极客时间算法训练营
第二十一课
实战技巧、总结与回顾

李煜东

《算法竞赛进阶指南》作者

目录

位运算

为什么需要位运算？

机器采用二进制对数值进行表示、存储和运算

在程序中恰当地使用二进制，可以提高运行效率

十进制与二进制的转化

- 短除法
- 幂次和
- $6_{(10)} = 1*2^2 + 1*2^1 + 0*2^0 = 110_{(2)}$

位运算符

含义	运算符	示例
按位与	&	$\begin{matrix}1011\\0011\end{matrix} \rightarrow 0011$
按位或	|	$\begin{matrix}1011\\0011\end{matrix} \rightarrow 1011$
按位取反	~	$0011 \rightarrow 1100$（4位二进制数）
按位异或	^	$\begin{matrix}1011\\0011\end{matrix} \rightarrow 1000$（相同得0 不同得1）
左移	<<	$0011 \rightarrow 0110$
右移	>>	$0011 \rightarrow 0001$

异或运算（xor）

相同为 0，不同为 1

也可用“不进位加法”来理解

异或运算的特点与应用：

- x ^ 0 = x
- x ^ x = 0
- 结合律：a ^ b ^ c = a ^ (b ^ c) = (a ^ b) ^ c
- 成对变换：x ^ 1（0^1=1，1^1=0；2^1=3，3^1=2；4^1=5，5^1=4）
- 交换两个数：a ← a ^ b，b ← a ^ b，a ← a ^ b

指定位置的位运算

二进制数的最右边为第 0 位

获取 x 在二进制下的第 n 位（0 或者 1）：(x >> n) & 1

将 x 在二进制下的第 n 位置为 1：x | (1 << n)

将 x 在二进制下的第 n 位置为 0：x & (~(1 << n))

将 x 在二进制下的第 n 位取反：x ^ (1 << n)

将 x 最右边的 n 位清零：x & (~0 << n)

将 x 最高位至第 n 位（含）清零：x & ((1 << n) - 1)

位运算实战要点

判断奇偶：

- x % 2 == 1 → (x & 1) == 1
- x % 2 == 0 → (x & 1) == 0

除以 2 的幂次：

- x / 2 → x >> 1
- mid = (left + right) / 2; → mid = (left + right) >> 1;

lowbit：

- 得到最低位的 1：x & -x 或 x & (~x + 1)
- 清零最低位的 1：x = x & (x - 1)

实战

位 1 的个数

https://leetcode-cn.com/problems/number-of-1-bits/

2 的幂

https://leetcode-cn.com/problems/power-of-two/

颠倒二进制位

https://leetcode-cn.com/problems/reverse-bits/

比特位计数

https://leetcode-cn.com/problems/counting-bits/

实战：位运算优化

Pow(x, n) —— 快速幂

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

N 皇后

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

解数独

https://leetcode-cn.com/problems/sudoku-solver/

期末串讲

高频主题

计算机科学基础

- 基本数据结构、树/图、DFS/BFS
- 第 1、2、3、5、6、9、10、16、18 课

状态空间与子问题划分

- 递归、分治、动态规划的设计
- 第 4、11、12、13、14 课

进阶主题

图论算法

- 第 16 课

高级数据结构

- 第 7、20 课

高级搜索

- 第 19 课

高端技巧

二分答案：求解转化为判定

- 第 8 课

优化技巧

- Pre-calculation：空间换时间（前缀和、记忆化等）
- 决策、候选集合、滑动窗口系列的多种优化方法，固定一端+移动一端+消除冗余的思想
- 第 14、17 课

离线批处理，关键事件思想

- 第 21 课

实战

天际线问题

https://leetcode-cn.com/problems/the-skyline-problem/

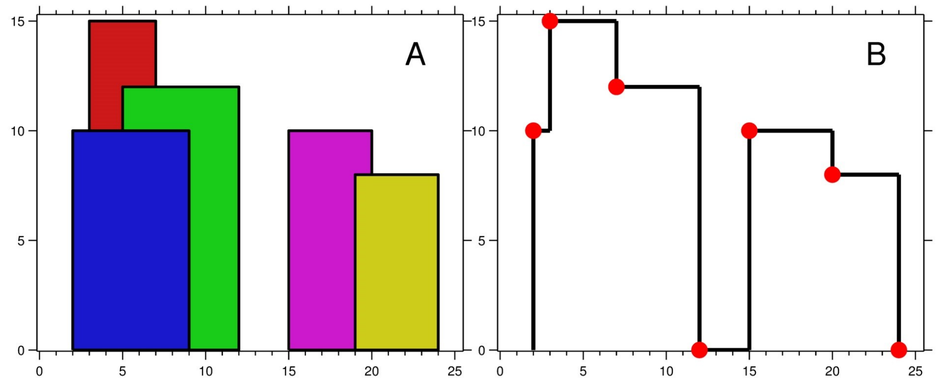
批处理 + 差分思想

每个建筑物拆成两个事件：

- 在 left[i] 处，height[i] 开始可用（插入）
- 在 right[i] 处，height[i] 不再可用（删除）

任何时刻只需要判断可用的 height 里哪个最高（查询）

排序 + 数据结构（map，或懒惰删除的二叉堆）



vs. 支持区间修改的线段树

实战

包含每个查询的最小区间

https://leetcode-cn.com/problems/minimum-interval-to-include-each-query/

批处理所有询问，得到答案后再统一返回

每个区间看作一个“限时任务”，在 left[i] 时产生，在 right[i] 时消失，任务代价是 length[i]

每个询问看作一个在 queries[i] 时发生的“询问”事件 —— 哪个可做的任务代价最小?

排序 + 堆 or map 维护

vs. 在线处理询问需要支持区间修改的线段树

THANKS