极客时间算法训练营
第六课
深度优先搜索、广度优先搜索

李煜东

《算法竞赛进阶指南》作者

目录

本课的重要性

- 本课将第一次归纳总结状态、状态空间和把问题抽象为树或图的方法
- 搜索是解决一切问题的万金油算法，众多没有多项式时间解法的问题都需要靠搜索求解
- 学会定义搜索框架，将极大地帮助你学习动态规划和图论算法
- 搜索题是训练代码能力最有效的题目类别

状态与状态空间

状态

什么是状态?

- 题面中涉及的所有数学信息
- 你在纸上人力计算时，关注的所有数据
- 一个函数访问的所有变量

例如最简单的计票问题

- 给 n 个名字，统计每个名字出现了多少次

你在纸上画“正”字统计的时候，关注了哪些数据?

- 名字（n 个字符串）
- 统计到哪个名字了（第 $1 \leq i \leq n$ 个名字）
- 画的“正”字（一个用于计数的数据结构，例如 Hash Map）

状态

写成程序：

```
for (int i = 0; i < n; i++)
    count[names[i]]++;
```

访问的变量：

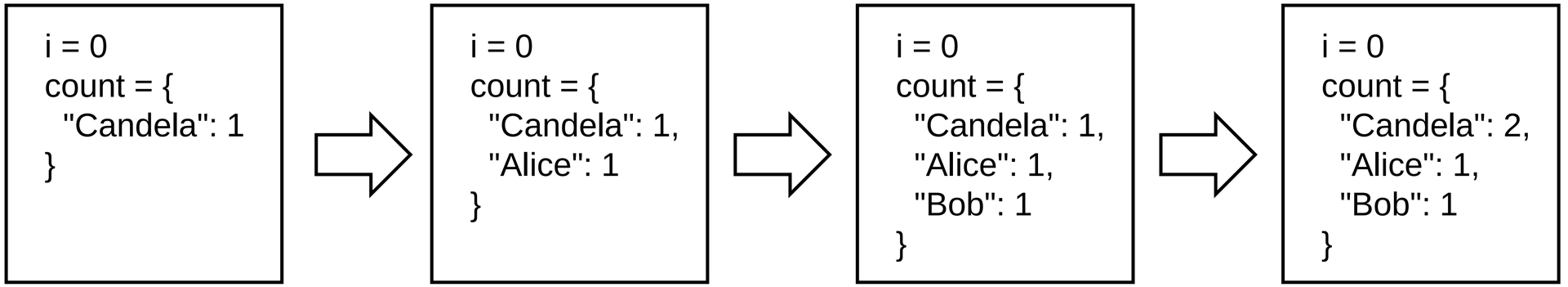
- ~~n~~
- ~~names~~
- i
- count

我们只关注其中动态变化的数据，也就是 i 和 count

状态

状态，就是程序维护的所有动态数据构成的集合

names = ["Candela", "Alice", "Bob", “Candela"]



状态空间 → 图

所有可能状态构成的集合就是一个问题的状态空间

把状态作为点，如果从一个状态可以到达另一个状态，就连一条边

这样就把整个状态空间抽象为了一张有向图

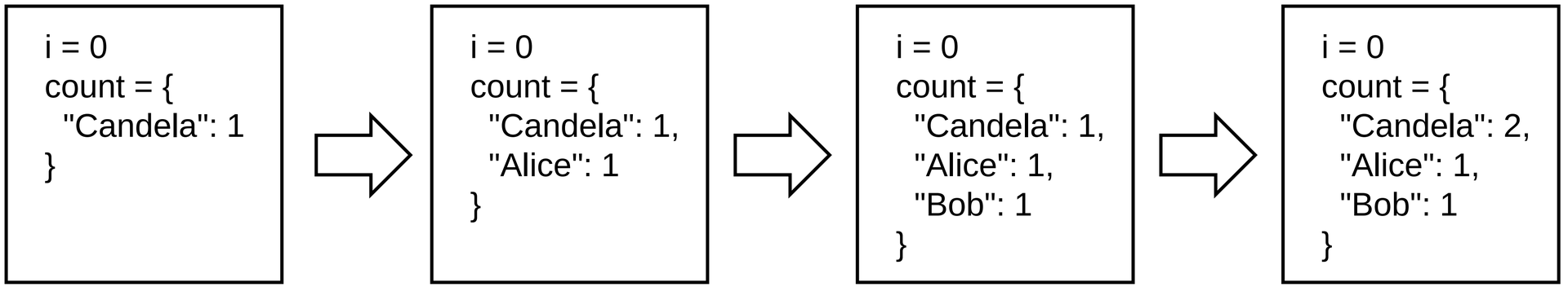
对问题的求解，就是对这张图的遍历

计票问题的状态空间由 n 个状态组成

可以看作一张 n 个点，n - 1 条边的有向图

整张图是一条链，自然就可以用一维循环解决了

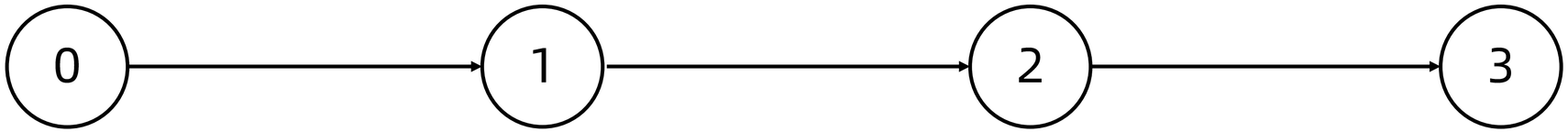
状态的简化



在这里，当 i 固定以后，counts 其实已经被决定了，它没有其他可能

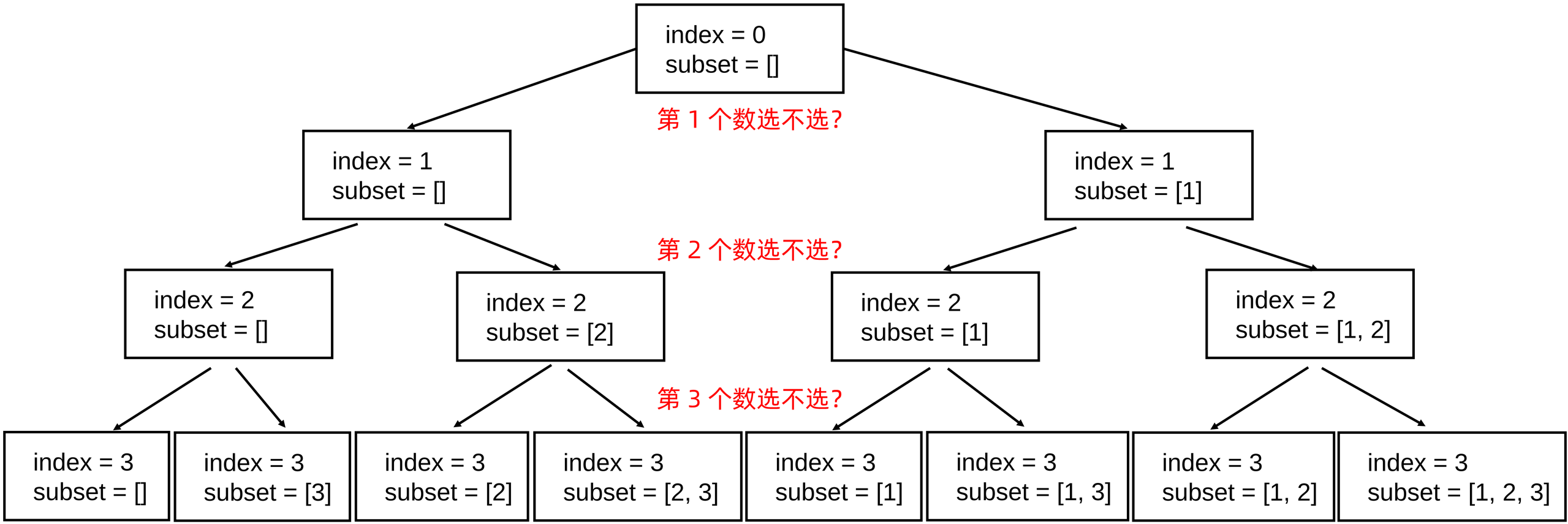
counts 对于 i 来说，只是一个附加信息，并不影响状态的规模

把可以由其它数据决定的信息从状态中排除，得到的最简状态决定了问题的复杂度



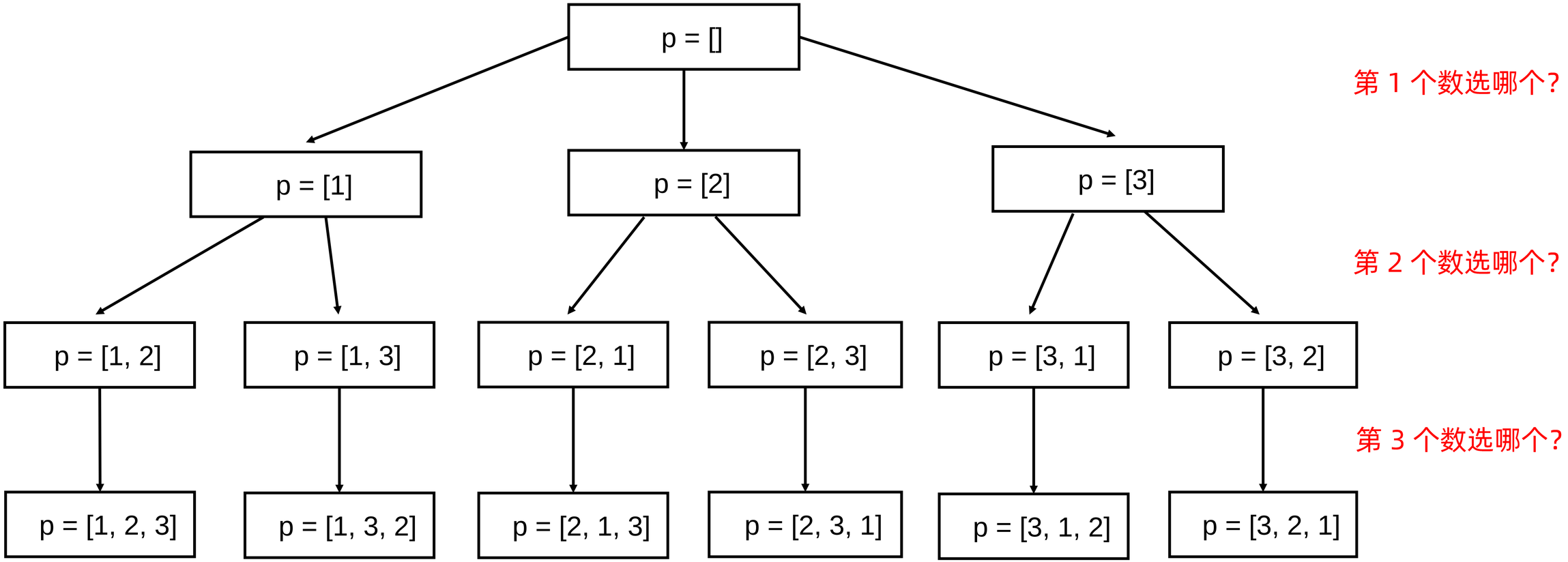
指数型状态空间（子集）

set = [1, 2, 3]， subsets = [[], [1], [2], [3], [1, 2], [1, 3], [2, 3], [1, 2, 3]]



排列型状态空间（全排列）

permutations = [[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]



深度优先搜索（DFS）
广度优先搜索（BFS）

搜索

搜索就是采用直接遍历整个状态空间的方式寻找答案的一类算法

根据遍历状态空间（图）方式的不同，可分为

- 深度优先搜索（DFS, depth first search）
- 广度优先搜索（BFS, breadth first search)

一般来说，每个状态只遍历一次

所以当状态空间是“图”而不是“树”时，要判重（记忆化）

搜索

搜索题的解题步骤：

1. 纸上模拟，提取信息
2. 定义状态
3. 确定遍历顺序（DFS、BFS）
4. 定义搜索框架
    - 如果是DFS，状态作为参数，确定递归边界，注意还原现场
    - 如果是BFS，状态用队列保存
    - 考虑是否需要判重
5. 程序实现

实战

电话号码的字母组合

https://leetcode-cn.com/problems/letter-combinations-of-a-phone-number/

N 皇后

https://leetcode-cn.com/problems/n-queens/

实战

岛屿数量

https://leetcode-cn.com/problems/number-of-islands/

被围绕的区域（Homework）

https://leetcode-cn.com/problems/surrounded-regions/

实战

最小基因变化

https://leetcode-cn.com/problems/minimum-genetic-mutation

实战

矩阵中的最长递增路径

https://leetcode-cn.com/problems/longest-increasing-path-in-a-matrix/

DFS vs BFS

DFS 更适合搜索树形状态空间

- 递归本身就会产生树的结构
- 可以用一个全局变量维护状态中较为复杂的信息（例：子集方案、排列方案）
- 不需要队列，节省空间

BFS 适合求“最小代价”、“最少步数”的题目

- BFS 是按层次序搜索，第 k 步搜完才会搜 k + 1 步，在任意时刻队列中至多只有两层

在状态空间为一般的“图”时（需要判重），DFS 和 BFS 差不多

- 按个人喜好选择就可以了

THANKS