utm\_source=desktop&utm\_medium=notes-included-collection)
- 攻城师 (/c/e500fdc6a0f2?utm\_source=desktop&utm\_medium=notes-included-collection)

(/p/efb2d79e2b0f?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)  
**Ford-Fulkerson 方法——最大流问题 (/p/efb2d79e2b0f?utm\_campaign=...**

最大流&&最小费用最大流&&最大二分匹配 Python 源码：https://github.com/edisonleohl/DataStructure-Algorithm/blob/master/Graph/MaxFlow 最大流问题 比喻：有一个自来水管道运输系统，起点...

廖少少 (/u/28b41b12cde8?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

高考3500 (/p/0bda5d804ee3?utm\_campaign=maleskine&utm\_content=...

A a (an) [ə, eɪ(ə)n] art. 一 (个、件、.....) abandon [əˈbændən] v.抛弃, 舍弃, 放弃 ability [əˈbɪlɪti] n. 能力; 才能 able [ˈeɪb(ə)] a. 能够; 有能力的 abnormal [æbˈnɔːm...

0涂桃子 (/u/02eb49244585?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

(/a  
utn

(11)图算法3: 所有节点对最短路径与最大流问题 (/p/71f5fe37e53c?utm\_ca...

所有结点对的最短路径问题 Floyd算法 前提条件: 可以有负权重边, 但是不能有负权重的环. 特点: 动态规划, V^3. 按照动态规划的步骤:最优子结构: d[i][j]表示结点vi至结点vj的最短路径, 而带上了上标d[i][j]<k>表示允...

陈码工 (/u/0a709b6a0a9a?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

算法笔记 (/p/ed373cc0e066?utm\_campaign=maleskine&utm\_content=...

算法 插入排序 每次将一个待排序的元素与已排序的元素进行逐一比较, 直到找到合适的位置按大小插入. 插入排序代码 注意[0,i-1]都是有序的. 如果待插入元素比arr[i-1]还大则无需再与[i-1]前面的元素进行比较了...

AkaTBS (/u/2dd5b9531cfc?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

(/p/367b7e80eabf?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

网络流之最大流 (Edmonds Karp算法) (/p/367b7e80eabf?utm\_campai...

题目描述 如题, 给出一个网络图, 以及其源点和汇点, 求出其网络最大流. 输入输出格式 输入格式 第一行包含四个正整数N、M、S、T, 分别表示点的个数、有向边的个数、源点序号、汇点序号. 接下来M行每行...

Ricardo\_Y\_Li (/u/1564475b3e73?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

(/p/679ab8674425?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

奋力担当职责, 当好储备人 (/p/679ab8674425?utm\_campaign=maleskin...

湖北局九三五处 崔玲玲 2017年11月5日至11月11日, 在国家局高度重视、江西局用心安排以及六七三处全体员工悉心照顾下, 国家物资储备系统2017年新录用工作人员第三期培训在井冈山和江西局六七三处顺利进...

张盼盼0113 (/u/5d094eff5c65?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

从今天起, 每日一篇文章 (/p/d34a0652b628?utm\_campaign=maleskine&ut...

是什么时候开始想写东西呢? 大概就是看书的时候吧. 同人, 就是最先的方向. 只不过是遗憾, 然后想要弥补罢了. 记得那时候再晋江混, 一篇文章, 一节, 很喜欢. 写了一篇八千字的同人..... 发给作者后, 作者...

晓风v清风 (/u/407fbcdceea5?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

(/p/b4fea5d8061a?




utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

我的妈呀, 我过敏了 (/p/b4fea5d8061a?utm\_campaign=maleskine&utm...


故事力 三个金人 (/p/69eef7c85127?utm\_campaign=maleskine&utm\_co...

曾经有个小国到中国来，进贡了三个一模一样的金人，金碧辉煌，把皇帝高兴坏了。可是这小国不厚道，同时出一道题目：这三个金人哪个最有价值？皇帝想了许多的办法，请来珠宝匠检查，称重量，看做工，都...

 姜姜讲 (/u/2dc55a64c4e7?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

羽尊记 (/p/862596cdce8b?utm\_campaign=maleskine&utm\_content=no...

第一章：落樱村 落樱村，这是一个毫不起眼的小山村。村里一百多户人都是靠打猎为生，妇人耕种。在积羽大陆上，这种小村庄随处可见。“嘎吱，嘎吱”这已是冬月中旬，外面下起了鹅毛大雪。寒风吹得呼呼地响...

 柠檬果冻茶\_\_ (/u/947dc2dd7cc2?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

