**为什么能用最大流解决棒球赛问题？**

**目标**：对于球队x，检查是否存在胜负分配使得x能够第一

**剩余比赛的胜负分配** 类似于 **流量分配**， 每场比赛的胜利只能分配给一个球队，并且每个球队的额外胜场有限制，

流量：每个单位流量代表一个胜场。从s到t的路径表示一个胜场从比赛分配到球队的过程

最大流算法核心作用：验证可行性。

源点的净流 = 剩余比赛数

汇点的净流 =

最大流值F 如果等于 剩余总比赛数T ——所有分区内比赛的结果均可分配，且满足最大胜场限制，因此x可能夺冠

如果F < T ,说明在x的最大胜场限制下无法分配所有比赛，因此x会被淘汰。

**模型设计：**

|  |  |  |
| --- | --- | --- |
| *s*→比赛节点 *i*-*j* | *g*[*i*][*j*] | 该场比赛的总数量 |
| 比赛节点 *i*-*j*→球队节点 *i* | ∞ | 比赛胜场可分配给球队 *i* |
| 比赛节点 *i*-*j*→球队节点 *j* | ∞ | 比赛胜场可分配给球队 *j* |
| 球队节点 *i*→*t* | *w*[*x*]+*r*[*x*]−*w*[*i*] | 球队 *i* 最多可增加的胜场数（不超过 *x* 的最大胜场） |

**最大流算法：**

**Ford–Fulkerson 增广**

依赖容量值：若容量为整数，**最坏时间复杂度为 O(E⋅f)**，其中 f 是最大流值。

**Edmonds–Karp 算法**

​**改进**​：用 ​**BFS**​ 寻找最短增广路径（按边数最少）。

​**时间复杂度**​：***O*(*V E ^* 2)。**

​**优势**​：避免 DFS 可能陷入长路径的问题。

**证明**：每次增广路径长度单调递增，最多 *O*(*VE*) 次增广。

**Dinic算法**

##### **改进​：**

##### 分层图（Level Graph）​​：BFS 构建节点的层次 *d*(*u*)。

##### ​阻塞流（Blocking Flow）​​：在分层图上 DFS 找到无法再增广的流。

##### ​**时间复杂度​：**

##### **一般：*O*(*V*2*E*)。**

##### **使用动态树优化：*O*(*VE*log*V*)。**

##### ​**优势​：**

##### 单次迭代找到多条增广路径。

##### 分层图减少无效搜索。

**Push-Relabel 算法族**

**核心思想​：模拟水流扩散（与增广路径法不同）。**

**​预流（Preflow）​​：允许节点流入 > 流出（超额流 e(u)）。**

**​高度标签（Height Label）​​：h(u) 表示节点高度，水流从高向低流。**

**​操作​：**

**​Push​：将超额流推送给更低邻居。**

**​Relabel​：若无更低邻居，增加高度。**

|  |  |  |
| --- | --- | --- |
| **​Highest-Label​** | ***O*(*V*2*E*​)** | **优先处理最高节点（实际最快）** |

**ISAP**

* **改进​：Dinic 的优化版，只构建一次分层图，动态更新高度。**
* **​时间复杂度​：*O*(*V*2*E*)**

**实际效率常优于 Dinic**