Shopping with Coupons

Group:xxx

Name:xxx

Date:2024-05-02

Chapter 1： Introduction

1 Description

Given **N items** and **N coupons**, each coupon can be used unlimited times, and each item can be purchased unlimited times, too. But for each item, each coupon can be used once and **only once**. Now with **D dollars**, how can you buy as many items as you can, and how much is left?

```
For example:
Coupon 1 used in Item 1, Coupon 1 used in Item 2, Coupon 2 used in Item 1 (OK)
Coupon 1 used in Item 1, Coupon 1 used in Item 1 again (WRONG)
```

2 Input

Each input file contains one test case, with **three** lines.

The first line: the number of items (and the coupons) N ($\leq 10^5$), and the amount of money D ($\leq 10^6$)

The second line: N positive prices

The third line: N positive coupon values

(Guaranteed: the highest value of coupons $\leq$ the lowest price of items, and the numbers in a line are separeted by spaces)

3 Output

The maximum number of items you can buy, and the maximum amount of money left.

(Print in a line, and separated by 1 space.)

```
[Example]
input:
4 30
12 20 15 10
9 6 7 8
output:
8 2
```

Chapter 2： Algorithm Specification

To maximize the number of items purchased, we need to start with items that cost as little as possible, so it's easy to think of using the **greedy algorithm** to solve this problem.

To build the greedy algorithm, we first use **bubble sorting** so that items are listed in ascending order (price) and coupons are listed in descending order (coupon). Add a pointer (index) of coupon for each item, and we can represent the lowest price to buy each item in the current round (`min_price = price[i] - coupons[index[i]]`, note that the array'coupons' is all initialized to 0, so when the pointer points to an address beyond n, there is no wrong visit, but rather a visit to a coupon with a value of 0). Compare these prices and choose the smallest to know what to buy and how much to spend.

Main structure	Description
`int price[MAX_NUM+5]`	the price of each item
`int coupons[MAX_NUM+5]`	the value of each coupon
`int index[MAX_NUM+5]`	the pointer of coupon for each item

```
bubble sort price[] in ascending order
bubble sort coupons[] in descending order

WHILE !flag
    min_price = INFINITY
    FOR i = 0 TO n-1
        IF min_price = lowest price of item i in current round
            min_price = lowest price of item i in current round
            item = i
    IF d < min_price
        flag = 1
        break
    ELSE
        d -= min_price
        num++
        index[item]++
```

(The code is so simple that pseudo code is no different from the source code, so descriptive language is used here to summarize it briefly.)

Chapter 3: Testing Results

Case	Expected result	Actual behavior	Status	Note

`4 30`
`12 20 15 10`
`9 6 7 8`
	`8 2`	`8 2`	Pass	Given by the problem

`1 10`
`6`
`4`
	`2 2`	`2 2`	Pass	n = 1 (coupon can use only once)

`5 150`
`21 35 18 62 45`
`3 5 1 15 9`
	`11 5`	`11 5`	Pass	Some items are never bought
				(the original price of $item_a$ is more cheaper)

`3 30`
`5 4 6`
`2 3 1`
	`10 0`	`10 0`	Pass	Buy every item at least once,
				and use every coupon at least once

Chapter 4: Analysis and Comments

This program has a time complexity of $O(N^2)$ and a space complexity of $O(N)$. And the analysis is as follows:

1 Time Complexity:

- **Reading Input Data:** $O(N)$ - It requires reading n prices and n coupons.
- **Bubble Sort:** $O(N^2)$ - Performing bubble sort on n prices and n coupons.
- **Item Purchase Loop:** Worst case is $O(N^2)$ - Iterating n times for each item to find the minimum price.

2 Space Complexity Analysis:

- **Integer Variables:** $O(1)$ - Occupies constant space.
- **Integer Arrays price, coupons, index:** $O(N)$ - Each array has a size of n.

Chapter 5: Source Code (in C)

```c
#include <stdio.h>
#define MAX_NUM 100000
#define INFINITY 1000000
void swap(int* a, int* b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
int main(void)
{

    int n;  // the number of items and the coupons (<= 10^5)
    int d;  // the amount of money (<= 10^6)
    int price[MAX_NUM+5];   // the price of each items
    int coupons[MAX_NUM+5] = {0};   // the coupons
    int num = 0;    // the max_number of the items we can buy

    int index[MAX_NUM+5] = {0};
    // index[1] = 3 means item2(ascending sort) use coupons4(descending) next time
    int min_price;  // the min price for next item we buy
    int item;       // pointer for item we buy
    int flag = 0;

    scanf("%d %d", &n, &d);
    for(int i = 0; i < n; i++)
        scanf("%d", &price[i]);
    for(int i = 0; i < n; i++)
        scanf("%d", &coupons[i]);

    for(int i = 1; i < n; i++){ // bubble sort
        for(int j = 0; j < n - i; j++)
        {
            if(price[j] > price[j+1]) swap(&price[j], &price[j+1]);
            // ascending order
            if(coupons[j] < coupons[j+1]) swap(&coupons[j], &coupons[j+1]);
            // descending order
        }
    }
    // for(int i = 0; i < n; i++) printf("%d ",coupons[i]); printf("(sort)\n");
    //* test bubble sort
    while(!flag)
    {
        min_price = INFINITY;
        for(int i = 0; i < n; i++)
        {
```

```
            if(min_price > price[i] - coupons[index[i]])
            // for every item, the min price is price[i] - coupons[index[i]]
            {
                min_price = price[i] - coupons[index[i]];
                item = i;
            }
        }
        if(d < min_price)   // can't buy item anymore
        {
            flag = 1;
            break;
        }else{
            // printf("buy item%d, cost: %d\n", item, min_price);
            //* test by print item we buy
            d -= min_price;
            num++;
            index[item]++;
        }
    }
    printf("%d %d", num, d);
    return 0;
}
```

Declaration

I hereby declare that all the work done in this project titled "Shopping with Coupons" is of my independent effort.