搜尋 Searching

陳俊安 Colten

2022 新化高中 x 嘉義高中 x 薇閣高中 資研社暑期培訓營隊

2022.07.04

這堂課你會學到什麼

- 建表（預處理）Preprocessing
- 雙指針 Two pointers
- 滑動窗口 Sliding Window
- 二分搜尋 Binary Search
- 折半枚舉 Meet in the Middle

建表（預處理）Preprocessing

建表（預處理）Preprocessing

- 預先處理好一個表
- 當要使用到某個資訊時，直接拿之前建的表的資料，就不用一直重複計算的技巧
- 有許多技巧也是以此為概念，像是倍增法

Ten Point Round #11 (Div. 2) pC. Gunjyo 與骰子

Gunjyo 有三顆 100 面骰，每顆骰子的點數為 $1 \sim 100$，現在 Gunjyo 會同時骰三顆骰子，假設這三顆的點數分別為 a, b, c，那麼接下來她可以選擇將所有點數變成 $a+b+c, a+b\times c, a\times b+c, a\times b\times c$，有 Q $(1 \le Q \le 2\times 10^5)$ 組詢問，每一組詢問將查詢 Gunjyo 共有幾種可能會使最後的點數變成 N

建表（預處理）Preprocessing

Ten Point Round #11 (Div. 2) pC. Gunjyo 與骰子

Gunjyo 有三顆 100 面骰，每顆骰子的點數為 $1 \sim 100$，現在 Gunjyo 會同時骰三顆骰子，假設這三顆的點數分別為 a, b, c，那麼接下來她可以選擇將所有點數變成 $a+b+c, a+b \times c, a \times b+c, a \times b \times c$，有 Q $(1 \leq Q \leq 2 \times 10^5)$ 組詢問，每一組詢問將查詢 Gunjyo 共有幾種可能會使最後的點數變成 N

- 如果每一次查詢我們都去重算一次的話時間複雜度會是 $O(100^3 \times Q)$ 很明顯會 TLE

Ten Point Round #11 (Div. 2) pC. Gunjyo 與骰子

Gunjyo 有三顆 100 面骰，每顆骰子的點數為 $1 \sim 100$，現在 Gunjyo 會同時骰三顆骰子，假設這三顆的點數分別為 a, b, c，那麼接下來她可以選擇將所有點數變成 $a+b+c, a+b \times c, a \times b+c, a \times b \times c$，有 Q $(1 \leq Q \leq 2 \times 10^5)$ 組詢問，每一組詢問將查詢 Gunjyo 共有幾種可能會使最後的點數變成 N

- 如果每一次查詢我們都去重算一次的話時間複雜度會是 $O(100^3 \times Q)$ 很明顯會 TLE
- 但如果我們預先算好 $1 \sim N$ 的所有答案，時間複雜度會變成 $O(100^3 + Q)$
- 這就是建表強大的地方！

建表（預處理）Preprocessing

Ten Point Round #11 (Div. 2) pC. Gunjyo 與骰子

Gunjyo 有三顆 100 面骰，每顆骰子的點數為 $1 \sim 100$，現在 Gunjyo 會同時骰三顆骰子，假設這三顆的點數分別為 a, b, c，那麼接下來她可以選擇將所有點數變成 $a+b+c, a+b \times c, a \times b+c, a \times b \times c$，有 Q $(1 \leq Q \leq 2 \times 10^5)$ 組詢問，每一組詢問將查詢 Gunjyo 共有幾種可能會使最後的點數變成 N

- 要怎麼預先算好所有的答案?

Ten Point Round #11 (Div. 2) pC. Gunjyo 與骰子

Gunjyo 有三顆 100 面骰，每顆骰子的點數為 $1 \sim 100$，現在 Gunjyo 會同時骰三顆骰子，假設這三顆的點數分別為 a, b, c，那麼接下來她可以選擇將所有點數變成 $a+b+c, a+b\times c, a\times b+c, a\times b\times c$，有 Q $(1 \le Q \le 2\times 10^5)$ 組詢問，每一組詢問將查詢 Gunjyo 共有幾種可能會使最後的點數變成 N

- 要怎麼預先算好所有的答案?
- 開一個陣列，索引值 i 表示點數最後變成 i 的可能性有幾種
- 我們可以跑三層迴圈，每一個迴圈分別代表其中一種骰子
- 接著將每一次三顆骰子可以變成的四種結果都記錄起來，假設這四種結果的點數分別為 a, b, c, d，那就表示組合出 a, b, c, d 這四種點數的可能性多了一種

建表（預處理）Preprocessing

Ten Point Round #11 (Div. 2) pC. Gunjyo 與骰子

Gunjyo 有三顆 100 面骰，每顆骰子的點數為 $1 \sim 100$，現在 Gunjyo 會同時骰三顆骰子，假設這三顆的點數分別為 a, b, c，那麼接下來她可以選擇將所有點數變成 $a+b+c, a+b \times c, a \times b+c, a \times b \times c$，有 Q $(1 \leq Q \leq 2 \times 10^5)$ 組詢問，每一組詢問將查詢 Gunjyo 共有幾種可能會使最後的點數變成 N

- 要怎麼預先算好所有的答案?
- 開一個陣列，索引值 i 表示點數最後變成 i 的可能性有幾種
- 我們可以跑三層迴圈，每一個迴圈分別代表其中一種骰子
- 接著將每一次三顆骰子可以變成的四種結果都記錄起來，假設這四種結果的點數分別為 a, b, c, d，那就表示組合出 a, b, c, d 這四種點數的可能性多了一種
- 最後查詢的時候就拿我們存在陣列的答案來輸出就可以了

建表（預處理）Preprocessing

```
15      for(int i=1;i≤100;i++)
16      {
17          for(int k=1;k≤100;k++)
18          {
19              for(int j=1;j≤100;j++)
20              {
21                  check[i*k*j]++;
22                  check[i+k+j]++;
23                  check[i+k*j]++;
24                  check[i*k+j]++;
25              }
26          }
27      }
28
29      int q;
30      cin >> q;
31
32      while(q--)
33      {
34          int n;
35          cin >> n;
36
37          cout << check[n] << "\n";
38      }
```

練習題

Codeforces Round #739 (Div. 3) pA. Dislike of Threes

有 t 組查詢，每一組查詢你必須回答 ≥ 1 且不是 3 的倍數中，第 k $(1 \leq k \leq 1000)$ 小的數字是誰（備注： 這題暴力也會過，但請大家用建表練習）

CSES Problem Set Common Divisors

給你一個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列，你的任務是從這個序列中挑出兩個不同位置的數字，請問這兩個數字的最大公因數最大可以是多少

雙指針 Two Pointers

雙指針 Two Pointers

- 枚舉的一種
- 但是在某些情況下，讓我們可以省略掉許多不必要的枚舉
- 實作上真的就是兩個指針在跑來跑去

選數字

給你兩個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列 a, b，你要在這兩個序列中各選一個元素，請問共有幾種選法會使你選擇的那兩個數字的和大於 m

選數字

給你兩個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列 a, b，你要在這兩個序列中各選一個元素，請問共有幾種選法會使你選擇的那兩個數字的和大於 m

應該不難想到 $O(n^2)$ 的做法吧，但是其實這題可以在 $O(n + n \log n)$ 的時間做完

選數字

給你兩個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列 a, b，你要在這兩個序列中各選一個元素，請問共有幾種選法會使你選擇的那兩個數字的和大於 m

- 我們先將這兩個序列都排序
- 接下來我們去枚舉 a 最小的元素最少需要加上 b 的哪一個位置的數字

選數字

給你兩個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列 a, b，你要在這兩個序列中各選一個元素，請問共有幾種選法會使你選擇的那兩個數字的和大於 m

- 我們先將這兩個序列都排序
- 接下來我們去枚舉 a 最小的元素最少需要加上 b 的哪一個位置的數字
- 如果現在排序後 a_1 最少需要加上 b_3 才會大於 m，那麼接下來我們要算 a_2 最少需要加上誰才會大於 m 時，是不是可以省略一些枚舉的步驟呢 ...
- 給大家 2 分鐘思考一下

選數字

給你兩個長度為 n $(1 \le n \le 2 \times 10^5)$ 的序列 a, b，你要在這兩個序列中各選一個元素，請問共有幾種選法會使你選擇的那兩個數字的和大於 m

- 既然 $a_1 + b_3 > m$
- 那麼很顯然的 $a_2 + b_3 > m, a_2 + b_4 > m, a_2 + b_5 > m$
- 所以我們在做的時候就不用從頭開始枚舉，從上一個元素枚舉到的位置繼續往下就可以了！
- 由於每一個元素最多只會被指針掃過一次，複雜度為 $O(n + n \log n)$
- 這題也可以用二分搜來做，時間複雜度一樣

Sum of Two Values

給定一個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列，請你找到兩個不同的位置 i, j，滿足 $a_i + a_j = x$，如果找不到則輸出 IMPOSSIBLE

這題也可以用 map 或二分搜做掉

Sum of Two Values

給定一個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列，請你找到兩個不同的位置 i, j，滿足 $a_i + a_j = x$，如果找不到則輸出 IMPOSSIBLE

這題也可以用 map 或二分搜做掉

Sum of Two Values

給定一個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列，請你找到兩個不同的位置 i, j，滿足 $a_i + a_j = x$，如果找不到則輸出 IMPOSSIBLE

- 我們一樣用枚舉的方式來做
- 先將整個序列由小到大排序好

Sum of Two Values

給定一個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列，請你找到兩個不同的位置 i, j，滿足 $a_i + a_j = x$，如果找不到則輸出 IMPOSSIBLE

- 我們一樣用枚舉的方式來做
- 先將整個序列由小到大排序好
- 去找當前排序後的 a_1 可以跟哪一個元素配
- 檢查 $a_1 + a_n$ 的結果，如果 $a_1 + a_n > x$ 那我們就去檢查 $a_1 + a_{n-1}$
- 因為 $a_1 + a_n > x$ 那麼 $a_2 + a_n$ 的結果一定也大於 x

Sum of Two Values

給定一個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列，請你找到兩個不同的位置 i, j，滿足 $a_i + a_j = x$，如果找不到則輸出 IMPOSSIBLE

- 我們一樣用枚舉的方式來做
- 先將整個序列由小到大排序好
- 去找當前排序後的 a_1 可以跟哪一個元素配
- 檢查 $a_1 + a_n$ 的結果，如果 $a_1 + a_n > x$ 那我們就去檢查 $a_1 + a_{n-1}$
- 因為 $a_1 + a_n > x$ 那麼 $a_2 + a_n$ 的結果一定也大於 x
- 如果 $a_1 + a_n < x$ 接下來就直接檢查 $a_2 + a_n$
- 因為 $a_1 + a_n < x$，那麼 $a_1 + a_{n-1}$ 也一定小於 x

CSES Sum of Two Values

給定一個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列，請你找到兩個不同的位置 i, j，滿足 $a_i + a_j = x$，如果找不到則輸出 IMPOSSIBLE

- 照這樣的規則下去，你會發現我們就是在控制兩個指針
- 當第一個指針的元素加上第二個指針的元素 $> m$ 時，移動第二個指針，讓總和變小一點，$< m$ 時移動第一個指針，讓總和變大一點
- 這樣兩個指針一直往內縮，直到找到答案為止
- 時間複雜度 $O(n + n \log n)$

Ten Point Round #14 一週年紀念賽 pH. Colten 與風原的餅乾大戰爭

現在有兩排餅乾，每一排有 N $(1 \leq N \leq 2 \times 10^5)$，第一排的餅該每一個販售 G_1 元，第二排的每一個販售 G_2 元，每一個餅乾都有一個滿足度，Colten 要買餅乾給 Koying，Koying 要求這些餅乾至少要有 K 的滿足度，請問 Colten 最少需要花多少錢，如果無法達成要求輸出 -1

給大家一點時間想想看這一題

Ten Point Round #14 一週年紀念賽 pH. Colten 與風原的餅乾大戰爭

現在有兩排餅乾，每一排有 N $(1 \leq N \leq 2 \times 10^5)$，第一排的餅該每一個販售 G_1 元，第二排的每一個販售 G_2 元，每一個餅乾都有一個滿足度，Colten 要買餅乾給 Koying，Koying 要求這些餅乾至少要有 K 的滿足度，請問 Colten 最少需要花多少錢，如果無法達成要求輸出 -1

你觀察出什麼了嗎？

Ten Point Round #14 一週年紀念賽 pH. Colten 與風原的餅乾大戰爭

現在有兩排餅乾，每一排有 N $(1 \leq N \leq 2 \times 10^5)$，第一排的餅該每一個販售 G_1 元，第二排的每一個販售 G_2 元，每一個餅乾都有一個滿足度，Colten 要買餅乾給 Koying，Koying 要求這些餅乾至少要有 K 的滿足度，請問 Colten 最少需要花多少錢，如果無法達成要求輸出 -1

你觀察出什麼了嗎？

- 每一排的餅乾價格都是固定的，那我一定優先買滿足度比較高的餅乾（這其實是貪心算法的概念）

Ten Point Round #14 一週年紀念賽 pH. Colten 與風原的餅乾大戰爭

現在有兩排餅乾，每一排有 N $(1 \leq N \leq 2 \times 10^5)$，第一排的餅該每一個販售 G_1 元，第二排的每一個販售 G_2 元，每一個餅乾都有一個滿足度，Colten 要買餅乾給 Koying，Koying 要求這些餅乾至少要有 K 的滿足度，請問 Colten 最少需要花多少錢，如果無法達成要求輸出 -1

你觀察出什麼了嗎？

- 每一排的餅乾價格都是固定的，那我一定優先買滿足度比較高的餅乾（這其實是貪心算法的概念）
- 我們先把餅乾照滿足度排序好，我們枚舉第一排的餅乾要買幾個

Ten Point Round #14 一週年紀念賽 pH. Colten 與風原的餅乾大戰爭

現在有兩排餅乾，每一排有 N $(1 \leq N \leq 2 \times 10^5)$，第一排的餅該每一個販售 G_1 元，第二排的每一個販售 G_2 元，每一個餅乾都有一個滿足度，Colten 要買餅乾給 Koying，Koying 要求這些餅乾至少要有 K 的滿足度，請問 Colten 最少需要花多少錢，如果無法達成要求輸出 -1

你觀察出什麼了嗎？

- 每一排的餅乾價格都是固定的，那我一定優先買滿足度比較高的餅乾（這其實是貪心算法的概念）
- 我們先把餅乾照滿足度排序好，我們枚舉第一排的餅乾要買幾個
- 接著我們去找看看，第二排的餅乾最少需要買多少才能讓滿足度 $\geq K$

Ten Point Round #14 一週年紀念賽 pH. Colten 與風原的餅乾大戰爭

現在有兩排餅乾，每一排有 N $(1 \leq N \leq 2 \times 10^5)$，第一排的餅該每一個販售 G_1 元，第二排的每一個販售 G_2 元，每一個餅乾都有一個滿足度，Colten 要買餅乾給 Koying，Koying 要求這些餅乾至少要有 K 的滿足度，請問 Colten 最少需要花多少錢，如果無法達成要求輸出 -1

你觀察出什麼了嗎？

- 接著我們去找看看，第二排的餅乾最少需要買多少才能讓滿足度 $\geq K$
- 如果在第一排買 C_1 個餅乾的情況下，第二排需要買 C_2 個餅乾才可以滿足題目要求
- 那麼當我第一排買 $C_1 + 1$ 個餅乾時，第二排需要的餅乾數量會更多？更少？

Ten Point Round #14 一週年紀念賽 pH. Colten 與風原的餅乾大戰爭

現在有兩排餅乾，每一排有 N $(1 \leq N \leq 2 \times 10^5)$，第一排的餅該每一個販售 G_1 元，第二排的每一個販售 G_2 元，每一個餅乾都有一個滿足度，Colten 要買餅乾給 Koying，Koying 要求這些餅乾至少要有 K 的滿足度，請問 Colten 最少需要花多少錢，如果無法達成要求輸出 -1

你觀察出什麼了嗎？

- 接著我們去找看看，第二排的餅乾最少需要買多少才能讓滿足度 $\geq K$
- 如果在第一排買 C_1 個餅乾的情況下，第二排需要買 C_2 個餅乾才可以滿足題目要求
- 那麼當我第一排買 $C_1 + 1$ 個餅乾時，第二排需要的餅乾數量會更多？更少？
- 很明顯當我們第一排拿得越多，第二排需要拿的就會越少
- 如此一來就出現了單調性（遞增或遞減的性質）

Ten Point Round #14 一週年紀念賽 pH. Colten 與風原的餅乾大戰爭

現在有兩排餅乾，每一排有 N $(1 \leq N \leq 2 \times 10^5)$，第一排的餅該每一個販售 G_1 元，第二排的每一個販售 G_2 元，每一個餅乾都有一個滿足度，Colten 要買餅乾給 Koying，Koying 要求這些餅乾至少要有 K 的滿足度，請問 Colten 最少需要花多少錢，如果無法達成要求輸出 -1

你觀察出什麼了嗎？

- 因此我們就可以用雙指針去維護兩排共拿幾個餅乾
- 一開始把第一個指針放在第一排的最前端，第二個指針放在第二排的最尾端
- 依序枚舉第一排要拿幾個餅乾 $(0 \sim N)$，然後移動第二個指針值到滿足度已經 $< K$，就可以確定當前第一排的餅乾拿的數量會需要幾個第二排的餅乾來補

Ten Point Round #14 一週年紀念賽 pH. Colten 與風原的餅乾大戰爭

現在有兩排餅乾，每一排有 N $(1 \leq N \leq 2 \times 10^5)$，第一排的餅該每一個販售 G_1 元，第二排的每一個販售 G_2 元，每一個餅乾都有一個滿足度，Colten 要買餅乾給 Koying，Koying 要求這些餅乾至少要有 K 的滿足度，請問 Colten 最少需要花多少錢，如果無法達成要求輸出 -1

你觀察出什麼了嗎？

- 因此我們就可以用雙指針去維護兩排共拿幾個餅乾
- 一開始把第一個指針放在第一排的最前端，第二個指針放在第二排的最尾端
- 依序枚舉第一排要拿幾個餅乾 $(0 \sim N)$，然後移動第二個指針值到滿足度已經 $< K$，就可以確定當前第一排的餅乾拿的數量會需要幾個第二排的餅乾來補
- 接著就繼續枚舉第一排的下一個數量，並利用上一次的資訊，移動第二個指針，如此一來時間複雜度就會是 $O(N + N \log N)$

雙指針 Two Pointers

```
36      int ans = 8e9,index = n; // 第二個指針 ( 拿了 index 個第二排的餅乾 )
37
38      for(int i=0;i≤n;i++) // 第一個指針 ( 拿了 i 個第一排的餅乾 )
39      {
40          /* 如果當拿了 i 個第一排的餅乾時，需要 index 個第二排的餅乾，滿足度才會 ≥ t */
41
42          while( a_total + b_total ≥ t && index ≥ 0 )
43          {
44              // 如果當前滿組度 ≥ t，那麼試著減少第二排所拿的數量
45
46              ans = min( i * g1 + index * g2 ,ans) // 這次的花費
47
48              if( index - 1 ≥ 0 ) b_total -= b[index-1]; // 更新當前第二排所拿的滿足度總和 ( 注意索引值邊界 )
49
50              index--; // 第二個指針向左移動一格
51          }
52
53          if( i < n ) a_total += a[i]; // 更新當前第一排所拿的滿足度總和
54      }
55
56      if( ans == 8e9 ) cout << -1 << "\n";
57
58      else cout << ans << "\n";
```

練習題

CSES Sum of Three Values

給你一組長度為 n $(1 \leq n \leq 5000)$ 的序列，請你找到三個不同的位置 (i, k, j) 滿足 $a_i + a_k + a_j = x$

CSES Sum of Four Values

給你一組長度為 n $(1 \leq n \leq 1000)$ 的序列，請你找到四個不同的位置 (i, k, j, p) 滿足 $a_i + a_k + a_j + a_p = x$

Educational Ten Point Round #1 嘉義高中 x 嘉義女中讀書會練習賽 pH. 如果心是近的，遙遠的路也會是短的

原題目敘述較複雜，請去原題閱讀題目

滑動窗口 Sliding Window

滑動窗口 Sliding Window

- 其實他實際上就是雙指針
- 但因為實際的過程就像是一個窗口一直不斷的平移，因此被叫做滑動窗口

CSES Problem Set Subarray Distinct Values

給定一個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列，請問這一個序列有幾個區間滿足間間內的相異元素數量不超過 k 個

最簡單的做法是暴力枚舉 $O(n^2)$，但是顯然是會 TLE 的，用滑動窗口的想法可以做到 $O(n)$

CSES Problem Set Subarray Distinct Values

給定一個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列，請問這一個序列有幾個區間滿足間間內的相異元素數量不超過 k 個

- 我們先找到從 1 開始的區間，最遠可以延長到哪裡

CSES Problem Set Subarray Distinct Values

給定一個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列，請問這一個序列有幾個區間滿足間間內的相異元素數量不超過 k 個

- 我們先找到從 1 開始的區間，最遠可以延長到哪裡
- 假設可以延長到 5，那麼就表示 $[1,6],[1,7],...,[1,n]$ 這幾個區間不相同的元素數量都 $> K$

CSES Problem Set Subarray Distinct Values

給定一個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列，請問這一個序列有幾個區間滿足間間內的相異元素數量不超過 k 個

- 我們先找到從 1 開始的區間，最遠可以延長到哪裡
- 假設可以延長到 5，那麼就表示 $[1,6],[1,7],...,[1,n]$ 這幾個區間不相同的元素數量都 $> K$
- 這時我們可以確定 $[1,1],[1,2],[1,3],[1,4],[1,5]$ 這 5 個區間會是合法的答案，因此答案數量 $+5$

CSES Problem Set Subarray Distinct Values

給定一個長度為 n $(1 \leq n \leq 2 \times 10^5)$ 的序列，請問這一個序列有幾個區間滿足間間內的相異元素數量不超過 k 個

- 我們先找到從 1 開始的區間，最遠可以延長到哪裡
- 假設可以延長到 5，那麼就表示 $[1,6],[1,7],...,[1,n]$ 這幾個區間不相同的元素數量都 $> K$
- 這時我們可以確定 $[1,1],[1,2],[1,3],[1,4],[1,5]$ 這 5 個區間會是合法的答案，因此答案數量 $+5$
- 接著我們要去找 2 最遠可以延伸到哪裡，我們可以很確定的是，既然 $[1,2],[1,3],[1,4],[1,5]$ 是合法的答案，那麼 $[2,2],[2,3],[2,4],[2,5]$ 也一定會是合法的答案，因此我們就不用重頭找起，從上一次的結果繼續往下延伸區間就可以了，如此一來就可以在 $O(n)$ 的時間枚舉完成，用 `map` 紀錄就多帶一個 `log`

二分搜尋 Binary Search

幸運數字

現在有 $1 \sim 10^{18}$ 這些數字，其中一個數字是幸運數字，請你找到他，你可以做 $\log 10^{18} + 1$ 次猜測，猜錯了會告訴你幸運數字比你猜測的大還是小，請問應該要用什麼策略猜測一定可以在有限的次數內猜到答案呢？

幸運數字

現在有 $1 \sim 10^{18}$ 這些數字，其中一個數字是幸運數字，請你找到他，你可以做 $\texttt{log}\, 10^{18} + 1$ 次猜測，猜錯了會告訴你幸運數字比你猜測的大還是小，請問應該要用什麼策略猜測一定可以在有限的次數內猜到答案呢？

- 假設現在答案落在區間 $[L, R]$

幸運數字

現在有 $1 \sim 10^{18}$ 這些數字，其中一個數字是幸運數字，請你找到他，你可以做 $\log 10^{18} + 1$ 次猜測，猜錯了會告訴你幸運數字比你猜測的大還是小，請問應該要用什麼策略猜測一定可以在有限的次數內猜到答案呢？

- 假設現在答案落在區間 $[L, R]$
- 那我們就猜測 $\frac{L+R}{2}$

幸運數字

現在有 $1 \sim 10^{18}$ 這些數字，其中一個數字是幸運數字，請你找到他，你可以做 $\log 10^{18} + 1$ 次猜測，猜錯了會告訴你幸運數字比你猜測的大還是小，請問應該要用什麼策略猜測一定可以在有限的次數內猜到答案呢？

- 假設現在答案落在區間 $[L, R]$
- 那我們就猜測 $\frac{L+R}{2}$
- 如此一來我們可以保證你就算猜錯，答案的可能區間也至少會縮小一半

幸運數字

現在有 $1 \sim 10^{18}$ 這些數字，其中一個數字是幸運數字，請你找到他，你可以做 $\log 10^{18} + 1$ 次猜測，猜錯了會告訴你幸運數字比你猜測的大還是小，請問應該要用什麼策略猜測一定可以在有限的次數內猜到答案呢？

- 假設現在答案落在區間 $[L, R]$
- 那我們就猜測 $\frac{L+R}{2}$
- 如此一來我們可以保證你就算猜錯，答案的可能區間也至少會縮小一半
- 由於每一次都縮小一半，因此我們可以很確定我們可以在題目限制的次數內找到答案，這就是二分搜的概念

二分搜尋 Binary Search

- 二分搜的實作簡單，但細節非常多
- 10 個人會有 10 種二分搜的寫法
- 大家只要平時都寫自己最不會寫錯的做法就可以了
- 我等等會提供我的二分搜寫法，但在這之前我們先來看看二分搜的常見 bug

二分搜的常見 bug

- 左界 + 右界的時候超出 int 範圍，發生 overflow
- 在序列上二分搜前忘記排序
- 沒有確定好答案落在右界還是左界（如果你的做法不會讓左界跟右界剛好重疊的話）
- 右界左界的初始值設定錯（我很常這樣 QQ）

我的二分搜做法

- 每次在移動的時候都將左界縮成 `mid + 1`，如果是移動右界則是將右界設定成 `mid - 1`
- 這種做法的缺點是你最後要判斷答案是左界還是右界
- 舉個例子: 如果現在 `mid` 是合法的答案，那麼你要找看看有沒有比 `mid` 還小而且合法的答案，那麼這種情況下，答案會落在左界 L

我的二分搜做法

- 每次在移動的時候都將左界縮成 `mid + 1`，如果是移動右界則是將右界設定成 `mid - 1`
- 這種做法的缺點是你最後要判斷答案是左界還是右界
- 舉個例子： 如果現在 `mid` 是合法的答案，那麼你要找看看有沒有比 `mid` 還小而且合法的答案，那麼這種情況下，答案會落在左界 L
- 但如果現在 `mid` 是合法的答案，那你要找有沒有比 `mid` 還大而且合法的答案，在這種情況下，答案會落在右界 R

分析答案是誰

- 如果現在 mid 是合法的答案，那麼你要找看看有沒有比 mid 還小而且合法的答案，那麼這種情況下，答案會落在左界 L
- 如果現在右界變成了 mid - 1，且 mid - 1 不是合法的答案的話，最後左界跟右界會同時在 mid - 1，但 mid - 1 不是合法的答案，因此左界會變成 (mid - 1) + 1，所以答案會是左界

分析答案是誰

- 如果現在 mid 是合法的答案，那你要找有沒有比 mid 還大而且合法的答案，在這種情況下，答案會落在右界 R
- 如果現在 mid + 1 不是合法的答案，最後左界跟右界會同時在 mid + 1
- 但因為 mid + 1 不是合法的答案，這個時候右界會變成 (mid + 1) - 1，因此答案會是右界
- 因此使用我這種做法的話大家務必要判斷清楚答案落在左界還是右界

STL 內建的二分搜工具

- `binary_search(L,R,val)`
- `lower_bound(L,R,val)`
- `upper_bound(L,R,val)`

binary_search

- binary_search(L,R,val)
- 在區間 [L,R) 之間尋找是否存在 val
- L,R 所放置的是迭代器位置
- 回傳 true 或 false 分別表示是否有找到指定的元素
- 記得就算是使用工具也是要先排序

lower_bound

- lower_bound(L,R,val)
- L,R 所放置的是迭代器位置
- 回傳在區間 [L,R) 之間尋找第一個 $\geq$ val 的迭代器位置
- 記得就算是使用工具也是要先排序

upper_bound

- upper_bound(L,R,val)
- L,R 所放置的是迭代器位置
- 回傳在區間 [L,R) 之間尋找第一個 $>$ val 的迭代器位置
- 記得就算是使用工具也是要先排序

使用工具的好處

- 超級方便省時，還不用怕自己寫錯

使用工具的好處

- 超級方便省時，還不用怕自己寫錯
- 額，沒了

工具超級好用之例題

Ten Point Round #5 第一屆初學者程式設計交流會團體賽 pD. 琳瑯滿目

NHDK 的圖書館裡面有 n $(1 \leq n \leq 2 \times 10^5)$ 本書，每一本書有一個編號 s_i $(1 \leq s_i \leq 10^9)$，接下來將有 Q $(1 \leq Q \leq 2 \times 10^5)$ 組詢問，每組詢問將詢問一個編號，你必須回答這一個編號的書有幾本

給大家一點時間想想看

工具超級好用之例題

Ten Point Round #5 第一屆初學者程式設計交流會團體賽 pD. 琳瑯滿目

NHDK 的圖書館裡面有 n $(1 \leq n \leq 2 \times 10^5)$ 本書，每一本書有一個編號 s_i $(1 \leq s_i \leq 10^9)$，接下來將有 Q $(1 \leq Q \leq 2 \times 10^5)$ 組詢問，每組詢問將詢問一個編號，你必須回答這一個編號的書有幾本

- 這題也可以用 map 來解決
- 但我們來看看怎麼用二分搜解決

工具超級好用之例題

Ten Point Round #5 第一屆初學者程式設計交流會團體賽 pD. 琳瑯滿目

NHDK 的圖書館裡面有 n $(1 \leq n \leq 2 \times 10^5)$ 本書，每一本書有一個編號 s_i $(1 \leq s_i \leq 10^9)$，接下來將有 Q $(1 \leq Q \leq 2 \times 10^5)$ 組詢問，每組詢問將詢問一個編號，你必須回答這一個編號的書有幾本

- 這題也可以用 `map` 來解決
- 但我們來看看怎麼用二分搜解決
- `upper_bound` 會回傳第一個大於查詢元素的位置
- `lower_bound` 會回傳第一個大於等於查詢元素的位置

工具超級好用之例題

Ten Point Round #5 第一屆初學者程式設計交流會團體賽 pD. 琳瑯滿目

NHDK 的圖書館裡面有 n $(1 \le n \le 2 \times 10^5)$ 本書，每一本書有一個編號 s_i $(1 \le s_i \le 10^9)$，接下來將有 Q $(1 \le Q \le 2 \times 10^5)$ 組詢問，每組詢問將詢問一個編號，你必須回答這一個編號的書有幾本

- 這題也可以用 `map` 來解決
- 但我們來看看怎麼用二分搜解決
- `upper_bound` 會回傳第一個大於查詢元素的位置
- `lower_bound` 會回傳第一個大於等於查詢元素的位置
- 不知道有沒有人已經發現 ... 我們直接把這兩個位置相減，就是這一個元素在這一個序列出現的數量了 !!!
- 時間複雜度 $O(n \log n + Q \log n)$

練習題

TIOJ 1044 [Interactive] Guess My Number

本題是互動題，請至原題了解題目

TNFSH OJ 47 magic spell

本題題敘讓我無法以三言兩語就帶過，請至原題觀看題敘 XDDD

對答案二分搜

對答案二分搜

- 顧名思義，對答案二分搜
- 當你發現你可以檢查某個答案是不是合法的，而且你也可以在判斷完時知道接下來答案會落在哪一邊，就可以對答案二分搜
- 有時候不想用數學來解題目，發現可以直接對答案二分搜，那就砸下去就對了，不用動腦 XD

對答案二分搜

APCS 2017 3 月第四題基地台

現在有 N $(1 \leq N \leq 50000)$ 個服務點，給你這 N 個服務點的座標，你現在可以隨意在任何地方架設 K 個基地台，請問基地台的服務直徑最少只需要多少服務範圍就可以到所有服務點（所有服務點都是在一條數線上），座標範圍最多到 10^9

如果我給你一個直徑，你能回答這一個直徑是不是 ok 的嗎？

對答案二分搜

APCS 2017 3 月第四題基地台

現在有 N $(1 \leq N \leq 50000)$ 個服務點，給你這 N 個服務點的座標，你現在可以隨意在任何地方架設 K 個基地台，請問基地台的服務直徑最少只需要多少服務範圍就可以到所有服務點（所有服務點都是在一條數線上），座標範圍最多到 10^9

如果可以，那現在如果我給你的這個直徑不 ok 那我們是不是就往更大的直徑去找?

對答案二分搜

APCS 2017 3 月第四題基地台

現在有 N $(1 \leq N \leq 50000)$ 個服務點，給你這 N 個服務點的座標，你現在可以隨意在任何地方架設 K 個基地台，請問基地台的服務直徑最少只需要多少服務範圍就可以到所有服務點（所有服務點都是在一條數線上），座標範圍最多到 10^9

如果這個直徑 ok，那我們是不是就可以找找看有沒有更小的直徑，你會發現有了這一個性質我們就可以對答案二分搜！

對答案二分搜

APCS 2017 3 月第四題基地台

現在有 N $(1 \leq N \leq 50000)$ 個服務點，給你這 N 個服務點的座標，你現在可以隨意在任何地方架設 K 個基地台，請問基地台的服務直徑最少只需要多少服務範圍就可以到所有服務點（所有服務點都是在一條數線上），座標範圍最多到 10^9

- 要怎麼檢查某一個直徑是不是 ok 的呢？

對答案二分搜

APCS 2017 3 月第四題基地台

現在有 N $(1 \leq N \leq 50000)$ 個服務點，給你這 N 個服務點的座標，你現在可以隨意在任何地方架設 K 個基地台，請問基地台的服務直徑最少只需要多少服務範圍就可以到所有服務點（所有服務點都是在一條數線上），座標範圍最多到 10^9

- 要怎麼檢查某一個直徑是不是 ok 的呢？
- 這裡會有一些貪心的概念，如果現在你要放下你的第一台基地台，你會放在哪裡？

對答案二分搜

APCS 2017 3 月第四題基地台

現在有 N $(1 \leq N \leq 50000)$ 個服務點，給你這 N 個服務點的座標，你現在可以隨意在任何地方架設 K 個基地台，請問基地台的服務直徑最少只需要多少服務範圍就可以到所有服務點（所有服務點都是在一條數線上），座標範圍最多到 10^9

- 要怎麼檢查某一個直徑是不是 ok 的呢？
- 這裡會有一些貪心的概念，如果現在你要放下你的第一台基地台，你會放在哪裡？
- 為了讓服務範圍越廣，我們可以確定我們在最佳的放置方式下，如果第一個服務點的位置是 P，當前檢查的直徑是 T，那我們第一個基地台最遠的涵蓋範圍就會是 $P \sim P + T$

對答案二分搜

APCS 2017 3 月第四題基地台

現在有 N $(1 \leq N \leq 50000)$ 個服務點，給你這 N 個服務點的座標，你現在可以隨意在任何地方架設 K 個基地台，請問基地台的服務直徑最少只需要多少服務範圍就可以到所有服務點（所有服務點都是在一條數線上），座標範圍最多到 10^9

- 要怎麼檢查某一個直徑是不是 ok 的呢？
- 這裡會有一些貪心的概念，如果現在你要放下你的第一台基地台，你會放在哪裡？
- 為了讓服務範圍越廣，我們可以確定我們在最佳的放置方式下，如果第一個服務點的位置是 P，當前檢查的直徑是 T，那我們第一個基地台最遠的涵蓋範圍就會是 $P \sim P + T$
- 如果當前的服務點已經超出服務範圍了，那我們就比照第一個基地台放置的方式，重新更新服務的範圍

對答案二分搜

APCS 2017 3 月第四題基地台

現在有 N $(1 \leq N \leq 50000)$ 個服務點，給你這 N 個服務點的座標，你現在可以隨意在任何地方架設 K 個基地台，請問基地台的服務直徑最少只需要多少服務範圍就可以到所有服務點（所有服務點都是在一條數線上），座標範圍最多到 10^9

我們會需要 $O(log 10^6)$ 的時間做二分搜，每一次二分搜需要花 $O(N)$ 的時間檢查，檢查前要先將服務點排序，總時間複雜度為 $O(N \log N + N \log 10^6)$

雖然我不是數學家，但這聽起來很不錯對吧

對答案二分搜

```
int mid = ( l + r ) / 2;
```

```
40          int now = a[0] + mid; // 第一個基地台可以涵蓋到的位置
41
42          int u = 1; // 需要的基地台數量
43
44          for(int i=1;i<n;i++)
45          {
46              if( a[i] ≤ now ) continue; // 已經被當前最新的基地台涵蓋到了
47
48              else // 當前基地台都涵蓋不到，所以要架一個新的
49              {
50                  u++;
51
52                  now = a[i] + mid; // 新的基地台可以涵蓋到的範圍
53              }
54          }
```

對答案二分搜

Ten Point Round #9 (Div. 1 + Div. 2) pD. Gunjyo 的點數分配問題

在一個風和日麗的早晨，Gunjyo 一如往常的一邊走路一邊想著她自己發明的 Gunjyo 演算法。

Gunjyo 是一個喜歡照顧小孩子的人，所以他某天到了一間幼稚園打工，而且萬萬沒想到，幼稚園裡的老師居然給了她一個難題。老師希望 Gunjyo 能夠幫他分配一些點數給兩位小朋友，而且這兩個小朋友要拿到的點數數量也不太一樣，因為老師會照著小朋友們的表現來去分配點數而且這些小朋友原本手上就有一些點數。這兩個小朋友我們這邊稱 A 與 B，由於他們的表現，所以老師決定 A 最終擁有的點數一定要比 B 多。而且老師給 Gunjyo 的點數一定要全部用完。Gunjyo 知道分配點數這件事其實很簡單，不過她想要知道，在依照規則的情況下，分配所有點數後，會有幾種的情況是符合老師的要求的。（也就是 A 拿到的點數比 B 還要多）請你寫一個程式，幫助 Gunjyo 算出有幾種情況吧！輸入：N, M, K $(0 \leq N, M, K \leq 10^9)$ 表示一開始 A 有 N 點 B 有 M 點，Gunjyo 有 K 點可以分給他們

對答案二分搜

- 這題可以用數學解，但我真的不是數學家
- 所以我們就來對答案二分搜吧

對答案二分搜

- 這題可以用數學解，但我真的不是數學家
- 所以我們就來對答案二分搜吧
- 我們去二分搜我們要給 A 多少的點數
- 當我們發現我們如果分配給 A 5 個點數結果會是合法的話，我們就試著去給 A 更少的點數
- 最後得到的結果會是我們最少會需要給 A 的點數數量，我們就可以確定我們給 A 多少點數會是合法的了

有負數的二分搜

有負數的二分搜

- 如果二分搜的值有負數，取 `mid` 的時候必須寫 $L + \frac{R-L}{2}$，而不是 $\frac{L+R}{2}$

有負數的二分搜

- 如果二分搜的值有負數，取 `mid` 的時候必須寫 $L + \frac{R-L}{2}$，而不是 $\frac{L+R}{2}$
- 我們平時取 `mid` 的時候都是 `floor`，也就是向下取整（例如：$\frac{2+3}{2} = 2$）

有負數的二分搜

- 如果二分搜的值有負數，取 `mid` 的時候必須寫 $L + \frac{R-L}{2}$，而不是 $\frac{L+R}{2}$
- 我們平時取 `mid` 的時候都是 `floor`，也就是向下取整（例如：$\frac{2+3}{2} = 2$）
- 但是在負數時可能會變成向上取整（例如：$\frac{-2+-3}{2} = -2$，正確應該要是 -3）

浮點數上的二分搜

浮點數上的二分搜

ITMO Academy: pilot course » Binary Search » Step 2 pB. Ropes

現在有 n $(1 \leq n \leq 10^4)$ 個線段，現在你需要 k $(1 \leq k \leq 10^4)$ 段長度相同的線段，你可以對這些線段做任意的切割操作，找到這 k 段線段的最大長度，剛開始的 n 條線段的長度最長到 10^7

浮點數上的二分搜

ITMO Academy: pilot course » Binary Search » Step 2 pB. Ropes

現在有 n $(1 \leq n \leq 10^4)$ 個線段，現在你需要 k $(1 \leq k \leq 10^4)$ 段長度相同的線段，你可以對這些線段做任意的切割操作，找到這 k 段線段的最大長度，剛開始的 n 條線段的長度最長到 10^7

很明顯是個對答案二分搜的題目對吧，但這次有浮點數了哦！
要怎麼做浮點數二分搜呢？

浮點數上二分搜

- 在學會怎麼寫浮點數二分搜前，我們要先介紹一個東西叫做 EPS（Epslion）
- 是一個用來修正浮點數誤差的東西

浮點數上二分搜

- 在學會怎麼寫浮點數二分搜前，我們要先介紹一個東西叫做 EPS（Epslion）
- 是一個用來修正浮點數誤差的東西
- 通常我們會開一個變數，並將這個變數設成一個超級小的值（通常是 10^{-7} 或 10^{-9}）
- 由於有時候誤差會導致我們在判斷兩個浮點數是否相等時發生問題，因此我們通常在判斷兩個浮點數是否相等時會直接檢查兩個數字的差距是否 $<$ EPS 來判斷

浮點數上二分搜

- 浮點數上二分搜的關鍵是我們什麼時候要停止

浮點數上二分搜

- 浮點數上二分搜的關鍵是我們什麼時候要停止
- 左界跟右界都夾在一起的時候要停止

浮點數上二分搜

- 浮點數上二分搜的關鍵是我們什麼時候要停止
- 左界跟右界都夾在一起的時候要停止
- 但由於是浮點數，判斷浮點數是否相等是一個很恐怖的事情，會被誤差搞死，因此我們才會需要 EPS

浮點數上二分搜

- 浮點數上二分搜的關鍵是我們什麼時候要停止
- 左界跟右界都夾在一起的時候要停止
- 但由於是浮點數，判斷浮點數是否相等是一個很恐怖的事情，會被誤差搞死，因此我們才會需要 EPS
- 所以我們的二分搜終止條件就會變成 $(r-l) \leq$ EPS 的時候停止，換句話說，當 $(r-l) >$ EPS 的時候我們繼續執行二分搜

浮點數上二分搜

- 浮點數上二分搜的關鍵是我們什麼時候要停止
- 左界跟右界都夾在一起的時候要停止
- 但由於是浮點數，判斷浮點數是否相等是一個很恐怖的事情，會被誤差搞死，因此我們才會需要 EPS
- 所以我們的二分搜終止條件就會變成 $(r-l) \leq$ EPS 的時候停止，換句話說，當 $(r-l) >$ EPS 的時候我們繼續執行二分搜
- 由於當 r 跟 l 的差距程度非常小的時候我們就可以當作他已經幾乎夾在一起了，因此停止二分搜

浮點數上二分搜

- 浮點數上二分搜的關鍵是我們什麼時候要停止
- 左界跟右界都夾在一起的時候要停止
- 但由於是浮點數，判斷浮點數是否相等是一個很恐怖的事情，會被誤差搞死，因此我們才會需要 EPS
- 所以我們的二分搜終止條件就會變成 $(r-l) \leq$ EPS 的時候停止，換句話說，當 $(r-l) >$ EPS 的時候我們繼續執行二分搜
- 由於當 r 跟 l 的差距程度非常小的時候我們就可以當作他已經幾乎夾在一起了，因此停止二分搜
- 這個時候有人會想要問，這樣可能會跟正確答案有一些小小的誤差吧，這樣還會 AC 嗎

浮點數上二分搜

- 好的題目通常都會在最後跟你說你最後的答案最大可以跟正確答案有多少誤差，因此就算跟答案有極小的誤差也是會被判定正確的
- 像是剛剛那一題例題誤差最大可到 10^{-6}

浮點數上二分搜

- 好的題目通常都會在最後跟你說你最後的答案最大可以跟正確答案有多少誤差，因此就算跟答案有極小的誤差也是會被判定正確的
- 像是剛剛那一題例題誤差最大可到 10^{-6}
- 我相信很多人的心聲是： EPS 好麻煩哦

浮點數上二分搜

- 好的題目通常都會在最後跟你說你最後的答案最大可以跟正確答案有多少誤差，因此就算跟答案有極小的誤差也是會被判定正確的
- 像是剛剛那一題例題誤差最大可到 10^{-6}
- 我相信很多人的心聲是： EPS 好麻煩哦，有時候還會控錯
- 好，沒關係，有一個更無腦的做法

浮點數上二分搜

- 好的題目通常都會在最後跟你說你最後的答案最大可以跟正確答案有多少誤差，因此就算跟答案有極小的誤差也是會被判定正確的
- 像是剛剛那一題例題誤差最大可到 10^{-6}
- 我相信很多人的心聲是： EPS 好麻煩哦，有時候還會控錯
- 好，沒關係，有一個更無腦的做法
- 我們乾脆就直接賭他二分搜 100 次後左界跟右界會夾在一起！這樣我們就不用擔心什麼 EPS 的問題了

浮點數上二分搜

```cpp
    vector <int> a(n);

    for(int i=0;i<n;i++) cin >> a[i];
    double l = 0 , r = 1e18;
    double ans = 0.0;

    for(int i=0;i<100;i++)
    {
        double mid = ( l + r ) / 2;

        int total = 0;

        for(int j=0;j<n;j++)
        {
            total += (int)(a[j] / mid);

            if( total ≥ k ) break;
        }

        if( total ≥ k ) l = mid;
        else r = mid;
    }

    cout << fixed << setprecision(10) << l << "\n";
}
```

經典題-對平均二分搜

對平均二分搜

ITMO Academy: pilot course » Binary Search Step 4. pA. Maximum Average Segment

給你一個長度 n $(1 \leq n \leq 10^5)$ 的序列 a，給一個整數 d $(1 \leq d \leq 10^5)$ 請你找到一個長度至少 d 的區間，這一個區間的數字的平均要越大越好

對平均二分搜

ITMO Academy: pilot course » Binary Search Step 4. pA. Maximum Average Segment

給你一個長度 n $(1 \leq n \leq 10^5)$ 的序列 a，給一個整數 d $(1 \leq d \leq 10^5)$ 請你找到一個長度至少 d 的區間，這一個區間的數字的平均要越大越好

先試著想想看，給你一個數字，你有辦法確定你能湊出這一個平均值嗎？

對平均二分搜

- 假設我們選了一個區間 $[L, R]$，那麼這一個區間的平均會是 $\dfrac{\sum_{i=L}^{R} a_i}{R - L + 1}$

對平均二分搜

- 假設我們選了一個區間 $[L, R]$，那麼這一個區間的平均會是 $\dfrac{\sum_{i=L}^{R} a_i}{R - L + 1}$
- 我們假設平均是 x

對平均二分搜

- 假設我們選了一個區間 $[L, R]$，那麼這一個區間的平均會是 $\dfrac{\sum_{i=L}^{R} a_i}{R - L + 1}$
- 我們假設平均是 x，也就是說 $\dfrac{\sum_{i=L}^{R} a_i}{R - L + 1} = x$
- 那麼把 $R - L + 1$ 乘過去右邊，式子就會會變成 $\sum_{i=L}^{R} a_i = (R - L + 1) \cdot x$

對平均二分搜

- 假設我們選了一個區間 $[L, R]$，那麼這一個區間的平均會是 $\dfrac{\sum_{i=L}^{R} a_i}{R-L+1}$
- 我們假設平均是 x，也就是說 $\dfrac{\sum_{i=L}^{R} a_i}{R-L+1} = x$
- 那麼把 $R-L+1$ 乘過去右邊，式子就會會變成 $\sum_{i=L}^{R} a_i = (R-L+1) \cdot x$
- 再把 $R-L+1$ 個 x 丟到左邊去就可以變成 $(a_L - x) + (a_{L+1} - x) + ... + (a_R - x) = 0$

對平均二分搜

- 假設我們選了一個區間 $[L, R]$，那麼這一個區間的平均會是 $\dfrac{\sum_{i=L}^{R} a_i}{R-L+1}$
- 我們假設平均是 x，也就是說 $\dfrac{\sum_{i=L}^{R} a_i}{R-L+1} = x$
- 那麼把 $R-L+1$ 乘過去右邊，式子就會會變成 $\sum_{i=L}^{R} a_i = (R-L+1) \cdot x$
- 再把 $R-L+1$ 個 x 丟到左邊去就可以變成 $(a_L - x) + (a_{L+1} - x) + ... + (a_R - x) = 0$
- 那麼我們就可以發現，如果我們要確定某一個平均是不是有可能被湊出來的，我們只需要檢查平均 x 是否滿足 $(a_L - x) + (a_{L+1} - x) + ... + (a_R - x) = 0$ 這一個條件了

對平均二分搜

- 既然我們知道要怎麼檢查了，那我們就回來這題看看
- 我們可以把 $a_i - x$ 建成一個前綴和，並用一個變數紀錄當前前綴和的最小值，如果發現當前的前綴和減掉當前前綴和的最小值的數字 ≥ 0 而且這樣子的區間長度 $\geq d$，那麼就表示最大平均一定 $\geq x$

對平均二分搜

- 既然我們知道要怎麼檢查了，那我們就回來這題看看
- 我們可以把 $a_i - x$ 建成一個前綴和，並用一個變數紀錄當前前綴和的最小值，如果發現當前的前綴和減掉當前前綴和的最小值的數字 ≥ 0 而且這樣子的區間長度 $\geq d$，那麼就表示最大平均一定 $\geq x$
- 因此把左界變成 x，否則如果找不到就表示最大平均一定 $< x$，我們就把右界變成 x

對平均二分搜

- 既然我們知道要怎麼檢查了，那我們就回來這題看看
- 我們可以把 $a_i - x$ 建成一個前綴和，並用一個變數紀錄當前前綴和的最小值，如果發現當前的前綴和減掉當前前綴和的最小值的數字 ≥ 0 而且這樣子的區間長度 $\geq d$，那麼就表示最大平均一定 $\geq x$
- 因此把左界變成 x，否則如果找不到就表示最大平均一定 $< x$，我們就把右界變成 x
- 最後用得到的最大平均值去構造答案就可以滿足題目要求了！

對平均二分搜

```cpp
double l = 0.0 , r = 100.0;

for(int i=0;i≠100;i++)
{
    double mid = ( l + r ) / 2.0;
    vector <double> v(n+1,0.0);
    double mn = 0;

    bool check = false;

    for(int i=0;i<n;i++)
    {
        if( i + 1 - d ≥ 0 )
        {
            if( v[i+1-d] < mn ) mn = v[i+1-d];
        }

        v[i+1] = v[i] + ( a[i] - mid );

        if( v[i+1] - mn ≥ 0 && i + 1 ≥ d ) check = true; // 這個平均是 ok 的
    }

    if( check == true ) l = mid; // mid 可能會是平均，我們試圖去找更大的
    else r = mid; // 平均要比 mid 還要小，往小的地方去
}
```

對平均二分搜

```
57      double ans = r;
58
59      vector<double> v(n+1,0.0);
60
61      double mn = 0;
62      int idx = 0;
63
64      for(int i=0;i<n;i++)
65      {
66          if( i + 1 - d ≥ 0 )
67          {
68              if( v[i+1-d] < mn ) mn = v[i+1-d] , idx = i - d + 1;
69          }
70
71          v[i+1] = v[i] + ( a[i] - r );
72          if( v[i+1] - mn + eps ≥ 0 && i + 1 ≥ d ) // 這邊有可能會發生誤差，我被坑過一次
73          {
74              cout << idx + 1 << " " << i + 1 << "\n";
75
76              exit(0);
77          }
78      }
79  }
```

練習題

- 給大家一點時間練習對平均二分搜這題，小心這一題最後構造答案的時候可能會發生誤差

折半枚舉 Meet in the middle

折半枚舉 Meet in the middle

- 暴力美學，不那麼暴力的暴力
- 為什麼會被叫做折半枚舉等一下就會知道了 XD

折半枚舉 Meet in the middle

- 暴力美學，不那麼暴力的暴力
- 為什麼會被叫做折半枚舉等一下就會知道了 XD
- 枚舉全部時間複雜度很差，那你有試過分成一半分開枚舉嗎 ><

CSES Problem Set Meet in the middle

給你一個長度為 n $(1 \leq n \leq 40)$ 的序列，請問你有幾種選擇方式可以使你選擇的數字和等於 x

折半枚舉 Meet in the middle

CSES Problem Set Meet in the middle

給你一個長度為 n $(1 \leq n \leq 40)$ 的序列，請問你有幾種選擇方式可以使你選擇的數字和等於 x

- 全部枚舉？$O(2^n)$ 顯然是不夠好的，那我們只枚舉一半呢？

折半枚舉 Meet in the middle

CSES Problem Set Meet in the middle

給你一個長度為 n $(1 \leq n \leq 40)$ 的序列，請問你有幾種選擇方式可以使你選擇的數字和等於 x

- 全部枚舉？$O(2^n)$ 顯然是不夠好的，那我們只枚舉一半呢？
- 我們先把其中一半枚舉的結果用一個資料結構存起來，接下來枚舉另一半

折半枚舉 Meet in the middle

CSES Problem Set Meet in the middle

給你一個長度為 n $(1 \leq n \leq 40)$ 的序列，請問你有幾種選擇方式可以使你選擇的數字和等於 x

- 全部枚舉？$O(2^n)$ 顯然是不夠好的，那我們只枚舉一半呢？
- 我們先把其中一半枚舉的結果用一個資料結構存起來，接下來枚舉另一半
- 假設當前另外一半枚舉出來的總和是 y，那麼我們就必須知道原本枚舉的那一半有幾種選法總和會是 $x - y$，就可以知道當前這一半如果這樣選，原本的另一半會有幾種選法可以跟當前這一個選法配

折半枚舉 Meet in the middle

CSES Problem Set Meet in the middle

給你一個長度為 n $(1 \leq n \leq 40)$ 的序列，請問你有幾種選擇方式可以使你選擇的數字和等於 x

- 全部枚舉？$O(2^n)$ 顯然是不夠好的，那我們只枚舉一半呢？
- 我們先把其中一半枚舉的結果用一個資料結構存起來，接下來枚舉另一半
- 假設當前另外一半枚舉出來的總和是 y，那麼我們就必須知道原本枚舉的那一半有幾種選法總和會是 $x - y$，就可以知道當前這一半如果這樣選，原本的另一半會有幾種選法可以跟當前這一個選法配
-

折半枚舉 Meet in the middle

- 那我們要怎麼找到原本的那一半有多少數字可以匹配呢?

折半枚舉 Meet in the middle

- 那我們要怎麼找到原本的那一半有多少數字可以匹配呢?
- 回憶一下，我們想要在一個序列找有幾個數字滿足 $x - y$

折半枚舉 Meet in the middle

- 那我們要怎麼找到原本的那一半有多少數字可以匹配呢?
- 回憶一下，我們想要在一個序列找有幾個數字滿足 $x - y$
- 沒錯！`upper_bound - lower_bound`
- 時間複雜度：$O(2^{\frac{n}{2}} \log 2^{\frac{n}{2}})$

折半枚舉 Meet in the middle

```
41 signed main(void)
42 {
43     Colten_AC;
44     cin >> n >> m;
45
46     for(int i=0;i<n;i++) cin >> a[i];
47
48     dfs1(0,0); // 枚舉左半部
49     sort(arr.begin(),arr.end()); // 為了二分搜
50     dfs2(n/2+1,0); // 枚舉右半部
51
52     cout << ans << "\n";
53 }
```

折半枚舉 Meet in the middle

```cpp
int n,m,a[45],ans;

vector <int> arr;

void dfs1(int index,int sum)
{
    if( sum > m ) return;

    if( index == n / 2 + 1 )
    {
        arr.push_back(sum);

        return;
    }

    dfs1(index+1,sum+a[index]); // 選這一個數字
    dfs1(index+1,sum); // 不選這一個數字
}
void dfs2(int index,int sum)
{
    if( sum > m ) return;

    if( index == n )
    {
        ans += upper_bound(arr.begin(),arr.end(),m-sum) - lower_bound(arr.begin(),arr.end(),m-sum);

        return;
    }

    dfs2(index+1,sum+a[index]); // 選這一個數字
    dfs2(index+1,sum); // 不選這一個數字
}
```

折半枚舉 Meet in the middle

Codeforces Round #498 pF. Xor-Paths

給你一個 $n \times m$ $(1 \leq n, m \leq 20)$ 的網格，每一個網格上有一個數字，一開始你在左上角 $(1, 1)$，你的目標是走到右下角 (n, m)，每一次你只能往右或往下移動，請問你總共有幾種走法會使你路徑上的所有數字 xor 起來剛好是 k

- 第一次看到這種題目的應該會很難想到折半枚舉
- 先給大家想一下，等一下我會一步一步帶大家解決這題

折半枚舉 Meet in the middle

- 如果一個網格是 $n \times m$，那麼總共會需要走 $n + m - 1$ 步
- 極限時，$20 + 20 - 1 = 39$

折半枚舉 Meet in the middle

- 如果一個網格是 $n \times m$，那麼總共會需要走 $n + m - 1$ 步
- 極限時 $20 + 20 - 1 = 39$
- 每一條從 $(1, 1)$ 走到 (n, m) 的路徑我們都把它拆成兩半

折半枚舉 Meet in the middle

- 如果一個網格是 $n \times m$，那麼總共會需要走 $n + m - 1$ 步
- 極限時 $20 + 20 - 1 = 39$
- 每一條從 $(1, 1)$ 走到 (n, m) 的路徑我們都把它拆成兩半
- 拆成兩半的那一個點如果是 (x, y)，那麼也就是說現在這一條路徑等價於先從 $(1, 1)$ 走到 (x, y)，再從 (x, y) 走到 (n, m)

折半枚舉 Meet in the middle

- 如果一個網格是 $n \times m$，那麼總共會需要走 $n + m - 1$ 步
- 極限時 $20 + 20 - 1 = 39$
- 每一條從 $(1, 1)$ 走到 (n, m) 的路徑我們都把它拆成兩半
- 拆成兩半的那一個點如果是 (x, y)，那麼也就是說現在這一條路徑等價於先從 $(1, 1)$ 走到 (x, y)，再從 (x, y) 走到 (n, m)
- 折半枚舉最重要過程在於枚舉的部分被分成兩半之後能夠快速的匹配答案

折半枚舉 Meet in the middle

- 如果一個網格是 $n \times m$，那麼總共會需要走 $n + m - 1$ 步
- 極限時 $20 + 20 - 1 = 39$
- 每一條從 $(1, 1)$ 走到 (n, m) 的路徑我們都把它拆成兩半
- 拆成兩半的那一個點如果是 (x, y)，那麼也就是說現在這一條路徑等價於先從 $(1, 1)$ 走到 (x, y)，再從 (x, y) 走到 (n, m)
- 折半枚舉最重要過程在於枚舉的部分被分成兩半之後能夠快速的匹配答案
- 我們先枚舉從 $(1, 1)$ 走到 (x, y) 的部分，並將路徑上所有數字 `xor` 起來的結果用一個資料結構存起來（像是二維 `map`）
- 定義 `mp[x][y][val]` 表示從 $(1, 1)$ 走到 (x, y) 路徑上所有數字 `xor` 起來的結果為 `val` 的走法有 `mp[x][y][val]` 種

折半枚舉 Meet in the middle

- 如果一個網格是 $n \times m$，那麼總共會需要走 $n + m - 1$ 步
- 極限時 $20 + 20 - 1 = 39$
- 每一條從 $(1, 1)$ 走到 (n, m) 的路徑我們都把它拆成兩半
- 拆成兩半的那一個點如果是 (x, y)，那麼也就是說現在這一條路徑等價於先從 $(1, 1)$ 走到 (x, y)，再從 (x, y) 走到 (n, m)
- 折半枚舉最重要過程在於枚舉的部分被分成兩半之後能夠快速的匹配答案
- 我們先枚舉從 $(1, 1)$ 走到 (x, y) 的部分，並將路徑上所有數字 xor 起來的結果用一個資料結構存起來（像是二維 map）
- 定義 mp[x][y][val] 表示從 $(1, 1)$ 走到 (x, y) 路徑上所有數字 xor 起來的結果為 val 的走法有 mp[x][y][val] 種
- 所以一開始我們就先枚舉前 $\frac{(n+m-1)}{2}$ 步的所有結果，並把結果記錄在 map 裡面

折半枚舉 Meet in the middle

- 所以一開始我們就先枚舉前 $\frac{(n+m-1)}{2}$ 步的所有結果，並把結果記錄在 map 裡面
- 接下來剩下的 $(n+m-1)-\frac{(n+m-1)}{2}$ 步我們從 (n,m) 往回走

折半枚舉 Meet in the middle

- 所以一開始我們就先枚舉前 $\frac{(n+m-1)}{2}$ 步的所有結果，並把結果記錄在 map 裡面
- 接下來剩下的 $(n+m-1)-\frac{(n+m-1)}{2}$ 步我們從 (n,m) 往回走
- 往回走時，把該走的都走完的時候，如果最後停留的點是 (x,y)，路徑上所有數字 xor 起來的結果是 P，那麼我們就必須要知道共有幾種走法是從 $(1,1)$ 走到 (x,y) 且路徑上所有數字 xor 起來的結果必須跟 P xor 起來等於 k

折半枚舉 Meet in the middle

- 所以一開始我們就先枚舉前 $\frac{(n+m-1)}{2}$ 步的所有結果，並把結果記錄在 map 裡面
- 接下來剩下的 $(n+m-1)-\frac{(n+m-1)}{2}$ 步我們從 (n,m) 往回走
- 往回走時，把該走的都走完的時候，如果最後停留的點是 (x,y)，路徑上所有數字 xor 起來的結果是 P，那麼我們就必須要知道共有幾種走法是從 $(1,1)$ 走到 (x,y) 且路徑上所有數字 xor 起來的結果必須跟 P xor 起來等於 k
- 換句話說我們要找到 $(1,1)$ 走到 (x,y) 有幾種走法路徑上所有數字 xor 起來的結果等於 T，這個 T 必須滿足 T xor $P=k$

折半枚舉 Meet in the middle

- 所以一開始我們就先枚舉前 $\frac{(n+m-1)}{2}$ 步的所有結果，並把結果記錄在 map 裡面
- 接下來剩下的 $(n+m-1)-\frac{(n+m-1)}{2}$ 步我們從 (n,m) 往回走
- 往回走時，把該走的都走完的時候，如果最後停留的點是 (x,y)，路徑上所有數字 xor 起來的結果是 P，那麼我們就必須要知道共有幾種走法是從 $(1,1)$ 走到 (x,y) 且路徑上所有數字 xor 起來的結果必須跟 P xor 起來等於 k
- 換句話說我們要找到 $(1,1)$ 走到 (x,y) 有幾種走法路徑上所有數字 xor 起來的結果等於 T，這個 T 必須滿足 T xor $P=k$
- xor 有一個特性，如果 T xor $P=k$ 則 $T=P$ xor k
- 那接下來我們就只要直接拿之前先存在 map 裡面的 mp[x][y][P xor k] 就可以知道前半部有幾條路徑可以跟這條後半部的路徑匹配了，將匹配成功的數量記錄在答案上，這題就完成了