

# 浅谈基于分层思想的 网络流算法

上海市延安中学  
王欣上

Email: [wxsxg@hotmail.com](mailto:wxsxg@hotmail.com)

- 最短路径增值 (MPLA)
- Dinic
- MPM

# 什么是剩余图?

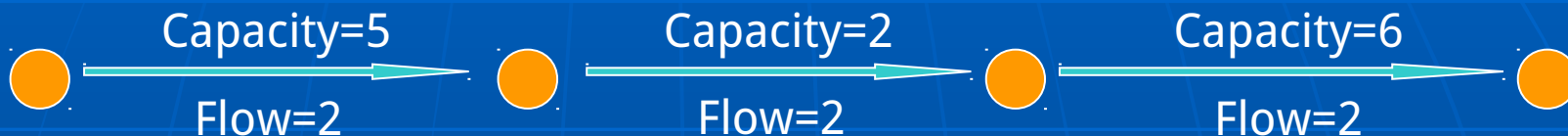
剩余图  $G'=(V,E')$

流量网络  $G=(V,E)$  中, 对于任意一条边  $(a,b)$ , 若  
 $\text{flow}(a,b) < \text{capacity}(a,b)$  or  $\text{flow}(b,a) > 0$   
则  $(a,b) \in E'$

可以沿着  $a \rightarrow b$   
方向增广

## 剩余图的权值代表能沿边增广的大小

有向图



剩余图

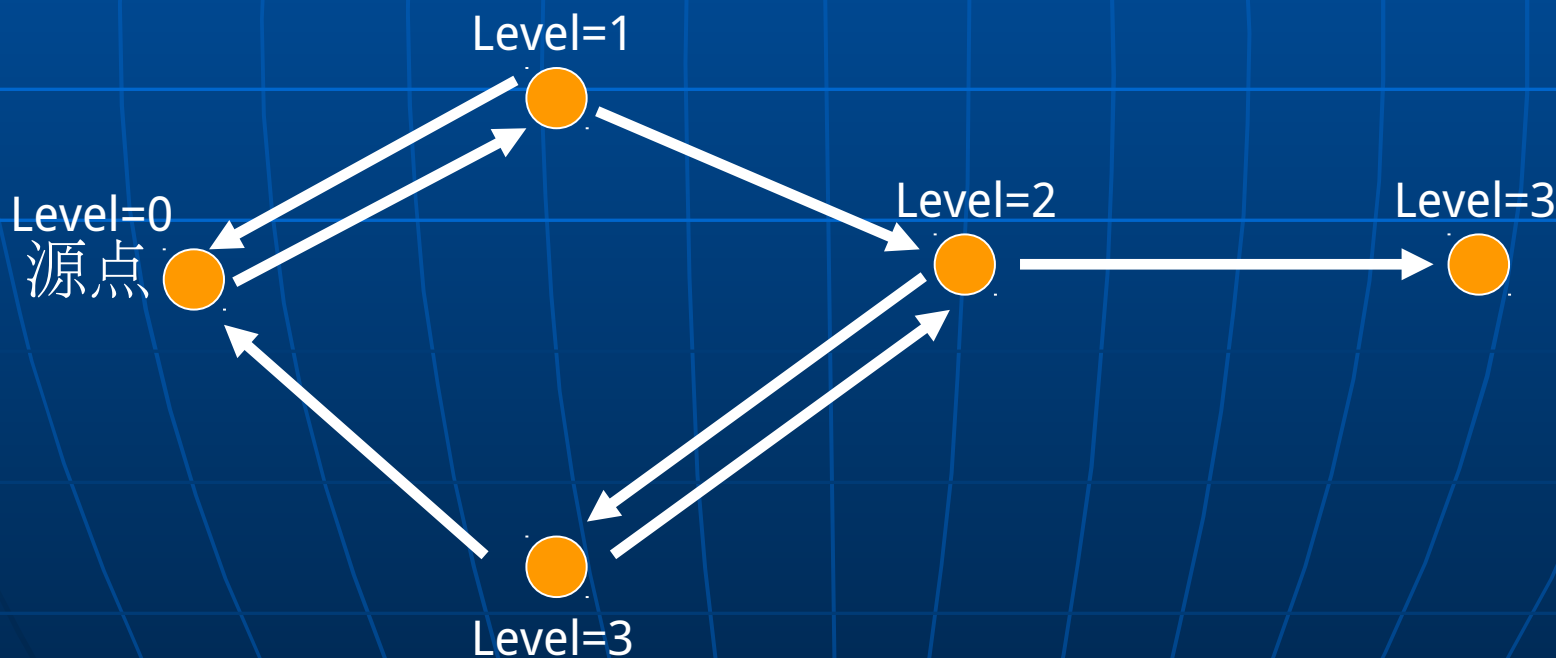


- 剩余图中，每条边都可以沿其方向增广
- 剩余图中，从源点到汇点的每一条路径都对应一条增广路

# 一、最短路径增值 (MPLA)

顶点  $u$  的层次:  $\text{level}(u)$  = 在剩余图中从源点到  $u$  所经过的最少边数

层次图: 对于剩余图中的任意一条边  $(a,b)$ , 当且仅当  $\text{level}(a)+1=\text{level}(b)$  时,  $(a,b)$  是层次图中的边



# 一、最短路径增值 (MPLA)

## 算法步骤

1、初始化流量，计算出剩余图

2、一次 **bfs** 对顶点标号，计算出层次图，如果汇点不在层次图内，那么算法结束

3、不断在层次图中寻找增广路进行增广，并修改剩余图 **多次 bfs**

4、转步骤 2

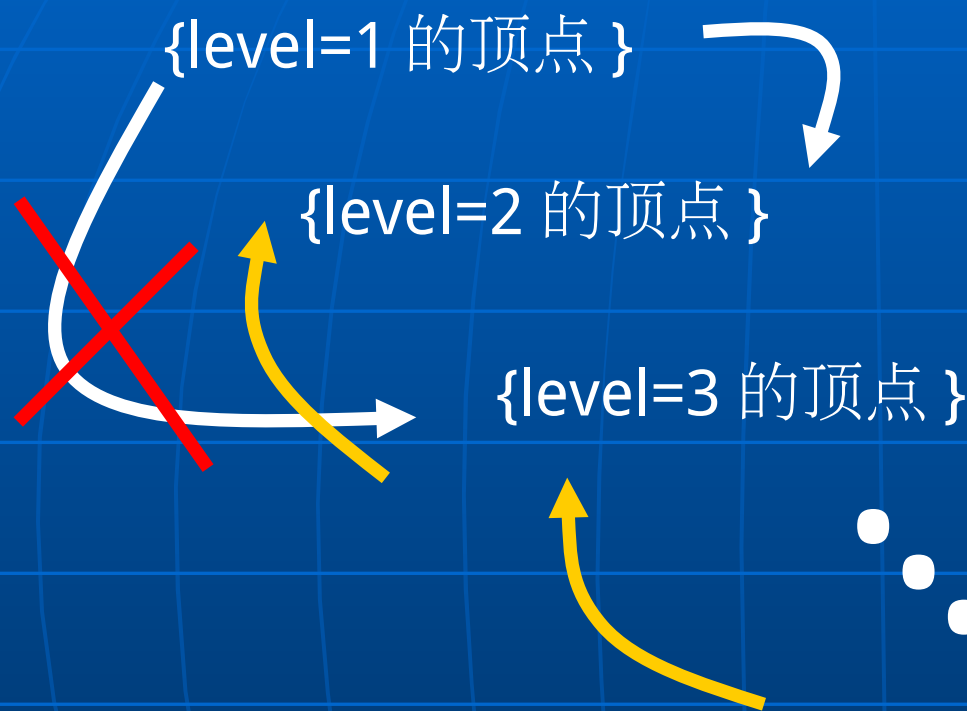
定理：对于有  $n$  个点的流量网络，在最短路径增值算法中，最多建立  $n$  次层次图。

证明这个定理有助于进行算法复杂度分析

在建立完层次图以后，假设从源点到汇点的最短路径长度为  $k$ ，我们将层次图中所有的点分到  $k+1$  个集合中，第  $i$  个集合为 { 顶点  $u \mid \text{level}(u)=i-1$  }

源点

在剩余图中，存在着 2 类边



第一类：从第  $i$  个集合中的顶点连到第  $i+1$  ( $1 \leq i \leq k$ ) 个集合中的顶点

第二类：从第  $i$  ( $1 \leq i \leq k+1$ ) 个集合中的顶点连到第  $j$  ( $1 \leq j \leq i$ ) 个集合中的顶点

在层次图中， $\text{level}$  存在第  $i$  类边，连到是  $\text{level}$  层次图性质决定的。

$\{\text{level}=k-1 \text{ 的顶点}\}$

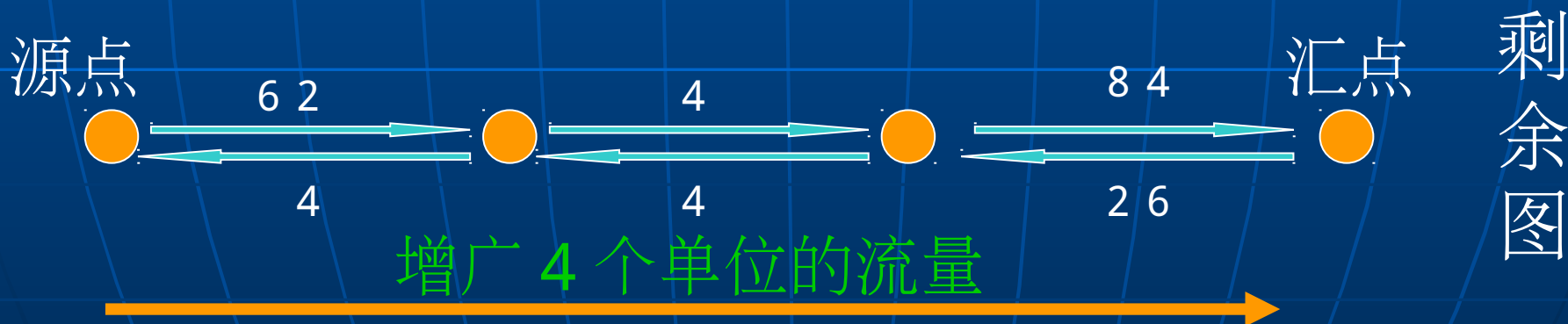
汇点



一次增广的效果:

- 删除一条或多条边
- 可能增加一条或多条回边

与增广路  
的方向相  
反



源点

{level=1 的顶点 }

{level=2 的顶点 }

{level=3 的顶点 }

- 必然会经过第二类边
- 经过的第一类边的数量  $\geq k$

{level=k-1 的顶点 }

汇点

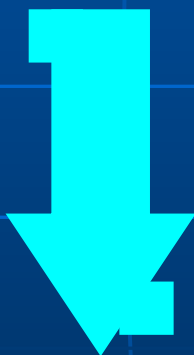
层次图中找完增广路径以后，  
剩余图中的最短路径：

从源点开始，往下一步一步走，走到某个集合后沿一条增广路径都回到源点。每走一步，类边向上走到某个集合，再继续一步一步向下走，到某个集合又向上退……直到走到汇点。

# 一、最短路径增值 (MPLA)

- 层次图中增广路径长度序列严格递增

$$1 \leq \text{路径长度} \leq n-1$$



- 最多建  $n$  次层次图

# 一、最短路径增值 (MPLA)

## 复杂度分析

最多有  $n$  层

建层次图: 每层做一次 bfs 标号  $O(m)$

$O(n*m)$

# 一、最短路径增值 (MPLA)

## 复杂度分析

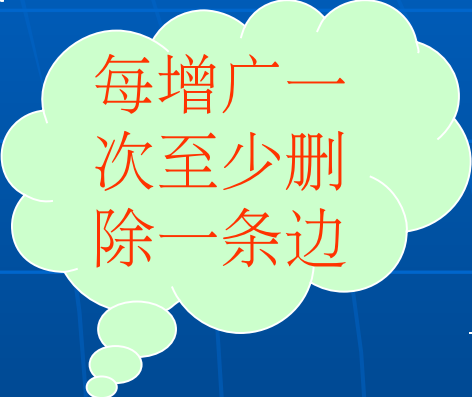
最多有  $n$  层

找增广路:

最多找  $m$  次增广路

找增广路 bfs  $O(m)$

$O(n*m*m)$



每增广一次至少删除一条边

# 一、最短路径增值 (MPLA)

- 复杂度  $O(n*m^2)$
- 程序简短
- 对于中小规模数据速度快

## 二、Dinic

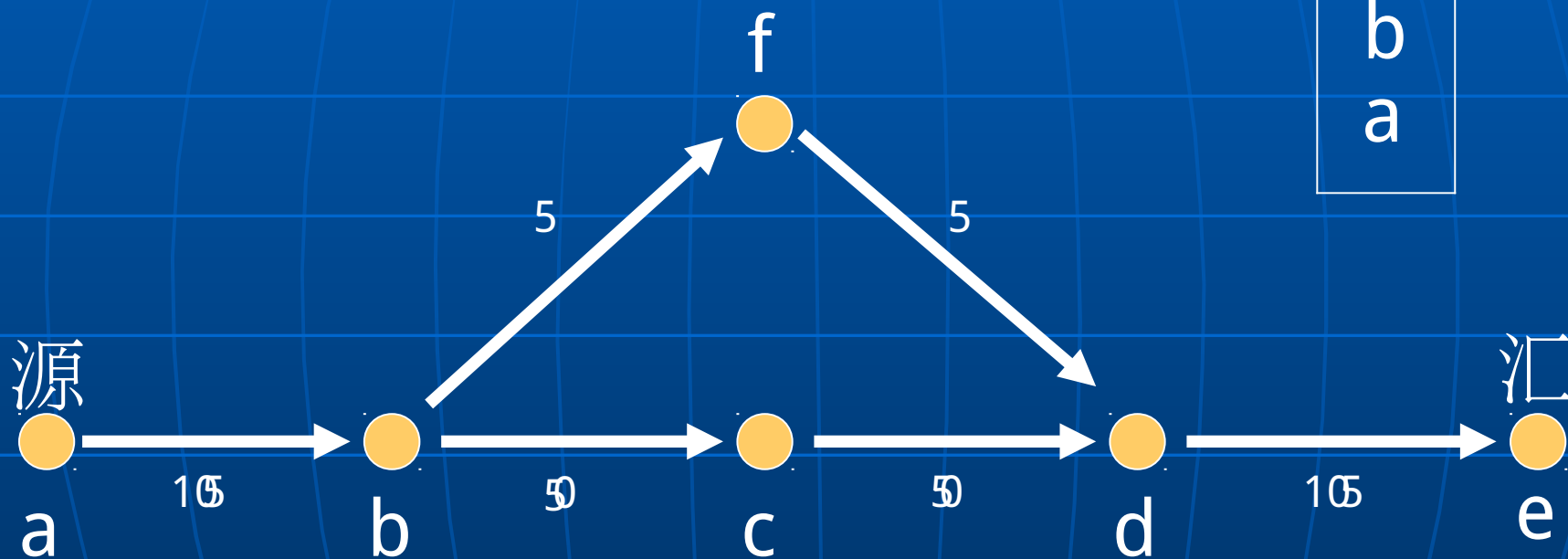
### 算法步骤

- 1、初始化流量，计算出剩余图
- 2、一次 **bfs** 对顶点标号，计算出层次图，如果汇点不在层次图内，那么算法结束
- 3、一次 **dfs** 过程找增广
- 4、转步骤 2

## 二、Dinic

栈

e  
d  
f  
b  
a



后退到原路径中从源点能够到达的最远点



```
p ← s;  
While 源点没有被删除  
    u ← p.top;  
    if u <> t  
        if outdegree(u) > 0  
            设 (u,v) 为层次图中的一条边;  
            p ← p,v;  
        else  
            从 p 和层次图中删除点 u,  
            以及和 u 连接的所有边;  
    else  
        增广 p (删除了 p 中的饱和边);  
        令 p.top 为 p 中从 s 可到达的最后顶点;  
end while
```

## 二、Dinic

复杂度分析

建层次图:  $O(n*m)$

+

dfs 找增广路:  $O(n*n*m)$

- 层次图中最多找  $m$  次增广路
- 每次在 dfs 中最多前进  $n$  次, 花费  $O(n)$
- 每次修改流量花费  $O(n)$
- 一次 Dfs 复杂度为  $O(m*n)$

## 二、Dinic

- 复杂度  $O(n^2 * m)$
- 程序简短
- 对于较大规模的数据实际速度很快

### 三、Dinic 的应用

Noi2006 最大获利:

一共有  $N$  个通讯信号中转站，建立第  $i$  个通讯中转站需要的成本为  $P_i (1 \leq i \leq N)$ 。

另有  $M$  个用户群，第  $i$  个用户会使用中转站  $A_i$  和中转站  $B_i$  进行通讯，公司可以获益  $C_i$ 。  
( $1 \leq i \leq M, 1 \leq A_i, B_i \leq N$ )。

要求选择建立一些中转站，使得净收益最大。

## 例题一：Profit 最大获利 (NOI2006)

解题简述：

建立一张共有  $n+m+2$  个的顶点、 $3*m+n$  条边的二分图，求网络的最大流。

（参考《算法艺术与信息学竞赛》 p317）

# NOI2006: Profit 最大获利

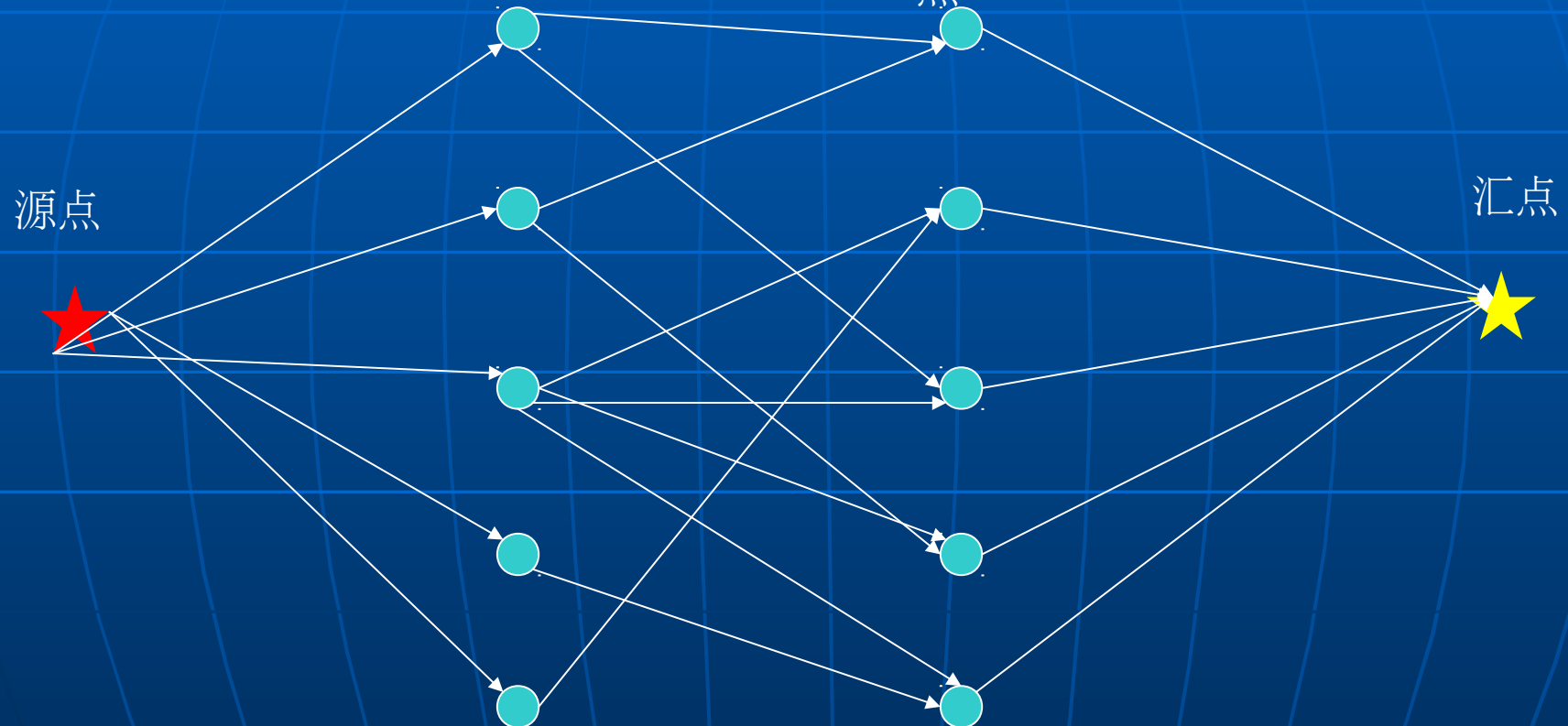
点数:  $N+M+2$

边数:  $M*3+N$      $N$  个点

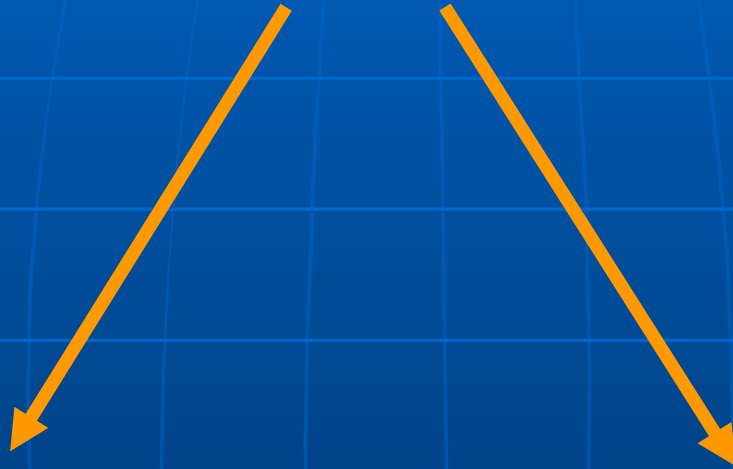
$M$  个点

$N \leq 5000$

$M \leq 50000$



Ac 它



贪心初始流

使用高效的网络流算法



# NOI2006: Profit 最大获利

## 算法的选择

	Test1~8	test9	test10
最短路径增值	<0.1s	>30s	>30s
Dinic	<0.03s	0.40s	0.37s
预流推进	<0.03s	0.53s	0.51s

$$\sigma = 0.02s$$

Dinic	<0.03s	0.22s	0.20s
-------	--------	-------	-------

## 例题二：矩阵游戏（2006 年江苏省选拔赛）

题目大意：对于一个  $n$  行、 $m$  列的 0-1 矩阵，规定在第  $i$  行中 1 的个数恰为  $R_i$  个（ $1 \leq i \leq n$ ）；在第  $j$  列中 1 的个数恰为  $C_j$  个（ $1 \leq j \leq m$ ）。每一行、每一列最多可以有一个格子指定为 0。

问是否存在一种满足条件的 0-1 矩阵。

数据范围：  $n, m \leq 1000$  每个测试点最多 10 组数据


























## 建立二分图网络流模型

- 把第  $i$  行作为点  $X_i$ ，从  $S$  至  $X_i$  连一条边，容量为  $R_i$
- 把第  $j$  列作为点  $Y_j$ ，从  $Y_j$  至  $T$  连一条边，容量为  $C_j$
- 若第  $i$  行第  $j$  列可以放 1，那么从  $X_i$  至  $Y_j$  连一条边，容量为 1

- 求网络最大流，判断流量值是否等于 1 的总数即可
- 边的总数达到了  $O(n*m)$
- 使用 MPLA 可以拿到 60% 的分数
- 使用 Dinic 可以拿到 80% 的分数

# 深入分析

- 首先不考虑指定为 0 的格子
- 因为将某 2 行或某 2 列交换，不影响问题的求解，我们不妨将  $R_i$  与  $C_i$  从大到小排序

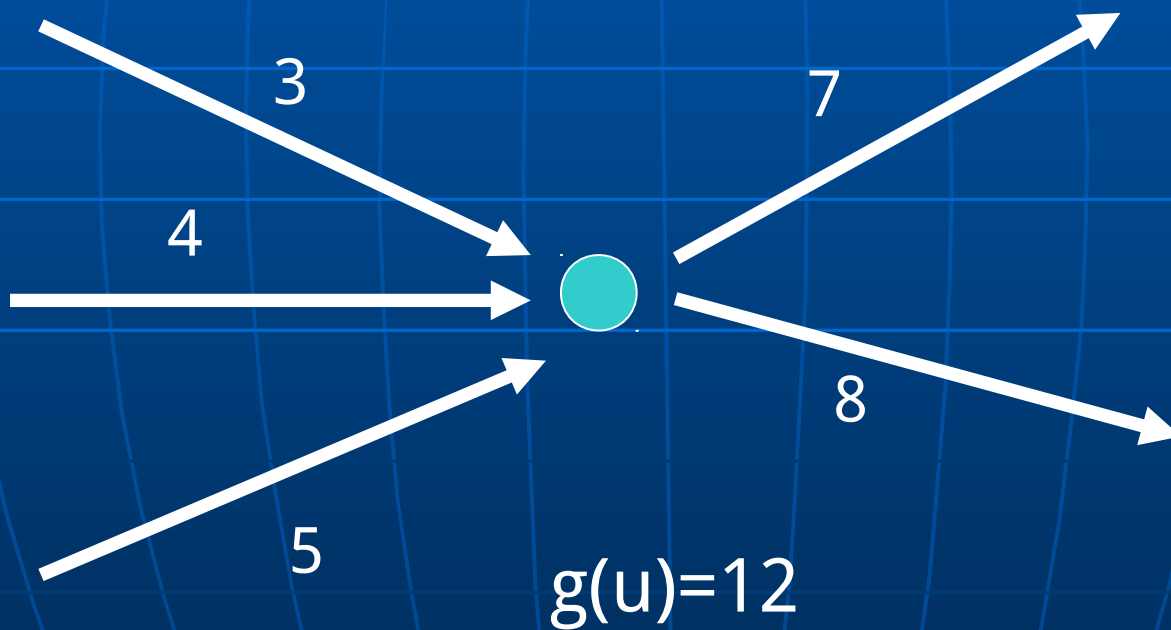
	6	5	5	3	3	0
6						
4						
4						
4						
2						
1						
1						

- 第一列，需要有 6 个 1，但有 7 行可以提供 1
- 将多余的 1 储存起来
- 第三列，需要 5 个 1，但只有 4 行可以提供 1

- 加入这个简单的判断后，MPLA 算法仍然只能过 60%
- 但是 Dinic 通过了 100% 的数据
- 其实这题的标准方法是贪心
- 使用高效的网络流算法节省了大部分的思考时间

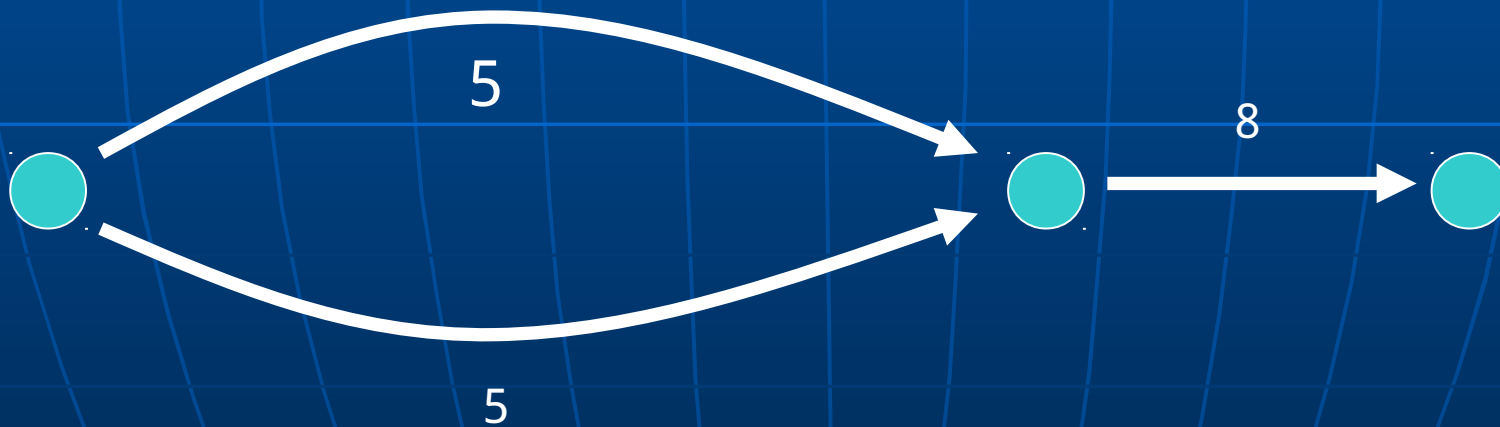
## 四、MPM

顶点  $u$  的通过量  $g(u)$  :  
剩余图中, 入边权值和与出边权重的较小值



## 四、MPM

- 增广时，每次找一个通过量最小的点  $v$ ，从点  $v$  向源点“推”大小为  $g(v)$  的流量
- 向汇点“拉”大小为  $g(v)$  的流量
- 尽量使剩余图中的边饱和





## 四、MPM

- 复杂度  $O(n^3)$
- 程序繁琐
- 实际效果并不理想

山是山，山非山；

其实困难与简单只是一线之隔

