模拟

crazy_cloud

SJTU

2021 年 10 月 1 日

目录

目录

例题

例 (NOIP 2011 - 瑞士轮)

$2n$ 名编号为 $1 \ldots 2n$ 的选手共进行 R 轮比赛。每轮比赛开始前，以及所有比赛结束后，都会按照总分从高到低对选手进行一次排名。
选手的总分为第一轮开始前的初始分数加上已参加过的所有比赛的得分和。总分相同的，约定编号较小的选手排名靠前。
每轮比赛的对阵安排与该轮比赛开始前的排名有关：第 1 名和第 2 名、第 3 名和第 4 名、……、第 $2k-1$ 名和第 $2k$ 名、……、第 $2n-1$ 名和第 $2n$ 名，各进行一场比赛。
每场比赛胜者得 1 分，负者得 0 分。也就是说除了首轮以外，其它轮比赛的安排均不能事先确定，而是要取决于选手在之前比赛中的表现。
现给定每个选手的初始分数及其实力值，试计算在 R 轮比赛过后，排名第 Q 的选手编号是多少。我们假设选手的实力值两两不同，且每场比赛中实力值较高的总能获胜。
$1 \leq n \leq 10^5, 1 \leq R \leq 50$

友情提醒：$O(Rn \log n)$ 会超时。

瑞士轮

每轮的失败者和胜利者内部有序，使用归并即可。
时间复杂度 $O(n \log n + Rn)$。

小技巧

使用归并来处理有序的子序列，可以降低时间复杂度。

例题

例 (小 W 学物理)

在二维坐标系中放了 n 面镜子（镜子坐标绝对值不超过 m），镜子均与坐标轴成 45° 角，所以一共有两种类型 / 和 \。原点不会有镜子，任意一点最多只有一面镜子。
镜子两个面都能反光，而中间不透光，例如，对于一个 / 型镜子，下方向射入的光线会被反射到右方向，左方向射入的光线会被反射到上方向。
现在有一条光线从原点沿 x 轴正方向射出，求走过 T 路程后所在位置。

直接模拟光线的路径。
循环了怎么办？第一次重复的长度就是循环节，把 T 用循环节模掉就好了。

例题

例 (小 W 学物理)

在二维坐标系中放了 n 面镜子（镜子坐标绝对值不超过 m），镜子均与坐标轴成 45° 角，所以一共有两种类型 / 和 \。原点不会有镜子，任意一点最多只有一面镜子。
镜子两个面都能反光，而中间不透光，例如，对于一个 / 型镜子，下方向射入的光线会被反射到右方向，左方向射入的光线会被反射到上方向。
现在有一条光线从原点沿 x 轴正方向射出，求走过 T 路程后所在位置。

小技巧

如果模拟过程会无限进行下去，可以考虑找循环节加速。

小技巧

网格图内的移动模拟，可以把四个方向（甚至八个）的位移向量存进数组，方便枚举。

例题

例 (NOIP 2006 - 作业调度方案)

我们现在要利用 m 台机器加工 n 个工件，每个工件都有 m 道工序，每道工序都在不同的指定的机器上完成。每个工件的每道工序都有指定的加工时间。

每个工件的每个工序称为一个操作，我们会给定对于个操作的一个安排顺序。

每个操作的安排都要满足以下的约束条件：首先，对于每一个工件，每道工序必须在它前面的工序完成之后才能开始；其次，同一时刻每一台机器之多能加工一个工件。另一方面，在安排后面的操作时，不能改动前面已安排的操作的工作状态。

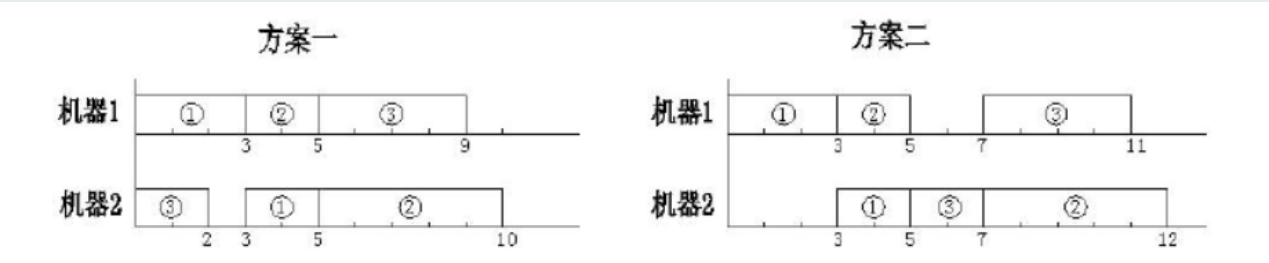
由于同一工件都是按工序的顺序安排的，因此，只按顺序给出工件号，就能推出完整的的安排顺序。

还要注意，“安排顺序”只要求按照给定的顺序安排每个操作。不一定是各机器上的实际操作顺序。在具体实施时，有可能排在后面的某个操作比前面的某个操作先完成。

例题

例 (NOIP 2006 - 作业调度方案)

假如，取 $n = 3, m = 2$，已知工件 1 各工序的机器号/加工时间为：$1/3$、$2/2$，工件 2：$1/2$、$2/5$，工件 3：$2/2$、$1/4$。
则对于安排顺序 112332，下图的两种实施方案都是正确的，但所需要的总时间分别是 10 与 12。



为了让实施方案唯一，我们约定安排过程中，在保证约束条件的前提下，操作尽量靠前插入空档。并且，我们还约定，如果有多个空档可以插入，就在保证约束条件的前提下，插入到最前面的一个机器的空档。这样，方案即唯一，如上面的例子只有方案一是合法的。
请计算完成所有任务需要的总时间。
$1 \leq m, n < 20$，每个操作耗时不超过 20。

作业调度方案

最大难度在于语文（雾）
提供一个参考实现思路：用布尔数组标记每个机器每个时间是否空闲，用一个数组储存每个机器前面的上一个工序的最后完成时间。
找空档直接暴力扫描即可。
如果一次操作的时间上限很大怎么办？
我们不能通过布尔数组来标记机器的空闲了，但是我们可以用双向链表之类的结构，按时间顺序存储每个机器上的工件，那么空档其实就是相邻的两个工件之间的以及最后空出来的。

例题

例 (COCI 2006-2007 #6 - KAMEN)

在一个 $r \times c$ 的方阵里，有些点是.，表示为空；有些点是 X，表示这里是一堵墙。
我们可以认为这个方阵在竖直方向放置。
有一个人在这 c 列的第一行会抛下 n 块石头，用 O 来表示。如果这一个石头由于重力作用会向下滚动。具体来说，就是会从第一行向最后一行滚动，规则如下：
如果下一个格子是空格，那么向下运动一格。
如果下一个格子是墙或者已经到了第 r 行，则停止滚动并停在原处。
如果下一个格子是一块停止的石头，则如果在左侧和左下方为空格时首选滚动到左侧的那一行，否则如果右侧和右下方为空格，则滚动到右侧的那一行。如果两侧都不为空，则石头静止不在移动。
只有前一块石头永久静止后，下一块石头才会被抛下。
请你输出最终方阵的状态。
$1 \leq r \leq 3 \times 10^4, 1 \leq c \leq 30, 1 \leq n \leq 10^5$

KAMEN

直接模拟时间复杂度是 $O(nr)$ 的，显然无法通过。
可以发现对于同一列，相邻两次扔的石头，路径会有一段前缀重合。
考虑对于每一列，保存上一次扔的石头下落的路径。
当我们从某一列扔下石头时，我们从上一次路径的结尾开始，先不断回退直到当前格子没有被石头占领，然后模拟下落。
时间复杂度怎么计算?
单独考虑每一列，每个格子一旦从路径中删除，永远不会出现第二次。
于是每个格子最对被每一列处理一次，时间复杂度 $O(rc^2 + n)$。

小技巧

有时需要观察题目性质来加速模拟，比如这题利用了路径的重复，跳过了很多不必要的步骤。
这里还用到了基础的均摊分析，即将每个元素的贡献单独考虑。

目录

差分与前缀和

在允许离线的前提下，区间加减操作可以很方便地执行。
我们先对整个数组差分。这样，对于区间 $[l, r]$ 的加 x 操作，就可以看作对 l 的加 x 和对 $r+1$ 的减 x。
最后我们只需要做一次前缀和，将差分数组还原为原数组。这样我们就在线性的时间里，求出了执行完这些操作之后的数组。

双指针是用于维护可行区间，查找元素的一个技巧。

例 (统计三角形)

给定一个序列 $\{a_n\}$，统计里面有多少个三元组能组成合法的三角形边长集合。
要求做到 $O(n^2)$。

将所有边按长度排序。
枚举长度最小的边，然后再枚举次小的边，两个指针动态维护第三条边的取值范围。

离散化

有时候元素值域很大，但我们只关心它们的相对顺序。
这时候我们可以用它们排序后的名次代指元素本身。
其实更常用于动态规划和数据结构中。

预处理

有些计算时间长，查询频次多，但是在整个算法流程中没有发生过变化的内容，在空间允许的前提下，可以提前计算，再完整或部分地存储下来，优化时间复杂度。

Worklist 算法

所谓工作表算法，就是用队列记录每个时刻活跃的事件。
即，一个操作要进行，依赖于其他操作产生的后果的触发。
如果我们每个模拟时刻都检查每一个事件是否活跃，可能会消耗大量的时间。
使用队列，每次出队前，将这个事件可能影响到的事件一一检查，如果被激活则放入队列。往往会得到线性复杂度的算法。
本质是宽搜。

STL

模拟题往往会需要数据结构辅助，例如查询集合极值、区间极值之类的。
善用 STL 能避免手动实现一些特别模板的数据结构和算法。
常用的有

- sort, lower_bound, upper_bound, nth_element, next_permutation, ...
- priority_queue, set/multiset, map, vector, bitset, stack, queue, unordered_map, deque, ...