泛型程式設計與標準模板函式庫
期末報告－－肥宅隊伍

陳守業 405262099 資工四乙
劉育騏 405262180 資工四乙
陳泓睿 405262324 資工四乙

故事

有一個熱鬧的商店街叫做東亞肥宅街，一年一度會有一次的大活動，為了避免人擠人，聰明的主辦單位想了一個方法，並有一個名單，叫做肥宅隊伍，這個方法給每個人編號，並分配給所有人一個福袋，福袋大小固定，但是裡面裝著價值不一定一樣的商品。而且這個編排方法，能夠使得福袋裡總價值較大的在隊伍的前頭(如果總價值一樣只能擇一)。也就是運氣最好的人會在最前面。但是主辦單位有些疏失，名字可能重複了沒注意到，而且編排方式也不是照著原本想要的價值大到小的編排方式。主辦單位想破了頭，請幫主辦單位給的名單刪除最少個名單項目，使得名單滿足肥宅隊伍的定義，名單上項目必須保持原本的順序，並將新的名單中的人名印出來。

設計理念

測資包含一個含有「名字」和「福袋內容物價值」的名單——

名單中每個人都有一個相應的、含有 n 個價值不一內容物（元素）的福袋

名單的總人數 m 和福袋的內容物數量 n 限制為 0 <= m, n <= 1000

名字長度不超過10個字元，福袋內容物的價值 p[i] 限制為 0 <= p[i] <= 10^9

(i = 0~n-1)

設計理念

每個項目的福袋內容物數量統一為 n 個

每個福袋的「總價值」定義為：

1. 假如是福袋中第一個物品，則把當前價值等於第一個物品

2. 接著，如果是之後的物品，加總的規則是：

    當前商品價值比現在總和大，則累加，否則減去當前商品價值

設計理念

對於每筆測資－－

1. 在不改變名單順序的前提下，把最少名單項目移除，
   使得名單符合一個「肥宅隊伍」的定義：
   　　價值由大到小，名字重複時只取第一個項目

2. 輸出新名單的項目數量，並將其中的名字依序輸出
   若有多組解，輸出「以輸入順序作為比較基準」的最小輸入序名單序列

必須使用STL容器與演算法完成，並使用到lambda expression，否則不於計分

輸入 / 輸出說明 — —題一〔測資產生〕

輸入說明：

輸入兩個整數 m, n (0 <= m, n <= 1000)

m 為名單項目數量，n 為福袋內容物數量

輸出說明：

總長為 m 的名單，每個名單項目以換行隔開

各項目：

開頭 ➔ 人名，length <= 10，各字元大小寫任意

後面 ➔ n 個正整數（各內容物的價值），0 <= n <= 10^9

輸出 0 0 表示名單末

輸入 / 輸出說明－－題一〔測資產生〕

輸入說明：

輸入兩個整數 m, n。m 為名單項目數量，n 為福袋內容物數量。

輸出說明：

產生總長為 m 的名單，每個名單項目以換行隔開。開頭為人名，最長 10 字元，各字元大小寫任意，後面接 n 個正整數，為該人福袋各內容物的價值，各內容物價值不得超過 10^9。0 <= m, n <= 1000。輸出 0 0 表示名單末。

輸入 / 輸出說明——題一〔測資產生〕

範例輸入：

4 5

範例輸出：

KinGboB 105 215 20 30 10

KinGboB 99 88 66 4599 787

Louis 777 999999 55 2 88

God 1 23 20 5 10

0 0

輸入 / 輸出說明——題二〔問題本體〕

輸入說明：題一的輸出。

輸出說明：

輸出名單符合以下規則。

一，對於名字重複的項目，只保留排序在最前面（最早被輸入）的。

二，以最少的項目刪除量，使名單符合依總價值遞減的排序。

總價值定義：初始價值總和為第一個物品，從第二個物品依序檢查，當前物品價值大於當前價值總和時累加，小於時減去。

首先，輸出新名單的項目數量，並進行一次換行。

然後，將其中的名字依序輸出，名字之間以“->”相隔。輸出最後接換行。

若有多組解，輸出「最小輸入序」的名單序列。

輸入 / 輸出說明－－題二〔問題本體〕

範例輸入：

KinGboB 105 215 20 30 10 // 總價值: 105+215-20-30-10=260

KinGboB 99 88 66 4599 787 // 這行不考慮，名字跟前一個一樣

Louis 777 999999 55 2 88 // 總價值: 777+999999-55-2-88=1000631

God 1 23 20 5 10 // 總價值: 1+23-20+5+10=19

0 0

範例輸出：

2 // 名單剩餘2個

KinGboB->God // 因為KinGboB的福袋比God的福袋總價值還要大，而且也是輸入順序最小的解

解題想法

- 暴力法: 2^N

- 枚舉所有可能，每個名單都可以刪除或不刪除。每個可能檢查是否滿足肥宅隊伍定義

解題想法

- 名單列表中先將重複名字去除
- 總價值是可以比較的
- 刪除最少名單項目 = 找到一個最長名單項目
- 找到最長名單排列(價值遞減) => LDS
- 紀錄最早出現的LDS

動態規劃: O(N^2) or O(NlogN)

Robinson-Schensted-Knuth Algorithm

- 時間複雜度: O(NlogN)
- 用貪心+二分搜找到LIS

- http://wiki.csie.ncku.edu.tw/acm/course/LIS

STL元件

```cpp
typedef long long LL; // short cut for long long
using namespace std;

int main()
{
    string str;
    set<string> st; // set for name
    unordered_map<LL, string> bagToName;
    vector<LL> arr;
    vector<LL> reverseLis; // just length
    vector<int> ind; // true index of lis

    while(getline(cin, str)){
        if(str == "0 0") break;

        stringstream ss(str); // split string
        string name;
        ss >> name;
        if(st.count(name)) continue; // already exist
        st.insert(name);

        int value;
        vector<int> bag; // bag for value
        while(ss >> value){
            bag.push_back(value);
        }

        LL total = accumulate(bag.begin(), bag.end(), 0, [](LL a, LL b){
                                return a<b ? a+b : a-b;
                            });
        arr.push_back(total);
        if(!bagToName.count(total))
            bagToName[total] = name; // trace name
    }

```

Associative Containers

Sequence Container

Accumulate累加函式，並自訂函式行為 Lambda

```
35      }
36
37      if(arr.size() == 0){
38          cout << 0 << endl;
39          return 0;
40      }
41
42      ind.resize((int)arr.size(), 0);
43
44      reverseLis.push_back(arr.back());
45      for(int i = arr.size()-2; i >= 0; --i){
46          if(reverseLis.back() < arr[i]){
47              reverseLis.push_back(arr[i]);
48              ind[i] = reverseLis.size()-1;
49          }
50          else{
51              auto it = lower_bound(reverseLis.begin(), reverseLis.end(), arr[i]);
52              ind[i] = it-reverseLis.begin();
53              *it = arr[i];
54          }
55      }
56
57      int ldsLen = reverseLis.size();
58      cout << ldsLen << endl; // new menu size
59      bool first = true;
60      for(int i = 0; i < (int)arr.size(); ++i){
61          if(ind[i] == ldsLen-1){
62              if(!first) cout << "->";
63              cout << bagToName[arr[i]];
64              ldsLen--;
65              first = false;
66          }
67      }
68
69      return 0;
70  }
```

做預處理

由後往前算LIS =
由前往後算LDS

Binary Search 查
找迭代器位置

反過來印，但
因為原本是反
過來做，所以
走訪還是從前
到後

時間複雜度

- 算LDS長度O(NlogN) + 找LDS O(N)
- 總時間複雜度 O(NlogN)

參考資料

http://wiki.csie.ncku.edu.tw/acm/course/LIS

https://en.wikipedia.org/wiki/Longest_increasing_subsequence