小向的試煉 Vol. 4 題解

hansonyu123

1 果實 (Fruit)

1.1 Subtask 1

對於每個點，直接判斷他到直線的距離是否不超過 0.5 即可。複雜度 $O(N|U-D|Q)$，如此可得 12 分。

需要注意的是，在接下來的 subtask 中，為了避免精度問題，我們希望使用整數運算來判斷距離。一個好的想法是利用 ax_1+by_1 的值來判斷 (x_1,y_1) 到直線 $ax+by=c$ 的距離。經計算可得到距離即為

$$\frac{|ax_1+by_1-c|}{\sqrt{a^2+b^2}}$$

所以在判斷是否不超過 0.5 時只需移項平方。在接下來的 subtask 中，為了確保乘法不會超過 long long 範圍，需要在平方之前判斷大小：如果太大的話就直接判斷，否則就平方後判斷。

1.2 Subtask 2

不難發現，對於同一個 x 座標的點們，只要看最上面的點「夠不夠上面」以及最下面的點「夠不夠下面」即可。也就是說，存在一個點到直線距離不超過 0.5 的充要條件是最上面的點在直線下方 0.5 的上面以及最下面的點在直線上方 0.5 的下面。詳細證明請讀者自己思考，大概的方法是從上面一步一步走到下面，看看什麼時候「穿越」直線。

既然如此，只要運用 Subtask 1 的整數運算方法判斷是不是夠上面以及夠下面就能解決一個 x 座標的詢問。因此總時間複雜度為 $O(NQ)$，如此可得 24 分。

1.3 Subtask 3

如果 Subtask 4 的寫法寫爛了就只會過這個 Subtask 跟 Subtask 1, 共 44 分。

1.4 Subtask 4

由 Subtask 2 的證明方法，可以發現對於任何一條直線，存在一個點到他的距離不超過 0.5 若且唯若「最上面的點」在直線下方 0.5 的上面以及「最下面的點」在直線上方 0.5 的下面。所謂「最上面」以及「最下面」是指讓 $ax + by$ 最大以及最小的點。注意這裡 $b \geq 0$，如果 $b < 0$ 則需要將 a, b, c 同時取負號。因此，我們只需要快速知道 $\max ax + by$ 以及 $\min ax + by$ 的值即可。

先處理 $b \neq 0$ 的情況。那麼只需要求 $\max rx + y, \min rx + y$，其中 $r = a/b$。而不難發現同一個 x 座標的點中，有可能是最大值的點只有 (i, U_i)，而有可能是最小值的點只有 (i, D_i)。所以最大/最小值的候選人各只有 N 個。除此之外，我們可以維護一個「有效區間」的結構，記錄每個人在 r 落在什麼區間內時會是最大值。這個可以使用單調隊列優化的技巧在 $O(N)$ 的時間內完成。因此初始化 $O(N)$，而詢問時只需要二分搜看看 a/b 落在哪個區間即可，單次詢問複雜度 $O(\log N)$，故總時間複雜度 $O(N + Q \log N)$。如果不甚理解這段的意思的話，請參考單調隊列優化以及斜率優化/凸殼優化。$b = 0$ 的情況則可以單獨處理。

實作上有一個小細節需要注意。為了避免精度問題，區間頭尾應使用分數形式記錄。這樣記錄還有一個好處，就是 $b = 0$ 的情況可以不用再單獨判斷（利用分數形式記錄時比大小可以完全避開除法運算)。由於這部分的細節高度依賴於實作方法，這裡不再多做描述，請讀者自行完成細節。

由於本題強制在線，故不能使用離線演算法。不過若允許離線，可以將詢問對斜率排序後 $O(1)$ 完成單次詢問（這部分和單調隊列優化是相同的)。總時間複雜度變為 $O(N + Q \log Q)$。除此之外，也可以找出 $2N$ 個候選人的凸包 (採用 monotone chain 可做到 $O(N)$) 後用旋轉卡尺判斷是否跟凸包「夠近」。同 Subtask 4 的想法，只要「夠近」就一定會有一個點到那條直線的距離不超過 0.5。複雜度同樣是 $O(N + Q \log Q)$。

2 魔法陣 (Magic Circle)

不難發現這題的最佳策略可以非常凌亂，所以一個直覺的方法是採用 DP。不過 $U > 0$ 時沒有 DP 結構，所以 Subtask 4 至 7 需要用其它的處理方法，而 $U = 0$ 時的 DP 方法則可以從 Subtask 1 到 Subtask 3 的想法慢慢建構出來。

本題第一個關鍵但又直覺的觀察是：在同樣的剩餘魔力遇到同樣模式的光牆時，可以永遠採用同樣的策略，而不隨著是第幾次遇到這光牆而改變。

2.1 Subtask 1

不難發現遇到最高機率破解的光牆時重來一次完全沒有好處，所以需要決定策略的只有剩下 $N-1$ 種模式的光牆。有 10 種剩餘魔力，所以總共有 $(2^{(}N-1))^K$ 種策略，每種策略只需要花 $O(K)$ 的時間計算成功機率 (利用一些無窮等比級數求和公式以及簡單 DP)，總複雜度 $O(K2^{K(N-1)})$。

2.2 Subtask 2

首先，我們可以先只考慮局部最佳策略，比如說只剩一次嘗試機會時的策略，再逐步往後推，計算最佳策略。也就是說由最後一次慢慢向後 DP。假設 $dp[i]$ 是指剩下 i 次嘗試機會使用最佳策略時的成功機率，那麼在 $dp[i-1]$ 算出來的前提下，我們只需要分配剩下 i 次嘗試機會時遇到 N 種光牆時應該重來一次還是直接嘗試即可。不難發現最佳策略一定是決定一個閾值 T，機率大於 T 的直接嘗試，機率小於 T 的重來一次。T 的取法有 N 種，而對於每種取法都需要 $O(N)$ 的時間計算 (同樣需要無窮等比級數求和)，總複雜度 $O(N^2K)$，如此可得 16 分。

2.3 Subtask 3

如果仔細計算就會發現取定一個 T 時，該策略的成功機率只和 $dp[i-1]$，機率小於 T 的模式個數以及機率和有關。因此只需要將 N 種模式依機率大小排序，再將 T 由小到大計算即可用每次 $O(1)$ 的時間計算該策略的成功機率。因為對於一個 $dp[i]$，需要從 N 種機率中取最大者，所以總複雜度 $O(NK)$，如此可得 32 分。

這樣的解法很難再繼續下去，因為當 $U>0$ 時 $dp[i-1]$ 的值依賴於 $dp[i]$，失去了最佳子問題結構，所以無法執行 DP。看起來計算出精確值不是辦法，不如用類似模擬的方法求近似值。

2.4 Subtask 4

同樣地我們可以分配在每個狀況下 (剩餘魔力個數以及遭遇的光牆模式) 使用的策略。決定好之後，可以利用 DFS 模擬各種情況，再將成功的機率加總。不過因為可能無限制地強制切換模式，所以直接 DFS 當然會 TLE。不過注意到每強制切換模式一次，成功機

率就至少乘以一次 $K/(K+1)$，只要乘很多次，成功機率就會遠小於 10^{-9} 即可忽略。因此只要讓 DFS 的深度有一個限制就可以計算出所有機率夠大的成功機率加總是多少。複雜度估計過於複雜且不重要，這裡略去。如此可得 8 分。

2.5 Subtask 5

結合 Subtask 3 中枚舉 T 的方法以及 Subtask 4 忽略機率太小的可能性即可通過本組測資。實作方法很多，複雜度估計也不太一樣，這裡同樣略去。如此可得 24 分。

2.6 Subtask 6

既然不用考慮強制切換模式太多次的可能性，那麼乾脆將強制切換模式很少次的所有可能性利用 DP 計算出來，直接以此近似值當作答案即可。

為此，令 $dp[i][j]$ 代表剩餘 i 次魔力，最多只能強制切換 j 次時的成功機率。不難知道在遇到破解成功機率為 $P[t]$ 的光牆時，成功機率為

$$\max(\frac{i}{K+1}dp[\max(0,i-u)][j-1], P[t]+(1-P[t])dp[i-1][j])$$

因此，

$$dp[i][j]=\frac{1}{n}\sum_{t=0}^{n-1}\max(\frac{i}{K+1}dp[\max(0,i-u)][j-1], P[t]+(1-P[t])dp[i-1][j])$$

所以如果要計算到最多只能切換 L 次時的成功機率，那麼時間複雜度為 $O(NKL)$，空間複雜度為 $O(NL)$。

接下來的目的是估計 L 的大小。當然亂試 L 的大小也是一種方法，不過不難發現只要取

$$\left(\frac{K}{K+1}\right)^L < 10^{-9}$$

即可。兩邊取 log 可得

$$L > \frac{9}{\log(1-\frac{1}{K+1})} = \frac{9}{\ln(1-\frac{1}{K+1})\times\log e} \approx 9(K+1)\ln 10$$

這裡用到了 $\ln(1+\frac{1}{x})\approx\frac{1}{x}$。因此可以取 $L=O(9K)$，時間複雜度 $O(9NK^2)$，空間複雜度 $O(9K^2)$。如此可得 76 分。

2.7 Subtask 7

同樣地，我們可以再使用 Subtask 3 的想法來優化 Subtask 6 的作法。我們只需要計算

$$\max(\frac{i}{K+1}dp[\max(0,i-u)][j-1], P[t]+(1-P[t])dp[i-1][j])$$

的閾值是多少，並將模式依機率排序之後二分搜得到有幾個模式機率小於閾值，再預處理前綴和就可以計算出 $dp[i][j]$。詳細的過程請讀者自行補完。因此計算 $dp[i][j]$ 的複雜度降為 $O(\log N)$，總時間複雜度降為 $O(9K^2 \log N)$，空間複雜度同樣是 $O(9K^2)$。這樣會有 MLE 的疑慮，不過不難發現可以對第二維滾動，空間複雜度降為 $O(K)$，這題便得到了完全的解決。

許多有關計算機率或期望值的題目都需要捨去貢獻過小的可能性以求效率，而如何有效地捨去過小的可能性以及計算剩餘可能性的機率或期望值所需要的估算能力是相當高的，建議大家多多練習。

3 叛逆的物語 (Hannkyaku No Monogatari)

有的人應該會發現這題和 1932(機上送餐問題) 是差不多的，只差在兩件事能不能在同一個時刻做。不過這點差異造成了機上送餐問題的解法在本題只能獲得部分分。要拿到整題的分數需要相當大程度的修改。

3.1 Subtask 2

$K = 1$ 時不難發現以下的貪婪策略是最佳的：在可以發射的時刻看看有沒有已注入魔力的子彈，有的話就發射，否則就注入魔力。複雜度 $O(10^9)$，會 TLE。然而只要做一點離散化的處理就可以做到複雜度 $O(N \log N)$，便可獲得 24 分。

除此之外，由於答案只能是 -1 或是 $2N$，故這也是個是非題，可以對輸入 hash 後二分搜，同樣獲得 24 分。

3.2 Subtask 3

一個相當直覺的方法是先將所有需要發射子彈的時間預留好，再安排注入魔力的時間。安排時要盡量讓注入魔力安排在發射子彈的時刻，如果不行的話再另外挪時刻注入魔力。這樣可以得到一個複雜度 $O(N \log N)$ 的作法，但想法會相當凌亂。

為了讓想法清淅，可以先將 $T_1, \ldots, T_N$ 由小到大排序後令 $dp[i][j]$ 為在時刻 i 時已注入 j 次魔力後至少需要解除幾次防護罩。如果時刻 i 時需要發射飛彈，那麼就檢查注入魔力的次數是否足夠，以及要發射的子彈是否不超過 K。若不然有兩種選擇：這個時刻不做任何事以及注入 K 次魔力。轉移複雜度 $O(1)$，故總時間複雜度 $O(TK)$。同樣地，可以離散化將狀態數降為 $O(NK)$，代價是轉移複雜度變為 $O(N/K)$(可以解除防護罩 1 秒, 2 秒, ..., N/K 秒)，故總時間複雜度 $O(N^2)$。如此可得到 16 分。

3.3 Subtask 4

針對 $K = 1$ 以及 $X = 0$ 兩種不同的特殊情況，我們想出了兩種貪婪法。然而在一般的情況，似乎找不到一個貪婪法則：給定所有局部狀態時，無法知道解除防護罩時應該要注入魔力還是發射子彈。所以需要另尋方法。

如果有思考貪婪法則一段時間，應該會發現主要的問題在於注入魔力的時間被發射時間給限制，而發射時間則被 T_i 限制，所以在貪婪時如果最優先化注入魔力，那麼有可能會錯失發射子彈的時機；如果最優先化發射子彈，那麼有可能因為延後注入魔力的時間而連帶影響發射子彈的時間。既然限制在於後方，不妨試著將問題反轉，倒著貪婪。也就是說，要先發射飛彈再注入魔力，而發射飛彈的時間則有一定的限制。

倒過來後，不難發現如果解除了防護罩，那麼好的貪婪法是：若解除了防護罩，就把該時段能發射的飛彈全發射完，再看看能注入多少子彈的魔力。因此可以令 $dp[i][j][k]$ 是將原題倒過來後，在時刻 i 已發射 j 顆子彈且注入 k 顆子彈的魔力後至少要解除多久的防護罩。狀態數 $O(TN^2)$，轉移複雜度 $O(K)$，故總時間複雜度 $O(N^2TK)$。雖然略大了點，但不難發現採用 top down 的 DP 時不會計算到所有狀態，因此常數會相當小，如此便可獲得 36 分。

3.4 Subtask 5

將 Subtask 4 中的時間離散化即可得到狀態數 $O(N^3)$，轉移複雜度 $O(N)$(可以解除防護罩 1 秒, 2 秒, ..., N/K 秒)。故總時間複雜度 $O(N^4)$，便可通過本題所有測資。

當正著做無法寫出任何好的 DP 轉移式時，將原題倒過來常常可以得到一些不錯的結果。尤其是需要 DP 優化的題目 (如 IOI 2002 的 Batch Scheduling, TIOJ 的烏龜疊疊樂)。