2023届CSUST ICPC集训队选拔赛题解

前置芝士汇总

- 算法基础
  - `1` 自定义排序
  - `2` 二分算法
  - `1` 离线处理
- 动态规划
  - `1` 概率类dp
  - `1` 计数类dp
  - `1` 0/1背包
- 数据结构
  - `2` st表(RMQ) / 跳表(二进制)
  - `1` DFS 序 / 树链剖分
  - `2` 线段树 / 树状数组
- 图论
  - `1` 最短路
  - `1` DFS / BFS
  - `1` 强连通分量
- 数学
  - `1` 素数筛&质因数分解
  - `1` 拓展欧几里得算法
  - `1` 概率&期望
  - `1` 乘法逆元

目录

干饭协会的入场券 🎫

前置芝士：素数筛，质因数分解。

由**唯一分解定理**可知：

我们可以将 $\prod_{i=1}^{n} a_i$ 分解成质数幂的乘积：

$$\prod_{i=1}^{n} a_i = p_1^{\alpha_1} \times p_2^{\alpha_2} \times \cdots \times p_k^{\alpha_k}$$

则其所有约数的个数就是：

$$(\alpha_1 + 1) \times (\alpha_2 + 1) \times \cdots \times (\alpha_k + 1)$$

因此直接质因数分解即可（用质数筛将所有不超过 $\lfloor \sqrt{10^7} \rfloor$ 质数预处理出来会跑得更快一点）。

时间复杂度 $\mathcal{O}\left(\sum_{i=1}^{n} \sqrt{\frac{a_i}{\log a_i}}\right)$，足以通过本题。

干饭协会的副本侠 😜

前置芝士：等比数列求和，乘法逆元，概率&期望，dp。

这类题通常被称为 概率dp，往往同时有正推和逆推两种做法。

首先，优先释放击杀概率更高的技能总是最优的。

这里考虑正推：

- 考虑当前有 i 只怪兽时，需要释放多少次技能才能将怪兽数量减少呢？
- 类似于题面中提示所考虑的特殊情形 $p_1 = p_2 = \frac{1}{2}$，对于每只怪兽，我们都可以计算杀死这只怪兽的期望步数。
- 但当 $p_1 \neq p_2$ 时，问题就变得难以处理。下面给出一种处理方法：
- 将 2 个技能看作一个整体，假设连续释放了 k 次 p_1 和 p_2 也没有杀死怪兽的概率是 P_0（共释放 $2k$ 次技能），则第 $2k+1$ 步杀死怪兽的概率为 $p_1 P_0$，第 $2k+2$ 步杀死怪兽的概率为 $p_2(1-p_1)P_0$。
- 对 $2k+1$ 步、$2k+2$ 步分别求期望，求和即为杀死一只怪兽需要释放技能次数的期望。显然求和式子为等比数列，代入求和公式即可。
- 因为先释放的技能可能是技能 1 或技能 2，所以用 dp 数组存概率时，可以加一维表示先释放的技能；也可以保持技能顺序不变进行转移，一旦释放第奇数次杀死了怪兽，就再释放一次，从而保证了单一的递推式。

总时间复杂度 $\mathcal{O}\left(\sum n\right)$，空间复杂度 $\mathcal{O}(\max\{n\})$（或使用滚动数组达到 $\mathcal{O}(1)$），足以通过本题。

bonus. 使用矩阵快速模幂可将时间复杂度降至 $\mathcal{O}\left(\sum \log n\right)$（但常数比较大）。

> 题目灵感来源：[《2023 年上海市大学生程序设计竞赛 - 一月赛》- Elden Remembrance]
> 借鉴了原题目中“一定概率杀死”的 idea，感觉很有意思。
> 原题为 PVP 模式 - 仨人互丢技能自相残杀。
> 本题为 PVE 模式 - 打副本杀怪兽惩奸除恶。

干饭协会的双子牌🎴

前置芝士：计数，dp。

对于每张牌，分 a_i 或 b_i 在右边两种情况，再 `dp` 转移：只要前一张牌右边的数字与当前这张牌左边的数字不相等，就可以累计起来。

总时间复杂度 $\mathcal{O}\left(\sum n\right)$，空间复杂度 $\mathcal{O}(\max\{n\})$（滚动数组可只开 $\mathcal{O}(1)$ 空间）。

> 题目灵感来源：[「AtCoder Beginner Contest 291」 D - Flip Cards]
>
> 这题只做了一点小改动，思路和做法都差不多——
> 原题是两面都有数字，只有一面朝上且相邻不相等。

干饭协会的彩虹猫😺

前置芝士：思维。

先说结论：悠米逃离失败当且仅当初始怪兽位于悠米的**正右上方**。

为了便与描述，这里假定 x 轴正方向为右方，y 轴正方向为上方。

- 当怪兽初始位置位于悠米的正右上方时：恶魔可以总是往怪兽初始位置的左方或下方放置怪兽——若悠米向上移动，则往左方放置；若悠米向右移动，则往右方放置。此时恶魔总能抓住悠米。
- 否则，悠米必然可以选择上方或右方的某一个方向，使得只要悠米一直沿着这个方向走，就不可能碰到怪兽（可以考虑哈密顿距离）。

时间复杂度 $\mathcal{O}(T)$，空间复杂度 $\mathcal{O}(1)$。

> 题目灵感来源：打算出一道跟喵星人有关的题，于是想到了悠米，然后翻B站看到了这个：
> 《当 魔法猫咪-悠米 奔跑带彩虹后... lol英雄联盟》——梓旋Fairym
>
> 之后经过一轮又一轮更新、改bug、润色，最终出了这道思维题。

干饭协会的扮装节🎩

前置芝士：最短路，离线处理。

全源最短路裸题（板子题），套了个 cosplay 的壳子。

精心调教过的数据范围使得 *Dijkstra* 和 *Floyd* 都可以顺利通过本题。

需要注意的就是一定要先离线算好答案再查询。

Dijkstra: 时间复杂度 $\mathcal{O}(nm\log m)$，空间复杂度 $\mathcal{O}(n^2+m)$ 或 $\mathcal{O}(n+m+q)$。

Floyd: 时间复杂度 $\mathcal{O}(n^3)$，空间复杂度 $\mathcal{O}(n^2+m)$。

> 题目灵感来源：网上看到一些卡通人物之间互相 cosplay 的图片，然后就突发奇想……
>
> 于是有了这道板子题。
>
> PS：那张皮卡丘 cos 小新的是某才子拿玩偶画出来的 XD

干饭协会的招聘书📕

前置芝士：自定义排序，背包。

显然做题顺序不影响答案。

其次，考虑任一合法的做题序列，如果有一道更难的题先于更简单的题被做完，那么，把这道更简单题，放到更难的题的前一题做完，所得到的序列依然合法；只要不断重复上述操作，直到最终得到的做题序列是按难度递增的，并且该序列也是合法的。

于是，考虑从简单的往难的做，选择一部分跳过，这样就能够算出所有可能值。因此为了求出最小值，可以预先排序，然后 dp 依次处理就行，本质是背包。

具体地，我们用 `dp[j]` 表示评分达到 j 的最小耗时，对于每道题，遍历 $x \in [r-d, m]$，用 `dp[x]` 更新 `dp[x + f[i]]` 即可。

时间复杂度 $\mathcal{O}(n(k+\log n))$，空间复杂度 $\mathcal{O}(m)$。

干饭协会的捏脸师👩

前置芝士：SCC（强连通分量）。

让我们来建立一张图。将脸型 u 捏成脸型 v 化为一个从 v 指向 u 的边，表示：只要能够捏出脸型 v，就可以免费捏出脸型 u。

首先让我们来看看关于图中 SCC 的结论：一旦捏过了 SCC 中的任意一张脸，那么该 SCC 剩余部分全部都能免费捏完。这一点由 SCC 的定义/性质即可得出。

那么第一步就是求 SCC 咯，只要把属于同一 SCC 的点缩成一个点，然后该点的价格定为 SCC 中原来各节点的最小价格。

接着让我们来考虑这样一个事实：如果对于某个 SCC 有入边，那么这个 SCC 就通通可以免费搞定啦！

这是为什么呢？因为如果一个 SCC 有入边，那么只要入边方向的那个 SCC 都捏完了脸，这个 SCC 就一定可以跟着全部免费捏完了；又因为由 SCC 缩点后的图是 DAG（有向无环图），入边方向的 SCC 是一定不会依赖于当前 SCC 的。

那么这样一来，只要我们把每个入度为 0 的 SCC 中最小的价格都加起来就是答案啦。

另外注意，如果一个 SCC 中没有边，本质上也就是如果没有人想要捏成这个脸型，那么就不需要累加该 SCC 的价格。（出题人一开始并没有注意到这一点，感谢指出此问题的巨巨验题人 `Kenshin2438`）

时间复杂度 $\mathcal{O}\left(\sum(n+m)\right)$，空间复杂度 $\mathcal{O}(\max\{n+m\})$。

> 题目灵感来源：上课摸鱼时在画脸，比如西瓜脸啊瓜子脸什么的，然后随便加了点箭头，突然灵感它就来了，于是就有了这道题。

干饭协会的排队论📚

前置芝士：倍增(st表) + 二分

用st表维护区间最大值，然后对于每一次询问，二分查找第一个比他高 k 的位置在哪里。

复杂度 $\mathcal{O}((n+q)\log n)$；另外，由于数据较弱，理论上无法通过的单调栈在赛时 AC 了本题。

干饭协会的整向量 ⬅️

前置芝士：数学 + 拓展欧几里得

可以列出方程：

$$\begin{cases} ax_1 + by_1 = z_1 \\ ax_2 + by_2 = z_2 \end{cases}$$

显然方程的解是：

$$\begin{cases} a = \dfrac{z_1y_2 - z_2y_1}{x_1y_2 - x_2y_1} \\ \\ b = \dfrac{x_1z_2 - x_2z_1}{x_1y_2 - x_2y_1} \end{cases}$$

接下来分类讨论：

- 如果 $x_1y_2 - x_2y_1 \neq 0$ ，直接解方程，判断 a, b 是否为整数。
- 如果 $x_1y_2 - x_2y_1 = 0$ ，
  - 如果 $x_1/x_2 = z_1/z_2$ ，就是说方程组所表示的直线重合，根据 gcd 判断解的存在性，然后再用拓展欧几里得解方程计算 $ax_1 + by_1 = z_1$ 或 $ax_2 + by_2 = z_2$ 。
  - 否则，方程组表示的俩直线平行，无解。

可以证明如果答案存在的话，就必然存在一组答案 a, b 满足 $-2 \cdot 10^{12} \leqslant a, b \leqslant 2 \cdot 10^{12}$。

干饭协会的欢乐树 🌳

前置芝士：DFS序/树链剖分 + 线段树/树状数组。

板子题一道，跟 DFS 序 1 差不多，是 DFS 序 4 的 easy 版本，不会的可以看 Tutorial。

复杂度 $\mathcal{O}(n \log n)$。

干饭协会的签到题 ✍️

前置芝士：思维。

答案为：最大元素 - 最小元素 + 其余元素。

具体操作方法为：先用 最小元素 依次减去除最大元素外的每一 其余元素，得到 $x =$ (最小元素 - 其余元素) ；最后用 最大元素 减去 x，得到 最大元素 - 最小元素 + 其余元素。

证明过程概述：最终的式子里总有元素要被减掉，因此能得到的最大值就是只有最小元素被减掉了。而上面的操作方法给出了一种这样的方案，因此得到的就是答案。

时间复杂度 $\mathcal{O}(n)$，空间复杂度 $\mathcal{O}(1)$。

> 题目灵感来源：Codeforces Round #695 (Div. 2) - C. Three Bags
> 日常训练时做到的题，觉得这题的思维挺有意思的。
> 考虑到原题对实现能力要求较高，对原题稍作简化即得此题。

干饭协会的金币树 🪙

前置芝士：DFS 或 BFS，偏思维。

本题有多种做法，这里介绍两种——

做法一：BFS 分层。由深至浅逐层遍历每一结点，记当前结点为 u，将 u 的所有子结点的金币数增加到它们的最大值，记该值为 x，添加后各子结点金币数相同，把这 x 个金币直接看成是结点 u 的即可。

做法二：DFS。可以借鉴上面 BFS 的思维来实现；或者也可以进行两趟 DFS，第一趟算出金币数最大的从根节点到叶子结点的路径，第二趟直接计算。

以上做法的时间复杂度、空间复杂度均为 $O(n)$。

> 题目灵感来源：码题集OJ-金币；原题是给的二叉树，稍加拓展即得此题。

干饭协会的金手帕 ✋

前置芝士：倍增跳点 / dp + 二进制。

我们知道任意一个数都有二进制表示。

例：假设 $k = 21$，其二进制表示为 10101，那么跳 k 步就可以看成跳 $2^0 + 2^2 + 2^4$ 步。

即对于任意一个 k，都可以拆分成若干个 2 的次幂的和。

因此只要预处理出每个点跳 2^n 步所到达的点即可。

时间复杂度 $\mathcal{O}((n+m)\log k)$，空间复杂度 $\mathcal{O}(n\log k)$。

干饭协会的魔法棒 🪄

前置芝士：线段树。

其实是线段树经典应用，其难点在于如何维护结点信息，以及上传操作 `push_up()` 怎么写——

这里分别用 `X.lzl, X.lz, X.zl, X.l, X.z` 表示区间 `X` 中 `lzl, lz, zl, l, z` 子序列的数量。假设要将两个不相交的子区间 `L` 和 `R` 合并成一个连续的大区间 `M`，且 `L` 在 `R` 的左边，那么：

```
M.l = L.l + R.l
M.z = L.z + R.z
M.lz = L.lz + L.l * R.z + R.lz
M.zl = L.zl + L.z * R.l + R.zl
M.lzl = L.lzl + L.lz * R.l + L.l * R.zl + R.lzl
```

因此，对于每一线段树结点，维护其中 `lzl, lz, zl, l, z` 子序列的总数即可。

时间复杂度 $\mathcal{O}(n+m\log n)$，空间复杂度 $\mathcal{O}(n)$。