Разбор задачи «Торговля акциями»

№1253

https://informatics.msk.ru/mod/statements/view3.php?chapterid=1253#1

Торговля акциями

Требуется написать программу, которая по имеющимся данным о стоимости акций в каждый из дней, найдет оптимальную стратегию покупки и продажи акций.

При этом для простоты будем считать, что за эти n дней купить акции можно не более одного раза и продать акции можно также не

Кроме этого, будем считать, что продажа и покупка будет осуществляться только с акциями одного типа. На начало этого периода вы располагаете суммой в x рублей. Для каждого из дней известна цена a_i по которой можно купить одну акцию, и цена b_i по которой можно одну акцию продать.

Разрешается продавать и покупать только целое число акций (например, если у вас есть 5 рублей, а акция стоит 2 рубля, то вы можете купить не более двух акций).

Торговля акциями

- Входные данные
- Первая строка входного файла содержит два целых числа n и x ($1 \le n \le 100000$, $1 \le x \le 10^6$).
- Вторая строка входного файла содержит n целых чисел $a_1, \ldots, a_m$. Третья строка входного файла содержит n целых чисел $b_1, \ldots, b_m$. Для каждого индекса i ($1 \le i \le n$) выполняются неравенства $1 \le bi \le a_i \le 1000$.
- Выходные данные
- В первой строке выходного файла выведите максимальную сумму, которой вы можете обладать по окончании рассматриваемого периода. Во второй строке выведите два числа — номер дня d_1, в который следует купить акции, и номер дня d_2, в который эти акции следует продать (должно выполняться неравенство $d_2 > d_1$). При этом подразумевается, что покупается столько акций, сколько их можно купить на x рублей, а потом они все продаются. Если в найденной вами стратегии продавать и покупать акции не требуется, то выведите выведите во второй строке «-1 -1».

Примеры

Примеры

входные данные
5 1000 2 3 1 4 3 1 2 1 2 3
выходные данные
3000 3 5

входные данные
5 1000 10 9 8 7 6 9 8 7 6 5
выходные данные
1000 -1 -1

Подсказки по синтаксису python

- Читаем первые 2 числа:

n,m = map(int, input().split())

- Читаем первый массив:

A = [int(i) for i in input().split()]

Если у вас падает на тестах №…

- № 27

Проверьте, не забыли ли вы прибавить остаток от деления при покупке акций

- № 7

В C++ если вы используете деление двух int, но не приводите к вещественному типу перед делением (к double или float), то 46/33 например, будет равно 46/41.
В python с этим проблем не возникает.

Осторожно, дальше разбор и спойлеры ;)

Подводные камни

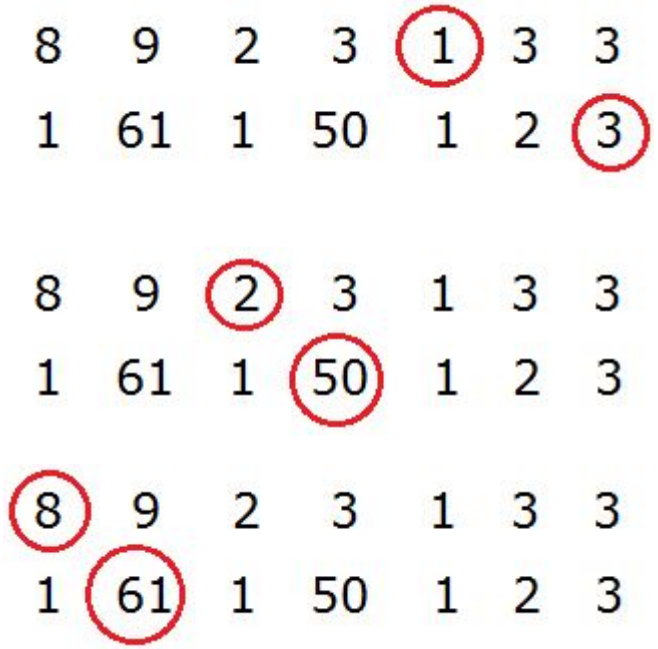
- Забыть прибавить к ответу остаток денег от x
- Жадный алгоритм тут не работает
- Перебор – «Превышено максимальное время работы»
- Будет ли оптимально умножить максимум цены в продаже на минимум до него в цене покупке? Может, и не будет оптимально? Может, в серединке был вариант получше? Придумайте свой тест на эту ситуацию.

Пример такого теста:

8 9 2 3 1 3 3
1 61 1 50 1 2 3

Как составить тест
чтобы сломать
свою программу?

То есть, возможно,
что продать по
самой большой
цене - не гарантия
самого большого
выигрыша



Алгоритм решения

1. Идём с конца массива b. Запоминаем b_max, если нашли большее, то начинаем рассматривать как кандидата на ответ

2. Для этого кандидата (обозначим день k, соответственно, рассматриваем b[k]) находим минимум в массиве a, из тех a[i], которые от первого дня до дня k. Запоминаем его день в переменную h. Вычисляем, сколько мы заработаем с этой парой чисел (результирующую функцию) и запоминаем это значение в F_rez_tmp. F_rez_tmp = x/a[h] * b[k]

Если оно оказалось больше, чем F_rez_max, найденное для предыдущих кандидатов, то у нас новая “пара-победитель”.

Перезаписываем F_rez_max, не забываем сохранить где-нибудь значения «победных» дней для покупки и для продажи. Пусть в h_best и k_best

Возвращаемся к пункту 1 и продолжаем просматривать b, пока он не закончится.

Когда он закончится, у нас будет и значение F_rez_max (к которому не забываем прибавить остаток денег от деления) и значения двух дней, в которые следует купить (h_best) и продать (k_best)

Решение на C++

```
int minai = n - 1;
int maxbi = n - 1;
int tempbi = maxbi;
for (int i = n - 2; i >= 0; i--)
{
    if (b[i]>b[tempbi]) {
        tempbi = i;
    }
    if (((x / a[i] * b[tempbi] + x%a[i]) >
        (x / a[minai] * b[maxbi] + x%a[minai])) && (a[i] <= x))
    {
        minai = i; maxbi = tempbi;
    }
}
if (b[maxbi] > a[minai] && a[minai] <= x)
{cout << x / a[minai] * b[maxbi] + x%a[minai] << endl
<< minai + 1 << ' ' << maxbi + 1;
}
else cout << x << endl << "-1" << ' ' << "-1";
```

Автор решения: Дегтярев Иван

Решение на Python

```
n,x=map(int,input().split())
A=list(map(int,input().split()))
B=list(map(int,input().split()))
bmax=0
amin=1001
ans=0
b_best=0
a_best=1001
for i in range(n-1,-1,-1):
    if B[i]>bmax:
        bmax=B[i]
        amin=i
        for j in range(i-1,-1,-1):
            if (A[j]<A[amin]) and (j!=i):
                amin=j
        if ((x//A[amin])*bmax>ans) and (amin!=i):
            ans=(x//A[amin])*bmax
            b_best=i
            a_best=amin
if ans<x:
    print(x)
    print(-1,-1)
else:
    print(int((ans)+(x-(A[a_best])*(x//A[a_best]))))
    print(a_best+1,b_best+1)
```

Автор решения: Баранов Андрей

Алгоритмическая сложность

Не O((n^2-n)/2), а меньше,

так как есть условие $a_i >= b_i$