小向的試煉 Vol. 2 題解

hansonyu123

1 洞穴 (Cave)

相信大家都看過這類問題，只是通常是問一堆螞蟻在一條線段上走路，全部的螞蟻走出線段要花多久。這個問題很簡單，因為只要把相撞的螞蟻當作是交換身份後穿越就好了。然而這題因為要找出最後走出去的螞蟻是哪一隻，所以「交換身份」這步就需要再多做處理了。

1.1 Subtask 1

直接模擬螞蟻們在洞穴內的移動狀況即可。複雜度 $O(NL)$(因為至多只需要 L 秒就可以走出去了)。如此可以得到 12 分。

1.2 Subtask 2

經過一些實驗後不難發現答案只跟向右的螞蟻數 N_1 以及向左的螞蟻數 N_2 有關，而跟他們之間的位置無關。再經由觀察可以發現最後從右邊走出去的螞蟻是從右邊數過來第 N_1 隻螞蟻，最後從左邊走出去的螞蟻是從左邊數來第 N_2 隻螞蟻。因此只需要看最後一隻螞蟻是從右還從左出去，而這可以用交換身份後穿越這個老方法直接判定：對每隻螞蟻，都只考慮他，看看他需要花多久走出去，再看花最久走出去的螞蟻是朝右還朝左就好。複雜度 $O(N \log N)$(排序) 或 $O(N)$(nth_element)。如此可獲得 12 分。詳細證明見 Subtask 4 的作法。

1.3 Subtask 3

$N \leq 10^3$，模擬的方法可能還行得通。然而一次走一格是行不通的。注意到螞蟻碰撞到的次數最多 $O(N^2)$ 次，如果每次找出下個相撞的螞蟻以及更新所有人的位置只需要花

$O(\log N)$ 的時間的話就可以用 $O(N^2 \log N)$ 的時間解決問題。因為我們在意的是相鄰兩個螞蟻之間的距離，所以可以考慮 $a_1, a_2, \ldots, a_{N-1}$，其中 a_i 代表左邊數來第 i 隻螞蟻及第 $i+1$ 隻螞蟻的距離。除此之外，我們還在意每過一秒，每個值的變動應該要是多少。我們可以利用 $b_1, b_2, \ldots, b_{N-1}$ 記錄。

接著就是轉換問題。首先，我們需要找到離下次碰撞還剩多久。容易知道這個時間長就是 $\min\{a_i | b_i = -1\}$。假設這個時間是 $a_j = T$，那麼就將 a_i 加上 $T \times b_i$。接著因為第 j 隻螞蟻以及第 $j+1$ 隻螞蟻相撞必須轉向，所以 b_{j-1} 減少 $1, b_{j+1}$ 減少 $1, b_j = 1$。而我們希望能在 $O(\log N)$ 的時間完成如此一次操作。

不難想像線段樹懶人標能達成要求。對於 $[l, r)$ 節點，儲存一個懶人標代表該節點尚未處理的時間長度 t，以及 $\min\{a_i | b_i = -1, i \in [l, r)\}, \min\{a_i | b_i = 0, i \in [l, r)\}, \min\{a_i | b_i = 1, i \in [l, r)\}$ 三個值。那麼時間長可由區間詢問得知，a_i 加上 $T \times b_i$ 可使用懶人標，而改變 b_{j-1}, b_{j+1}, b_j 則是三次單點修改。單次操作複雜度 $O(\log N)$，故總複雜度 $O(N^2 \log N)$。如此可獲得 48 分。

1.4 Subtask 4

試圖證明 Subtask 2 中的發現，可以獲得整題想法的啟發。以下提供兩種方法。

第一種想法是觀察 $N_1 = 1$ 的情況。觀察過程，可以發現對向左走的螞蟻來說，向右走的那隻螞蟻看起來就像是開始了一個「擾動」，而這個擾動不斷地傳到隊伍尾端。對向左走的螞蟻隊列 D 來說，最後 D 的前端多了原本向右走的螞蟻，而少了尾端。

如果左邊有很多隻螞蟻向右走，那麼整個過程不再像只有一隻螞蟻般單純。然而因為螞蟻都是等速前進的，所以兩個「擾動」之間不會互相影響。因此不妨將所有向右走的螞蟻造成的影響逐次計算：每次都推入 D 的前端，將 D 的尾端拔除。如此便可證得 Subtask 2 中的觀察。

由上面的想法可以給出下面的算法：先對所有螞蟻的座標排序，再由右開始維護向走左的螞蟻隊列 D。如果掃到的螞蟻向左，推入 D 的前端。如果向右，那麼推入 D 的前端，再將 D 的尾端拔除。由此可以得到向左走出最晚的螞蟻 (即最後 D 的尾端)。向右也可以同樣處理。複雜度 $O(N \log N)$。

另一個想法是：不難發現螞蟻之間不會交換順序，而最終狀態很好求出：所求的螞蟻一定是由左數來第 N_2 隻螞蟻或由右數來第 N_1 隻螞蟻。因此排序或使用 nth_element 後便可得出所求的螞蟻。複雜度 $O(N \log N)$ 或 $O(N)$。也就是說 Subtask 2 的解法如果沒有加上特判的話是可以獲得 100 分的。

2 平行世界 (Parallel universe)

其實每個平行世界連接的魔力隧道最多只能被縮短 2 條的意思就是將樹上的一堆不相交鏈縮短。然而採用前者的定義思考對此題較有幫助。

2.1 Subtask 1

最直接的作法是枚舉每個邊要不要被縮短，再用樹分治/樹 DP 判斷該次的方案是否合法以及最大共享難度。複雜度 $O(N2^N)$。如此可得 12 分。

2.2 Subtask 2

將樹上的一堆不相交鏈縮短，直覺上最大共享難度應該會非常小。因此我們可以試著估計最大共享難度最小值的最大可能值。

因為縮短的邊會構成很多不相交的鏈，所以一個自然的想法是如果採用「重鏈剖分」的話答案會是多少呢？也就是說，對於每個點，都縮短他到最大子樹之間的邊。容易發現這樣縮短邊是符合題目條件的。除此之外，對於任一個點，他到祖先的路徑中，每經過一條沒被縮短的邊就代表當前節點的子樹大小至少翻倍。因此沒被縮短的邊數 $= O(\log N)$。這告訴我們從 0 開始測試最大共享難度是該數的方案有多少個，直到為非零為止是一個負擔不會太大的作法。如此便可嘗試樹 DP。(事實上最大值發生在奇數層有一個兒子，偶數層有兩個兒子的樹。)

在進行樹 DP 時，枚舉根附近被縮短的魔力隧道方案個數。假設目前要算的是最大共享難度最小值是 n 的方案個數 $dp[n][0]$。對於根的每個小孩 s，如果他的父邊沒被縮短了，那麼他對這個情況的方案個數貢獻會多乘一個 $dp[n-1][s]$。如果父邊被縮短，那麼要考慮的是 s 的其它邊最多只能再縮短一個小孩的方案個數。為此，引進 $dp1[n][s]$，代表以 s 為根且 s 往下只能被縮短一條邊時，最大共享難度最小值是 n 的方案個數。

找出所有所需的狀態後，餘下的只有列出轉移式。不難看出：

$$dp1[n][s] = \prod_{s_1\text{是}s\text{的小孩}} dp[n-1][s_1] + \sum_{s_1\text{是}s\text{的小孩}} dp1[n][s_1] \prod_{s_2\text{是}s\text{的小孩},s_2\neq s_1} dp[n-1][s_2]$$

$$dp[n][s] = dp1[n][s] + \sum_{s_1,s_2\text{是}s\text{的小孩},s_1<s_2} dp1[n][s_1]dp1[n][s_2] \prod_{s_3\text{是}s\text{的小孩},s_3\neq s_1,s_2} dp[n-1][s_3]$$

故假設 $dp[n-1][\cdot], dp1[n-1][\cdot]$ 均算出時，算出 $dp[n][\cdot], dp1[n][\cdot]$ 的複雜度只有 $O(N^3)$。因為 n 的最大值只有 $O(\log N)$，所以總複雜度 $O(N^3 \log N)$。如此可獲得 28 分。

雖然如此的作法只能得出 $O(N^3 \log N)$ 的作法似乎有點令人氣餒，但是接下來的目標非常清晰：只需要快速地算出兩個 sigma 的值即可。Subtask 4 以及 Subtask 5 分別採取了兩種不同的方法計算。

2.3 Subtask 3

在不知道答案大小為 $O(\log N)$ 的情況下使用 Subtask 5 的 DP 方法即可得到 $O(N^2)$ 的作法。如此可得 48 分。

2.4 Subtask 4

這個作法用到了 1000000007 是質數的事實。

假設小孩的 $dp[n-1][\cdot]$ 乘積 X 模 1000000007 不為零，那麼將兩個 sigma 提出 X 後只需要計算

$$\sum_{s_1\text{是}s\text{的小孩}} dp1[n][s_1]dp[n-1][s_1]^{-1}$$

以及

$$\sum_{s_1,s_2\text{是}s\text{的小孩},s_1<s_2} (dp1[n][s_1]dp[n-1][s_1]^{-1})(dp1[n][s_2]dp[n-1][s_2]^{-1})$$

前者可用 $O($ 小孩個數 $)$ 算出，而後者可藉由和的平方扣掉平方和除以 2 得到，也就是 $O($ 小孩個數 $)$。因此再考慮計算模逆元的複雜度，總複雜度為 $O(N \log N \log P)$，其中 $P = 1000000007$。

然而若 X 為零，就不能再提出 X 了。不過在這個情況，不難發現需要求的兩項中有很多消為零。比如說如果恰一個 $dp[n-1][s_1]$ 是零，那麼第一坨只剩一項，而第二坨則化約成前一種狀況 $(X \neq 0)$ 的第一坨。如果有兩個是零，那麼第一坨是零，第二坨只剩一項。如果至少有三個是零，那麼全部都是零。因此不管是哪種狀況，都可以用 $O($ 小孩個數 $)$ 的複雜度計算出來。加上模逆元的複雜度，總複雜度為 $O(N \log N \log P)$。如此可以獲得 72 分。

2.5 Subtask 5

Subtask 4 的算法的問題只有計算模逆元過於耗時。注意到 Subtask 4 的作法中，計算兩坨的方法類似於 DP 的精神，故可考慮使用 DP 求解。而這個算法不依賴於模的數是否為質數。

想法非常簡單。令 s 的小孩為 $s_1, s_2, \ldots, s_t$ 且 $dp'[i][j]$ 代表前 j 個小孩中選 i 個，令他們的貢獻為 $dp1[n][\cdot]$，剩下的貢獻為 $dp[n-1][\cdot]$ 相乘後再對所有選法加起來的值。可見所求的即是 $dp'[0][t], dp'[1][t], dp'[2][t]$。轉移式也不難寫：

$$dp'[0][j] = dp'[0][j-1] \times dp[n-1][s_j]$$

$$dp'[i][j] = dp'[i][j-1] \times dp[n-1][s_j] + dp'[i-1][j-1] \times dp1[n][s_j](1 \le i \le 2)$$

對於一個節點，複雜度為 $O($ 小孩個數 $)$，故總複雜度為 $O(N \log N)$。然而空間複雜度也為 $O(N \log N)$，會 MLE。不過不難發現可以對第一維滾動，所以空間複雜度降為 $O(N)$，此題便得到了完整的解決。

事實上，因為模 1000000007 是 0 不代表沒有方法，所以還需要一個 bool 陣列求是否存在一個合法縮短的方法。這個 bool 的 DP 方法是完全類似的，這裡不再贅述。不過因為測資構造困難，本題測資裡並沒有這種測資。

3 轉！(Rotate!)

首先不難觀察到所有貪婪的換法都是最佳換法。換句話說，每次挑定一個不在位置上的球，把該顆球換到他所應該在的位置，持續做下去一定是最佳策略。然而證明這個需要費點口舌。沒有興趣的可以直接跳到 Subtask 1 的解決方法。如果對置換群略有研究的話應該知道這是個顯見的事實所以也跳到 Subtask 1 吧。

先來看一個常用的事實：

引理 1. 對於 1 到 N 到任何一個排列，如果存在兩種交換的方案 (規定每次都一定要換相異的數)，並且分別用了 n_1 以及 n_2 次交換，那麼 n_1, n_2 同奇偶。

證明: 考慮排列的逆序數對個數。不難證明對於每次交換，逆序數對個數的奇偶性一定改變。因此 n_1, n_2 都和原排列的逆序數對個數同奇偶 (注意到升冪排列的逆序數對個數為 0)，也就是說 n_1, n_2 同奇偶。 □

接下來的這個事實看起來很顯然，不過需要稍加說明。

引理 2. 對於 1 到 N 的某個排列，若 i 已在正確的位置上，且某個方案用了 n 次的交換換回原位並且 i 被換過，那麼存在一個交換至多 $n-2$ 次的方案。

證明: 首先，對於原本的方案，假設有相鄰兩次交換是先交換 a, b 再交換 c, d 且 $\{a, b\} \cap \{c, d\} = \phi$，那麼先交換 c, d 再交換 a, b 的效用是一樣的。假設 $\{a, b\} \cap \{c, d\} \neq \phi$，不妨假設 $a = c$，那麼先交換 b, d 再交換 a, b 的效用是一樣的。不論如何，都可以藉由這種交換交換的次序將所有和 i 有關的交換往前挪。

假設和 i 有關的交換有 k 個，那麼有和 i 交換過的數至多只有 k 個，所以由貪婪的換法知道存在一個至多只換 $k-1$ 次的方法將這些數換到原本 k 個交換後的結果。由引理 1 知這個換法的奇偶性和 k 一樣，所以至多只換了 $k-2$ 次。將一開始的 k 次交換換成這至多只換 $k-2$ 次的交換，那麼因為除了 i 之外的其它數在兩方案的位置都相同，所以 i 也在原位。因此我們成功地節省了兩次交換，也就證明了這個命題。 □

利用引理 2 可以證出原本的宣稱。假設已有一個最佳方案 A，我們需要證明對於一個不在原位的 i 存在一個最佳方案 B 使得第一次交換將 i 換回原位。假設他所應該在的位置上的數是 j，那麼先交換兩次 i, j 再開始使用最佳方案 A 的換法。考慮第一次後的所有交換，因為此時 i 已在本來的位置上，所以可以節省至少兩次交換。因此得到的新方案 B 不會比原本的方案 A 還差，也就是說新方案 B 也是最佳方案。如此一來貪婪法的正確性便得證。

3.1 Subtask 1

直接模擬貪婪的過程，每次都看看 N 在不在位置上，不在的話就換到位置上。N 遞減後繼續貪婪。複雜度 $O(N)$。如此可獲得 8 分。

3.2 Subtask 2

不難發現每次題目被交換一次後答案只會增加 1 或減少 1。故這實際上是 5 題是非題，可以對輸入 hash 後二分搜。複雜度 $O(N)$。如此可獲得 20 分。

3.3 Subtask 3

其實根本可以每交換一次之後當作新問題解決。複雜度 $O(MN)$，32 分。

3.4 Subtask 4

會不斷被改變位置的人其實有限。如果我們用好的貪婪順序或許可以得到好結果。

對於 N，看看他的位置上是誰。假設是 i，再看看 i 的位置上是誰。一直這樣找下一個人，直到找回 N 為止。可以對這些人貪婪，發現需要花 (個數)-1 次才能將這些人換回原位。對其它人也這樣做，便可以發現答案就是 $N-$(圈的個數)。所以我們只需要知道圈的個數之變動即可。

注意到因為只會換到前面的 10^4 個人，而整體的圈個數變動也就是只考前 10^4 個人的圈個數變動，故可以將前 10^4 的人用 Subtask 3 的作法找出答案是 +1 還是 -1，再更新真正的答案。複雜度 $O(N + \max i \times M)$。如此可獲得 56 分。

3.5 Subtask 5

不難觀察並證明出答案 +1 若且唯若所換的兩個球在不同的圈，而答案 -1 若且唯若兩個球在一樣的圈。而且如果兩球分屬不同的圈，那麼交換後兩個圈會合併。如果兩球在同樣的圈，那麼交換後圈會依兩個球的位置分裂成兩個圈。為了支援分裂和合併，看起來 treap 是最適合的工具。

現在對每個圈都用一個 treap 代表。合併時需要先將兩個圈「轉」到正確的位置。而「轉」到正確的位置就是將 treap 從選到的球分裂成兩個 treap，再將兩個 treap 逆序黏合。分裂時需要將分裂的圈「轉」到正確的位置再分裂。看起來 treap 的確支援這兩個操作。不過有幾個小小的問題。首先，球並沒有二分搜結構，我們不能依球的編號分裂 treap；再者，我們需要快速得知兩顆球所在的 treap 是哪一棵。一個相當簡易的作法是讓所有的節點都記錄其祖先以及子樹大小。如此一來，便可以邊往上走找到祖先邊記錄所要的球是從左邊數來第幾顆球。這些實作細節都相當容易，這裡不再多做著墨。

可見每次詢問都只需要花 $O(\log N)$ 的時間解決。再加上初始化，總複雜度為 $O((N + M) \log N)$。