Week2 出題練習 HW - 偉大性格的射手

第 16 組

Due: 2025/9/17

1 Problem

1.1 說明

今年是英雄聯盟中國賽區（LPL）最有希望的一年，身為熱門實況主的 JackeyLove 正在他人生中第一次的世界賽舞台上對決來自北美賽區（LCS）的 Team Liquid，在隊上擔任射手（AD carry）這個職位的他，對於金錢的運用一定要嚴謹，與對手對線到殘血的他正要回城出裝，因為身上的錢 M 不足以出一件大裝，但為了提升自己的能力而需要出兩件小裝作為過渡，商城裡總共有 N 件小裝，價格分別為 $n_0, n_1, n_2, \ldots, n_{N-1}$，為了讓自己數值最大化，請找出兩件裝備的價格之和等於自己身上的錢 M 的唯一組合。

1.2 Input Format and Constraints

- $2 \leq N \leq 5 * 10^4$
- $0 \leq n_0, n_1, \ldots, n_{N-1} < M \leq 10^9$
- 第一行有兩個數字 N 與 M 用一個空格隔開，分別代表商城內的裝備總數與自己身上有多少錢。
- 第二行有 N 個數字 $n_0, n_1, n_2, \ldots, n_{N-1}$ 代表這 N 件裝備的售價，且每件裝備的售價都不一樣。

1.3 Output Format

只有一行，由小到大輸出兩件裝備的 index A 與 B (0-based)，使得 $n_A + n_B = M$ 為唯一解，A 與 B 之間用一個空格隔開。

1.4 Sample Input 1

```
3 5
2 3 1
```

1.5 Sample Output 1

```
0 1
```

1.6 Sample Input 2

```
5 10
5 9 6 3 4
```

1.7 Sample Output 2

```
2 4
```

1.8 Sample Input 3

```
10 20
19 10 2 4 17 3 12 15 6 9
```

1.9 Sample Output 3

```
4 5
```

2 Solution

這題的核心觀念是使用一個類似 hash table 功能的陣列 ht，來紀錄每個元素的 index (因為根據題目，每個元素都會是 unique 的)。

但此題的 n_i 太大了，所以我們需要透過一些手段來節省使用的 memory，在官方解答中，我們會找出 n_i 中的 minimum 和 maximum，然後存每個元素的 index 時，扣掉 minimum (後面稱作 offset)。例如我們存 n_i 的 index 為 i 時，會紀錄為 $ht[n_i - offset] = i + 1$。這樣可以讓 `calloc` 的大小限制在 `maximum-minimum+1` (單位是 `int`)

最後，我們 iterate 過陣列的每個元素，然後看 index 是 $m - n_i - offset$ 的元素是否存在 (即不為 0，因為 `calloc` 會給 ht 每個元素 0 的初始值，所以是 0 代表沒這個元素) ，且 index 不為 $i + 1$ (即 n_i 自己)，如果存在即回傳該答案。

整體而言，此演算法的時間複雜度會是 $O(N)$，因為我們必須遍歷過常數次的陣列，空間複雜度則為 $O(\max n_i - \min n_i)$，$\max n_i$ 與 $\min n_i$ 分別為 $n_i, i \in \mathbb{N}, i \in [0, N-1]$ 的最大值與最小值，因為我們必須建立 hash table ht，其大小為 $\max n_i - \min n_i + 1$ 單位的 32-bit 整數。

3 Code

```c
#include <stdio.h>
#include <stdlib.h>
```

```
3
4  int *twoSum(int *nums, int n, int m)
5  {
6      int max = nums[0], min = nums[0];
7      for (int i = 1; i < n; i++)
8      {
9          if (nums[i] > max)
10             max = nums[i];
11         else if (nums[i] < min)
12             min = nums[i];
13     }
14     int htsize = max - min + 1;
15     int *ht = calloc(htsize, sizeof(int));
16     int offset = min;
17
18     for (int i = 0; i < n; i++)
19     {
20         ht[nums[i] - offset] = i + 1;
21     }
22     int *retarray = malloc(2 * sizeof(int));
23     for (int i = 0; i < n; i++)
24     {
25
26         int leftidx = m - nums[i] - offset;
27         if (leftidx >= 0 && leftidx < htsize && ht[leftidx] && ht[leftidx] !=
           ↪  (i + 1))
28         {
29             retarray[1] = ht[leftidx] - 1;
30             retarray[0] = i;
31             free(ht);
32             return retarray;
33         }
34     }
35     return 0;
36 }
37 int main()
38 {
39     int n, m;
40     scanf("%d%d", &n, &m);
41     int *nums = malloc(n * sizeof(int));
42     for (int i = 0; i < n; i++)
43     {
```

```
        scanf("%d", &nums[i]);
    }
    int *ans = twoSum(nums, n, m);
    printf("%d %d\n", ans[0], ans[1]);

    return 0;
}
```

4 TestData

本題我們共安排 7 筆測試資料，它們被設計的目的如下所示:

(a) subtask 0-3 (`[0-3].in`): 檢驗程式基本邏輯是否正確、能否給出正確的輸出。(Brute force is acceptable.) (50%，$N < 100, M < 5 * 10^4, n_i < 10^4$)

(b) subtask 4-6 (`[4-6].in`): 輸出正確以外，也檢驗程式的空間安排是否正確，在 N 與 n_i 較大的情況下，不只使用 hash table 的概念來改善時間複雜度，也有使用 solution 中 offset 的手法來節省空間以符合 memory 的規範。(50%, 除測資符合題目範圍外無任何限制)

5 Teamwork

- 題目設計/包裝與題解: 林有諒、陳昫叡
- 測資設計、驗題: 李國鈺、吳宥勳
- 校對測試、書面報告: 賴昱錡

6 Reference

本題的發想來自 LeetCode 1.Two Sum，但修改了原本題目的條件 (每個元素變成都要是 unique 的)、constraints、測資範圍與題目包裝。