2022“杭电杯”中国大学生算法设计超级联赛（3）题解

By Claris

2022 年 7 月 26 日

1 Equipment Upgrade

Shortest judge solution: 2275 Bytes.

设 E_i 表示从等级 i 强化至等级 n 的期望花费，则有

$$\begin{aligned}
E_n &= 0 \\
E_i &= c_i + p_i \cdot E_{i+1} + \frac{(1-p_i)}{\sum_{j=1}^{i} w_j} \cdot \left(\sum_{j=1}^{i} w_j \cdot E_{i-j} \right) \\
p_i \cdot E_{i+1} &= E_i - c_i - \frac{(1-p_i)}{\sum_{j=1}^{i} w_j} \cdot \left(\sum_{j=1}^{i} w_j \cdot E_{i-j} \right) \\
E_{i+1} &= \frac{E_i - c_i - \frac{(1-p_i)}{\sum_{j=1}^{i} w_j} \cdot \left(\sum_{j=1}^{i} w_j \cdot E_{i-j} \right)}{p_i}
\end{aligned}$$

根据上式，如果已知 $E_0, E_1, \ldots, E_i$，即可计算出 E_{i+1} 的值。但是由于 E_0 未知，我们无法从前往后进行递推。在这里，设 $E_i = a_i \cdot E_0 + b_i$，则

$$\begin{aligned}
a_0 &= 1 \\
b_0 &= 0 \\
a_{i+1} &= \frac{a_i - \frac{(1-p_i)}{\sum_{j=1}^{i} w_j} \cdot \left(\sum_{j=1}^{i} w_j \cdot a_{i-j} \right)}{p_i} \\
b_{i+1} &= \frac{b_i - c_i - \frac{(1-p_i)}{\sum_{j=1}^{i} w_j} \cdot \left(\sum_{j=1}^{i} w_j \cdot b_{i-j} \right)}{p_i}
\end{aligned}$$

此时 a 和 b 都是可以从前往后递推计算的，且计算过程是卷积的形式，可以用分治 +NTT 求出，最后根据 $a_n \cdot E_0 + b_n = E_n = 0$ 解出 E_0 的值即为答案。

时间复杂度 $O(n \log^2 n)$。

2 Boss Rush

Shortest judge solution: 1050 Bytes.

二分答案，转化为判断 T 帧内能否打败 BOSS，即求出 T 帧内能打出的最高伤害，判断是否大于等于 H。

从前往后依次发动若干个技能，则下一个技能可以发动的时刻等于之前发动过的技能的演出时间之和，因此只和之前发动过哪些技能有关。设 f_S 表示发动了 S 集合的技能，在 T 帧内最多能结算多少伤害，枚举不在 S 中的某个技能 x 作为下一个技能进行转移，由于技能发动时刻已知，因此可以 $O(1)$ 计算出在 T 帧内下一个技能可以结算多少伤害。

时间复杂度 $O(n2^n \log ans)$。

3 Cyber Language

Shortest judge solution: 347 Bytes.

签到模拟，遍历每个字符，如果一个字符是小写字母且前一个字符是空格或者它是第一个字符，那么把它转大写输出。

4 Divide the Sweets

Shortest judge solution: 2623 Bytes.

令 sum_S 表示 S 集合的箱子的糖果数之和，$f_{i,S}$ 表示把 S 集合的箱子分给 i 个孩子的最小平方和，则 $f_{i,S} = \min\left(f_{i-1,T} + (sum_S - sum_T)^2\right)$，其中 T 是 S 的子集。直接枚举子集转移的时间复杂度为 $O(m3^n)$，不能接受。

方便起见，令 $g_S = f_{i,S}, h_S = f_{i-1,S}$，则

$$\begin{aligned} g_S &= \min\left(h_T + (sum_S - sum_T)^2\right) \\ &= \min\left(h_T + sum_S^2 + sum_T^2 - 2 \cdot sum_S \cdot sum_T\right) \\ &= \min\left((h_T + sum_T^2) - 2 \cdot sum_S \cdot sum_T\right) + sum_S^2 \end{aligned}$$

将表示集合的 n 位二进制数拆成前 $\frac{n}{2}$ 位和后 $\frac{n}{2}$ 位来考虑。枚举转移中 T 的前 $\frac{n}{2}$ 位 A，再枚举 A 的所有超集 A' 作为 S 的前半部分，然后枚举 S 的后 $\frac{n}{2}$ 位 B'，此时 $S = A'B'$，我们需要找到一个 B' 的子集 B 作为 T 的后半部分（即 $T = AB$），满足 $(h_T + sum_T^2) - 2 \cdot sum_S \cdot sum_T$ 最小。这是经典斜率优化的形式，对于每个 T，将其看作直线 $y = sum_T \cdot x + (h_T + sum_T^2)$，若能得到 B' 子集的所有直线形成的凸壳，则在凸壳上询问 $x = -2sum_S$ 时 y 的最小值即可完成状态转移。假设已经有了凸壳，为了保证询问的均摊复杂度为 $O(1)$，需要按照 sum_S 递增（或递减）的顺序去询问，由于 $sum_S = sum_{A'} + sum_{B'}$，所有的 S 对应的 B' 都相等，因此按照 $sum_{A'}$ 递增（或递减）的顺序去枚举 A' 即可保证询问 x 坐标的单调性。

剩下的问题是如何得到 B' 子集的凸壳。类似于高维前缀和，对于每个状态 B 保存其凸壳。一开始对于每个状态 B，它的凸壳大小为 1，对应 $T = AB$。接下来依次枚举后半部分的每一位，假设当前枚举到了第 x 位，对于一个集合 B，如果它的第 x 位为 1，则它缺少 `B xor (1<<x)` 的信息，暴力将 `B xor (1<<x)` 和 B 的凸壳二路归并成一个，作为 B 的新凸壳。最坏情况下，每次合并后凸壳大小翻倍，总代价为最终凸壳大小的两倍，所有状态的凸壳大小不超过子集数，因此预处理子集凸壳的时间复杂度为 $O(3^{\frac{n}{2}})$。

总时间复杂度分析：一共要进行 $O(m)$ 轮，每轮需要枚举 $O(2^{\frac{n}{2}})$ 个 A，然后枚举它的超集 A'，总计 $O(3^{\frac{n}{2}})$，每个 A' 又需要接着枚举 $O(2^{\frac{n}{2}})$ 个后半部分 B'，然后花费均摊 $O(1)$ 的代价在子集的凸壳上进行询问，这部分每轮的总时间复杂度为 $O(3^{\frac{n}{2}} \cdot 2^{\frac{n}{2}}) = O(6^{\frac{n}{2}})$。而枚举完

A 后，又需要支付 $O(3^{\frac{n}{2}})$ 的代价预处理出所有子集的凸壳，这部分每轮的总时间复杂度也为 $O(6^{\frac{n}{2}})$。因此，最终得到总时间复杂度为 $O(m6^{\frac{n}{2}})$。

5 Spanning Tree Game

Shortest judge solution: 2335 Bytes.

题意即对于每个 k $(0 \le k \le m)$，从数组 a 中选取 k 个边权，从数组 b 中选取 $m-k$ 个边权，并最大化最小生成树的边权和。

对于一条边 (u, v, a, b)，有以下两种情况：

- $a < b$：拆成两条边 $(u, v, a, 1)$ 和 $(u, v, b, 0)$。
- $a \ge b$：拆成两条边 $(u, v, b, -1)$ 和 $(u, v, a, 0)$。

根据最小生成树的 Kruskal 算法，将拆后的 $2m$ 条边按边权从小到大排序，边权相同时，将类型 (-1 或 0 或 1) 为 0 的边排在后面，从前往后依次考虑每条边：

- 若是 $(u, v, w, 1)$，则这条边可选可不选，若选了则 a 数组中选取的边数要增加 1，此时若 u 和 v 不连通，则 MST 的边权和要增加 w。
- 若是 $(u, v, w, -1)$，则这条边可选可不选，若选了则 a 数组中选取的边数要减少 1，此时若 u 和 v 不连通，则 MST 的边权和要增加 w。
- 若是 $(u, v, w, 0)$，则这条边对应的另一条边在此之前已经考虑过。若另一条边选了，则这条边选上不会对 MST 产生多余的错误贡献；若另一条边没选，则这条边必须要选。因此，此时如果判断出 u 和 v 不连通，则 MST 的边权和要增加 w。

上述过程中每条边可能会有两种选择，使得整个图中 n 个点的连通情况不同，由此设计出状态 $f_{i,S,j}$ 表示考虑了排序后前 i 条边，n 个点连通性的最小表示为 S，有 j 条边边权来自数组 a 时，最小生成树边权和的最大值是多少。预处理转移后可以做到 $O(1)$ 转移。

状态数分析：i 和 j 都是 $O(m)$ 个，S 有 $Bell(n)$ 个，其中 $Bell(9) = 21147$。

时间复杂度 $O(m^2 Bell(n))$。

6 Dusk Moon

Shortest judge solution: 3349 Bytes.

对于一个点集，它的凸包覆盖住了所有点，且最小覆盖圆覆盖住了凸包，因此仅保留凸包的顶点不会影响答案。

由于点的坐标在给定的正方形范围内随机，因此一个点集的凸包的期望顶点数为 $O(\log n)$，使用线段树直接记录区间凸包点集，然后对于 $O(\log n)$ 个点运行最小圆覆盖算法即可。

时间复杂度 $O(q \log^2 n)$。

7 Shallow Moon

Shortest judge solution: 3236 Bytes.

按 w 行为一块，将 $m \times m$ 的矩阵从上到下分为 $\lceil \frac{m}{w} \rceil$ 个 $w \times m$ 的块。其中最后一块可能不足 w 行，方便起见下面都忽略这种情况。

由于每块的行数和每个矩形障碍的行数相等，因此每个矩形障碍要么完全属于某一块，要么经过相邻的两块。一个矩形障碍对于某一块的影响只能是下列两种情况之一：

- 块内第 b 列至第 $b+h-1$ 列的前若干行是障碍。
- 块内第 b 列至第 $b+h-1$ 列的后若干行是障碍。

对于某块中固定的某一列，只需要考虑上述第一种情况中下边界最靠下的限制，以及第二种情况中上边界最靠上的限制，那么这一列要么都是障碍，要么能走的行数是一个连续的区间。假设某一块中有 k 个矩形障碍，那么按照每个障碍的左右边界将这一块 $w \times m$ 的区域从左往右离散成 $O(k)$ 个行数为 w 的区域，每个区域中最多只有一个连续的行区间可以行走。由于所有障碍的列数都为 h，可以通过单调队列在 $O(k)$ 时间内求出每个区域的可走区间。至此，我们成功地将一个包含 k 个矩形障碍的块的可走区域从左往右划分成了 $O(k)$ 个矩形。

由于每块至少贡献一个区域，而总障碍数为 $O(n)$，因此上述方法会得到 $O(n+\frac{m}{w})$ 个矩形区域，当 $\frac{m}{w}$ 较大时会退化，这是因为很多块一个障碍都没有，此时需要把连续的不含障碍的块合并成一个，则最终我们得到了 $O(n)$ 个矩形区域。

对这 $O(n)$ 个矩形区域建图，每个点代表一个区域，点权为区域的面积，若两个区域相邻则连边，答案即为每个连通块的点权和的平方之和。在这里，连边有以下两种情况：

- 同一块内左右相邻的两个区域需要连边，边数显然为 $O(n)$。
- 相邻两块内上下相邻的两个区域需要连边，可以通过双指针实现，边数为相邻两块区域数之和，总边数仍为 $O(n)$。

时间复杂度 $O(n \log n)$，使用基数排序可以做到 $O(n)$。

8 Laser Alarm

Shortest judge solution: 1415 Bytes.

三个不共线的点可以确定一个平面。对于任意一个平面，将其调整至经过三个顶点，结果不会变差。因此枚举三个顶点得到平面，然后 $O(n)$ 计算触碰了该平面的线段数，更新答案即可。所有点都共线的情况需要特判。

时间复杂度 $O(n^4)$。

9 Package Delivery

Shortest judge solution: 976 Bytes.

考虑 r 最小的那个区间 k，第一次取快递放在第 r_k 天一定不会使结果变差。此时可能有很多区间覆盖了 r_k，那么为了尽量延后下一次取快递的日期，此时的最优策略应该是选择覆盖 r_k

且 r 值最小的 k 个区间，使用堆找到并去掉这些区间后，问题就递归了。重复上述过程直至处理完所有 n 个区间。

时间复杂度 $O(n\log n)$。

10 Range Reachability Query

Shortest judge solution: 1549 Bytes.

离线询问，设 $f_{i,j}$ 表示 i 点能否仅通过编号在 $[l_j, r_j]$ 之间的边走到第 j 个询问的目的地 v_j，共 $O(nq)$ 个 01 状态，可以使用 bitset 存储。第 j 个询问的答案即为 $f_{u_j,j}$。

考虑一条边 $u \to v$ 的转移，假设它的编号为 k，令所有满足 $l \le k \le r$ 的询问的集合为 S，则有 `f[u] |= f[v] & S`。接下来考虑如何得到集合 S，一个直接的想法是从 1 到 m 依次考虑每条边，维护覆盖当前边的询问集合，每个询问拆成两个事件：

- 在 l 处加入集合。
- 在 $r+1$ 处离开集合。

上述方法需要保存 m 个 bitset，空间复杂度过高。节省空间的方法是每 $O(\sqrt{q})$ 个事件保存一次 bitset，这样只需保存 $O(\sqrt{q})$ 个 bitset，然后每次要得到集合 S 时，再往对应 bitset 的副本中暴力模拟 $O(\sqrt{q})$ 个事件。

总时间复杂度 $O(\frac{mq}{w} + m\sqrt{q})$。

11 Taxi

Shortest judge solution: 1088 Bytes.

如果没有 w 的限制，那么是经典问题。根据 $|x| = \max(x, -x)$，有

$$
\begin{aligned}
& \max\left\{|x' - x_i| + |y' - y_i|\right\} \\
= \ & \max\left\{\max(x' - x_i, -x' + x_i) + \max(y' - y_i, -y' + y_i)\right\} \\
= \ & \max\left\{x' - x_i + y' - y_i, -x' + x_i + y' - y_i, x' - x_i - y' + y_i, -x' + x_i - y' + y_i)\right\} \\
= \ & \max\left\{(x' + y') + (-x_i - y_i), (x' - y') + (-x_i + y_i), (-x' + y') + (x_i - y_i), (-x' - y') + (x_i + y_i)\right\}
\end{aligned}
$$

分别记录 $x_i + y_i$、$x_i - y_i$、$-x_i + y_i$、$-x_i - y_i$ 的最大值即可在 $O(1)$ 时间内求出所有点到 (x', y') 的曼哈顿距离的最大值。

现在考虑加入 w 的限制。将所有城镇按照 w 从小到大排序，并记录排序后每个后缀的 $x_i + y_i$、$x_i - y_i$、$-x_i + y_i$、$-x_i - y_i$ 的最大值，用于 $O(1)$ 求给定点 (x', y') 到该后缀中所有点的距离最大值。

选取按 w 排序后的第 k 个城镇，$O(1)$ 求出给定点 (x', y') 到第 $k..n$ 个城镇的距离最大值 d，有两种情况：

- $w_k < d$，那么第 $k..n$ 个城镇对答案的贡献至少为 w_k。用 w_k 更新答案后，由于第 $1..k$ 个城镇的 w 值均不超过 w_k，因此它们不可能接着更新答案，考虑范围缩小至 $[k+1, n]$。

- $w_k \geq d$，那么第 $k..n$ 个城镇对答案的贡献为 d。用 d 更新答案后，考虑范围缩小至 $[1, k-1]$。

容易发现每次考虑的范围都是一个区间，如果每次取 k 为区间的中点，那么迭代 $O(\log n)$ 次即可得到最优解。

时间复杂度 $O((n+q)\log n)$。

12 Two Permutations

Shortest judge solution: 1494 Bytes.

首先特判序列 S 中每个数字出现次数不都为 2 的情况，此时答案为 0。

动态规划，设 $f_{i,j}$ 表示 P 的前 i 项匹配上了 S，且 P_i 匹配 S 中数字 P_i 第 j 次出现的位置时，有多少种合法的方案。由于 S 中每个数字出现次数都为 2，因此状态数为 $O(n)$。转移时枚举 P_{i+1} 匹配哪个位置，那么 P_i 匹配的位置与 P_{i+1} 匹配的位置中间的那段连续子串需要完全匹配 Q 中对应的子串，使用字符串 Hash 进行 $O(1)$ 判断即可。

时间复杂度 $O(n)$。