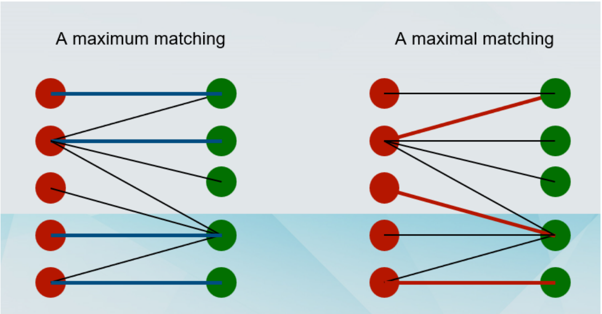
13 - Modeling with Flows

二分匹配（Bipartite Matching）

- 最大二分图匹配 或 在二分图 G 上的最大匹配
  - 匹配：边集合 E 的一个子集 M，使得 V 中的每个节点 v 在 M 中最多出现在一条边 e 上
  - 最大匹配：具有最大基数的匹配，即 $|M|$ 被最大化



最大匹配与极大匹配

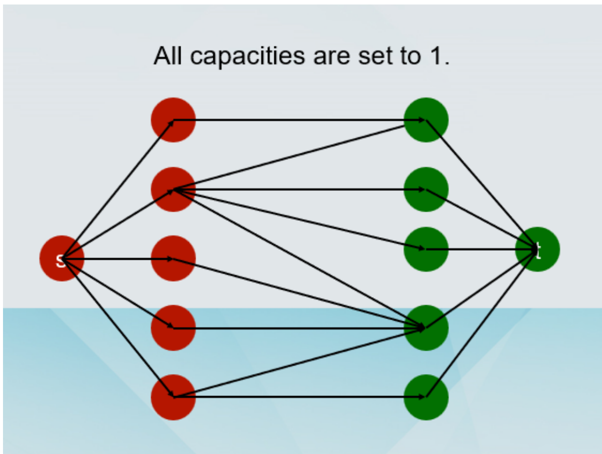
- 最大匹配（A maximum matching）：
  - 这个图展示了一种匹配，其中边的数量最多，即达到了可以在这个二分图中获得的最大匹配数
- 极大匹配（A maximal matching）：
  - 这个图展示了一种匹配，其中没有办法再添加边而不违反匹配的定义，但它不一定是最大的匹配

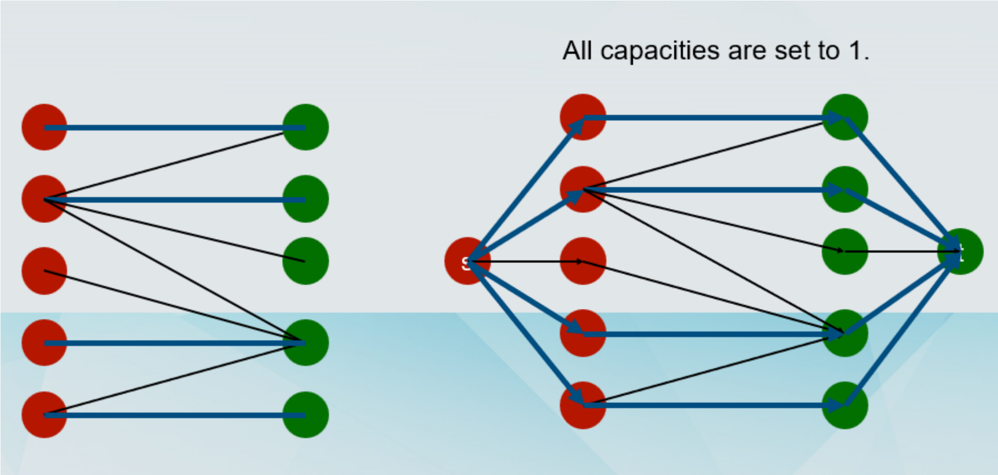
从匹配到流

假设在图 G（所有边的容量都设为 1）上有一个大小为 k 的匹配 M。那么在 G^f 中有一个值为 k 的流 f（从源节点到汇节点的总流量为 k）

- 考虑匹配 $M = \{(u_1, v_1), \ldots, (u_k, v_k)\}$
- 考虑流满足条件
  - $f(s, u_i) = f(u_i, v_i) = f(v_i, t) = 1$，对于所有的 $i = 1, \ldots, k$
  - $f(e) = 0$，其他情况

这是一个可行的流，并且其值显然为 k

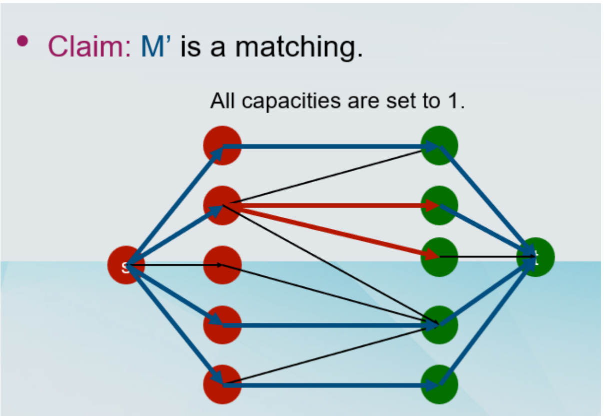


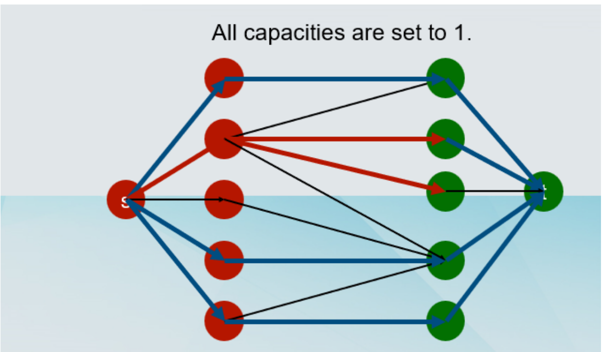


从流到匹配

假设在 G^f 中有一个值为 k 的流 f，那么在图 G 上有一个大小为 k 的匹配 M

- 对于一条边 e，$f(e)$ 要么是 0 要么是 1
- 考虑边集合 M'，其中 $f(e)=1$





最大流与最大匹配

- 图 G 中最大匹配 M 的带下等于图 G^f 中最大流 f 的值
- 最大匹配 M 的边是图 G^f 中从 A 到 B 的流动边
- 关键部分是整合定理

时间复杂度

- 根据 E-F 算法，是 $O(nm^2)$
- 根据 F-F，是 $O(mF)$

- F 的大小至多是 $\min(|A|,|B|)$
- 因此其实是 $O(nm)$

棒球淘汰问题

讨论了在棒球联赛中确定一个队伍是否还能有机会（甚至平局）赢得比赛的方法

情况 1

- **给定条件：**
  - 在棒球联赛中，有4支队伍，它们的胜场数如下：
    - New York: 92 胜
    - Baltimore: 91 胜
    - Toronto: 91 胜
    - Boston: 90 胜
- **假设条件：**
  - Boston赢下剩下的所有比赛
  - Baltimore和Toronto必须各赢下一场比赛
  - New York必须输掉剩下的所有比赛
  - Baltimore或Toronto必须再赢下一场比赛（Baltimore vs Toronto）
- **剩余比赛：**
  - NY vs BLT, NY vs TOR, BLT vs TOR, BLT vs BOS, TOR vs BOS
- **问题：**
  - Boston是否能（可能打成平局）排名第一？
  - 答案是No

情况 2

- **给定条件：**
  - 在棒球联赛中，有4支队伍，它们的胜场数如下：
    - New York: 90 胜
    - Baltimore: 88 胜
    - Toronto: 87 胜
    - Boston: 87 胜
- **剩余比赛：**
  - 赛季中还有12场比赛：
    - NY vs BLT
    - NY vs TOR（6场比赛）
    - BLT vs TOR
    - BOS vs ANY（4场比赛，共12场）
- **问题：**
  - Boston是否能（可能打成平局）排名第一？

一般情况

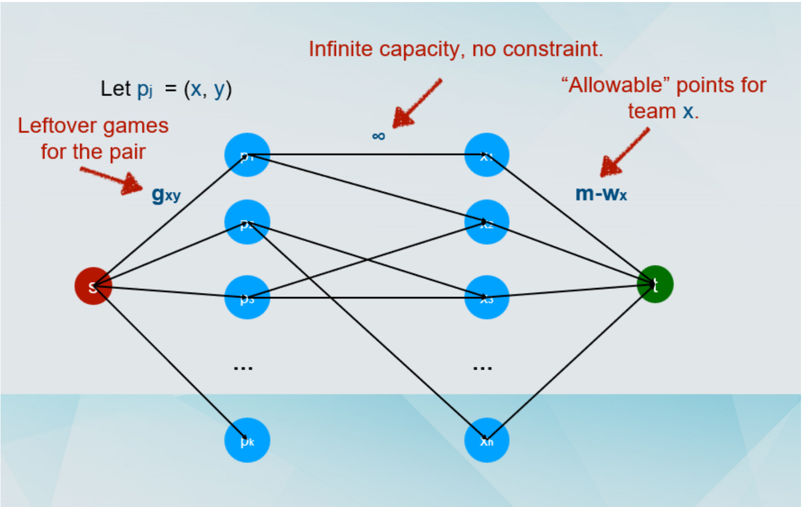
- **一般情况：**
  - 我们有一个球队集合 S

    - 对于集合中的每支队伍 x，当前的胜场数为 w_x
    - 对于集合中的每对队伍 x 和 y，它们还需对战 g_{xy} 场比赛
- 给定条件：
    - 我们指定一个队伍 z
- 问题：
    - z 能否赢得比赛（可能打成平局）？

从棒球淘汰到流

Observation（观察）：

- 如果有一种方式可以使得 z 队排名第一，那么也存在一种当 z 队赢得所有剩余比赛从而排名第一的方式
  假设：
- 假设最后 z 队有 m 胜
  我们在寻找什么？
- 是否存在一种分配方式，使得所有剩余的 g^* 场比赛（在其他队伍之间）分配后，没有队伍的胜场数超过 m 胜



其中

- $pj=(x,y)$：表示一对队伍 x 和 y 之间剩余的比赛，节点代表这对队伍
- $x_1,x_2,\ldots,x_n$：表示不同队伍的节点
- g_{xy}：容量，表示 x 和 y 之间剩余的比赛数量
- ∞：对赛对节点到队伍节点没有约束
- w_x：是队伍 x 现在已经赢得的场次
- $m-w_x$：表示队伍 x 允许再赢的场次（z 队伍最后有 m 胜）

我们需要找到网络中的最大流，这个最大流至少是 g^* 吗？

这个算法是有效的

假设算法给出肯定的答案

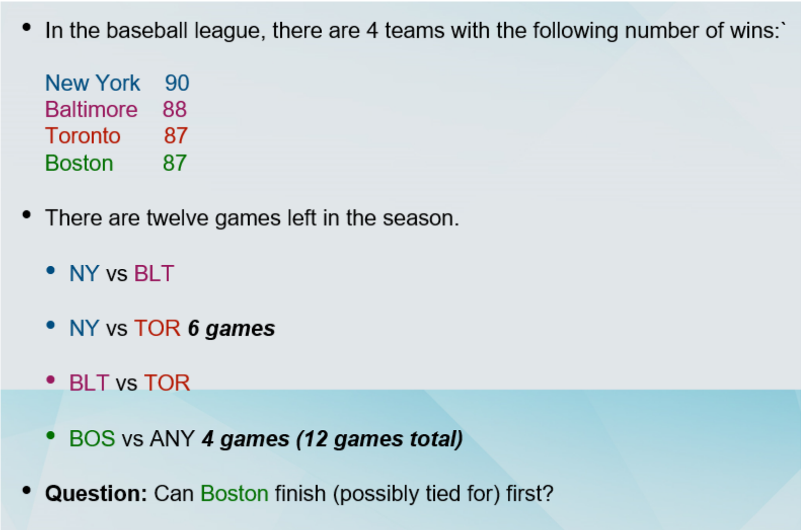
- 流的值等于剩余比赛的数量
    - 因为所有剩余的比赛都被考虑在内，并且我们正在寻找一种能够最大化流量的分配方式
- 并且我们可以得到：
    - 每对 (x,y) 将恰好进行 g_{xy} 场比赛

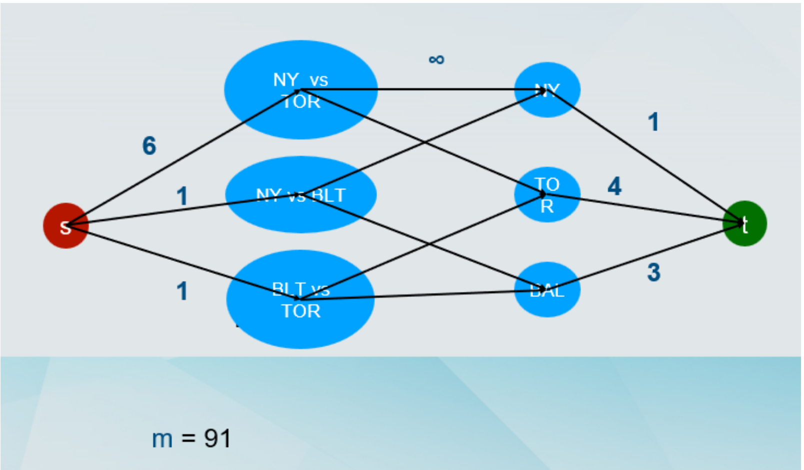
- 每个队伍 x 最多会赢 $m - w_x$ 场比赛
- z 队伍将会赢

假设算法给出的答案是否定的

- 最大流的值小于 g^*
  - 这是剩余比赛的总数
- 不可能在不使某个队伍 x 的胜场超过 $m - w_x$ 的情况下进行所有剩余比赛
  - z 不会赢

例子



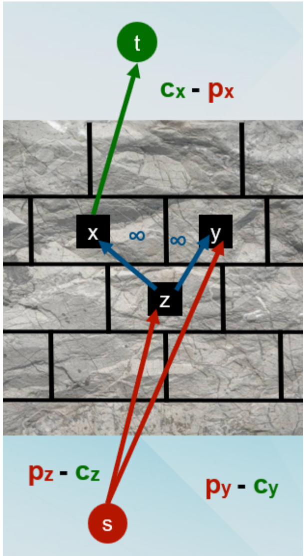


露天采矿

- 采矿目标：
  - 我们从地表提取土壤块，试图找到黄金
- 每个采矿块的属性：
  - 值 p_z：每个我们开采的土壤块 z 都有一个值 p_z
  - 开采成本 c_z：每个我们开采的土壤块 z 都有一个开采成本 c_z
- 约束条件：
  - 我们不能开采一个土壤块 z，除非我们已经开采了其上方的两个土壤块 x 和 y

4. 目标：

- 我们希望赚取尽可能多的钱



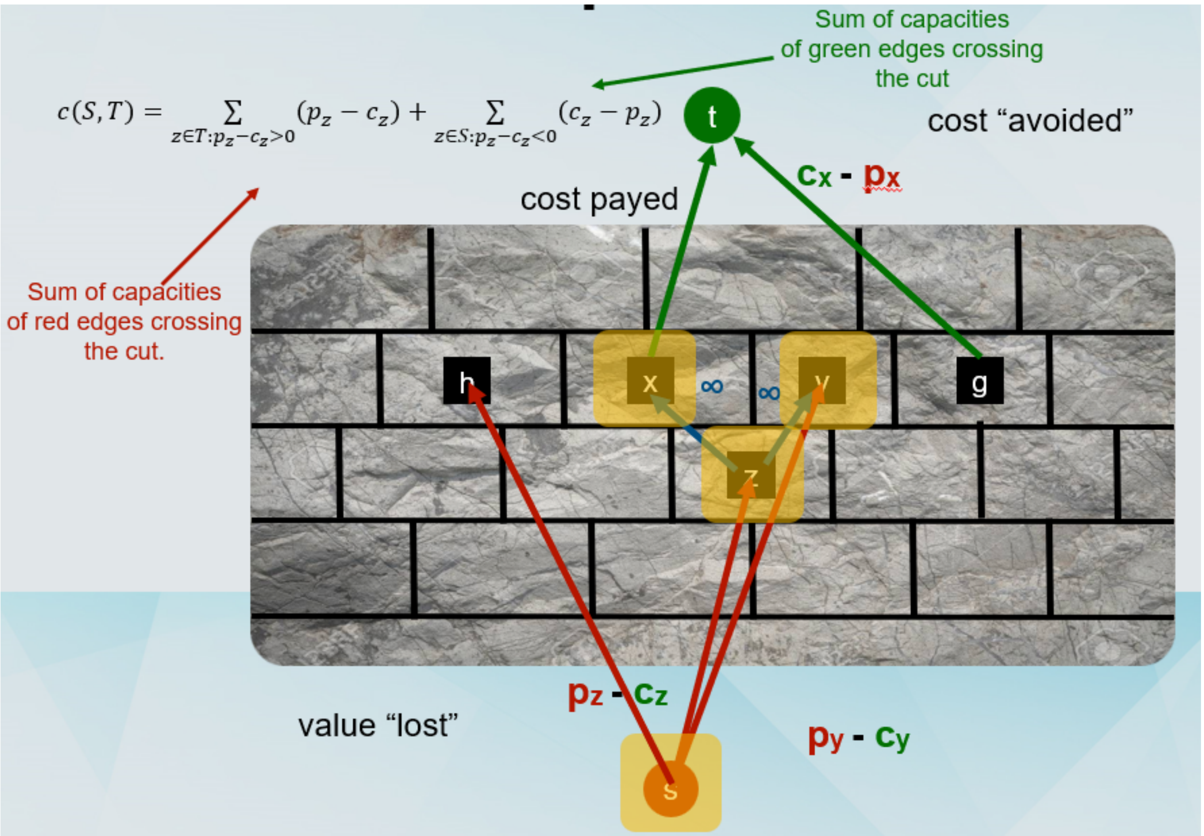
从采矿到割

考虑 (S,T) 的一个割 C

- 如果 C 的容量最小，则S 或 T 必须包含所有与无限容量边相连的节点
    - 如果 S 或 T 包含 z，他必须包含 x 和 y
        - 这是因为开采 z 的前提是我们已经开采了其上方的块
- 我们将开采 $S-\{s\}$
    - 从集合 S 中去除源节点 s 后的所有节点表示我们将要开采的土壤块
    - 这种方式保证了开采的可行性

成本公式：

$$c(S,T) = \sum_{z \in T:\, p_z - c_z > 0} (p_z - c_z) + \sum_{z \in T:\, p_z - c_z < 0} (c_z - p_z)$$



采矿集合的最优性

切割成本：

$$c(S,T) = \sum_{z \in T:\, p_z - c_z > 0} (p_z - c_z) + \sum_{z \in T:\, p_z - c_z < 0} (c_z - p_z)$$
$$c(S,T) = \sum_{z \in T:\, p_z - c_z > 0} (p_z - c_z) - \sum_{z \in T:\, p_z - c_z < 0} (p_z - c_z)$$

添加与减去常数项：

$$\sum_{z \in V:\, p_z - c_z > 0} (p_z - c_z)$$

简化：

$$c(S,T) = \sum_{z \in V:\, p_z - c_z > 0} (p_z - c_z) - \sum_{z \in S} (p_z - c_z)$$

最后的式子中，左边是一个常数，右边是我们采矿的利润，V 是所有节点的集合
这证明了我们选择的采矿集合是最优的

采矿总结

- 构建流网络
- 运行 F-F 找到最大流
- 使用 G^f 找到最小割
- 开采在 (S,T) 中 S 部分的块