小向的試煉 Vol. 1 題解

hansonyu123

1 猜謎遊戲 (Guessing)

如果只有一個 0 或 1 要猜的話，明顯地使用「三戰兩勝」式的猜法可以在 3 次詢問以內回答出正確答案。因此將 2^{17} 位的每位視為獨立的一個問題，使用 3×2^{17} 次一定可以回答出來。然而這樣無法拿到任何分數。

1.1 Subtask 1

注意到對於某一位，如果前兩次的詢問得到的答案都一樣，那就代表這兩次都不是謊話，也就不用問第三次了。套用「三戰兩勝」的想法，就是勝負一旦決定就不用比第三場了。因為至多說謊一次，2^{17} 位中頂多只有一個位置需要花 3 次詢問才能回答，剩下的只需要兩次，也就是說 $2 \times 2^{17} + 1$ 次詢問內可以回答出答案。如此可獲得 16 分。

1.2 Subtask 2

前面的作法是將每一位視作一個獨立問題。如果將每兩位視作一個獨立問題呢？

一開始同樣先詢問第一位（假設獲得的答案是 a）以及第二位（假設獲得的答案是 b)。為了確認前兩次有沒有說謊，兩位同時詢問（假設獲得的答案 c)。因為至多只能說謊一次，所以如果 $a \oplus b \oplus c = 0$，則代表前三次的詢問得到的答案都是對的。特別地，兩個位置的值就確定了。接下來只需要考慮 $a \oplus b \oplus c = 1$ 的情況。此種情況下，前三個詢問恰有一者說謊。因為之後的詢問一定會得到正確答案，所以也可以三個詢問再重新問一次，不過注意到如果有某個詢問得到的答案和前一次得到的答案不一樣的話，我們就能確定哪個詢問說謊了。除此之外，如果之後的兩次詢問都和前一次一樣的話，那一定是還沒測試的那次詢問說謊了。也就是說，其實只需要 5 次就可以得出正確答案。

簡而言之，如果只有兩位的話，不說謊可以只花 3 次，說謊可以只花 5 次得出正確答案。同樣地，因為最多只會說謊一次，所以可以在 $3 \times 2^{16} + 2$ 次內回答出答案。如此可獲

得 36 分。

1.3 Subtask 3

在 Subtask 2 中我們發現若要試圖驗證某幾次詢問的正確性，可以考慮這些詢問的 xor，並再次詢問以驗證正確性。事實上，這類似於偶同位元檢查。而這樣的工具可以幫助我們二分搜出可能說謊的候選人。

一開始先將所有位置詢問完。接著，假設可能說謊的單一位置詢問僅落在 $[l, r)$。先詢問 $[l, m)$ (這裡 $m = (l + r)/2$)：如果答案和 $[l, m)$ 的所有詢問答案 xor 值不同，代表可能說謊的單一位置詢問一定落在 $[l, m)$ 中，繼續二分搜；如果相同，再詢問 $[m, r)$，不同的話同樣繼續二分搜，相同的話代表所有單一位置的詢問都沒有說謊，回傳答案。易見詢問次數至多 $2^{17} + 34$，可以獲得 64 分。

1.4 Subtask 4

可以發現 Subtask 3 中的二分搜事實上是有些冗贅的。重點在於當驗證失敗時，我們無法辨別是受驗者中有其中一個說謊還是驗證的該次詢問說謊。為了確定說謊的是否是單一位置詢問，先花兩次驗證單一位置詢問中是否有說謊的：如果兩次得到的答案不同，代表單一位置詢問一定都沒說謊；如果兩次答案相同，代表驗證的結果即為單一位置詢問中是否有人說謊。如此一來，若單一位置詢問中有人說謊，之後的二分搜便只需 17 次就可找到說謊的詢問。也就是說詢問次數至多 $2^{17} + 19$。

基於 Subtask 3 的二分搜方法還可提出一個另解：假設可能說謊的單一位置詢問僅落在 $[l, r)$。先詢問 $[l, m)$：如果答案和 $[l, m)$ 的所有詢問答案 xor 值不同，代表可能說謊的單一位置詢問一定落在 $[l, m)$ 中，繼續二分搜；如果相同，代表 $[l, m)$ 中的單一位置詢問全都沒有說謊，故可直接二分搜 $[m, r)$。持續二分搜直到區間內只剩一個候選人，即可用三戰兩勝決定該位置的正確答案。由於只需再多兩次（第一次已經在將所有位置詢問一次時詢問)，詢問次數至多 $2^{17} + 19$。

不難發現上面那種作法，最後的位置其實只要再一次詢問就可以了。所以其實可以用 $2^{17} + 18$ 次詢問就可以達成目的了。

(註：事實上不論時限的話，若採用固定策略（詢問不隨著得到的答案改變)，保證詢問次數的理論最小值為 $2^{17} + 18$。)

2 橋 (Bridge)

不難發現，只需要解決以下的問題即可：

給定一個 0 到 $n-1$ 的排列 $a_0, \ldots, a_{n-1}$，對每個 i，找出集合 $\{a_j | j < i, a_j > a_i\}$ 的中位數。

2.1 Subtask 1

簡單的計算可以發現 N 足夠大時原題的答案都是 3。不過需要注意 $N = 2$ 時答案為 1，$N = 3$ 時答案是 2。如此可獲得 8 分。

2.2 Subtask 2

從這裡以後只考慮簡化後的問題。

最直接的作法是把 $a_0, \ldots, a_{i-1}$ 排序後選出大於 a_i 的人再取中位數。然而如此複雜度是 $O(N^2 \log N)$，稍嫌危險。若採用 counting sort 即可將複雜度降為 $O(N^2)$。如此可獲得 16 分。

2.3 Subtask 3

延續 counting sort 的精神，問題轉為有 n 個數 $b_0, b_1, \ldots, b_{n-1}$。每次需要求 $\sum_{j=0}^{a_i-1} b_j$，求最小的 k 使得 $\sum_{j=a_i+1}^{k} b_j$ 超過給定值以及將 b_{a_i} 加 1。除了求最小的 k 之外都是 BIT 支援的操作。除此之外，最小的 k 可由 BIT 的詢問搭配二分搜得出。複雜度 $O(N \log^2 N)$，48 分。

2.4 Subtask 4

延續 Subtask 3 的作法，問題只在用 $O(\log N)$ 的時間找出 k。不難發現使用線段樹並在樹上二分搜的話可達成要求，不過時限稍嫌危險。改用常數較小的 BIT 樹上二分搜便有充裕的時間完成任務。由於 BIT 的限制，BIT 中能 $O(1)$ 取得的區間和只有 $[n - \text{lowbit}(n), n)$，故二分搜時只能利用這些區間二分搜，也就是說分點位置離二分搜區間左界的距離必須是 2 的冪次。詳細的實作說明不在此列出。

另外，不難發現所有排名樹都可支援本題所需的操作，因此使用 treap 也可在

$O(N \log N)$ 的時間內找出答案，不過常數過大很難在時限內完成目的。

3 森林 (Forest)

3.1 Subtask 1

最直接的作法是枚舉所有點，計算其子樹大小（利用樹 DP)。複雜度 $O(N^2)$，24 分。

3.2 Subtask 3

事實上只需要計算子樹大小一次即可。對於每一個點，其分割出的子樹除了小孩們的子樹之外，還有祖先的子樹：前者可由 DP 所得的結果得知，後者可由 $n -$ (子樹大小) 得知。故 DP 同時可得知移除某個點後最大子樹大小。對所有點取最小值即得到答案。複雜度 $O(N)$。