Segment Tree

線段樹

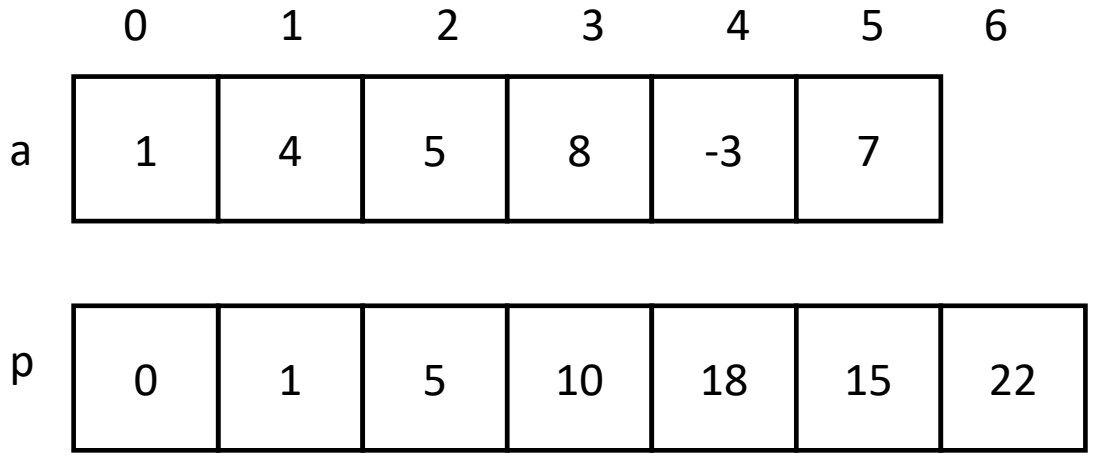
情境 - 大量查詢

- 給你 N（N < 1e5）個數字 a[0] ~ a[N – 1]，和 q 個詢問（q < 3e5）
  - 每次詢問有兩個數字 l、r，輸出 a[l] + a[l + 1] + a[l + 2] + … + a[r]

- 你該怎麼處理？

情境 - 大量查詢

- Naive Solution：每次都計算，最差複雜度 O(Nq)
- Better Solution：維護前綴和
  - 計算從 a[1] + a[2] + a[3] + a[4] + a[5] = p[6] – p[1] = 22 – 1 = 21
  - 複雜度 = 製作前綴和 + 每次查詢O(1) = O(N + q)



情境 - 大量區間加值

- 給你 N（N < 1e5）個數字 a[0] ~ a[N – 1]，和 q 個修改（q < 3e5）
  - 每次修改有三個數 l、r、v，請將 a[l] += v；a[l+1] += v；……；a[r] += v；
- 之後會再輸入一個數字 p，代表 p 次詢問（p < 3e5）
  - 每次詢問有兩個數字 l、r，輸出 a[l] + a[l + 1] + a[l + 2] + … + a[r]

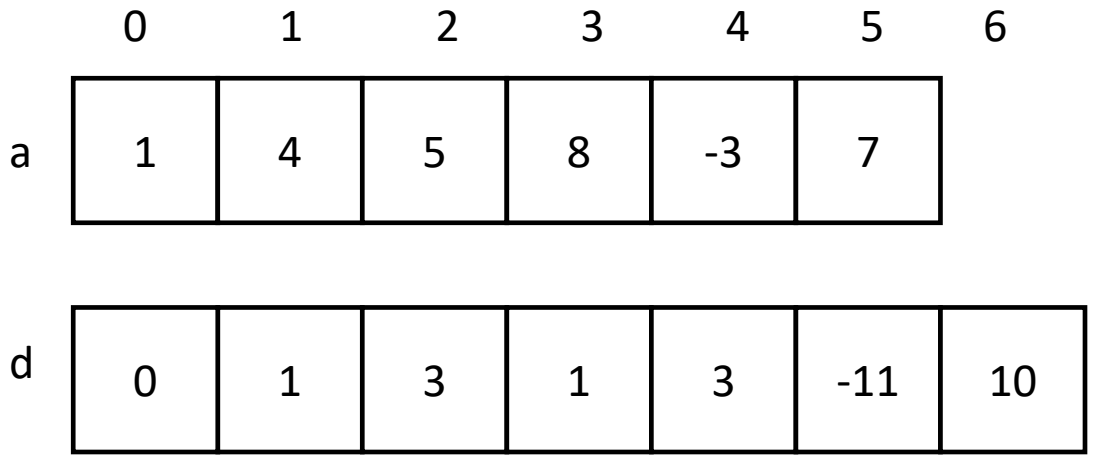
- 你該怎麼處理？

情境 - 大量區間加值

- 每次都計算區間加值，最後把陣列統計出來再計算前綴和？
- 複雜度：每次區間加值 = O(N) * q；處理詢問 O(N + p)
- 總複雜度 = O(Nq + p)
  - 很爛

情境 - 大量區間加值

- 更好的解：差分數列
  - 我們會注意到差分數列的前綴和是原數列
  - 區間 [l, r] 加值 v 等價於讓 d[l] 加上 v，再讓 d[r + 1] 扣掉 v
  - 之後再計算前綴和就會發現該數列 [l, r] 都被加上了 v。



複雜度分析：前綴序列、差分序列

- 前綴序列允許我們 O(N) 更新、O(1) 查詢區間和
- 差分序列允許我們 O(1) 更新、O(N) 查詢區間、O(N) 查詢元素

- 所以我們可以透過前綴和差分處理大量的分開查詢、修改操作

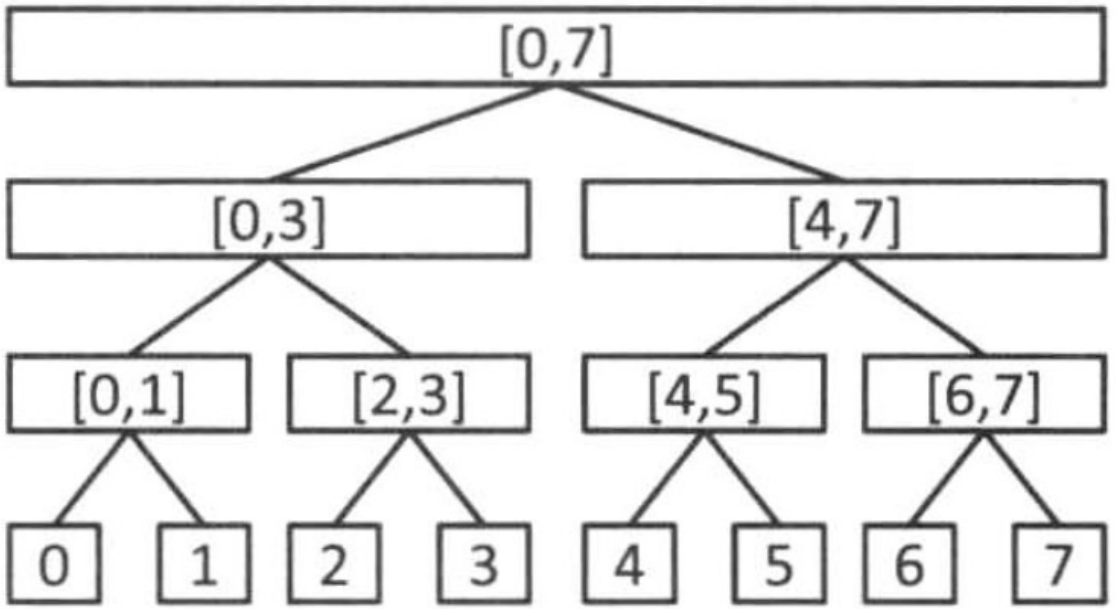
情境 - 大量修改+大量查詢

- 給你 N（N < 1e5）個數字 a[0] ~ a[N – 1]，和 q 個操作（q < 3e5）
- 有兩種操作
  - 修改：給定 i、v，請將 a[i] += v
  - 查詢：給定 l、r，輸出 a[l] + a[l + 1] + a[l + 2] + … + a[r]

- 你該怎麼處理？
  - 顯然沒辦法只靠前綴和差分的技巧處理

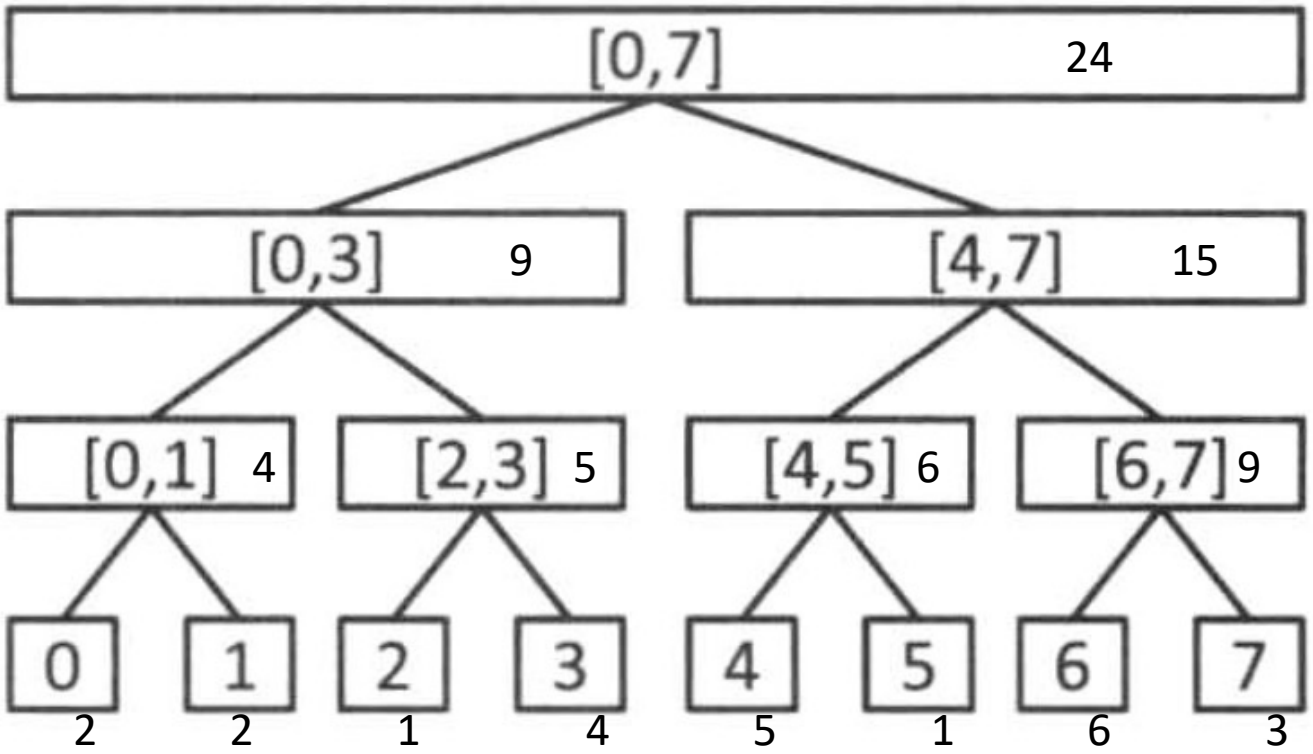
線段樹

- 我們可以用一種特殊的資料結構，叫做線段樹
  - 每一個節點代表一個區間的答案
    - 父節點會被拆為連續的兩段區間
    - 概念圖如右邊



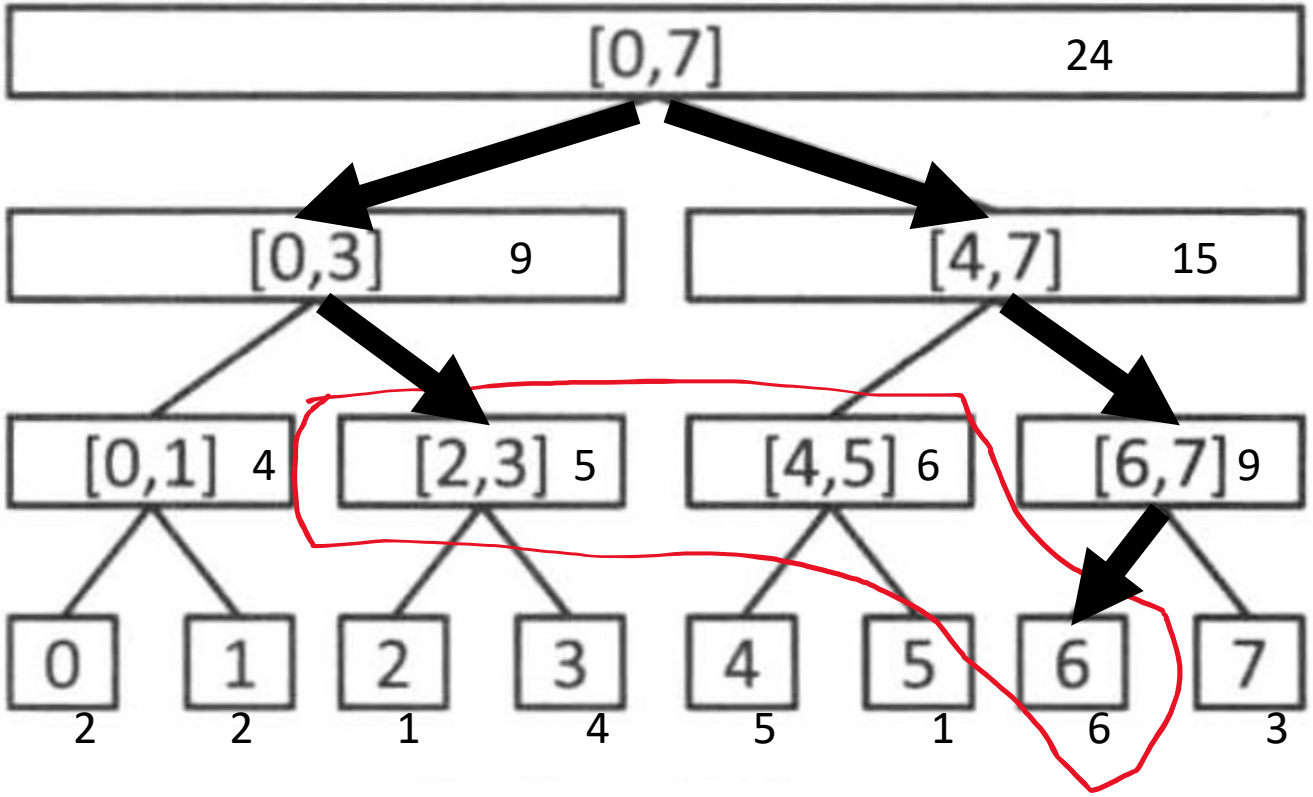
線段樹

- 假設 a = [2, 2, 1, 4, 5, 7, 6, 3]



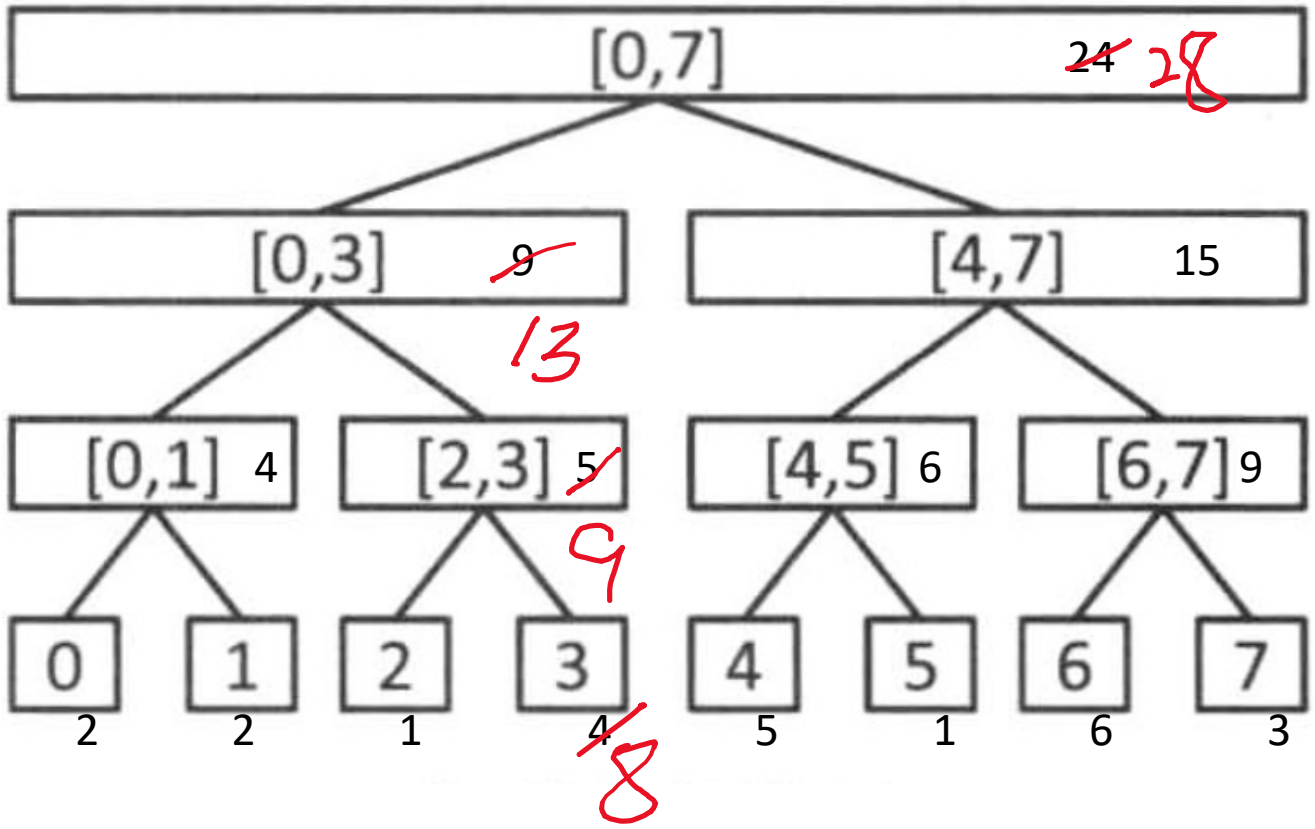
線段樹

- 假設我要查詢 [2, 6] 的區間和，答案就是 5 + 6 + 6



線段樹

- 假設我要更新 a[3] 成 8

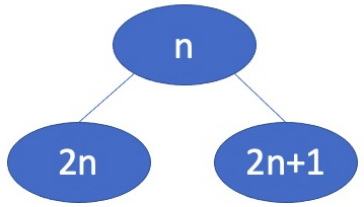


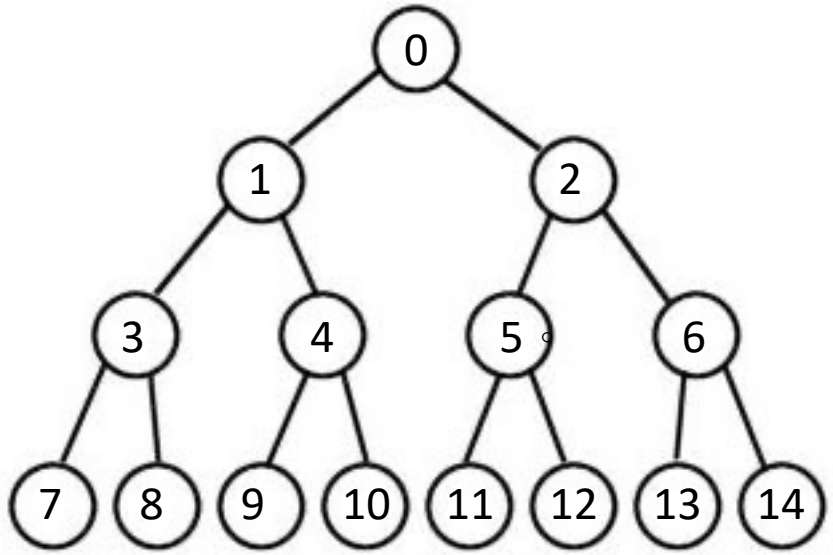
線段樹 – 複雜度分析

- 假設我要修改某個元素
  - 那會修改從那個元素到 root 總共 logN 個點
  - 因此複雜度是 O(logN)
- 假設我要查詢某個區間
  - 我會從 root 開始找，每次要嘛往左，要嘛往右，要嘛同時往下
    - 可以思考一下，如果同時往下，那往左的之後必然向右
    - 往右的之後必然往左，所以最多分岔成 2 條路
  - 複雜度是 O(2logN) = O(logN)

線段樹 – 實作說明

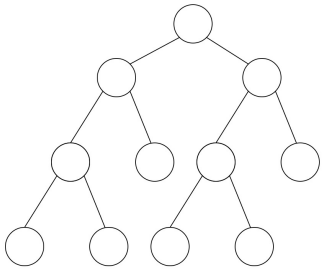
- 我們要先介紹用一個一維陣列來儲存一個二元樹的方式
  - 如果我們按照順序編號一顆滿二元樹的點（如右圖），我們會發現：
    - X 號點的左孩子是 2*X + 1，右孩子是 2*X + 2（如左圖）
  - 於是我們會開一個陣列，來儲存每個點的值

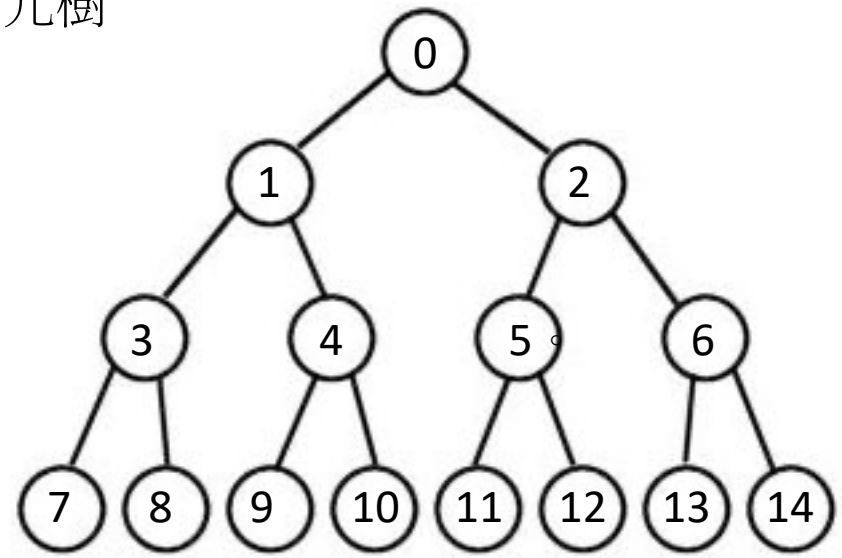




線段樹 – 實作說明

- 右下角的圖是由一個長度為 8（N=8） 的陣列所構成的線段樹
  - 我們可以知道實際上最多會有不超過 2*N – 1 個點
    - 最多 logN 層，2^0 + 2^1 + 2^2 + … + 2^(logN) = 2*N – 1（方便計算假設 N 是 2 的冪次
  - 那我們線段樹陣列能不能只開到 2 * N？
    - 不行，因為樹雖然很平衡，但並不是很完美的滿二元樹
      - 因為 N 不總是 2 的冪次，因此可能會多一點點出來
      - 比如 N = 6（左下圖），實際上會用到第 13 個點
      - 因此其實 4 倍是最保險的
    - 因此我們會開 4 * N 的陣列來儲存線段樹





線段樹 – 實作說明

- 我們會用一個陣列 int val[4 * N] 來儲存我們的線段樹本體
  - val[idx] 是編號 idx 節點所儲存的答案

- 我們會用兩個 function 來維護我們的線段樹本體
  - `void add(int idx, int l, int r, int i, int v)`
    - 當前節點編號 idx，負責 [l, r] 區間，希望為陣列中第 i 項加上 v
  - `int query(int idx, int l, int r, int i, int j)`
    - 當前節點編號 idx，負責 [l, r] 區間，想要查詢
    - 陣列 [i..j] 的總和

線段樹 – 模板題（https://loj.ac/p/130）

给定数列 $a_1, a_2, \ldots, a_n$，你需要依次进行 q 个操作，操作有两类：

- `1 i x`：给定 i, x，将 a_i 加上 x；
- `2 l r`：给定 l, r，求 $\sum_{i=l}^{r} a_i$ 的值（换言之，求 $a_l + a_{l+1} + \cdots + a_r$ 的值）。

输入格式

第一行包含 2 个正整数 n, q，表示数列长度和询问个数。保证 $1 \leq n, q \leq 10^6$。
第二行 n 个整数 $a_1, a_2, \ldots, a_n$，表示初始数列。保证 $|a_i| \leq 10^6$。
接下来 q 行，每行一个操作，为以下两种之一：

- `1 i x`：给定 i, x，将 $a[i]$ 加上 x；
- `2 l r`：给定 l, r，求 $\sum_{i=l}^{r} a_i$ 的值。

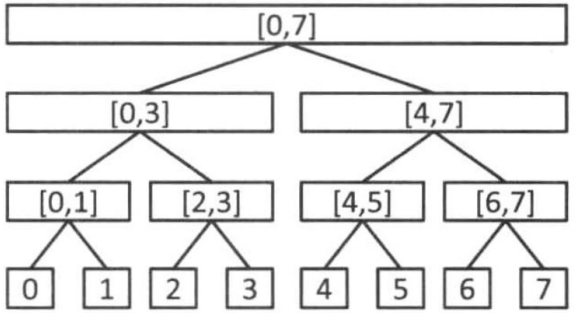
保证 $1 \leq l \leq r \leq n, |x| \leq 10^6$。

输出格式

对于每个 `2 l r` 操作输出一行，每行有一个整数，表示所求的结果。

線段樹 – 模板題（https://loj.ac/p/130）

```
 5 const int MAX = 1e6 + 5;
 6 int val[4 * MAX];
 7
 8 void add(int idx, int l, int r, int i, int v){
 9     if(l == r){ // we reach the leaf we should modify
10         val[idx] += v;
11         return;
12     }
13     int mid = (l + r) / 2;
14     if(i <= mid) add(2 * idx + 1, l, mid, i, v);
15     else add(2 * idx + 2, mid + 1, r, i, v);
16     val[idx] = val[2 * idx + 1] + val[2 * idx + 2];
17 }
```

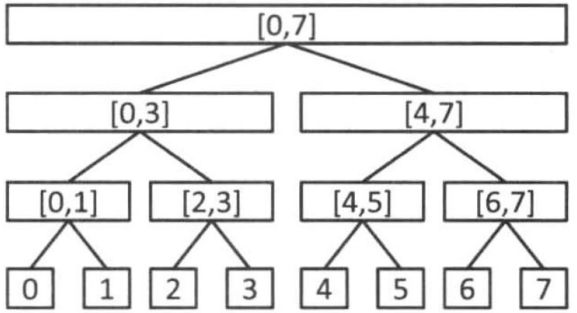


線段樹 – 模板題（https://loj.ac/p/130）

```
19 int query(int idx, int l, int r, int i, int j){
20     if(i <= l and r <= j) return val[idx];
21     if(r < i or j < l) return 0;
22     int mid = (l + r) / 2;
23     return query(idx * 2 + 1, l, mid, i, j) + query(idx * 2 + 2, mid + 1, r, i, j);
24 }
```

當 [l, r] 在 [i, j] 裡時（即 i <= l 且 r <= j），我們當前這個節點所負責的區間被覆蓋
- 我們就可以直接使用 val[idx] 來表示這個區間的答案

當 [l, r] 與 [i, j] 不重疊時候（可能 [l, r] < [i, j] 即 r < i 或 [i, j] < [l, r] 即 j < l）
- 我們就不能計算這個區間的答案，必須回傳一個不影響答案的數值
  - 在計算總和時，0 不影響答案
  - 在計算乘積時，1 不影響答案
  - 在計算 gcd 時， 0 不影響答案
  - 在計算最大值時，負無限大不影響答案
  - ……



線段樹 – 模板題（https://loj.ac/p/130）

```
26 int32_t main(){
27     int n, q; cin >> n >> q;
28     for(int i = 0; i < n; i++){
29         int x; cin >> x;
30         add(0, 0, n - 1, i, x);
31     }
32     while(q--){
33         int opr, x, y; cin >> opr >> x >> y;
34         if(opr == 1) add(0, 0, n - 1, x - 1, y);
35         else cout << query(0, 0, n - 1, x - 1, y - 1) << endl;
36     }
37     return 0;
38 }
```

線段樹 – 模板題（https://loj.ac/p/130）

```cpp
#include<bits/stdc++.h>
#define int long long int
using namespace std;

const int MAX = 1e6 + 5;
int val[4 * MAX];

void add(int idx, int l, int r, int i, int v){
    if(l == r){ // we reach the leaf we should modify
        val[idx] += v;
        return;
    }
    int mid = (l + r) / 2;
    if(i <= mid) add(2 * idx + 1, l, mid, i, v);
    else add(2 * idx + 2, mid + 1, r, i, v);
    val[idx] = val[2 * idx + 1] + val[2 * idx + 2];
}

int query(int idx, int l, int r, int i, int j){
    if(i <= l and r <= j) return val[idx];
    if(r < i or j < l) return 0;
    int mid = (l + r) / 2;
    return query(idx * 2 + 1, l, mid, i, j) + query(idx * 2 + 2, mid + 1, r, i, j);
}

int32_t main(){
    int n, q; cin >> n >> q;
    for(int i = 0; i < n; i++){
        int x; cin >> x;
        add(0, 0, n - 1, i, x);
    }
    while(q--){
        int opr, x, y; cin >> opr >> x >> y;
        if(opr == 1) add(0, 0, n - 1, x - 1, y);
        else cout << query(0, 0, n - 1, x - 1, y - 1) << endl;
    }
    return 0;
}

```

線段樹 – 模板題（https://zerojudge.tw/ShowProblem?problemid=d539）

內容

給一個數列T1,T2,T3....Tn ，求 Ta到Tb之間(涵蓋 Ta、Tb)的最大值。

輸入說明

每組測資輸入的第一行有一個整數N (1≦N≦50,0000) ，接下來會有N個正整數(int 範圍內)代表T1,T2,T3....Tn

再接下來會有一個整數M(1≦M≦N)，代表接下來會有M行詢問的a,b。(a, b 沒說誰大誰小 !)

輸出說明

輸出T[a,b]之間的MAX。

測資資訊：

記憶體限制： 512 MB
公開 測資點#0 (10%): 1.0s , <1K
公開 測資點#1 (10%): 1.0s , <1M
公開 測資點#2 (20%): 1.0s , <1M
公開 測資點#3 (20%): 1.0s , <1M
公開 測資點#4 (20%): 1.0s , <10M
公開 測資點#5 (20%): 1.0s , <10M

範例輸入 #1

```
10
3 2 4 5 6 8 1 2 9 7
7
1 5
3 5
1 10
5 8
6 6
2 4
2 9
```

範例輸出 #1

```
6
6
9
8
8
5
9
```

線段樹 – 模板題（https://zerojudge.tw/ShowProblem?problemid=d539）

```cpp
#include<bits/stdc++.h>
using namespace std;

const int MAX = 5e5 + 5;

int val[4 * MAX];

void setv(int idx, int l, int r, int i, int v){
    if(l == r){ // we reach the leaf we should modify
        val[idx] = v;
        return;
    }
    int mid = (l + r) / 2;
    if(i <= mid) setv(2 * idx + 1, l, mid, i, v);
    else setv(2 * idx + 2, mid + 1, r, i, v);
    val[idx] = max(val[2 * idx + 1], val[2 * idx + 2]);
}

int query(int idx, int l, int r, int i, int j){
    if(i <= l and r <= j) return val[idx];
    if(r < i or j < l) return -1; // input numbers are always positive
    int mid = (l + r) / 2;
    return max(query(idx * 2 + 1, l, mid, i, j), query(idx * 2 + 2, mid + 1, r, i, j));
}

int32_t main(){
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    int n; cin >> n;
    for(int i = 0; i < n; i++){
        int x; cin >> x;
        setv(0, 0, n - 1, i, x);
    }
    int m; cin >> m;
    while(m--){
        int x, y; cin >> x >> y;
        if(x > y) swap(x, y);
        cout << query(0, 0, n - 1, x - 1, y - 1) << "\n";
    }
    return 0;
}
```

同學們要注意：
這題輸入輸出量比較大，要記得做 IO 優化

1. ios::sync_with_stdio(false) 來關掉同步
2. cin.tie(0) 和 cout.tie(0) 來解除綁定
3. 不要用 endl，改用 “\n” 防止 flush

關於 IO 優化的那些事，請參考 zswu 的貼文（https://shorturl.at/lmGKT）

線段樹 – 比較難的模板題

給定一個陣列 a，長度 N < 1e5，給 q 個查詢 q < 3e5
每次查詢會輸入兩個數字 l, r，試求 gcd(a[l], a[l + 1], ..., a[r])

提示：差分、輾轉相除法、線段樹

線段樹 – 進階版（區間修改線段樹）

- 給你 N（N < 1e5）個數字 a[0] ~ a[N – 1]，和 q 個操作（q < 3e5）
- 有兩種操作
  - 修改：給定 l、r、v，請將 a[l] += v；a[l + 1] += v；……；a[r] += v
  - 查詢：給定 l、r，輸出 a[l] + a[l + 1] + a[l + 2] + … + a[r]

- 你該怎麼處理？
  - 顯然沒辦法只靠普通的線段樹
  - 需要懶標（lazy tag）的技巧
  - We do not discuss this today
  - 如果想知道作法，可以參考 https://cp.wiwiho.me/segment-tree/

線段樹 – 應用題（https://zerojudge.tw/ShowProblem?problemid=f315）

內容

輸入一個長度為 $2n$ 的陣列，其中 1 ~ n 的每個數字都剛好各 2 次。

i 的低窪值的定義是兩個數值為 i 的位置中間，有幾個小於 i 的的數字。

以 $[3, 1, 2, 1, 3, 2] 為例，1的低窪值為 0, 2 的低窪值值為 1, 3 的低窪值為 3。

請對於每個 1 ~ n 的數字都求其低窪值（兩個相同的數字之間有幾個數字比它小），輸出低窪值的總和，答案可能會超過 C++ int 的上限。

輸入說明

第一行有一個正整數 n

第二行有 $2n$ 個正整數，以空格分隔，保證 1 n 每個數字都恰好出現兩次。

配分

- 20%: $1 \leq n \leq 1000$
- 40%: $1 \leq n \leq 40000$
- 40%: $1 \leq n \leq 100000$

輸出說明

輸出 1 ~ n 每個數字的低窪值總和。

測資資訊：

記憶體限制： 250 MB
公開 測資點#0 (10%): 1.0s , <1M
公開 測資點#1 (10%): 1.0s , <1M
公開 測資點#2 (10%): 1.0s , <1M
公開 測資點#3 (10%): 1.0s , <1M
公開 測資點#4 (10%): 1.0s , <1M
公開 測資點#5 (10%): 1.0s , <1M
公開 測資點#6 (10%): 1.0s , <10M
公開 測資點#7 (10%): 1.0s , <10M
公開 測資點#8 (10%): 1.0s , <10M
公開 測資點#9 (10%): 1.0s , <10M

範例輸入 #1

```
3
3 1 2 1 3 2
```

範例輸出 #1

```
4
```

線段樹 –應用題（https://zerojudge.tw/ShowProblem?problemid=f315）

- 想法：
  - 紀錄每個數字的第一次出現和第二次出現
  - 從數字 1 開始
    - 把該數字在陣列上出現的位置設為 1
    - 計算該數字第一次出現和第二次出現的兩個位置之間，目前有多少 1（即中間有多少比他小的數字）
    - 加總到答案上

線段樹 –應用題（https://zerojudge.tw/ShowProblem?problemid=f315）

```cpp
#include<bits/stdc++.h>
#define int long long int

using namespace std;

const int MAX = 1e5 + 5;

int val[4 * 2 * MAX], seen[MAX], first[MAX], last[MAX];

void setv(int idx, int l, int r, int i, int v){
    if(l == r){
        val[idx] = v;
        return;
    }
    int mid = (l + r) / 2;
    if(i <= mid) setv(2 * idx + 1, l, mid, i, v);
    else setv(2 * idx + 2, mid + 1, r, i, v);
    val[idx] = val[2 * idx + 1] + val[2 * idx + 2];
}

int query(int idx, int l, int r, int i, int j){
    if(i <= l and r <= j) return val[idx];
    if(r < i or j < l) return 0;
    int mid = (l + r) / 2;
    return query(idx * 2 + 1, l, mid, i, j) + query(idx * 2 + 2, mid + 1, r, i, j);
}

int32_t main(){
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    int n; cin >> n;
    int ans = 0;
    for(int i = 0; i < 2 * n; i++){
        int x; cin >> x;
        if(seen[x] == 0) first[x] = i, seen[x] = 1;
        else last[x] = i;
    }
    for(int i = 1; i <= n; i++){
        ans += query(0, 0, 2 * n, first[i], last[i]);
        setv(0, 0, 2 * n, first[i], 1);
        setv(0, 0, 2 * n, last[i], 1);
    }
    cout << ans << endl;

    return 0;
}
```

注意：答案要開 long long

謝謝各位