|  |  |
| --- | --- |
| **Exam I Review Report** | |
| 學號: 109062318 | 姓名: 簡弘哲 |

**補題表格**

|  |  |  |
| --- | --- | --- |
| 題號 | 賽中是否有通過 | 是否要補題 |
| PA | O | X |
| PB | O | X |
| PC | X | O |
| PD | O | X |
| PE | X | O |
| PF | X | O |

**PC**

1. **解題報告**

(1)解題想法:，你是如何解決這個問題的?用了甚麼演算法?

資料結構使用了multiset(因為有重複的數字)與unordered map(記錄每個數字的個數，不在乎排序)。為了實作上的方便，我在隊伍的兩頭加入不可能出現的數字(-1e9, INT\_MAX)當作wall；接著把隊伍中每一個人的位置都塞進multiset “posi”(position)中；multiset “neg”(negative)的初始方式是將posi中的每個數字加上負號後，再塞進neg中，neg的存在是為了「能更快速的取得某個位置的左右鄰居」(原本是用multiset的iterator透過prev( ), next( )去找某個位置p的左右鄰居，但這會TLE)。另外，為了能更快速地取得任兩人之間的距離，還有一個multiset “dist”(distance)用來記錄目前隊伍中的間隔情況。因此如果隊伍是[1, 1, 2, 1e9, 1e9]，則

posi = [-1e9(wall), 1, 1, 2, 1e9, 1e9, INT\_MAX(wall)]

neg = [-INT\_MAX(wall), -1e9, -1e9, -2, -1, -1, 1e9(wall)]

dist = [0, 0, 1, (1e9 – 2), (1e9 + 1), (INT\_MAX - 1e9) ]

當有人加入或離開隊伍時，都要更新posi, neg, dist，才能在詢問社交距離時給出正確的答案，以上述的隊伍[1, 1, 2, 1e9, 1e9]為例，當3要加入時，先找出3加入後的左右鄰居(2和1e9)，將距離(1e9 - 2)從dist中移除，並加入新距離1 (3-2)與(1e9 - 3)。有人離開隊伍時，也是運用類似的想法。

(2)複雜度分析，這樣的時間複雜度能通過是不是合理的?

讀入資料與前置處理: n個數字n次emplace + sorting => O(nlgn)。

Q次詢問: 有使用到multiset的erase( ), find( ), lower\_bound( ) => O(qlgn)。

詢問社交距離: 使用size( ), begin( )，皆為constant time。

總時間複雜度: O( (n+q)lgn )，n, q <= 1e5，可以過。

(3)在比賽中為甚麼沒有寫出這題? (簡述即可)

這題是在比賽後期才開始想，腦袋思考效率差。賽中雖然有寫出code的大概架構，但因為沒有考慮好edge case導致有bug，且時間上也來不及de。

1. **補題 AC 連結網址**

https://codeforces.com/gym/486864/submission/233763585

**PE**

1. **解題報告**

(1)解題想法:，你是如何解決這個問題的?用了甚麼演算法?

演算法用了BFS，資料結構有queue<pair<int,int>>，其中pair的第一個int為棋盤的格子編號、第二個int為到該格所需的骰子投擲次數；另外也維護一個長度為1e6的visited陣列，每走到一個新的格子就將visited[該格編號]設為true，避免之後重複走訪。

從queue中取出一個pair時，如果到x了就印出當前的步數，否則就枚舉骰子的每個面，如果新的著陸點還沒被走訪過，我才會將它塞入queue中。

(2)複雜度分析，這樣的時間複雜度能通過是不是合理的?

因為visited陣列會記錄有哪些格子被走過，所以每個格子最多只會被走過一次，時間複雜度為O(n)，n=格子數量。

(3)在比賽中為甚麼沒有寫出這題? (簡述即可)

賽中看完題目完全沒看出它是BFS，還以為是遞迴，因為骰子最多50面並沒有很多。

1. **補題 AC 連結網址**

<https://codeforces.com/gym/486864/submission/233774399>

**PF**

1. **解題報告**

(1)解題想法:，你是如何解決這個問題的?用了甚麼演算法?

類似課程中所講解的數獨例子，盤面的每一格都有一個index (0~nm-1)，從index=0的格子開始枚舉，一直到index=nm時(已放完index=nm-1)就檢查盤面，如果合法的話就算一種擺法。每一格枚舉放或不放的可能性，有些格子沒辦法放炸彈就可以直接枚舉下一格，例如「目前這格是0」或是「這格的周圍九宮格內有0」的情況；另外，假設目前格子的row & col index為i與j，我們可以檢查目前(row, col) = (i-1, j-2)這個格子中的數字有沒有跟輸入的一模一樣，如果沒有的話，那我們就可以進行剪枝，因為在枚舉格子(i, j)的時候，接下來已經沒有任何格子能影響到(i-1, j-2)那格了。

(2)複雜度分析，這樣的時間複雜度能通過是不是合理的?

對於每一格都有放或不放兩種選擇，因此最差複雜度為O(2nm)，但因為有剪枝與優化，所以執行時間可以壓在兩秒之內。

(3)在比賽中為甚麼沒有寫出這題? (簡述即可)

讀完題目後發現是遞迴，估計要花上許多時間就先跳過了。

1. **補題 AC 連結網址**

<https://codeforces.com/gym/486864/submission/235674581>