

TOI推廣計畫

解題-滿意度調查



Icon made by [<https://www.flaticon.com/authors/flat-icons>] from www.flaticon.com

題目

專長統計學的林羊老師最近出了一個題目，要**算各種問卷調查滿意度(0~9)各選項的出現頻率**，進而探討不同行銷策略的制定方針。討厭數學的文組小姐看到一整串數字的時候總是會頭痛，因此想要追求她的宅男工程師決定**運用程式設計課所學**，一起算出問卷滿意度各選項出現的頻率，成功約她出去玩！

給定一連串數字，代表各題的調查滿意度，接著算出各選項出現的次數，最後由大至小排序各選項出現頻率。

請使用 *long long int* 型態宣告變數，以 *%lld* 讀入數字。

輸入格式

第一行有一個正整數 N ($1 \leq N \leq 2^{63} - 1$)，代表該問卷各題的調查滿意度。

輸出格式

對於每筆測資輸出 K ($1 \leq K \leq 10$ ，拆解後0 ~ 9有出現過的數字總數) 個正整數，代表數字出現頻率排序，由左至右、由高至低，頻率一樣則先輸出數字小者。彼此間以空白間隔。

輸入範例	輸出範例
2425264426558	2 4 5 6 8



解題重點:

1. 數字拆解

數字運算、存入陣列

2. 陣列操作

陣列頻率排序

Icon made by [<https://www.freepik.com/>]from www.flaticon.com



1. 數字拆解

數字運算、存入陣列

Icon made by [<https://www.flaticon.com/authors/flat-icons>] from www.flaticon.com

◆ 1. 數字拆解

int (32 bits) : $-2^{31} \sim 2^{31} - 1$

long long int (64bits) : $-2^{63} \sim 2^{63} - 1$

➤ 數字運算

- ① 讀取數字(long long int)
 - `scanf("%lld", &N);`
- ② 數字拆解(重複計算)
 - `while( N > 0 )`

取餘數%10 → 除以10

範例 : $N = 2425264$

$N > 0 \rightarrow N \% 10 = 4$ (242526**4**)

4 出現次數 + 1

算下一位 $N/10 = 242526$

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	0	0	1	0	0	0	0	0

◆ 1. 數字拆解

int (32 bits) : $-2^{31} \sim 2^{31} - 1$

long long int (64bits) : $-2^{63} \sim 2^{63} - 1$

➤ 數字運算

① 讀取數字(long long int)

- scanf("%lld", &N);

② 數字拆解(重複計算)

- while(N > 0)

取餘數%10 → 除以10

N = 242526

N > 0 → N%10 = 4 (24252**6**)

6 出現次數 + 1

算下一位 N/10 = 24252

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	0	0	1	0	1	0	0	0

◆ 1. 數字拆解

int (32 bits) : $-2^{31} \sim 2^{31} - 1$

long long int (64bits) : $-2^{63} \sim 2^{63} - 1$

➤ 數字運算

① 讀取數字(long long int)

- scanf("%lld", &N);

② 數字拆解(重複計算)

- while(N > 0)

取餘數%10 → 除以10

N = 24252

N > 0 → N%10 = 2 (2425**2**)

2 出現次數 + 1

算下一位 N/10 = 2425

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	1	0	1	0	1	0	0	0

◆ 1. 數字拆解

int (32 bits) : $-2^{31} \sim 2^{31} - 1$

long long int (64bits) : $-2^{63} \sim 2^{63} - 1$

➤ 數字運算

① 讀取數字(long long int)

- scanf("%lld", &N);

② 數字拆解(重複計算)

- while(N > 0)

取餘數%10 → 除以10

N = 2425

N > 0 → N%10 = 5 (242**5**)

5 出現次數 + 1

算下一位 N/10 = 242

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	1	0	1	1	1	0	0	0

◆ 1. 數字拆解

int (32 bits) : $-2^{31} \sim 2^{31} - 1$

long long int (64bits) : $-2^{63} \sim 2^{63} - 1$

➤ 數字運算

① 讀取數字(long long int)

- scanf("%lld", &N);

② 數字拆解(重複計算)

- while(N > 0)

取餘數%10 → 除以10

N = 242

N > 0 → N%10 = 2 (24²)

2 出現次數 + 1

算下一位 N/10 = 24

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	2	0	1	1	1	0	0	0

◆ 1. 數字拆解

int (32 bits) : $-2^{31} \sim 2^{31} - 1$

long long int (64bits) : $-2^{63} \sim 2^{63} - 1$

➤ 數字運算

① 讀取數字(long long int)

- scanf("%lld", &N);

② 數字拆解(重複計算)

- while(N > 0)

取餘數%10 → 除以10

N = 24

N > 0 → N%10 = 4 (24)

4 出現次數 + 1

算下一位 N/10 = 2

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	2	0	2	1	1	0	0	0

◆ 1. 數字拆解

int (32 bits) : $-2^{31} \sim 2^{31} - 1$

long long int (64bits) : $-2^{63} \sim 2^{63} - 1$

➤ 數字運算

① 讀取數字(long long int)

- scanf("%lld", &N);

② 數字拆解(重複計算)

- while(N > 0)

取餘數%10 → 除以10

N = 2

N > 0 → N%10 = 2 (2)

2 出現次數 + 1

算下一位 N/10 = 0

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	3	0	2	1	1	0	0	0

◆ 1. 數字拆解

int (32 bits) : $-2^{31} \sim 2^{31} - 1$

long long int (64bits) : $-2^{63} \sim 2^{63} - 1$

➤ 數字運算

- ① 讀取數字(long long int)
 - scanf("%lld", &N);
- ② 數字拆解(重複計算)
 - while(N > 0)

N = 0

N = 0 → 運算結束

取餘數%10 → 除以10

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	3	0	2	1	1	0	0	0

N = 2425264 數字拆解計次結果

◆ 1. 數字拆解

int (32 bits) : $-2^{31} \sim 2^{31} - 1$

long long int (64bits) : $-2^{63} \sim 2^{63} - 1$

➤ 存入陣列

① 存各數字出現次數

- **num[10]** : 0~9出現的次數
- 每次 $N \% 10$ 拆解出數字 : $tmp = N$, 是N的暫存值(對tmp運算不會更改N)
→ **num[tmp%10] += 1**

num[10]

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	3	0	2	1	1	0	0	0

◆ 1. 數字拆解

int (32 bits) : $-2^{31} \sim 2^{31} - 1$

long long int (64bits) : $-2^{63} \sim 2^{63} - 1$

➤ 數字運算

- ① 讀取數字(long long int)
 - scanf("%lld", &N);
- ② 數字拆解(重複計算)
 - while(N > 0)

取餘數%10 → 除以10

➤ 存入陣列

- ① 存各數字出現次數
 - num[10] : 0~9出現的次數
 - 每%10拆解出數字 :
→ num[tmp%10] += 1

```
long long int N = 0;
scanf("%lld", &N);

int num[10] = {};
long long int tmp = N;
while(tmp > 0)
{
    num[tmp%10] += 1;
    tmp /= 10;
}
```

2. 陣列操作

陣列頻率排序



Icon made by [<https://www.flaticon.com/authors/dinosoftlabs>] from www.flaticon.com

◆ 2. 陣列操作

➤ 排序概念

N = 2425264

- ① 搜尋沒排序過的數字
- ② 每輪找出出現次數最多的數字
- ③ 存進排名陣列frequency[10]
- ④ 搜尋到的數字設為已找過

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	3	0	2	1	1	0	0	0
有無排序過	X	X	X	X	X	X	X	X	X	X
frequency										

N = 2425264 數字拆解計次結果

◆ 2. 陣列操作

➤ 排序概念

- ① 搜尋沒排序過的數字
- ② 每輪找出出現次數最多的數字
- ③ 存進排名陣列frequency[10]
- ④ 搜尋到的數字設為已找過

N = 2425264

第一輪排序：
數字2，共出現3次

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	3	0	2	1	1	0	0	0
有無排序過	X	X	X	X	X	X	X	X	X	X
frequency										

N = 2425264 數字拆解計次結果

◆ 2. 陣列操作

➤ 排序概念

- ① 搜尋沒排序過的數字
- ② 每輪找出出現次數最多的數字
- ③ 存進排名陣列 `frequency[10]`
- ④ 搜尋到的數字設為已找過

N = 2425264

第一輪排序：
數字2，共出現3次
`frequency[0] = 2`(第一名)

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	3	0	2	1	1	0	0	0
有無排序過	X	X	X	X	X	X	X	X	X	X
frequency			[0]							

N = 2425264 數字拆解計次結果

◆ 2. 陣列操作

➤ 排序概念

- ① 搜尋沒排序過的數字
- ② 每輪找出出現次數最多的數字
- ③ 存進排名陣列 `frequency[10]`
- ④ 搜尋到的數字設為已找過

N = 2425264

第一輪排序：
數字2，共出現3次
`frequency[0] = 2`

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	3	0	2	1	1	0	0	0
有無排序過	X	X	O	X	X	X	X	X	X	X
frequency			[0]							

N = 2425264 數字拆解計次結果

◆ 2. 陣列操作

➤ 排序概念

N = 2425264

- ① 搜尋沒排序過的數字
- ② 每輪找出出現次數最多的數字
- ③ 存進排名陣列frequency[10]
- ④ 搜尋到的數字設為已找過

第二輪排序：

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	3	0	2	1	1	0	0	0
有無排序過	X	X	O	X	X	X	X	X	X	X
frequency			[0]							

N = 2425264 數字拆解計次結果

◆ 2. 陣列操作

➤ 排序概念

- ① 搜尋沒排序過的數字
- ② 每輪找出出現次數最多的數字
- ③ 存進排名陣列frequency[10]
- ④ 搜尋到的數字設為已找過

N = 2425264

第二輪排序：
數字4，共出現2次

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	3	0	2	1	1	0	0	0
有無排序過	X	X	O	X	X	X	X	X	X	X
frequency			[0]							

N = 2425264 數字拆解計次結果

◆ 2. 陣列操作

➤ 排序概念

- ① 搜尋沒排序過的數字
- ② 每輪找出出現次數最多的數字
- ③ 存進排名陣列 `frequency[10]`
- ④ 搜尋到的數字設為已找過

N = 2425264

第二輪排序：
數字4，共出現2次
`frequency[1] = 4` (第二名)

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	3	0	2	1	1	0	0	0
有無排序過	X	X	O	X	X	X	X	X	X	X
frequency			[0]		[1]					

N = 2425264 數字拆解計次結果

◆ 2. 陣列操作

➤ 排序概念

- ① 搜尋沒排序過的數字
- ② 每輪找出出現次數最多的數字
- ③ 存進排名陣列 `frequency[10]`
- ④ 搜尋到的數字設為已找過

N = 2425264

第二輪排序：
數字4，共出現2次
`frequency[1] = 4` (第二名)

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	3	0	2	1	1	0	0	0
有無排序過	X	X	O	X	O	X	X	X	X	X
frequency			[0]		[1]					

N = 2425264 數字拆解計次結果

以此類推

◆ 2. 陣列操作

➤ 排序概念

- ① 搜尋沒排序過的數字
- ② 每輪找出出現次數最多的數字
- ③ 存進排名陣列 `frequency[10]`
- ④ 搜尋到的數字設為已找過

N = 2425264

第三輪排序：
數字5，共出現1次
`frequency[2] = 5` (第三名)

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	3	0	2	1	1	0	0	0
有無排序過	X	X	O	X	O	O	X	X	X	X
frequency			[0]		[1]	[2]				

N = 2425264 數字拆解計次結果

◆ 2. 陣列操作

每個數字都排序過→結束排序

➤ 排序概念

- ① 搜尋沒排序過的數字
- ② 每輪找出**出現次數最多的數字**
- ③ 存進排名陣列**frequency[10]**
- ④ 搜尋到的數字設為已找過

N = 2425264

第四輪排序：
數字6，共出現1次
frequency[3] = 6 (第四名)

數字	0	1	2	3	4	5	6	7	8	9
出現次數	0	0	3	0	2	1	1	0	0	0
有無排序過	X	X	O	X	O	O	O	X	X	X
frequency			[0]		[1]	[2]	[3]			

N = 2425264 數字拆解計次結果

◆ 2. 陣列操作

➤ 排序

- ① 排序總數 → 排序輪數
 - **AppearNum** :
待排序數字總數
- ② 排序結果紀錄
 - **frequency[10]** 排序結果
 - **count** 當前排列位置

(1) **AppearNum** 紀錄出現的數字總數

```
int AppearNum = 0;  
for(int i=0; i<10; i+=1)  
{  
    if(num[i] != 0)  
    {  
        AppearNum += 1;  
    }  
}
```

(2) **frequency[10]** 存頻率排列後結果
count 存排列到第幾個位置

```
int frequency[10]={};  
int count = 0;
```


◆ 2. 陣列操作

➤ 排序概念

- ① 搜尋沒排序過的數字
- ② 每輪找出現次數最多的數字
- ③ 存進排名陣列 `frequency[10]`
- ④ 搜尋到的數字設為已找過

`AppearNum` 排列輪數

`frequency[10]` 存頻率排列後結果

`count` 存排列到第幾個位置

```
int frequency[10]={};  
int count = 0;
```

```
for(int i=0;i<AppearNum;i+=1)  
{  
    int max = 0;  
    int maxpos = -1;  
    for(int j=0;j<10;j+=1)  
    {  
        if(num[j] > max)  
        {  
            max = num[j];  
            maxpos = j;  
        }  
    }  
    frequency[count] = maxpos;  
    count += 1;  
    num[maxpos] = -1;  
}
```

◆ 2. 陣列操作

➤ 每輪找出現次數最多數字

- ① max 存最大值
- ② maxpos 存最大值位置
- ③ 在 num[1~10] 找 max, maxpos
- ④ frequency 存每輪max排列結果
- ⑤ num[maxpos] 設為已找過
(下輪不再搜尋)

AppearNum 排列輪數

frequency[10] 存頻率排列後結果

count 存排列到第幾個位置

```
for(int i=0;i<AppearNum;i+=1)
{
    int max = 0;
    int maxpos = -1;
    for(int j=0;j<10;j+=1)
    {
        if(num[j] > max)
        {
            max = num[j];
            maxpos = j;
        }
    }
    frequency[count] = maxpos;
    count += 1;

    num[maxpos] = -1;
}
```


◆ 範例程式

```
#include<stdio.h>

int main()
{
    long long int N = 0;
    scanf("%lld",&N);

    int num[10] = {};
    long long int tmp = N;
    int AppearNum = 0;
    while(tmp > 0)
    {
        num[tmp%10] += 1;
        tmp /= 10;
    }
    for(int i=0;i<10;i+=1)
    {
        if (num[i] != 0)
        {
            AppearNum += 1;
        }
    }
}
```

```
int frequency[10]={};
int count = 0;
for(int i=0;i<AppearNum;i+=1)
{
    int max = 0;
    int maxpos = -1;
    for(int j=0;j<10;j+=1)
    {
        if(num[j] > max)
        {
            max = num[j];
            maxpos = j;
        }
    }
    frequency[count] = maxpos;
    count += 1;

    num[maxpos] = -1;
}

for(int i=0;i<count;i+=1)
{
    printf("%d ",frequency[i]);
}
}
```