The 3rd Universal Cup

Stage 29 (Metropolis) 出題組

February 9th, 2025

預期難度

- Easy: I, C, B
- Easy-Medium: G, D, E
- Medium: L, J, F
- Medium-Hard: H, M
- Hard: K, A

Problem A

Problem A – 題目大意

- 給定平面上一些互不交的簡單多邊形
- 問從無限遠所有方向打光的陰影面積
- 總點數 $N \leq 90$

Problem A – 解法

- 任意一個點若是在陰影內，那就沒有辦法打射線不穿過任何邊
- 反之，我們可以稍微旋轉射線方向直到碰到某個頂點
- 換言之，若一個點 P 不在陰影內則有另一個頂點 A 滿足
  - $\overline{PA}$ 不與其他邊相交
  - $\overrightarrow{PA'}$ 不與其他邊相交，其中 A' 是 A 對 P 的對稱點

Problem A – 解法

- 對於每個頂點，算好 ban 掉的區域，取聯集，並取補集，再減去原先的多邊形就是答案
- 對於每個頂點，將所有點與其對稱點加入，做極角排序
- 對於一個極角區間，如果往反向打射線沒有交到點，則 ban 掉這個極角區間與最近撞到線段所形成的三角形
- Naive 做也行，這部份複雜度應是 $O(N^3)$，會產出 $O(N^2)$ 個三角形

Problem A – 解法

- 簡單多邊形的聯集以 $O(V^2 \log V) = O(N^4 \log N)$ 的做法足敷使用
- 兩次交點（算三角形、聯集）精確度有機會不夠
- 但交點能預處理 $O(N^4 \log N)$，實際上這會讓聯集只須 $O(N^4)$ 且精確度高、跑得又快
- 事實上除了算答案以外能全整，但誤差允許下應當有些還是能浮點數的

Problem A – 解法

- 出題隊的標程 6845 bytes（去除空白）
- 可能是寫的雜了，但祝各位實作順利！

Problem B

Problem B – 題目大意

1. 給定一張無向圖和要求的 dfs 順序，問最少要加幾條邊才能達成該順序
2. 關鍵詞：圖論

Problem B – 解法

1. 本題有兩種做法
2. 做法 1
   1. 直接照著給定的順序 dfs，每次在「真的找不到下一個點，而且有其他的人要被 dfs 了」的時候加一條邊，然後繼續照著給定的順序 dfs。
   2. 複雜度瓶頸是排序所有人的鄰居，但這可以在線性時間內完成。
3. 做法 2
   1. 對於每個點 i，找到他連到自己前面、且最後面的點 x_i （找不到令 x_i 為 0）。
   2. 觀察到每個點 i 需要一條往前指一格的邊若且唯若 (x_i, i) 開區間內有指到比 i 後面的點。
   3. 每個點維護往後指到的最後面的點，若沒有就是指到自己。
   4. 由左往右維護大到小的單調隊列，每個點進來之前二分搜一下判斷前面的觀察有沒有合法就好。
   5. 可以在 $O(n \log n + m)$ 時間內通過，常數極小。

Problem C

Problem C – 題目大意

- 給定 l, r，一開始 $x = l$
- Alice 和 Bob 輪流做操作，一個人的操作過程如下：
  1. 選擇 $[l, r]$ 中被 x 整除且尚未被選過的數
  2. $x \leftarrow x + 1$
- 選擇不了任何數的人輸，問誰會贏

Problem C – 解法

- 觀察到有一個人永遠只能選擇偶數
- 討論 l 是奇數還是偶數

Problem C – 解法

- Case 1: l 是奇數
- $r < 2l$ 時，兩人都干擾不了對方，最佳策略就是拿走 x，因此直接判斷 $r - l$ 奇偶性即可
- $r \geq 2l$ 時，Alice 一開始直接拿走 $2l$ 後，接下來只要一直拿走 x 就一定贏
  - 第一步後，場上剩下的偶數數量不超過奇數數量，而且 Bob 拿不了偶數

Problem C – 解法

- Case 2: l 是偶數
- $r < 2(l+1)$ 時，兩人都干擾不了對方，最佳策略就是拿走 x，因此直接判斷 $r-l$ 奇偶性即可
- $r \geq 2(l+1)$ 時，Bob 第一次做操作時直接拿走 $2(l+1)$ 後，接下來只要一直拿走 x 就一定贏
  - 原因和 l 是奇數的情況類似

Problem D

Problem D – 題目大意

- 給定一個陣列 $a_1, a_2, \ldots, a_n$，每個元素都是 0 或 10^{18}
- 每次操作可以選一個 i 使得 $1 \le i \le n-2$，然後把 a_i 減 1 並且交換 a_{i+1} 和 a_{i+2}
- 問能不能把陣列變全 0

Problem D – 解法

- 先考慮一個簡單的情況
- 如果 $a_{n-1} = a_n = 0$，那 i 從 $n-2$ 掃到 1 就能把陣列變全 0
- 對於一個每一項都是非負整數且長度為 n 的陣列 c，定義 c 的特徵陣列為 $b_1, b_2, \ldots, b_k$ 滿足 $n-3 \geq b_1 > b_2 > \cdots > b_k$ 為所有 index b_j 使得 $1 \leq b_j \leq n-3$ 而且 $c_{b_j} > 0$

Problem D – 解法

- 定義集合 S，包含了所有陣列 c 使得
  - c 的長度為 n
  - c 的每一項都是非負整數
  - c 的最後兩項裡至少有一項不是 0
  - 定義 x 為最小的 i 使得 $n-2 \leq i \leq n$ 而且 $c_i > 0$，注意到根據條件 3，x 必存在
  - 令 $b_1, b_2, \ldots, b_k$ 為 c 的特徵陣列，那麼對於所有 b_j，都滿足 $b_j < x - 2j$
- 對任意 S 中的陣列做任意操作後形成的陣列都還是在 S 裡面
- 故如果 $a \in S$ 那不能把陣列變全 0

Problem D – 解法

- 如果 $a \notin S$，那有兩種情形
- 如果 $a_{n-1} = a_n = 0$，那能把陣列變全 0
- 否則定義 x 為最小的 i 使得 $n-2 \leq i \leq n$ 而且 $a_i = 10^{18}$，令 $b_1, b_2, \ldots, b_k$ 為 c 的特徵陣列，那存在 t 使得 $b_t \geq x - 2t$
- 令 r 是滿足條件的最小的 t

Problem D – 解法

- Case 1: $r = 1$
  - 如果 $b_1 = n - 3$，那 $x = n - 2$ 或 $n - 1$，那用以下操作能把陣列變全 0
    - $(10^{18}, 10^{18}, 10^{18}, 0) \to (10^{18}, 0, 10^{18}, 0) \to (10^{18} - 1, 10^{18}, 0, 0)$
    - $(10^{18}, 10^{18}, 10^{18}, 10^{18}) \to (10^{18}, 0, 10^{18}, 10^{18}) \to (10^{18} - 1, 10^{18}, 0, 10^{18}) \to (10^{18} - 1, 10^{18} - 1, 10^{18}, 0) \to (10^{18} - 2, 10^{18}, 10^{18} - 1, 0) \to (10^{18} - 2, 0, 10^{18} - 1, 0) \to (10^{18} - 3, 10^{18} - 1, 0, 0)$
    - $(10^{18}, 10^{18}, 0, 10^{18}) \to (10^{18}, 10^{18} - 1, 10^{18}, 0) \to (10^{18} - 1, 10^{18}, 10^{18} - 1, 0) \to (10^{18} - 1, 0, 10^{18} - 1, 0) \to (10^{18} - 2, 10^{18} - 1, 0, 0)$
    - 注意到 10^{18} 是偶數
  - 否則 $b_1 = n - 4$ 且 $x = n - 2$，那用以下操作能把陣列變全 0
    - $(10^{18}, 0, 10^{18}, 10^{18}, 0) \to (10^{18} - 1, 10^{18}, 0, 10^{18}, 0) \to (10^{18} - 1, 10^{18} - 1, 10^{18}, 0, 0)$
    - $(10^{18}, 0, 10^{18}, 0, 10^{18}) \to (10^{18}, 0, 10^{18} - 1, 10^{18}, 0) \to (10^{18} - 1, 10^{18} - 1, 0, 10^{18}, 0) \to (10^{18} - 1, 10^{18} - 2, 10^{18}, 0, 0)$
    - $(10^{18}, 0, 10^{18}, 10^{18}, 10^{18}) \to (10^{18} - 1, 10^{18}, 0, 10^{18}, 10^{18}) \to (10^{18} - 1, 10^{18} - 1, 10^{18}, 0, 10^{18}) \to (10^{18} - 1, 10^{18} - 1, 10^{18} - 1, 10^{18}, 0) \to (10^{18} - 1, 0, 10^{18}, 10^{18} - 1, 0) \to (10^{18} - 2, 10^{18}, 0, 10^{18} - 1, 0) \to (10^{18} - 2, 10^{18} - 1, 10^{18} - 1, 0, 0)$

Problem D – 解法

- Case 2: $r > 1$
  - 因為 $b_r \geq x - 2r$ 且 $b_{r-1} < x - 2(r-1) = x - 2r + 2$，所以 $b_r = x - 2r$
  - 可以將陣列的第 $b_r = x - 2r$ 項到第 x 項變成 $d_1, 0, d_2, 0, \ldots, 0, d_{r+1}$，其中 d_i 是正整數
  - 方法是每次選最小的 i $(x - 2r \leq i \leq x - 2)$ 使得 $a_i, a_{i+1} > 0$ 而且 $a_{i+2} = 0$，對 index i 做操作
  - 由於 10^{18} 很大，正整數操作完還是正整數
  - 做完之後用類似 $r = 1, b_1 = n - 4$ 的構造能把陣列變全 0
- 故如果 $a \notin S$，那能把陣列變全 0
- 用 $O(n)$ 判斷 a 是否在 S 裡面即可通過

Problem E

Problem E – 題目大意

- 給一張 N 點 M 邊的連通簡單不帶權無向圖，求是否所有簡單環長度都一樣。

Problem E – 解法

- 一個簡單環上必定同屬於一個點雙連通分量，只要每一個點雙連通分量中，簡單環都一樣長，就只要最後再檢查這些長度是不是都一樣即可。
- 因此我們接下來會把重點放在同一個點雙連通分量上。

Problem E – 解法

- 先看看一個簡單環都一樣長的圖會長什麼樣子，假設這個長度是 a，考慮 DFS 生成樹，每一條 back edge 兩端的樹上距離必須恰好是 $a-1$。
- 對於一條 back edge，假設它兩的端點之間在樹上的路徑從祖先到子孫是節點 $1, 2, \ldots, a$，如果 $2, \ldots, a-1$ 有伸出別的 back edge，那只能是這兩種：
  - 從 $a/2$ 往 a 的方向，連到一個和 $a/2$ 距離是 $a-1$ 的節點 b，而且 $a/2, b$ 之間的樹上路徑經過 a。
  - 從 $a/2+1$ 往 1 的方向，連到一個和 $a/2+1$ 距離是 $a-1$ 的節點 b。$a/2+1, b$ 之間的樹上路徑當然會經過 1。

  不然就會出現長度不是 a 的環。
- 而且這兩種還不能同時出現，不然會出現一個長度是 $a+2$ 的環。第一種的可以有很多條，第二種的因為往祖先方向只能連到同一個點，有兩條這張圖就不簡單了，所以根據限制只會出現一條。

Problem E – 解法

- 畫一下就會發現，一個點雙連通分量就只能長成：
  - 它就只有一條邊，或它就只是一個環。
  - 有恰兩個度數 > 2 的節點，剩下是一堆連接這兩個點、有 $a/2$ 條邊的路徑，除此之外沒有其他任何邊。(所以 a 在這個 case 裡得是偶數。)
- 檢查是不是每個點雙連通分量都長這樣，以及環長度是不是都一樣，可以 $O(n)$ 達成。
  - 實作時，要注意檢查一個點雙連通分量時，不可以「對一個點枚舉它的所有原圖鄰點，判斷是不是在這個點雙連通分量裡面」，這樣複雜度會退化成 $O(n^2)$。

Problem F

Problem F – 題目大意

1. 給定一個特定的隨機排序演算法：每次將序列切割成盡量多塊，使得每一塊內包含所有各自該有的數字，每一塊獨立區間 shuffle 後遞迴呼叫該演算法。
2. 求排序完成的期望區間 shuffle 呼叫次數。

Problem F – 解法

1. 若直接打亂整個長度 n 的序列再使用原演算法，答案會是一個固定的值，令這個答案是 $f(n)$，顯然我們只需要解這個 $f(n)$
2. 列出 $f(n)$ 的轉移式

$$f(n) = \sum_{S^n} \left(p(S^n) \cdot \sum f(S_i^N) \right) + 1, \forall n > 1$$

   其中 S^n 跑遍所有 n 的整數有序分割，$p(S^n)$ 是 S^n 出現的機率
3. 這邊 $p(S^n) \cdot n! \neq \prod S_i^N!$，因為每一塊必須要不可分割
4. 令大小 k 的 permutation 不可分割的方法數為 $g(k)$（OEIS A003319），則有轉移式

$$g(k) = k! - \sum_{i=1}^{k-1} g(i) \cdot (k-i)!$$

5. 因此 $p(S^n) \cdot n! = \prod g(S_i^N)$

Problem F – 解法

1. 將 $g(k)$ 帶回原式得

$$f(n) = \sum_{S^n} \left(\frac{1}{n!} \prod g(S_i^N) \cdot \sum f(S_i^N) \right) + 1$$

2. 令 $h(n) = f(n) \cdot n!, \forall n > 1$ 和 $h(1) = 1$，得

$$\begin{aligned} h(n) - n! &= \sum_{S^n} \left(\prod g(S_i^N) \cdot \sum f(S_i^N) \right) \\ &= g(n) \cdot f(n) + \sum_{j=1}^{n-1} \sum_{S^{n-j}} \left(g(j) \prod g(S_i^{n-j}) \cdot (f(j) + \sum f(S_i^{n-j})) \right) \\ &= g(n) \cdot f(n) + \sum_{j=1}^{n-1} (g(j) \cdot f(j) \cdot (n-j)! + g(j)(h(n-j) - (n-j)!)) \end{aligned}$$

3. 其中有一步用到 $n! = \sum_{S^n} \prod g(S^n)$
4. 以上全部都可以用分治 +NTT 在 $O(n \log^2 n)$ 的時間內算完

Problem G

Problem G – 題目敘述

- 給定 n 條紅直線與 n 條藍鉛直線，定義一紅線與一藍線配對的價值為它們交點的 y 座標
- 請將他們兩兩配對，並最大化價值們的中位數
- $n \leq 10^5$

Problem G – 題解

- 考慮二分答案，並考慮計算最多能有幾個價值 $\geq z$
- 若紅線斜率為正，與之配對的藍線 x 座標需至少為某值
- 若紅線斜率為負，與之配對的藍線 x 座標需至多為某值
- 特判紅線斜率為零（水平線）

Problem G – 題解

- 易知將 x 座標大的藍線與斜率為正的紅線配對最優。將 x 座標小的藍線與斜率為負的紅線配對最優
- sort 完後簡單 greedy
- 總複雜度 $O(n \log^2 n)$

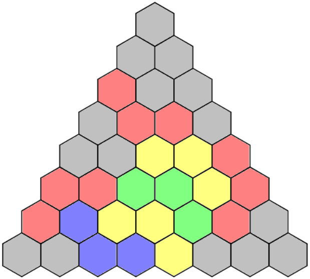
Problem G – 注意細節

- 負數時的整數除法
- 二分搜上下界，需包含 $[-2 \times 10^{18}, 2 \times 10^{18}]$

Problem H

Problem H – 題目敘述

- 給定大小為 n 三角形狀的六邊形棋盤，在上面放盡量多個 V 型拼圖。拼圖有 26 種顏色，相鄰的不能同色

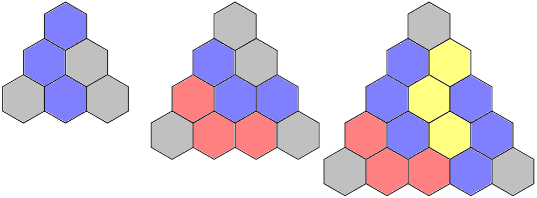


Problem H – 題解

- 數歸一下可以發現對於每條邊界一定都會有一個六邊形放不了
- 三條邊界 $\implies$ 至少有兩個六邊形放不了
- 令 x 為棋盤總六邊形數量，則有答案上界 $\lfloor \frac{x-2}{3} \rfloor$

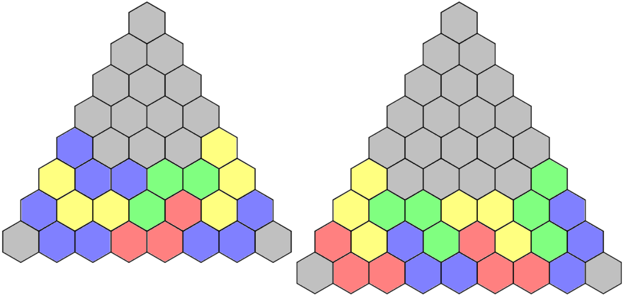
Problem H – 題解

- 手玩一下小測資：

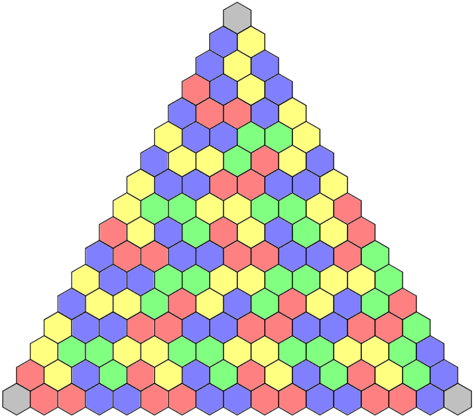


Problem H – 題解

- 對於大的 n 考慮以下歸約到 $n-3$ 的構造：



Problem H – 題解



Problem H – 塗色方法

- greedy 亂塗
- 找出該拼圖 x 座標最小的值與 y 座標最小的值，用兩者的座標模 4 作為顏色

Problem I

Problem I – 題目大意

- 給定 n 個非負整數和 $k \geq 2$
- 一次操作可以把其中 k 個整數拿走換成它們的乘積
- 輸出最後所剩的最大數的最大值
- 輸出要 $\bmod 998244353$

Problem I – 觀察

- 一定是對目前最大的 k 個數做操作
- 如果無論如何都有 0 參與操作就不要做操作

Problem I – 解法

- 將所有數字按大到小排序，先把最大的數字拿出來作為 ans
- 反覆考慮往後看的 $k-1$ 個數，只要有 0 就 break
- 否則將 ans 乘上這 $k-1$ 個數
- 陷阱 1：記得 ans 要一直取模，連一開始也要
- 陷阱 2：判斷 $k-1$ 個數中有沒有 0 不能先在 $\bmod 998244353$ 下乘起來判是否為 0

Problem J

Problem J – 題目大意

- 以 properly nested 矩形給定一棵樹，支援兩種操作
- 1. 切換一個點（開到關或關到開）
- 2. 詢問深度為 k 的所有點，有多少點的子樹內有點是開的
- $n, q \leq 5 \times 10^5$

Problem J – 解法

- parse 樹形不難，以 `std::set` 就能做完
- 考慮改變一個點就是從點到根全 $+1$ 或 -1，詢問等價問同深度有多少點的權重 > 0
- 直接維護 > 0 的區域不難：每次改變都是加入或刪除一條路徑
  - 從修改點開始至修改點到其他所有標記點的最深 LCA
  - 不含右端點
- 使用 `std::set` 就能以 DFS 序輕易維護 LCA 詢問，修改只須一棵 BIT
- 預期時間複雜度 $\mathcal{O}((n+q)\log n)$

Problem K

Problem K – 題目大意

1. 給定一張二分完全圖去掉 m 條邊，所有剩下的邊滿足 u, v 之間的邊權為 $a_u + a_v$ 模 998244353
2. 對於每個 $1 \leq i \leq k$，求匹配 i 條邊的最大權重
3. 關鍵詞：圖論、匹配

Problem K – 解法

1. 先來看一遍整題怎麼做，會需要用到三個 Claim
2. Claim1: 對於二分圖左側的每個點，只需要保留前 k 大的鄰邊
3. Claim2: 在 Claim1 篩選過的圖裡，對於二分圖右側的每個點，只需要保留前 k 大的鄰邊
4. Claim3: 對於 Claim2 留下來的邊，只有前 $(2k-1)(k-1)+1$ 大的那些是有用的

Problem K – 解法

1. 邊權本身沒什麼特殊性質，重點只是要能拿到每個點前 k 大的鄰邊，這可以用雙指針線性得到。
2. 好好使用 `nth_element` 之類的東西也就可以在線性時間內完成 Claim2 和 Claim3 要求的篩選
3. 最後可以留下 $O(k^2)$ 個點和邊，使用 Dijkstra 版本的費用流就可以在 $O(k^3 \log k)$ 的時間內得到答案，當然這裡使用 SPFA 實作也是能通過的

Problem K – 證明

1. Claim1: 對於二分圖左側的每個點，只需要保留前 k 大的鄰邊
2. Claim2: 在 Claim1 篩選過的圖裡，對於二分圖右側的每個點，只需要保留前 k 大的鄰邊
3. 以上兩個 Claim 的正確性是顯然的，若匹配到的邊是被丟掉的邊，那就會有更好的選擇

Problem K – 證明

1. Claim3: 對於 Claim2 留下來的邊，只有前 $(2k-1)(k-1)+1$ 大的那些是有用的
2. 這是因為每個點的度數在此時不超過 k，因此對於每個匹配邊，他至多會額外擋住 $2k-2$ 條邊，加上匹配自己的邊就是一口氣消耗掉 $2k-1$ 條邊
3. 當匹配 $k-1$ 條後，至多會消耗 $(2k-1)(k-1)$ 條邊，因此多保留一條就能找到第 k 條匹配，證畢

Problem K – 假解

1. 本題存在許多假解，包含
2. 直接挑選最大的 $O(k^2)$ 條邊 $\Rightarrow$ 這顯然是錯的
3. 對於每個點，保留其前 k 大的鄰邊，並直接找出這些邊當中的前 $O(k^2)$ 大條邊 $\Rightarrow$ 這會失去每個點度數不超過 k 的保證
4. 只保留 $(2k-1)(k-1)$ 條邊 $\Rightarrow$ 順著前一頁的證明就能構造出一組不夠的測資，這個界是緊的

Problem L

Problem L – 題目敘述

- n 個人圍著一個轉盤，一開始第 $[(i + x) \bmod n]$ 個人面前有第 i 道菜，所有菜都在轉盤上
- 花費一秒鐘可以將轉盤順時針或逆時針轉動一格
- 有 m 個條件，每個條件會是某個人想要在某個時間之前吃到某一道菜
- 對於所有 $0 \le x < n$，輸出能不能讓滿足所有條件
- $n \le 5000, m \le 10^5$

Problem L – 題解

- 將目前的第 0 個人面前的菜的編號當成狀態，每個條件會變成「在某個時間點前到達某一段環狀連續區間的任一個狀態」，轉動轉盤會變成狀態 +1/-1
- 拜訪過的狀態會是一段環狀區間
- $dp[k][l][r]$ 代表至多要在多少時間之前將環狀區間 $[l..r]$ 的狀態都拜訪過才能符合所有條件，且若 $k = 0$，最後會停在狀態 l，$k = 1$ 最後會停在狀態 r
- base case：$dp[*][i][(i-1) \bmod n] = \infty$，轉移考慮最後一步是從哪裡轉過來即可。而若 $dp[0][i][i] \geq 0$ 第 i 個答案為 1
- 先用條件去預處理每個環狀區間必須要在多少時間之前都拜訪過才不會違背條件就可以很輕鬆的轉移
- 總複雜度 $O(n^2 + m)$

Problem M

Problem M – 題目大意

- 給定三維上的點，求是否有直圓柱的表面通過所有點
- 直圓柱的底面要在 x-y 平面上
- $n \leq 10^5$

Problem M – 解法

- 圓柱的方向已經給定
- 底面 $z = 0$，若有解頂面可調整至 $z = \max z_i$
- 問題轉換為：給定兩個 2D 點集 A, B
  - A 需在圓上
  - B 需在圓內或圓上

Problem M – 解法

- Case 1: $|A| = 0$
  - 永遠有解
- Case 2: $|A| = 1$，令 $A = \{P\}$
  - 問是否有個半平面通過 P 使得 B 都在平面嚴格的某一側
  - 各種解法：蓋凸包判斷或判以 P 為原點的極角
- Case 3: $|A| \geq 3$
  - 任抓三點 $P, Q, R \in A$ 以外接圓檢查
  - P, Q, R 共線則無解

Problem M – 解法

- Case 4: $|A| = 2$，令 $A = \{P, Q\}$
  - 事先檢查 $\overleftrightarrow{PQ}$ 上的點皆需落在 $\overline{PQ}$
  - 將平面沿 $\overleftrightarrow{PQ}$ 切半分為左右
  - 檢查左側「最遠點」與右側「最遠點」是否能在一個圓內
  - 「最遠點」：找到點 X 最小化 $\angle PXQ$
  - 事實上對同側點，若 PXQ 外接圓包含 Y 那 $\angle PXQ \leq \angle PYQ$
  - 可以直接使用 Case 3 的判斷點是否在三點外接圓內

Problem M – 解法

- 事實上，判斷點是否在三點外接圓內以浮點數算可能會有精度問題
- 使用一點技巧可以純整：
  - 圓內接四邊形對角相加 180°，用算 $\sin$ 避掉浮點數
  - 托勒密定理（能判是否剛好在圓上）
  - 若 a, b, c 逆時針，則能算以下行列式的號決定：

$$D = \begin{vmatrix} a_x & a_y & a_x^2 + a_y^2 & 1 \\ b_x & b_y & b_x^2 + b_y^2 & 1 \\ c_x & c_y & c_x^2 + c_y^2 & 1 \\ p_x & p_y & p_x^2 + p_y^2 & 1 \end{vmatrix}$$

Source: Math SE

Problem M – 解法

- 全整作法可能需要使用 128-bit 整數（`__int128`）但瞎寫大數應該還是能通過：時限非常寬裕
- 好好處理浮點數誤差，或乾脆用四倍精度（`__float128`）
- 複雜度 $\mathcal{O}(n \log n)$ 或 $\mathcal{O}(n)$

Problem M – Side Note

- 本題命題的時候對值域相當為難：低值域的圓相當無趣，但值域大全整作法塞不進 64-bit
- 測資有努力卡掉精確度不夠好的作法，例如一些外心的算法實際上不夠穩定