ZeroJudge f579 1. 購物車

ZeroJudge f579 1. 購物車

https://zerojudge.tw/ShowProblem?problemid=f579

109/07/04 APCS #1 近似題

丁培毅

110/08

問題描述

- 有一個商場中每台購物車都有安裝感應器，能夠得知每位顧客放入或拿出購物車的商品以及件數

問題描述

- 有一個商場中每台購物車都有安裝感應器，能夠得知每位顧客放入或拿出購物車的商品以及件數
- 感應器以正整數 **x** 表示編號 **x** 的商品被放入購物車，以負整數 **-x** 代表編號 **x** 的商品被移出購物車

問題描述

- 有一個商場中每台購物車都有安裝感應器，能夠得知每位顧客放入或拿出購物車的商品以及件數
- 感應器以正整數 **x** 表示編號 **x** 的商品被放入購物車，以負整數 **-x** 代表編號 **x** 的商品被移出購物車
- **輸**入的第一列是兩個正整數 **a, b**, $1 \le a,b \le 100$, 這是指定要觀察的兩個商品的編號

Sample Input
1 8

問題描述

- 有一個商場中每台購物車都有安裝感應器，能夠得知每位顧客放入或拿出購物車的商品以及件數
- 感應器以正整數 x 表示編號 x 的商品被放入購物車，以負整數 -x 代表編號 x 的商品被移出購物車
- **輸**入的第一列是兩個正整數 **a, b**, 1≤a,b≤100, 這是指定要觀察的兩個商品的編號，第二列的正整數 **n**, 1≤n≤100 代表顧客人數

Sample Input
1 8
5

問題描述

- 有一個商場中每台購物車都有安裝感應器，能夠得知每位顧客放入或拿出購物車的商品以及件數
- 感應器以正整數 x 表示編號 x 的商品被放入購物車，以負整數 -x 代表編號 x 的商品被移出購物車
- 輸入的第一列是兩個正整數 a, b, 1≤a,b≤100, 這是指定要觀察的兩個商品的編號，第二列的正整數 n, 1≤n≤100 代表顧客人數，接下來有 n 列，每一列有一連串的整數，代表放入或是取出購物車的商品，每一列最後一個數字都是 0，表示該位顧客的購物紀錄結束。

Sample Input
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0

問題描述

- 有一個商場中每台購物車都有安裝感應器，能夠得知每位顧客放入或拿出購物車的商品以及件數
- 感應器以正整數 x 表示編號 x 的商品被放入購物車，以負整數 -x 代表編號 x 的商品被移出購物車
- **輸入**的第一列是兩個**正整數 a, b**, 1≤a,b≤100, 這是指定要觀察的兩個商品的編號，第二列的正整數 **n**, 1≤n≤100 代表顧客人數，接下來有 n 列，每一列有一連串的整數，代表放入或是取出購物車的商品，每一列最後一個數字都是 0，表示該位顧客的購物紀錄結束。
- **輸出**一個整數，表示有幾位顧客同時購買商品 a 與商品 b

Sample Input
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0

Sample Output
2

問題描述

- 有一個商場中每台購物車都有安裝感應器，能夠得知每位顧客放入或拿出購物車的商品以及件數
- 感應器以正整數 x 表示編號 x 的商品被放入購物車，以負整數 -x 代表編號 x 的商品被移出購物車
- **輸入**的第一列是兩個**正整數 a, b**, 1≤a,b≤100, 這是指定要觀察的兩個商品的編號，第二列的正整數 **n**, 1≤n≤100 代表顧客人數，接下來有 n 列，每一列有一連串的整數，代表放入或是取出購物車的商品，每一列最後一個數字都是 0，表示該位顧客的購物紀錄結束。
- **輸出**一個整數，表示有幾位顧客同時購買商品 a 與商品 b

Sample Input
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0

Sample Output
2

Sample Input
3 9
2
3 9 -3 3 9 0
3 3 -3 -3 9 0

問題描述

- 有一個商場中每台購物車都有安裝感應器，能夠得知每位顧客放入或拿出購物車的商品以及件數
- 感應器以正整數 x 表示編號 x 的商品被放入購物車，以負整數 -x 代表編號 x 的商品被移出購物車
- **輸入**的第一列是兩個**正整數 a, b**, 1≤a,b≤100, 這是指定要觀察的兩個商品的編號，第二列的正整數 **n**, 1≤n≤100 代表顧客人數，接下來有 n 列，每一列有一連串的整數，代表放入或是取出購物車的商品，每一列最後一個數字都是 0，表示該位顧客的購物紀錄結束。
- **輸出**一個整數，表示有**幾位**顧客同時購買商品 a 與商品 b

Sample Input
```
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0
```

Sample Output
```
2
```

Sample Input
```
3 9
2
3 9 -3 3 9 0
3 3 -3 -3 9 0
```

Sample Output
```
1
```

問題分析與基本解法

- 「簡化問題」：簡化過程中常常可以讓你更清楚地看到問題的根源，也看到基本的解法

問題分析與基本解法

- 「簡化問題」：簡化過程中常常可以讓你更清楚地看到問題的根源，也看到基本的解法
- 怎麼簡化呢? 這是大家一定會覺得困惑的，最直接的就是把看到的數字變小、變少、省略要求

問題分析與基本解法

- 「簡化問題」：簡化過程中常常可以讓你更清楚地看到問題的根源，也看到基本的解法
- 怎麼簡化呢? 這是大家一定會覺得困惑的，最直接的就是把看到的數字變小、變少、省略要求
- 例如 **1.** 只觀察一個商品 **2.** 不考慮移除商品 **3.**只有一個顧客
  也就是右上方範例輸入裡的藍字部份

Sample Input
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0

問題分析與基本解法

Sample Input
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0

- 「簡化問題」：簡化過程中常常可以讓你更清楚地看到問題的根源，也看到基本的解法
- 怎麼簡化呢? 這是大家一定會覺得困惑的，最直接的就是把看到的數字變小、變少、省略要求
- 例如 **1.** 只觀察一個商品 **2.** 不考慮移除商品 **3.**只有一個顧客
  也就是右上方範例輸入裡的藍字部份
- 處理的方法就變得比較直接了：

問題分析與基本解法

Sample Input
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0

- 「簡化問題」：簡化過程中常常可以讓你更清楚地看到問題的根源，也看到基本的解法
- 怎麼簡化呢? 這是大家一定會覺得困惑的，最直接的就是把看到的數字變小、變少、省略要求
- 例如 **1.** 只觀察一個商品 **2.** 不考慮移除商品 **3.**只有一個顧客
  也就是右上方範例輸入裡的藍字部份
- 處理的方法就變得比較直接了：

  1. 讀入需要觀察的商品編號....**1**

問題分析與基本解法

Sample Input
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0

- 「簡化問題」：簡化過程中常常可以讓你更清楚地看到問題的根源，也看到基本的解法
- 怎麼簡化呢? 這是大家一定會覺得困惑的，最直接的就是把看到的數字變小、變少、省略要求
- 例如 **1.** 只觀察一個商品 **2.** 不考慮移除商品 **3.**只有一個顧客
  也就是右上方範例輸入裡的藍字部份
- 處理的方法就變得比較直接了：
  1. 讀入需要觀察的商品編號....**1**
  2. 以一個 while 迴圈讀入顧客所有放進購物車的商品編號... **1 8 0**

問題分析與基本解法

Sample Input
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0

- 「簡化問題」：簡化過程中常常可以讓你更清楚地看到問題的根源，也看到基本的解法
- 怎麼簡化呢? 這是大家一定會覺得困惑的，最直接的就是把看到的數字變小、變少、省略要求
- 例如 **1.** 只觀察一個商品 **2.** 不考慮移除商品 **3.**只有一個顧客
  也就是右上方範例輸入裡的藍字部份
- 處理的方法就變得比較直接了：

  1. 讀入需要觀察的商品編號....**1**
  2. 以一個 while 迴圈讀入顧客所有放進購物車的商品編號... **1 8 0**

     迴圈結束的條件是讀到的商品編號為 **0**

問題分析與基本解法

Sample Input
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0

- 「簡化問題」：簡化過程中常常可以讓你更清楚地看到問題的根源，也看到基本的解法
- 怎麼簡化呢? 這是大家一定會覺得困惑的，最直接的就是把看到的數字變小、變少、省略要求
- 例如 **1.** 只觀察一個商品 **2.** 不考慮移除商品 **3.**只有一個顧客
  也就是右上方範例輸入裡的藍字部份
- 處理的方法就變得比較直接了：

  1. 讀入需要觀察的商品編號....**1**
  2. 以一個 while 迴圈讀入顧客所有
     放進購物車的商品編號... **1 8 0**

     迴圈結束的條件是讀到的商品編號為 **0**

     比對每一個讀到的商品編號和步驟 1 得到的指定商品編號

問題分析與基本解法

Sample Input
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0

- 「簡化問題」：簡化過程中常常可以讓你更清楚地看到問題的根源，也看到基本的解法
- 怎麼簡化呢? 這是大家一定會覺得困惑的，最直接的就是把看到的數字變小、變少、省略要求
- 例如 **1.** 只觀察一個商品 **2.** 不考慮移除商品 **3.**只有一個顧客
  也就是右上方範例輸入裡的藍字部份

Sample Output
yes

- 處理的方法就變得比較直接了：

1. 讀入需要觀察的商品編號....**1**
2. 以一個 while 迴圈讀入顧客所有
   放進購物車的商品編號... **1 8 0**

   迴圈結束的條件是讀到的商品編號為 **0**

   比對每一個讀到的商品編號和步驟 1 得到的指定商品編號

   此顧客有購買一次指定商品就輸出一個 **yes**

問題分析與基本解法

Sample Input
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0

- 「簡化問題」：簡化過程中常常可以讓你更清楚地看到問題的根源，也看到基本的解法
- 怎麼簡化呢? 這是大家一定會覺得困惑的，最直接的就是把看到的數字變小、變少、省略要求
- 例如 **1.** 只觀察一個商品 **2.** 不考慮移除商品 **3.**只有一個顧客
  也就是右上方範例輸入裡的藍字部份

Sample Output
yes

- 處理的方法就變得比較直接了：

1. 讀入需要觀察的商品編號....**1**
2. 以一個 while 迴圈讀入顧客所有放進購物車的商品編號... **1 8 0**

```
1 int a, x;
2 scanf("%d", &a);
3 while ((1==scanf("%d", &x))
4           && (x!=0))
5    if (x==a) printf("yes\n");
```

迴圈結束的條件是讀到的商品編號為 **0**

比對每一個讀到的商品編號和步驟 1 得到的指定商品編號

此顧客有購買一次指定商品就輸出一個 **yes**

擴充問題並逐步延伸解法

- 觀察兩種商品，顧客同時購買此兩種商品時印出 yes

Sample Input

```
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0
```

Sample Output

```
yes
```

擴充問題並逐步延伸解法

- 觀察兩種商品，顧客同時購買此兩種商品時印出 yes

1. 讀取資料的地方很容易擴充成讀取兩個資料

Sample Input

```
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0
```

Sample Output

```
yes
```

擴充問題並逐步延伸解法

- 觀察兩種商品，顧客同時購買此兩種商品時印出 yes

```
1 int a, b, x;
2 scanf("%d%d", &a, &b);
3 while ((1==scanf("%d", &x))
4             && (x!=0))
5     if (x==a) printf("a\n");
6     else if (x==b) printf("b\n");
```

Sample Input

```
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0
```

Sample Output

```
yes
```

1. 讀取資料的地方很容易擴充成讀取兩個資料

擴充問題並逐步延伸解法

- 觀察兩種商品，顧客同時購買此兩種商品時印出 yes

```
1 int a, b, x;
2 scanf("%d%d", &a, &b);
3 while ((1==scanf("%d", &x))
4           && (x!=0))
5     if (x==a) printf("a\n");
6     else if (x==b) printf("b\n");
```

Sample Input

```
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0
```

Sample Output

```
yes
```

1. 讀取資料的地方很容易擴充成讀取兩個資料
2. 但是怎樣才能夠知道顧客有沒有同時購買指定的兩種商品呢?

擴充問題並逐步延伸解法

- 觀察兩種商品，顧客同時購買此兩種商品時印出 yes

```
1 int a, b, x;
2 scanf("%d%d", &a, &b);
3 while ((1==scanf("%d", &x))
4           && (x!=0))
5     if (x==a) printf("a\n");
6     else if (x==b) printf("b\n");
```

Sample Input

```
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0
```

Sample Output

```
yes
```

1. 讀取資料的地方很容易擴充成讀取兩個資料
2. 但是怎樣才能夠知道顧客有沒有同時購買指定的兩種商品呢?
3. 在讀進一個商品的時候，只能夠檢查它是 a 還是 b

擴充問題並逐步延伸解法

- 觀察兩種商品，顧客同時購買此兩種商品時印出 yes

```
1 int a, b, x;
2 scanf("%d%d", &a, &b);
3 while ((1==scanf("%d", &x))
4           && (x!=0))
5     if (x==a) printf("a\n");
6     else if (x==b) printf("b\n");
```

Sample Input

```
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0
```

Sample Output

```
yes
```

1. 讀取資料的地方很容易擴充成讀取兩個資料
2. 但是怎樣才能夠知道顧客有沒有同時購買指定的兩種商品呢?
3. 在讀進一個商品的時候，只能夠檢查它是 a 還是 b，要判斷是不是商品 a 和商品 b 都在購物車中，需要把購物車裡的商品全部清點一次

擴充問題並逐步延伸解法

- 觀察兩種商品，顧客同時購買此兩種商品時印出 yes

```
1 int a, b, x;
2 scanf("%d%d", &a, &b);
3 while ((1==scanf("%d", &x))
4           && (x!=0))
5     if (x==a) printf("a\n");
6     else if (x==b) printf("b\n");
```

Sample Input

```
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0
```

Sample Output

```
yes
```

1. 讀取資料的地方很容易擴充成讀取兩個資料
2. 但是怎樣才能夠知道顧客有沒有同時購買指定的兩種商品呢?
3. 在讀進一個商品的時候，只能夠檢查它是 a 還是 b，要判斷是不是商品 a 和商品 b 都在購物車中，需要把購物車裡的商品全部清點一次，需要一張紙來記錄商品 **a** 和商品 **b** 的個數

擴充問題並逐步延伸解法

- 觀察兩種商品，顧客同時購買此兩種商品時印出 yes

```
1 int a, b, x;
2 scanf("%d%d", &a, &b);
3 while ((1==scanf("%d", &x))
4           && (x!=0))
5     if (x==a) printf("a\n");
6     else if (x==b) printf("b\n");
```

Sample Input

```
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0
```

Sample Output

```
yes
```

1. 讀取資料的地方很容易擴充成讀取兩個資料
2. 但是怎樣才能夠知道顧客有沒有同時購買指定的兩種商品呢?
3. 在讀進一個商品的時候，只能夠檢查它是 a 還是 b，要判斷是不是商品 a 和商品 b 都在購物車中，需要把購物車裡的商品全部清點一次，需要一張紙來記錄商品 **a** 和商品 **b** 的個數，清點完以後如果兩個商品的個數都大於 0 才印出 yes

擴充問題並逐步延伸解法

- 觀察兩種商品，顧客同時購買此兩種商品時印出 yes

```
1 int a, b, x;
2 scanf("%d%d", &a, &b);
3 while ((1==scanf("%d", &x))
4           && (x!=0))
5     if (x==a) printf("a\n");
6     else if (x==b) printf("b\n");
```

Sample Input

```
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0
```

Sample Output

```
yes
```

1. 讀取資料的地方很容易擴充成讀取兩個資料
2. 但是怎樣才能夠知道顧客有沒有同時購買指定的兩種商品呢?
3. 在讀進一個商品的時候，只能夠檢查它是 a 還是 b，要判斷是不是商品 a 和商品 b 都在購物車中，需要把購物車裡的商品全部清點一次，需要一張紙來記錄商品 **a** 和商品 **b** 的個數，清點完以後如果兩個商品的個數都大於 0 才印出 yes

```
01 int a, b, x, counta=0, countb=0;
02 scanf("%d%d", &a, &b);
03 while ((1==scanf("%d", &x))
04           && (x!=0))
05     if (x==a)
06         counta++;
07     else if (x==b)
08         countb++;
09 if ((counta>0)&&(countb>0))
10     printf("yes\n");
```

n 個顧客

- 題目擴充到 n 個顧客的時候，我們要檢查每一個顧客是不是同時購買 a 和 b 兩種商品，同時輸出有幾個顧客滿足這個條件

n 個顧客

$1 \le a,b,n \le 100$

- 題目擴充到 n 個顧客的時候，我們要檢查每一個顧客是不是同時購買 a 和 b 兩種商品，同時輸出有幾個顧客滿足這個條件

Sample Input

```
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0
```

Sample Output

```
2
```

n 個顧客

$1 \le a,b,n \le 100$

- 題目擴充到 n 個顧客的時候，我們要檢查每一個顧客是不是同時購買 a 和 b 兩種商品，同時輸出有幾個顧客滿足這個條件
- 在前一個程式裡，增加讀取 n 的敘述，然後增加一個迴圈來處理每一個顧客的購物車內容，並且增加一個變數 m 來紀錄有幾個顧客滿足條件

```
01 int a, b, x, counta, countb, n, m=0, i;
02 scanf("%d%d%d", &a, &b, &n);
03 for (i=0; i<n; i++) {
04     counta = countb = 0;
05     while ((1==scanf("%d", &x)) && (x!=0)) {
06         counta += (x==a);
07         countb += (x==b);
08     }
09     m += (counta>0)&&(countb>0);
10 }
11 printf("%d\n", m);
```

Sample Input

```
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0
```

Sample Output

```
2
```

n 個顧客

1≤ a,b,n ≤100

- 題目擴充到 n 個顧客的時候，我們要檢查每一個顧客是不是同時購買 a 和 b 兩種商品，同時輸出有幾個顧客滿足這個條件
- 在前一個程式裡，增加讀取 n 的敘述，然後增加一個迴圈來處理每一個顧客的購物車內容，並且增加一個變數 m 來紀錄有幾個顧客滿足條件

Sample Input
```
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0
```

Sample Output
```
2
```

```
01 int a, b, x, counta, countb, n, m=0, i;
02 scanf("%d%d%d", &a, &b, &n);
03 for (i=0; i<n; i++) {
04     counta = countb = 0;
05     while ((1==scanf("%d", &x)) && (x!=0)) {
06         counta += (x==a);
07         countb += (x==b);
08     }
09     m += (counta>0)&&(countb>0);
10 }
11 printf("%d\n", m);
```

注意: 更換一種程式寫法
在 C/C++ 程式裡
關係與邏輯運算結果
為 1 代表「真」;
為 0 代表「偽」

n 個顧客

1≤ a,b,n ≤100

- 題目擴充到 n 個顧客的時候，我們要檢查每一個顧客是不是同時購買 a 和 b 兩種商品，同時輸出有幾個顧客滿足這個條件
- 在前一個程式裡，增加讀取 n 的敘述，然後增加一個迴圈來處理每一個顧客的購物車內容，並且增加一個變數 m 來紀錄有幾個顧客滿足條件

Sample Input
```
1 8
5
1 8 0
5 6 0
2 7 0
8 1 0
33 22 0
```

Sample Output
```
2
```

```
01 int a, b, x, counta, countb, n, m=0, i;
02 scanf("%d%d%d", &a, &b, &n);
03 for (i=0; i<n; i++) {
04     counta = countb = 0;
05     while ((1==scanf("%d", &x)) && (x!=0)) {
06         counta += (x==a);
07         countb += (x==b);
08     }
09     m += (counta>0)&&(countb>0);
10 }
11 printf("%d\n", m);
```

注意: 更換一種程式寫法
在 C/C++ 程式裡
關係與邏輯運算結果
為 1 代表「真」;
為 0 代表「偽」

注意: a 與 b 萬一相等時
程式表現仍然合理

購物車中可移除商品

- 題目再擴充「負整數 -x 代表編號 x 的商品被移出購物車」

購物車中可移除商品

- 題目再擴充「負整數 -x 代表編號 x 的商品被移出購物車」

Sample Input

```
3 9
2
3 9 -3 3 9 0
3 3 -3 -3 9 0
```

Sample Output

```
1
```

購物車中可移除商品

- 題目再擴充「負整數 -x 代表編號 x 的商品被移出購物車」
- 擴充這個功能時，只需要修改一點點程式，需要比對 **–a** 及 **–b**，滿足時將商品個數減 **1**

Sample Input
3 9
2
3 9 -3 3 9 0
3 3 -3 -3 9 0

Sample Output
1

購物車中可移除商品

- 題目再擴充「負整數 -x 代表編號 x 的商品被移出購物車」
- 擴充這個功能時，只需要修改一點點程式，需要比對 **–a** 及 **–b**，滿足時將商品個數減 **1**

Sample Input
3 9
2
3 9 -3 3 9 0
3 3 -3 -3 9 0

Sample Output
1

```
01 int a, b, x, counta, countb, n, m=0, i;
02 scanf("%d%d%d", &a, &b, &n);
03 for (i=0; i<n; i++) {
04     counta = countb = 0;
05     while ((1==scanf("%d", &x)) && (x!=0)) {
06         counta += (x==a); counta -= (x==-a);
07         countb += (x==b); countb -= (x==-b);
08     }
09     m += (counta>0)&&(countb>0);
10 }
11 printf("%d\n", m);
```

購物車中可移除商品

- 題目再擴充「負整數 -x 代表編號 x 的商品被移出購物車」
- 擴充這個功能時，只需要修改一點點程式，需要比對 **–a** 及 **–b**，滿足時將商品個數減 **1**

Sample Input
3 9
2
3 9 -3 3 9 0
3 3 -3 -3 9 0

Sample Output
1

```
01 int a, b, x, counta, countb, n, m=0, i;
02 scanf("%d%d%d", &a, &b, &n);
03 for (i=0; i<n; i++) {
04     counta = countb = 0;
05     while ((1==scanf("%d", &x)) && (x!=0)) {
06         counta += (x==a); counta -= (x==-a);
07         countb += (x==b); countb -= (x==-b);
08     }
09     m += (counta>0)&&(countb>0);
10 }
11 printf("%d\n", m);
```

或是
if (x==-a) counta--;

如果指定觀察多種商品

- 再擴充一點，假設指定觀察 **k** 種商品，想要知道有購買其中 3 種以上商品的顧客數量

如果指定觀察多種商品

- 再擴充一點，假設指定觀察 **k** 種商品，想要知道有購買其中 3 種以上商品的顧客數量

Sample Input
7
3 21 9 8 2 7 18
2
3 9 -3 3 9 8 8 0
3 3 -3 -3 9 7 12 4 0

Sample Output
1

如果指定觀察多種商品

- 再擴充一點，假設指定觀察 **k** 種商品，想要知道有購買其中 3 種以上商品的顧客數量
- 如果延續前面的解題方法，使用「if 敘述檢查」來製作程式，會遇見兩個問題：**1.** k 不是固定的數值，**2.** k 值很大導致程式很長

Sample Input
7
3 21 9 8 2 7 18
2
3 9 -3 3 9 8 8 0
3 3 -3 -3 9 7 12 4 0

Sample Output
1

如果指定觀察多種商品

- 再擴充一點，假設指定觀察 **k** 種商品，想要知道有購買其中 3 種以上商品的顧客數量
- 如果延續前面的解題方法，使用「if 敘述檢查」來製作程式，會遇見兩個問題：**1.** k 不是固定的數值，**2.** k 值很大導致程式很長
- 這個時候又出現了「稀疏的對照表」可以
  1. 把 **k** 種商品編號先讀進陣列裡

Sample Input
7
3 21 9 8 2 7 18
2
3 9 -3 3 9 8 8 0
3 3 -3 -3 9 7 12 4 0

Sample Output
1

如果指定觀察多種商品

- 再擴充一點，假設指定觀察 **k** 種商品，想要知道有購買其中 3 種以上商品的顧客數量
- 如果延續前面的解題方法，使用「if 敘述檢查」來製作程式，會遇見兩個問題：**1.** k 不是固定的數值，**2.** k 值很大導致程式很長
- 這個時候又出現了「稀疏的對照表」可以
  1. 把 **k** 種商品編號先讀進陣列裡
  2. 把資料排序

Sample Input
7
3 21 9 8 2 7 18
2
3 9 -3 3 9 8 8 0
3 3 -3 -3 9 7 12 4 0

Sample Output
1

如果指定觀察多種商品

- 再擴充一點，假設指定觀察 **k** 種商品，想要知道有購買其中 3 種以上商品的顧客數量
- 如果延續前面的解題方法，使用「if 敘述檢查」來製作程式，會遇見兩個問題：**1.** k 不是固定的數值，**2.** k 值很大導致程式很長
- 這個時候又出現了「稀疏的對照表」可以
  1. 把 **k** 種商品編號先**讀進陣列**裡
  2. 把資料**排序**
  3. 每次讀進一個 x 時，用**二分法快速確定**是這 **k** 種裡哪一種

Sample Input
7
3 21 9 8 2 7 18
2
3 9 -3 3 9 8 8 0
3 3 -3 -3 9 7 12 4 0

Sample Output
1

如果指定觀察多種商品

- 再擴充一點，假設指定觀察 **k** 種商品，想要知道有購買其中 3 種以上商品的顧客數量
- 如果延續前面的解題方法，使用「if 敘述檢查」來製作程式，會遇見兩個問題：**1.** k 不是固定的數值，**2.** k 值很大導致程式很長
- 這個時候又出現了「稀疏的對照表」可以
  1. 把 **k** 種商品編號先**讀進陣列**裡
  2. 把資料**排序**
  3. 每次讀進一個 x 時，用**二分法快速確定**是這 **k** 種裡哪一種
  4. **購買商品的數量**用一個 **k** 個元素的陣列來紀錄

Sample Input
7
3 21 9 8 2 7 18
2
3 9 -3 3 9 8 8 0
3 3 -3 -3 9 7 12 4 0

Sample Output
1

如果指定觀察多種商品

- 再擴充一點，假設指定觀察 **k** 種商品，想要知道有購買其中 3 種以上商品的顧客數量
- 如果延續前面的解題方法，使用「if 敘述檢查」來製作程式，會遇見兩個問題：**1.** k 不是固定的數值，**2.** k 值很大導致程式很長
- 這個時候又出現了「稀疏的對照表」可以
  1. 把 **k** 種商品編號先**讀進陣列**裡
  2. 把資料**排序**
  3. 每次讀進一個 x 時，用**二分法快速確定**是這 **k** 種裡哪一種
  4. **購買商品的數量**用一個 **k** 個元素的陣列來紀錄
     - C 的 **stdlib.h** 裡面的 **bsearch()**

Sample Input
7
3 21 9 8 2 7 18
2
3 9 -3 3 9 8 8 0
3 3 -3 -3 9 7 12 4 0

Sample Output
1

如果指定觀察多種商品

- 再擴充一點，假設指定觀察 **k** 種商品，想要知道有購買其中 3 種以上商品的顧客數量
- 如果延續前面的解題方法，使用「if 敘述檢查」來製作程式，會遇見兩個問題：**1.** k 不是固定的數值，**2.** k 值很大導致程式很長
- 這個時候又出現了「稀疏的對照表」可以
  1. 把 **k** 種商品編號先**讀進陣列**裡
  2. 把資料**排序**
  3. 每次讀進一個 x 時，用**二分法快速確定**是這 **k** 種裡哪一種
  4. **購買商品的數量**用一個 **k** 個元素的陣列來紀錄
     - C 的 **stdlib.h** 裡面的 **bsearch()**
     - C++ 的 **algorithm** 裡面的 **lower_bound()**

Sample Input
7
3 21 9 8 2 7 18
2
3 9 -3 3 9 8 8 0
3 3 -3 -3 9 7 12 4 0

Sample Output
1

如果指定觀察多種商品

- 再擴充一點，假設指定觀察 **k** 種商品，想要知道有購買其中 3 種以上商品的顧客數量
- 如果延續前面的解題方法，使用「if 敘述檢查」來製作程式，會遇見兩個問題：**1.** k 不是固定的數值，**2.** k 值很大導致程式很長
- 這個時候又出現了「稀疏的對照表」可以
  1. 把 **k** 種商品編號先**讀進陣列**裡
  2. 把資料**排序**
  3. 每次讀進一個 x 時，用**二分法快速確定**是這 **k** 種裡哪一種
  4. **購買商品的數量**用一個 **k** 個元素的陣列來紀錄
     - C 的 **stdlib.h** 裡面的 **bsearch()**
     - C++ 的 **algorithm** 裡面的 **lower_bound()**
     - 自己撰寫等效的 **lower_bound()**

Sample Input
7
3 21 9 8 2 7 18
2
3 9 -3 3 9 8 8 0
3 3 -3 -3 9 7 12 4 0

Sample Output
1

```
01 int k, i, v[100], c[100], *ptr, x, y, n, m=0, sum;
02 scanf("%d", &k);
03 for (i=0; i<k; i++)
04     scanf("%d", &v[i]);
05 qsort(v, k, sizeof(int), comp);
06 scanf("%d", &n);
07 while (n--) {
08     for (i=0; i<k; i++) c[i] = 0;
09     while ((1==scanf("%d", &x)) && (x!=0)) {
10         y=x; if (y<0) y=-x;
11         if ((ptr=(int*)bsearch(&y, v, k, sizeof(int), comp))!=NULL)
12             c[ptr-v] += 2*(x>0)-1;
13     }
14     for (i=sum=0; i<k&&sum<3; i++)
15         sum += c[i]>0;
16     m += sum==3;
17 }
```

ple Input

1 9 8 2 7 18

3 3 9 8 8 0
3 -3 9 7 12 4 0

ple Output

重裡哪一種

4. 購買商品的數量用一個 k 個元素的陣列來紀錄

- C 的 **stdlib.h** 裡面的 **bsearch()**
- C++ 的 **algorithm** 裡面的 **lower_bound()**
- 自己撰寫等效的 **lower_bound()**

```
01 int k, i, v[100], c[100], *ptr, x, y, n, m=0, sum;
02 scanf("%d", &k);
03 for (i=0; i<k; i++)
04     scanf("%d", &v[i]);
05 sort(v, v+k);
06 scanf("%d", &n);
07 while (n--) {
08     for (i=0; i<k; i++) c[i] = 0;
09     while ((1==scanf("%d", &x)) && (x!=0)) {
10         y=x; if (y<0) y=-x;
11         if ((ptr=lower_bound(v, v+k, y))&&(*ptr==y))
12             c[ptr-v] += 2*(x>0)-1;
13     }
14     for (i=sum=0; i<k&&sum<3; i++)
15         sum += c[i]>0;
16     m += sum==3;
17 }
```

ple Input

1 9 8 2 7 18

3 3 9 8 8 0
3 -3 9 7 12 4 0

ple Output

NULL)

重裡哪一種

紀錄

- C 的 **stdlib.h** 裡面的 **bsearch()**
- C++ 的 **algorithm** 裡面的 **lower_bound()**
- 自己撰寫等效的 **lower_bound()**

```
01 int k, i, v[100], c[100], *ptr, x, y, n, m=0, sum;
02 scanf("%d", &k);
03 for (i=0; i<k; i++)
04     scanf("%d", &v[i]);
05 qsort(v, k, sizeof(int), comp);
06 scanf("%d", &n);
07 while (n--) {
08     for (i=0; i<k; i++) c[i] = 0;
09     while ((1==scanf("%d", &x)) && (x!=0)) {
10         y=x; if (y<0) y=-x;
11         if ((ptr=lower_bound(v, v+k, y))&&(*ptr==y))
12             c[ptr-v] += 2*(x>0)-1;
13     }
14     for (i=sum=0; i<k&&sum<3; i++)
15         sum += c[i]>0;
16     m += sum==3;
17 }
```

```
01 int *lower_bound(int *begin, int *end, int target) {
02     int mid, *result=end;
03     while (begin<end) {
04         mid = (end-begin)/2;
05         if (*(begin+mid)>=target)
06             result = end = begin+mid;
07         else
08             begin += mid+1;
09     }
10     return result;
11 }
```

ple Input

1 9 8 2 7 18

3 3 9 8 8 0
3 -3 9 7 12 4 0

NULL)

==y))

ple Output

- C 的 **stdlib.h** 裡
- C++ 的 **algorithm**
- 自己撰寫等效的 **l**

觀察多種商品 (續)

- 接下來非常重要的實作方法就是運用 STL 的 **map** 了

觀察多種商品 (續)

- 接下來非常重要的實作方法就是運用 STL 的 **map** 了

  如果要觀察的商品很多，編號又很分散，map 是很好的選擇

觀察多種商品 (續)

- 接下來非常重要的實作方法就是運用 STL 的 **map** 了

  如果要觀察的商品很多，編號又很分散，map 是很好的選擇

```
01 #include <cstdio>
02 #include <map>
03 using namespace std;
   …
```

觀察多種商品 (續)

- 接下來非常重要的實作方法就是運用 STL 的 **map** 了

  如果要觀察的商品很多，編號又很分散，map 是很好的選擇

```
01 #include <cstdio>
02 #include <map>
03 using namespace std;
   …
```

```
04 int k, i, v, x, y, n, m=0, sum;
05 map<int,int> c;
06 scanf("%d", &k);
07 for (i=0; i<k; i++)
08     scanf("%d", &v), c[v]=0;
```

觀察多種商品 (續)

- 接下來非常重要的實作方法就是運用 STL 的 **map** 了

  如果要觀察的商品很多，編號又很分散，map 是很好的選擇

```
01 #include <cstdio>
02 #include <map>
03 using namespace std;
   …
```

```
04 int k, i, v, x, y, n, m=0, sum;
05 map<int,int> c;
06 scanf("%d", &k);
07 for (i=0; i<k; i++)
08     scanf("%d", &v), c[v]=0;
09 scanf("%d", &n);
10 while (n--) {
11     for (auto &e:c) e.second=0;
19 }
```

觀察多種商品 (續)

- 接下來非常重要的實作方法就是運用 STL 的 **map** 了

  如果要觀察的商品很多，編號又很分散，map 是很好的選擇

```
01 #include <cstdio>
02 #include <map>
03 using namespace std;
   …
```

```
04 int k, i, v, x, y, n, m=0, sum;
05 map<int,int> c;
06 scanf("%d", &k);
07 for (i=0; i<k; i++)
08     scanf("%d", &v), c[v]=0;
09 scanf("%d", &n);
10 while (n--) {
11     for (auto &e:c) e.second=0;
12     while ((1==scanf("%d", &x)) && (x!=0)) {
13         y=x; if (y<0) y=-x;
14         if (c.find(y) != c.end())
15             c[y] += 2*(x>0)-1;
16     }


19 }
```

觀察多種商品 (續)

- 接下來非常重要的實作方法就是運用 STL 的 **map** 了

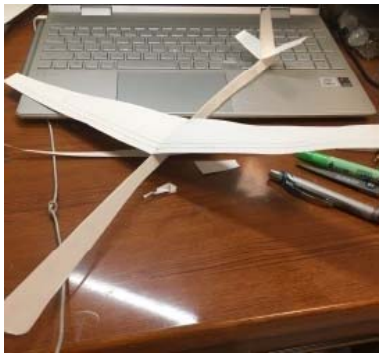
  如果要觀察的商品很多，編號又很分散，map 是很好的選擇

```
01 #include <cstdio>
02 #include <map>
03 using namespace std;
   …
```

```
04 int k, i, v, x, y, n, m=0, sum;
05 map<int,int> c;
06 scanf("%d", &k);
07 for (i=0; i<k; i++)
08     scanf("%d", &v), c[v]=0;
09 scanf("%d", &n);
10 while (n--) {
11     for (auto &e:c) e.second=0;
12     while ((1==scanf("%d", &x)) && (x!=0)) {
13         y=x; if (y<0) y=-x;
14         if (c.find(y) != c.end())
15             c[y] += 2*(x>0)-1;
16     }
17     sum = 0; for (auto &e:c) sum += e.second>0;
18     m += sum>=3;
19 }
```

看清楚題目

- APCS 很多模擬型式的題目，描述常常很長
- 題目都很會清楚地規範問題的規模，會影響到解題的方法



- 「東京工業大學紙飛機大賽」
  - 紙飛機大賽的規則：
    1. 以 2 張 A3 製圖紙以內的材料製作
    2. 只可使用製圖紙、膠水與配重物
    3. 配重物上限為 20 個迴紋針，超過即失去參賽資格
    4. 綜合得分將以飛行距離與飛行時間兩方面計算
    5. 飛行距離以機體完全停止的地面標示距離為準
    6. 飛行時間計算以碼表測量，以機體任何部位接觸到地面為止
    7. 不可使用外力發射裝置
    8. 時間與距離的綜合得分將以固定公式計算之

看清楚題目 (續)

- 紙飛機大賽的規則非常簡單，僅針對材料做出具體規範，外型上卻沒有任何限制

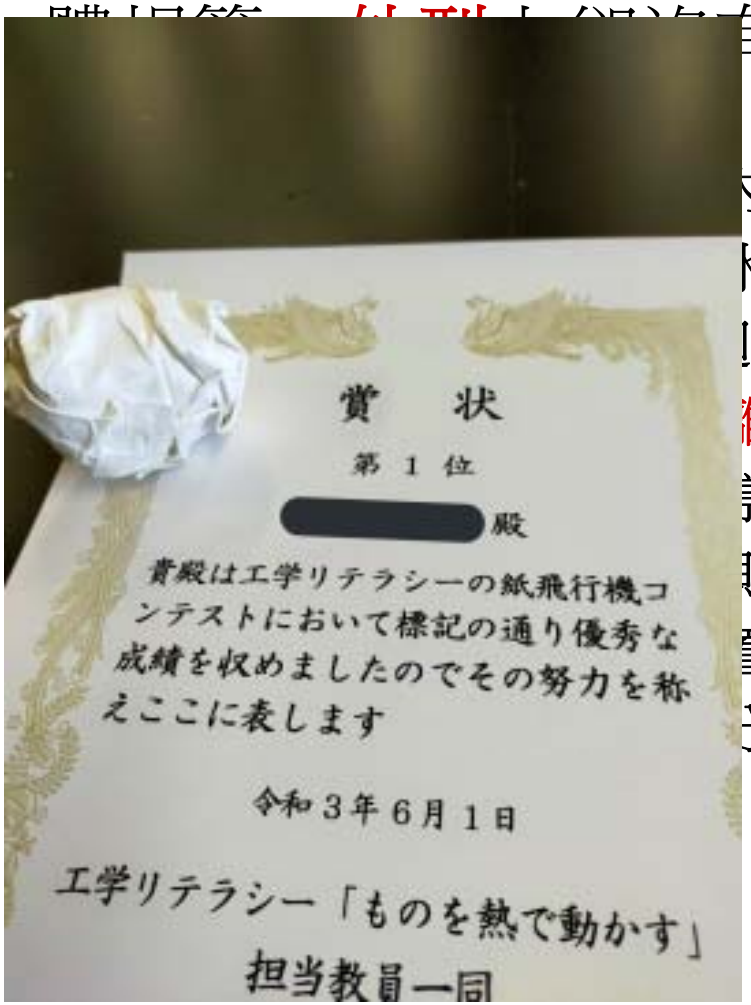
看清楚題目 (續)

- 紙飛機大賽的規則非常簡單，僅針對材料做出具體規範，外型上卻沒有任何限制

1. 以 **2** 張 **A3** 製圖紙以內的材料製作
2. 只可使用製圖紙、膠水與配重物
3. 配重物上限為 **20** 個迴紋針，超過即失去參賽資格
4. 綜合得分將以飛行距離與飛行時間兩方面計算
5. 飛行距離以機體完全停止的地面標示距離為準
6. 飛行時間計算以碼表測量，以機體任何部位接觸到地面為止
7. 不可使用外力發射裝置
8. 時間與距離的綜合得分將以固定公式計算之

看清楚題目 (續)

- 紙飛機大賽的規則非常簡單，僅針對材料做出具
  有任何限制



的材料製作

與配重物

紋針，超過即失去參賽資格

準與飛行時間兩方面計算

停止的地面標示距離為準

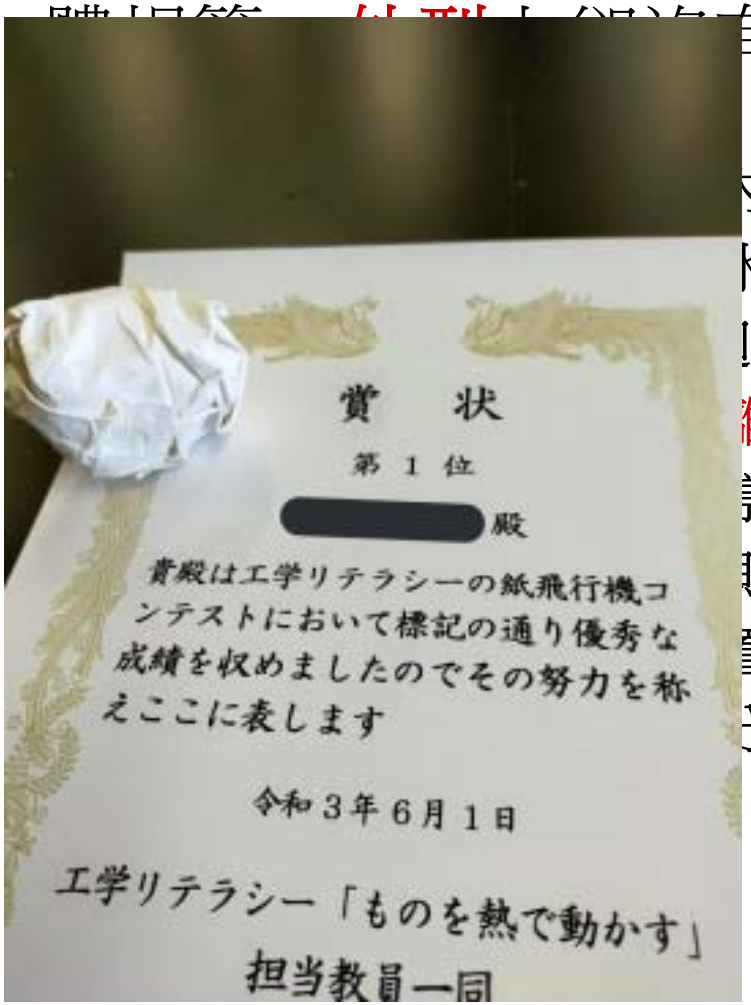
測量，以機體任何部位接觸到地面為止

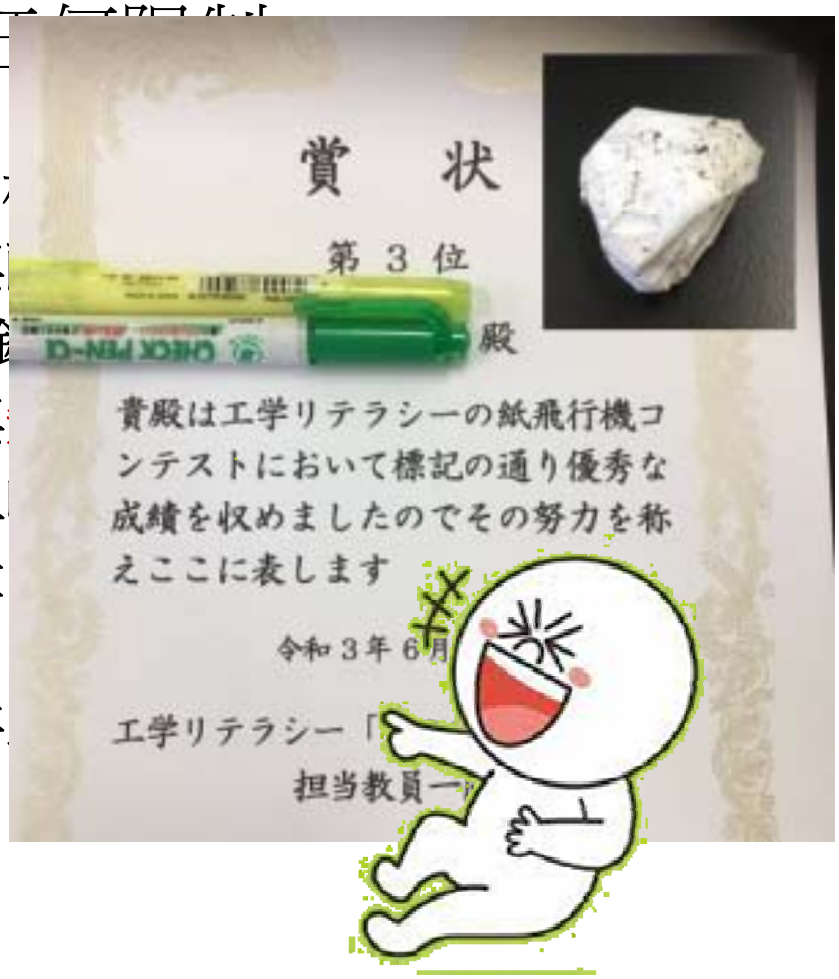
置

分將以固定公式計算之

看清楚題目 (續)

- 紙飛機大賽的規則非常簡單，僅針對材料做出具





看清楚題目 (續)

- 紙飛機大賽的規則非常簡單，僅針對材料做出具