彰化高中112學年度資訊學科能力競賽校內初選 題解

A.祭典 *At Most 3*

> 出題者 ***Derek0***

subtask1: $35\%\ 1 \le W, N \times A_i \le 10^9$

三個迴圈搭配std::set即可

subtask2: 65% ***As statement***

記得開long long

B.翻譯家 *French*

> 出題者 ***Derek0***

task: 100% ***As statement***

依照題意模擬

C.村長 *King*

> 出題者 ***Derek0***

Subtask

subtask1: 20%: $1 \le X, Y \le 10^9$

使用迴圈枚舉每一位的數字

subtask2: 30%: $1 \le X, Y \le 10^{18}$

開long long

subtask3: 50%: ***As statement***

使用std::string輸入即可

D.選糖果 *Pick*

> 出題者 ***ysh***

Subtask

subtask1: $10\%\ 1 \le n, m \le 10$

唬爛用，甚至 $O(2^n \cdot m)$ 都會過。

subtask2: $10\%\ 1 \le n \le 10^5, 1 \le m \le 10$

我們可以使用貪心法：將陣列 a 排序後，從前面一個一個試過來，每次檢查是否超過背包所能承受的重量。

總時間複雜度 $O(nlog_2^n + nm)$

subtask3: $30\%\ 1 \le n \le 10^5, 1 \le m \le 10^5$, $(\forall 1 \le i \le n,\ 1 \le a_i \le 1000)$

這個子題有點酷，我們可以發現 n , m 都大得不像話，唯一可以動手腳的，大概就只有 $a_i \leq 1000, \forall\ 1 \leq i \leq n$

既然每個值都小於等於 1000 那我們完全可以開個陣列並記錄每個數字出現的次數。

然後從 1 數到 1000 ，以除法的方式判斷目前背包使否裝得下。

總時間複雜度 $O(1000m)$ 。

subtask4: 50% ***As statement***

我們使用**前綴和**和**二分搜**來優化。

先對 < a > 做前綴和，而後對於每一筆詢問，只需要花 $O(\log_2^n)$ 的時間即可完成查詢。

總時間複雜度 $O((n+m) \cdot \log_2^n)$ 。

E.更多的糖果 *Pick-II*

> 出題者 ***ysh***

subtask1: 10% $1 \leq n, m \leq 10$

唬爛用，甚至 $O(n^5)$ 都會過。

subtask2: 30% $1 \leq n, m \leq 100$

這就需要使用**前綴和**了

假設今天給你一個平面 < a > :

```
1 1 1
1 1 1
1 1 1
```

如果想用暴力取總和的話，複雜度會是 $O(n^2)$ ，但是如果我們先造出另外一個序列 < g > 的話...

$$\text{Define } g_{ij} = \sum_{k=1}^{i} \sum_{l=1}^{j} a_{kl}$$

我們可以發現 < g > 為:

```
1 2 3
2 4 6
3 6 9
```

那要如何利用 < g > 取得 $\sum\limits_{k=x_0}^{x} \sum\limits_{l=y_0}^{y} a_{kl}$ 呢?

答案就是...

$$\sum_{k=x_0}^{x} \sum_{l=y_0}^{y} a_{kl} = g_{xy} - g_{x_0 y} - g_{x y_0} + g_{x_0 y_0}$$

看出來了嗎
其實就是~~用湊的~~

因此，我們只需要花**一次** $O(n^2)$ 的時間建表，即可用坐享 $O(1)$ 的查詢速度。
對於這題來說，只要從小到大依次列舉 r ，每次再用 $O(nm)$ 來窮舉右下角的座標，最後用 $O(1)$ 的時間算出矩形和即可通過此題。
總時間複雜度 $O(nm \cdot min(m,n))$ 。

subtask3: 10% $1 \le n,m \le 500$

這個子題本來應該沒什麼用，但後來發現好像可以卡忘了開***long long***的解。

subtask4: 50% ***As statement***

solution 1

可以發現 n 跟 m 最大可以來到 2500 ，如果我們使用上面的解法，最多需要進行
$nm \cdot min(m,n) \to 2500^3 = 1.5625 \times 10^{10}$ 次運算，遠超過機器兩秒所能進行的運算。

這題其實存在二分搜解法，因為答案具有單調性。所以我們可以對答案進行二分搜。

這可以讓時間複雜度降至 $O(nm \cdot \log_2^{min(m,n)})$ ，機器只需進行約莫
$nm \cdot \log_2^{min(n,m)} \to 2500^2 \cdot \lceil \log_2^{2500} \rceil = 7.5 \times 10^7$ 次運算，故能在兩秒內完成。

solution 2

直接使用***sliding window***技巧做掉這題。

同樣事先求出前綴和。

從 $(1,1),(2,1),(3,1)\dots(n,1)$ 這些點往右下 $+x,+y$ 方向做，
再從 $(1,1),(1,2),(1,3)\dots(1,m)$ 這些點往右下做。

就可以求出答案了。

因為對角線方向上每個點最多進入、出去一次***window***，所以複雜度均攤 $O(nm)$ 。

F.循環小數 *Repeating Decimal*

> 出題者 ***gamic1234***

subtask1: 30% $1 \le q < 10,$ 且循環長度保證不超過10^5

觀察到只要除數固定，循環長度就會是一個定值，於是事先算好答案直接回答

subtask2: 70% ***As statement***

直接模擬除法，用一個陣列紀錄哪些被除後的餘數有出現過且出現在哪裡，遇到有出現過的結束模擬，算出循環長度。
順帶一提，用map記錄會多一個log導致TLE

G.運算子 *Operator*

> 出題者 ***ysh***

這邊定義 $True^n$ 代表 n 個 $True$ 互相進行 ***XOR*** 運算。
$False^n$ 代表 n 個 $False$ 互相進行 ***XOR*** 運算。
$True \cdot False$ 為 $True$ 對 $False$ 進行 ***XOR*** 運算。

$True^0 \cdot False^0$ 為 0 。

subtask1: 40% $0 \le a, b \le 10^5$

直接模擬

subtask2: 60% ***As statement***

當數字變大後，我們可以發現執行速度越來越慢。

但是其實 $False$ 完全不影響計算結果，會影響答案的，只有 $True$ 數量的奇偶性，以下為證明。

$$\text{Proof: } When\ a\ \equiv 1\ (mod\ 2),\ True^a \cdot False^b = True$$

首先，我們有 $False^n = False$ ， 因為由歸納法原理：
$False^n = False \cdot False \cdot False^{n-2} = False^{n-1}$ ， $False^1 = False$ 。

接著，我們有 $True^a = True, a \equiv 1\ (mod\ 2)$ ，因為：
$True^n\ = True \cdot True \cdot True^{n-2}\ = False \cdot True^{n-2}\ = False \cdot True \cdot True^{n-3}\ = True \cdot True^{n-3}\ = True^{n-2}$ ， $True^1 = True$ 。

因此， $True^a \cdot False^b = True^{a\%2} \cdot False = True^{a\%2}$ ，得證。
複雜度 $O(1)$ 。