一些重要事項宣佈

- 有學員反應 NEOJ 795 出了點技術問題，現在已經修好 + rejudge 了， 這題的 deadline 會延後兩天
- 除了 795 這題之外，judge 在 6/2 左右，出現了技術問題。從那個時候開始傳的 submission 都有可能踩到怪雷，如果擔心自己的 submission 踩到雷的話，可以重傳一次

  我們不會 rejudge 之前的 submission
- 136 是折半枚舉的題目，他已經被出在第九週的加分題了

一些重要事項宣佈

- 認證考在 6/19 (六), 線上作答
  - 帳號密碼會在 6/18 前寄信給各位
  - 出題範圍以第二階段講授的內容為主
  - 可以參考任何資料(紙本、線上), 唯獨不能跟其他人通訊
- 團體賽在 6/26 (六), 同樣也是線上作答
  - 組隊表單今天會釋出
  - 題目很酷
  - 敬請期待:)

根號算法

credit by nkhg (t1016d)
modified by yp155136
2021/06/12

Q & A

- 大家有任何關於影片的問題嗎 >//////<

上課大綱

- 數三角形 Review
- 值域分塊
- 塊狀鍊表
- 按照序列長度分 case
- 操作分塊

Counting Triangles

- 定義輕點是 degree < sqrt(M) 的點，重點是 degree >= sqrt(M) 的點
- 假設我們可以 O(1) 得知一個 pair (x, y) 之間有沒有邊
  - 會在根據不同的情況，給予不同的作法！
- 在這個題目中，我們得到兩種看待問題的方法：
  - 從點的角度來看
  - 從邊的角度來看

Counting Triangles

- 從點的角度來看：
- 給定一個點，可以 O(M) 算出有多少個三角形包含這個點
- 給定一個點，可以 O(d^2) 算出有多少個三角形包含這個點

- 輕點用 O(d^2)，重點用 O(M)，複雜度為 O(M sqrt(M))

- 對於每個重點，開一個 adjacency list，達成 O(1) 查詢
  - 時間、空間複雜度為 O(M * sqrt(M))
- 對於每個輕點，離線把詢問存起來，一起查詢

Counting Triangles

- 如何離線 O(1) 查詢？

- 把圖上的點標記起來
- 依序走過所有的詢問
- 把圖上的點取消標記

- 總花費時間，均攤下來是 O(M + 詢問數)

```
int ans = 0;
for (int i = 0; i < n; ++i) {
    // 這邊就是離線均攤 O(1) 的作法！
    for (int j: G[i]) {
        vis[j] = true;
    }
    for (int j: query[i]) {
        ans += vis[j];
    }
    for (int j: G[i]) {
        vis[j] = false;
    }
}
printf("%d\n", ans / 3);
```

Counting Triangles

- 從邊的角度來看:
- 假設邊 (u, v) 的 d_u < d_v
- 可以花 O(min(d_u, d_v)) 的時間查詢
- 整體複雜度為 O(M * sqrt(M))

中國人插隊問題

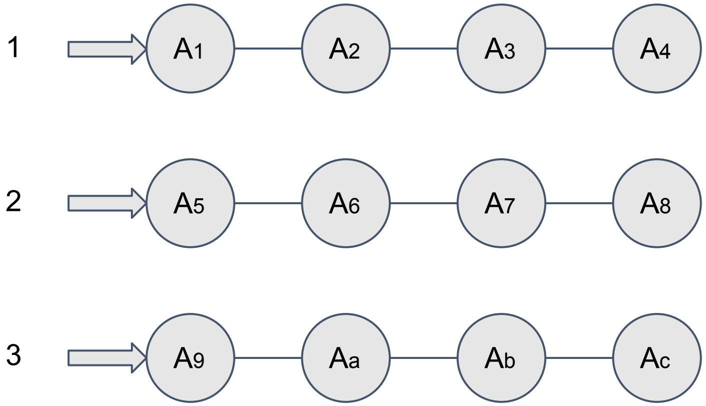
- 一開始給一個長度 N 的序列(index 1 到 N)
- 接著 Q 次操作，每次可能是
  a. 請在第 i 個位子插入數字 x(index >= i 的數字往後一格)
  b. 拔掉第 i 個位子的數字(index > i 的數字往前一格)
  c. 請你回答第 i 個位子的數字是多少
- 例如
  - 初始:1, 2, 3, 4, 5
  - 操作:(a, 1, 6), (a, 7, 7), (c, 4), (b, 3), (b, 3), (c, 5)
  - 過程:
    - 6, 1, 2, 3, 4, 5
    - 6, 1, 2, 3, 4, 5, 7（輸出 3）
    - 6, 1, 3, 4, 5, 7
    - 6, 1, 4, 5, 7（輸出 7）

中國人插隊問題

- 想想看怎麼用 RMQ 類似的方法來分塊吧～
- 雖然這題用平衡樹炸下去是可以 O(N log N) ，不過這裡我們先試試看分塊XD

中國人插隊問題

- 嘗試讓每塊儲存了 K 個數字（若 K = 4）

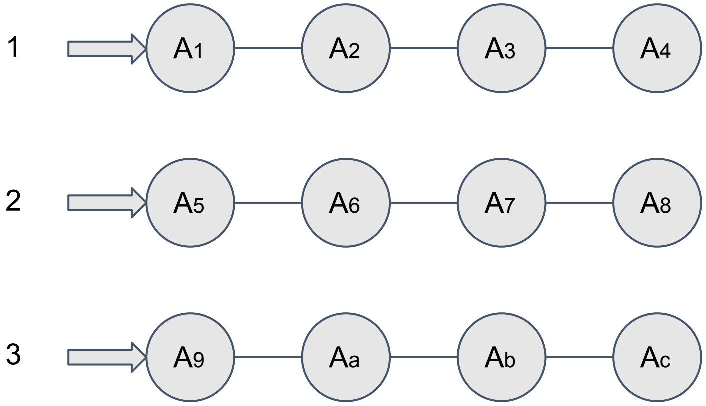


中國人插隊問題

- 一開始給一個長度 N 的序列(index 1 到 N)
- 接著 Q 次操作，每次可能是
  a. **請在第 i 個位子插入數字 x(index >= i 的數字往後一格)**
  b. 拔掉第 i 個位子的數字(index > i 的數字往前一格)
  c. 請你回答第 i 個位子的數字是多少
- 例如
  - 初始:1, 2, 3, 4, 5
  - 操作:(a, 1, 6), (a, 7, 7), (c, 4), (b, 3), (b, 3), (c, 5)
  - 過程:
    - 6, 1, 2, 3, 4, 5
    - 6, 1, 2, 3, 4, 5, 7 (輸出 3)
    - 6, 1, 3, 4, 5, 7
    - 6, 1, 4, 5, 7 (輸出 7)

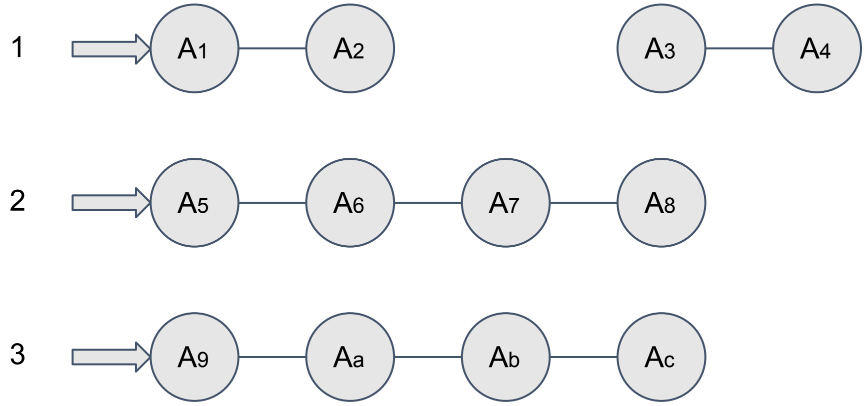
中國人插隊問題

- 增加一個數字(第 3 個位置插入 B)



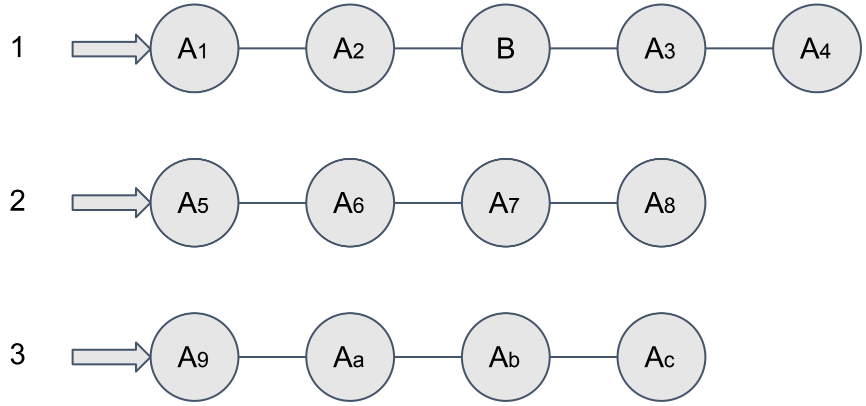
中國人插隊問題

- 增加一個數字（第 3 個位置插入 B）



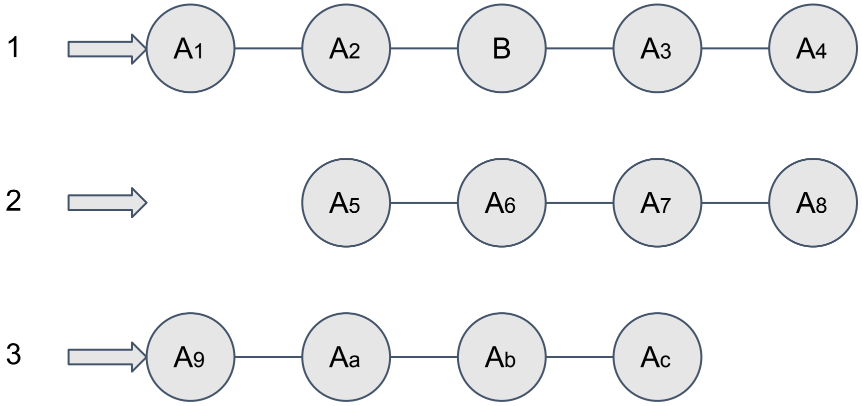
中國人插隊問題

- 增加一個數字(第 3 個位置插入 B)



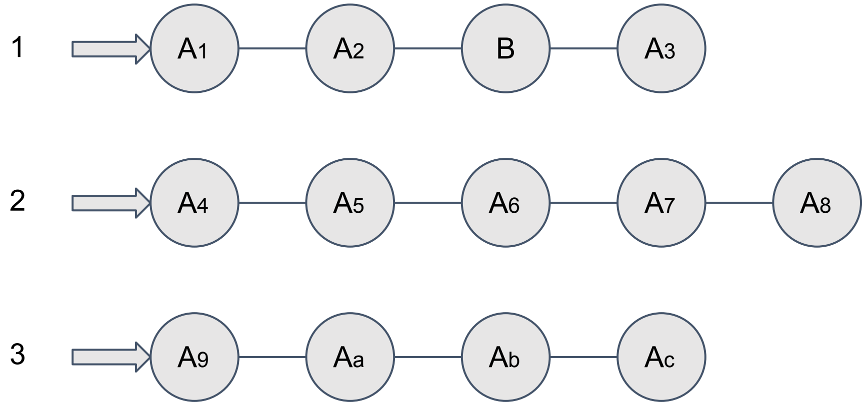
中國人插隊問題

- 增加一個數字(第 3 個位置插入 B)



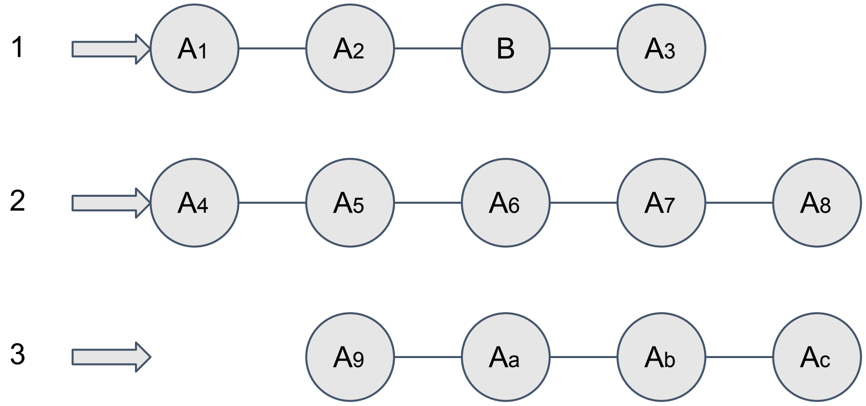
中國人插隊問題

- 增加一個數字(第 3 個位置插入 B)



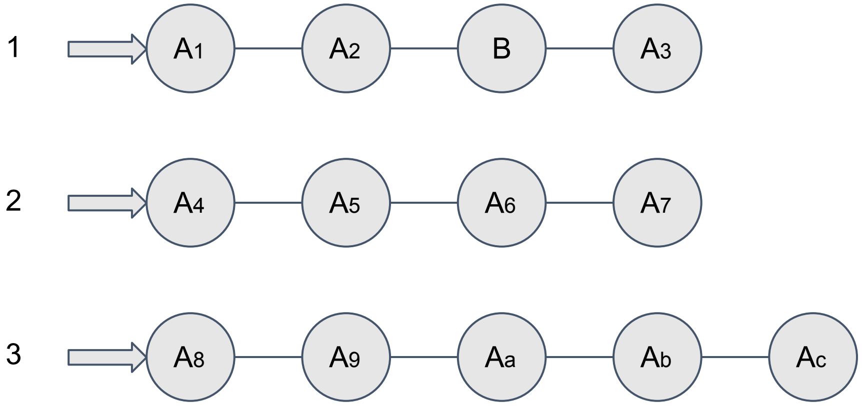
中國人插隊問題

- 增加一個數字(第 3 個位置插入 B)



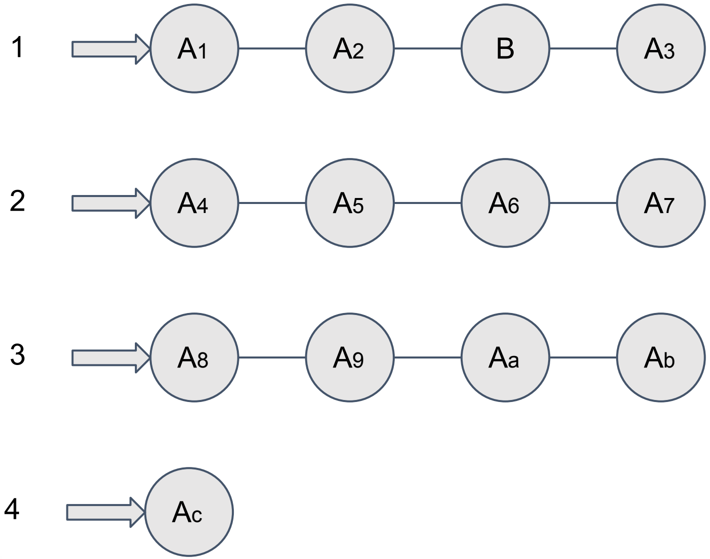
中國人插隊問題

- 增加一個數字（第 3 個位置插入 B）



中國人插隊問題

- 增加一個數字(第 3 個位置插入 B)

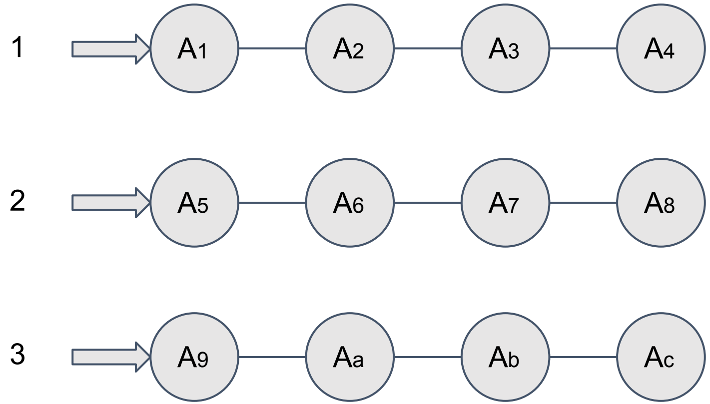


中國人插隊問題

- 一開始給一個長度 N 的序列(index 1 到 N)
- 接著 Q 次操作，每次可能是
  a. 請在第 i 個位子插入數字 x(index >= i 的數字往後一格)
  b. 拔掉第 i 個位子的數字(index > i 的數字往前一格)
  c. 請你回答第 i 個位子的數字是多少
- 例如
  - 初始:1, 2, 3, 4, 5
  - 操作:(a, 1, 6), (a, 7, 7), (c, 4), (b, 3), (b, 3), (c, 5)
  - 過程:
    - 6, 1, 2, 3, 4, 5
    - 6, 1, 2, 3, 4, 5, 7（輸出 3）
    - 6, 1, 3, 4, 5, 7
    - 6, 1, 4, 5, 7（輸出 7）

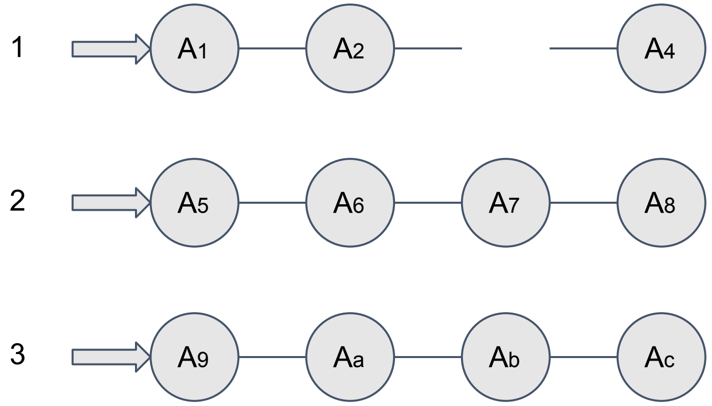
中國人插隊問題

- 拔掉一個數字(拔掉第 3 個位置)



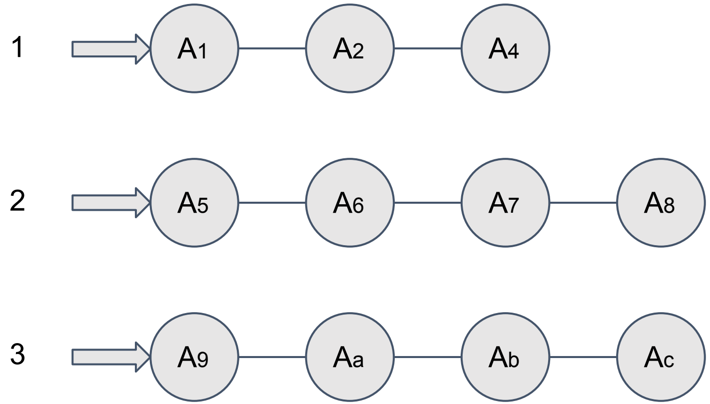
中國人插隊問題

- 拔掉一個數字（拔掉第 3 個位置）



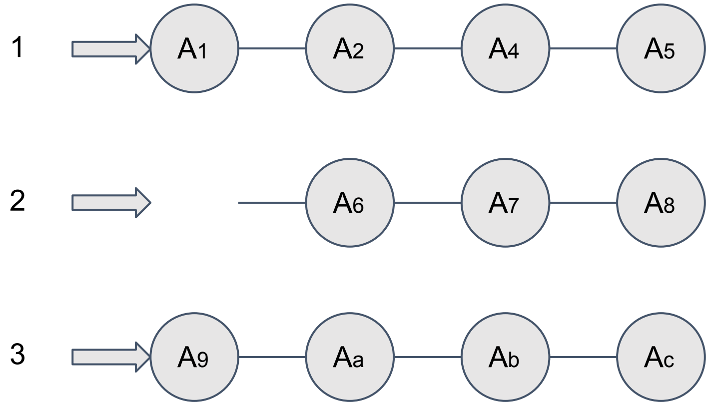
中國人插隊問題

- 拔掉一個數字(拔掉第 3 個位置)



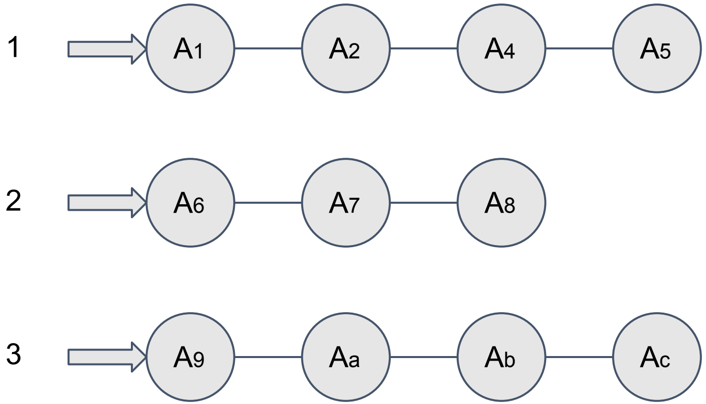
中國人插隊問題

- 拔掉一個數字（拔掉第 3 個位置）



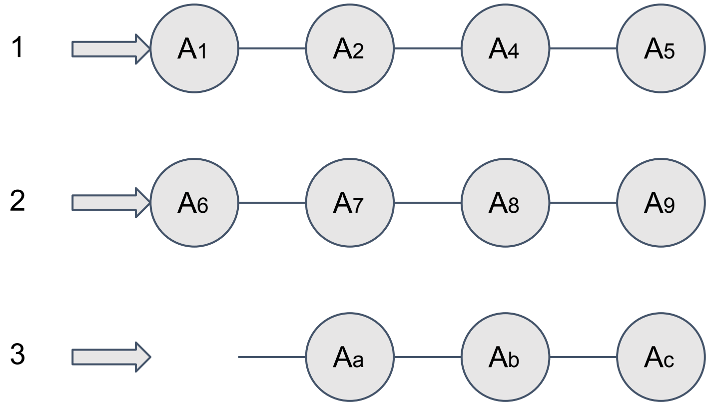
中國人插隊問題

- 拔掉一個數字（拔掉第 3 個位置）



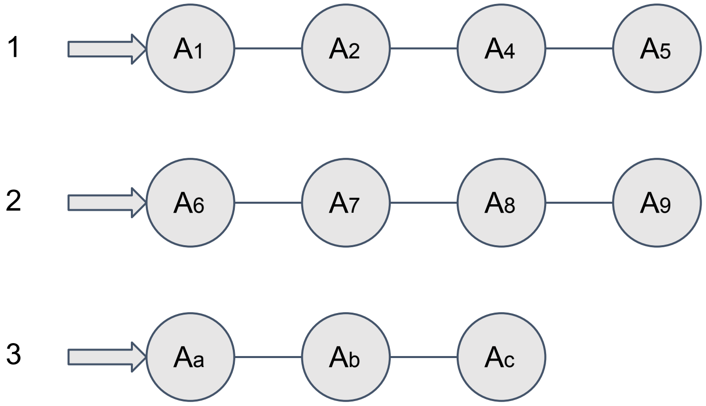
中國人插隊問題

- 拔掉一個數字（拔掉第 3 個位置）



中國人插隊問題

- 拔掉一個數字(拔掉第 3 個位置)

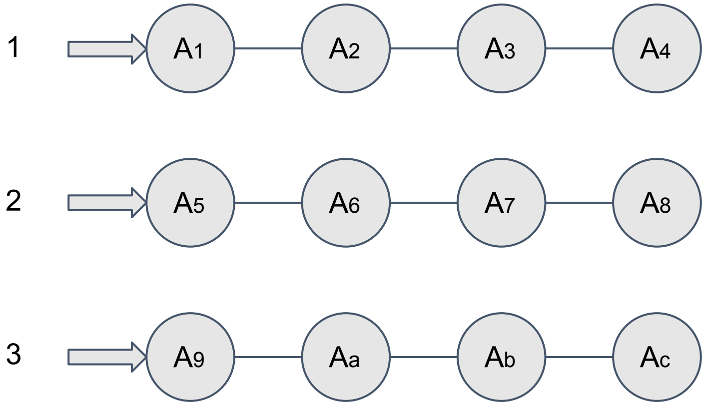


中國人插隊問題

- 一開始給一個長度 N 的序列(index 1 到 N)
- 接著 Q 次操作，每次可能是
  a. 請在第 i 個位子插入數字 x(index >= i 的數字往後一格)
  b. 拔掉第 i 個位子的數字(index > i 的數字往前一格)
  c. 請你回答第 i 個位子的數字是多少
- 例如
  - 初始:1, 2, 3, 4, 5
  - 操作:(a, 1, 6), (a, 7, 7), (c, 4), (b, 3), (b, 3), (c, 5)
  - 過程:
    - 6, 1, 2, 3, 4, 5
    - 6, 1, 2, 3, 4, 5, 7 (輸出 3)
    - 6, 1, 3, 4, 5, 7
    - 6, 1, 4, 5, 7 (輸出 7)

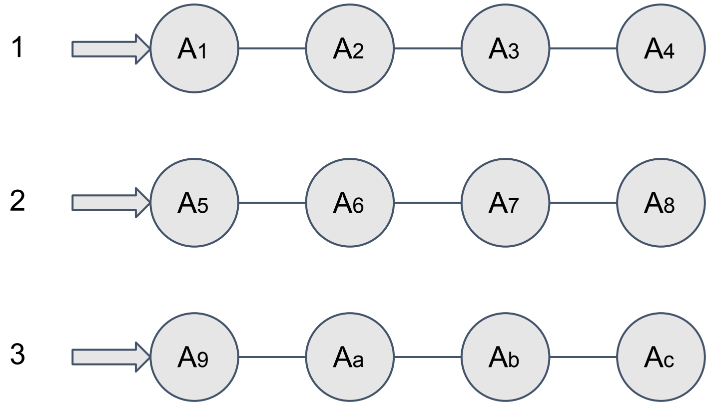
中國人插隊問題

- 第 i 個數字:第 ________________ 塊的第 ____________ 個



中國人插隊問題

- 第 i 個數字:第 (i - 1) / K + 1 塊的第 (i - 1) % K 個



中國人插隊問題

- 如果有一個可以 O(1) 從頭尾兩端刪除或增加的資料結構來維護每一塊的話
  (內部的查找、刪除都是與儲存的大小成線性)
  - 插入:該塊內部 O(K)、之後每塊都要修一下 O(N / K)
  - 刪除:該塊內部 O(K)、之後每塊都要修一下 O(N / K)
  - 詢問:該塊內部 O(K)
- 找到第 i 塊?
  - 把一堆這種資料結構開成陣列:O(1)
  - 把一堆這種資料結構串起來(linked list):O(N / K)
- 總之全部合起來就是 O(K + N / K) => 又是根號了!

中國人插隊問題

- **如果**有一個可以 O(1) 從頭尾兩端刪除或增加的資料結構來維護每一塊的話
（內部的查找、刪除都是與儲存的大小成線性）
- 方法很多，例如 雙向鏈結串列(Doubly linked list)，或者是STL的std::deque
- 這邊的實做就交給大家囉～可以試試看前面提到用 /K 知道位於哪一塊的作法～

第 Z 大

- 你有一個容器，一開始是空的
- 題目有三種操作，操作總數量是 Q：
  - 加值：把 y 個數字 x 丟到容器裡面
  - 減值：把 y 個數字 x 從容器中丟掉
  - 查詢：詢問這個容器的第 Z 大的元素

- $1 \le Q, x \le 10^5$， $y \le 10^9$ ，Z保證合法

第 Z 大

- 當然，這題可以用平衡樹、線段樹輕鬆過
- 不過老天爺還是希望你可以使用分「塊」
- 不過是什麼「塊」呢？

第 Z 大

- 對題目的「值域」分塊
- 對這題來講，就是對數字 x 去分塊
- 假設把值域 K 個做一塊，並且維護好每一塊的總和
  - 也就是維護每一塊中，每一個數字出現過得次數和
- 加值、減值操作：
  - O(1) 更新那個數字出現的次數和，以及相對應塊的總和
- 查詢操作：
  - 先花 O(C/K) 的時間，找到相對應的數字在哪一塊
    - C代表值域，本題來講C = 10^5
  - 再花 O(K) 的時間，確認那個數字是哪一個數字
- 總複雜度為O(Q(C/K + K))，取 K = sqrt(C)

小咲的玩具

- NPSC 2019 國中組初賽 pB
- 你有 K 個 vector<int> v[K]，vector 裡面總共會有 N 的元素
- Q 筆詢問，每筆詢問給兩個 vector<int> 的 index a, b，請求出

```
long long ans=0;
for (int i:v[a]) for (int j:v[b]) ans += min(i,j)
```

小咲的玩具

- 不是序列題，要怎麼分塊QAQ
- 搞不好跟影片中的第一個應用題有點像（?）

小咲的玩具

- 先不管這個，現在假設詢問的 vector<int> 的 size 分別是 x, y，並且 x < y。
- 有沒有什麼方法可以得到答案(？)
- 方法一： O(xy) 暴力枚舉
- 方法二： O(x + y) ，使用 雙指針(two pointer)
  方法二不知道的話沒關係^^
- 方法三： O(x log y)
  先維護好前綴和(prefix sum)，之後對於 x 的每個元素，在 y 二分搜出哪個位置比他小

小咲的玩具

- 方法三: O(x log y)
  先維護好前綴和(prefix sum), 之後對於 x 的每個元素 z ,
  在 y 二分搜出哪個位置比他小
- 比 z 小的元素, 題目所求就是那些元素的總和
- 比 z 大的元素, 題目所求就是 z * 比他大的元素的個數

```
for (int i:v[x]) {
    int pos = lower_bound(v[y].begin(),v[y].end(),i) - v[y].begin();
    --pos;
    if (pos < 0) ans += i*1ll*SZ(v[y]);
    else ans += pre[y][pos] + i*1ll*(SZ(v[y])-pos-1);
}
```

小咲的玩具

- 把那先 vector<int> 分成兩類
  - 胖 vector ： size > K
  - 瘦 vector ： size <= K

小咲的玩具

- 把那先 vector<int> 分成兩類
    - 胖 vector ： size > K
    - 瘦 vector ： size <= K
- 先預處理詢問是兩個胖 vector 的所有答案

小咲的玩具

- 把那先 vector<int> 分成兩類
  - 胖 vector ： size > K
  - 瘦 vector ： size <= K
- 先預處理詢問是兩個胖 vector 的所有答案
  - 複雜度為: $O(\ (N/K)^2\ K \log N\ ) = O(N^2/K \log N)$

小咲的玩具

- 把那先 vector<int> 分成兩類
  - 胖 vector ： size > K
  - 瘦 vector ： size <= K
- 先預處理詢問是兩個胖 vector 的所有答案
  - 複雜度為: O((N/K)^2 K log N) = O(N^2/K log N)
- 對於其他的詢問(胖&瘦 或者 瘦&瘦), 就直接算答案

小咲的玩具

- 把那先 vector<int> 分成兩類
  - 胖 vector ： size > K
  - 瘦 vector ： size <= K
- 先預處理詢問是兩個胖 vector 的所有答案
  - 複雜度為: O((N/K)^2 K log N) = O(N^2/K log N)
- 對於其他的詢問(胖&瘦 或者 瘦&瘦), 就直接算答案
  - 複雜度為: O(N K log N)

小咲的玩具

- 把那先 vector<int> 分成兩類
  - 胖 vector ： size > K
  - 瘦 vector ： size <= K
- 先預處理詢問是兩個胖 vector 的所有答案
  - 複雜度為: O((N/K)^2 K log N) = O(N^2/K log N)
- 對於其他的詢問(胖&瘦 或者 瘦&瘦),就直接算答案
  - 複雜度為: O(N K log N)
- 總複雜度為: O(N^2/K log N + NK log N)

小咲的玩具

- 把那先 vector<int> 分成兩類
  - 胖 vector ： size > K
  - 瘦 vector ： size <= K
- 先預處理詢問是兩個胖 vector 的所有答案
  - 複雜度為: O((N/K)^2 K log N) = O(N^2/K log N)
- 對於其他的詢問(胖&瘦 或者 瘦&瘦), 就直接算答案
  - 複雜度為: O(N K log N)
- 總複雜度為: O(N^2/K log N + NK log N)
- 取 K = sqrt(N) , 可得複雜度為 O(N^1.5 log N)

小咲的玩具

- 如果跟 counting triangles 一樣，每次拿 size 比較小的 vector，去對 size 比較大的 vector 做二分搜
- 複雜度會是好的嗎？
- 如果不是好的話，瓶頸在哪邊？要如何改進？

操作分塊

- 顧名思義，就是把「操作」分成一塊一塊的
- 直接來看一題例題吧～

操作分塊

- 給你一張 N 個點 M 條邊的無向圖，邊有邊權
- 接下來，有 Q 筆操作需要做處理，操作的內容分別如下：
- 1. 修改：把圖上某一條邊的邊權做修改
- 2. 詢問：給你（s, v），請你輸出從 s 出發，在只經過邊權 >= v 的邊的情況下，s 可以到達的點的數量。

- N <= 50000, M, Q <= 100000

操作分塊

- 給你一張 N 個點 M 條邊的無向圖，邊有邊權
- 接下來，有 Q 筆操作需要做處理，操作的內容分別如下：
- 1. 修改：把圖上某一條邊的邊權做修改
- 2. 詢問：給你（s, v），請你輸出從 s 出發，在只經過邊權 >= v 的邊的情況下，s 可以到達的點的數量。

- N <= 50000, M, Q <= 100000

- 簡單版：沒有修改，詢問的 v 非嚴格遞減

操作分塊

- 題目限制：沒有修改，詢問的 v 非嚴格遞減
- 考慮使用並查集
- 把邊從大到小排序
- 詢問 v 時，把 >= v 的邊通通 union 起來
- 複雜度 O(M log M + M alpha(M))

操作分塊

- 既然我們現在在講操作分塊，不如我們來試著把操作 K 個切成一塊吧
- 可以把圖上的邊分成兩個部份
  - 1. 在這 K 個操作中，有變動的邊
  - 2. 在這 K 個操作中，沒有變動的邊
- 第一種邊最多只有 K 條
- 想想看有沒有什麼好性質發生 ^^

操作分塊

- 可以把圖上的邊分成兩個部份
  - 1. 在這 K 個操作中，有變動的邊（這種邊至多 K 個）
  - 2. 在這 K 個操作中，沒有變動的邊

- 針對第二種邊，可以考慮剛剛簡化版的作法
  - 離線排序後，每次加邊加到詢問的 v 為止
  - 每 K 個操作，要花 O(M log M) 的時間
- 針對第一種邊，因為至多只有 K 條，在處理玩第二種邊後，暴力花 O(K) 的時間做 Union 就行了

- 複雜度呢？

操作分塊

- 可以把圖上的邊分成兩個部份
  - 1. 在這 K 個操作中，有變動的邊（這種邊至多 K 個）
  - 2. 在這 K 個操作中，沒有變動的邊
- 針對第二種邊，可以考慮剛剛簡化版的作法
  - 離線排序後，每次加邊加到詢問的 v 為止
  - 每 K 個操作，要花 O(M log M) 的時間
- 針對第一種邊，因為至多只有 K 條，在處理完第二種邊後，暴力花 O(K) 的時間做 Union 就行了
- O((Q / K) M log M + QK)，取 K = sqrt(Q)，即可得到 O(sqrt(Q) * (M log M + K))

更多根號算法

- 以下是本次課堂沒有涉及的一些算法:
  - 數論上的分塊
  - 莫隊演算法, 以及各種噁心變形
  - ....
- 投影片最後面有 2019 年當時的莫隊演算法講解, 有興趣的學員歡迎參考～

Mo’s algorithm

https://zerojudge.tw/ShowProblem?problemid=b417

- 先來看一道經典問題:「區間 種樹(X) 眾數(O)」
- 給你長度為 N 的序列，並且有 Q 筆詢問，每筆詢問的內容是詢問 [L, R] 出現最多次數的數字出現的個數。

- N, Q <= 10^5

Mo’s algorithm

- 發明者：中國隊隊長莫濤
- 莫濤在解決「小Z的襪子」這題是，想到的演算法 https://www.lydsy.com/JudgeOnline/problem.php?id=2038
- 於是乎，這個算法就被稱為「莫隊算法」，簡稱「莫隊」、「Mo's algo」

Mo’s algorithm

- 莫隊算法(Mo's algorithm)是一個離線的算法
- 先把詢問「按照某種順序」排序過後，開始暴力硬算

- 題目的詢問從 [L, R] 轉移到 [L, R+1 or R-1] 時，必須在一個很短的時間內轉移。Ex: O(1) or O(log N)

- 題目的詢問從 [L, R] 轉移到 [L+1 or L-1, R] 時，必須在一個很短的時間內轉移。Ex: O(1) or O(log N)

Mo’s algorithm

- 假設上面那張投影片的所有轉移時間都是 O(1)
- 那在「經過某種順序排序」後的詢問之下，時間複雜度就是 O(L指針移動的次數) + O(R指針移動的次數) + O(Q * 得到答案的複雜度)

Mo’s algorithm

- 假設上面那張投影片的所有轉移時間都是 O(1)
- 那在「經過某種順序排序」後的詢問之下，時間複雜度就是 O(L指針移動的次數) + O(R指針移動的次數) + O(Q * 得到答案的複雜度)
- 莫隊算法就是擬定一個好的排序方法，使得 O(L指針移動的次數) = O(R指針移動的次數) = O(N sqrt(N))

Mo’s algorithm

- 先把序列每 K 個分成一塊。

Mo’s algorithm

- 先把序列每 K 個分成一塊。
- 排序詢問的方法為:
  - 先按照 L 所在的塊排序
  - 如果 L 所在的塊相同，則按照 R 排序

Mo’s algorithm

- 先把序列每 K 個分成一塊。
- 排序詢問的方法為:
  - 先按照 L 所在的塊排序
  - 如果 L 所在的塊相同，則按照 R 排序
- 認真分析一下複雜度:
  - 對於 L 指針:
  - 對於 R 指針:

Mo’s algorithm

- 先把序列每 K 個分成一塊。
- 排序詢問的方法為:
  - 先按照 L 所在的塊排序
  - 如果 L 所在的塊相同，則按照 R 排序
- 認真分析一下複雜度:
  - 對於 L 指針:
    O(N * K + N)
  - 對於 R 指針:
    O(N/K * N + N)

Mo’s algorithm

- 先把序列每 K 個分成一塊。
- 排序詢問的方法為:
  - 先按照 L 所在的塊排序
  - 如果 L 所在的塊相同，則按照 R 排序
- 認真分析一下複雜度:
  - 對於 L 指針:
    O(N * K + N)
  - 對於 R 指針:
    O(N/K * N)
- 取 K = sqrt(N) ，大勝利～
- 於是乎，大家就得到一個 O(N sqrt (N))的區間眾數的解法

Mo's algorithm

- 參考實做方式:詢問的排序

```
struct Query {
  int L, R, block, i;
  bool operator<(const Query& q) const {
    return block == q.block ? R < q.R : block < q.block;
  }
} query[maxm];
```

Mo’s algorithm

- 參考實做方式:指針的移動方式

```
for (int i = 0, L = 1, R = 0; i < M; ++i) {
  while (R < query[i].R) add(++R);
  while (R > query[i].R) sub(R--);
  while (L > query[i].L) add(--L);
  while (L < query[i].L) sub(L++);
  ans[query[i].i] = make_pair(most, freq[most]);
}
```

Mo’s algorithm

- 參考實做方式：指針移動的加值/減值

```
void add(int bound) {
  freq[cnt[S[bound]]]--;
  cnt[S[bound]]++;
  freq[cnt[S[bound]]]++;
  while (freq[most + 1]) most++;
}

void sub(int bound) {
  freq[cnt[S[bound]]]--;
  cnt[S[bound]]--;
  freq[cnt[S[bound]]]++;
  while (!freq[most]) most--;
}
```

Mo’s algorithm

- 進階版:帶修改莫隊
- 複雜度是 O(N^(5/3))
- 就留給大家上網查資料囉

Mo’s algorithm

- 許多序列操作題，如果一直在跟 O(N log N)，O(N (log N)^2) 奮鬥的話
- 不如可以考慮寫寫莫隊，搞不好有更優秀的解答～

題目

[SRM 675 Div.1 Middle]

- 給你 n, x0, a, b，表示有一個長度 n 的數列：
  - x[0] = x0
  - for (int i = 1; i < n; i++)
    - x[i] = (x[i - 1] * a + b) % (10^9 + 7)
- 你要回答 Q 個詢問，每個詢問給你一個 k (0 <= k < n)，請你回答數列 x 中第 k 小的數字(k=0 為最小)
- n <= 5 * 10^6
- Q <= 100
- 時限 10sec、額外儲存空間限制 1MB

題目

- 給你長度為 n 的序列 a_1, a_2, ..., a_n，以及 q 個詢問
- 每筆詢問給你 L_1, R_1, L_2, R_2，請你算出符合底下限制的 (i, j) 數量
  - L_1 <= i <= R_1
  - L_2 <= j <= R_2
  - a_i == a_j
- n, q, a_i <= 10^5