Стажировка весна-лето 2023 | бэкенд

12 фев 2023, 16:54:31
старт: 12 фев 2023, 16:02:11
финиш: 12 фев 2023, 21:02:11
до финиша: 04:07:37
начало: 1 фев 2023, 00:00:00
длительность: 05:00:00

B. Горе от перфекционизма

	Все языки	GNU C++20 10.2	Clang14 C++20
Ограничение времени	3 секунды	1 секунда	1 секунда
Ограничение памяти	256Mb	256Mb	256Mb
Ввод	стандартный ввод или input.txt		
Вывод	стандартный вывод или output.txt		

Объявление: если у вас есть **жалобы / претензии / вопросы** насчет задач, то советуем для начала ознакомиться с расширенной инструкцией, содержащей в том числе ответы на самые частые вопросы.
Начало условия: К Новому-преновому году работники Тындекса построили N ледяных скульптур, i-я скульптура состоит из a_i килограмм льда.

Но они не посоветовались с Кузей! А ведь Кузя знает, что **идеальная** скульптура состоит из ровно X килограмм льда, не больше и не меньше.

Новый-преновый год уже совсем скоро, до него осталось всего T минут. За одну минуту Кузя может выбрать одну скульптуру и добавить или удалить ровно 1 килограмм льда из неё.

Вас, как отличника художественной школы, Кузя просит найти максимальное количество идеальных скульптур в момент наступления праздника.

Формат ввода

В первой строке вводятся три целых числа N, X, $T\left(1 \le N \le 2 \cdot 10^5; 0 \le X \le 10^9; 0 \le T \le 3 \cdot 10^{14}\right)$ — количество скульптур, идеальное количество льда в скульптуре и оставшееся количество минут до наступления праздника.
Во второй строке вводятся через пробел N целых чисел $a_i\left(1 \le a_i \le 10^9\right)$ — количество килограмм льда в i-й скульптуре.

Формат вывода

В первой строке выведите целое число $K(0 \le K \le N)$ — максимально возможное количество идеальных скульптур в момент наступления праздника.
Во второй строке выведите через пробел K различных целых чисел $b_i(1 \le b_i \le N)$ — номера скульптур, которые будут идеальными в момент наступления Нового-пренового года.

Скульптуры нумеруются с 1 в порядке ввода.

Если **оптимальных** ответов несколько, то выведите **любой** из оптимальных.

Пример 1

Ввод	Вывод
3 5 2 5 10 6	2 1 3

Пример 2

Ввод	Вывод
5 19 32 36 10 72 4 50	2 2 4

Пример 3

Ввод	Вывод
4 25 10 1 10 42 9	0

Примечания

Пояснение к **первому** тестовому примеру.
До нового года остаётся 2 минуты, а идеальная скульптура должна содержать ровно 5 килограмм льда.

1. Первая скульптура идеальна сразу, поэтому Кузя не тратит времени на её исправление.
2. Кузя может сделать идеальной третью скульптуру за $|6 - 5| = 1$ минуту. После этого у него в запасе останется $2 - 1 = 1$ минута.
3. Кузя не сможет сделать идеальной вторую скульптуру, так как на её исправление необходимо $|10 - 5| = 5$ минут.

Пояснение ко **второму** тестовому примеру.

До нового года остаётся 32 минуты, а идеальная скульптура должна содержать ровно 19 килограмм льда.

Рассмотрим, сколько требуется времени на «идеализацию» фигур:

1. $|19 - 36| = 17$ минут;
2. $|19 - 10| = 9$ минут;
3. $|19 - 72| = 53$ минуты;
4. $|19 - 4| = 15$ минут;
5. $|19 - 50| = 31$ минута.

Итого получаются три возможных сценария с двумя идеальными фигурами:

1. Первая и вторая за $17 + 9 = 26$ минут;
2. Первая и четвертая за $17 + 15 = 32$ минуты - обратите внимание, что в данном сценарии Кузя потратит полностью время, оставшееся до события;
3. Вторая и четвертая за $9 + 15 = 24$ минуты.

Хотя Кузя может сделать идеальной пятую фигуру, но на неё одну потребуется почти всё время (31 из 32 минут), поэтому Кузя не рассматривает такие сценарии.

Пояснение ко **третьему** тестовому примеру.

До нового года остаётся 10 минут, а идеальная скульптура должна содержать ровно 25 килограмм льда.

Кузя не успеет сделать ни одну из фигур идеальной, так как на каждую из них требуется больше, чем 10 минут:

1. $|1 - 25| = 24 > 10$;
2. $|10 - 25| = 15 > 10$;
3. $|42 - 25| = 17 > 10$;
4. $|9 - 25| = 14 > 10$.

Язык GNU C++20 10.2

Набрать здесь | Отправить файл

```cpp
#include<iostream>
#include<vector>
#include<math.h>
#include<algorithm>
using namespace std;

long long x;

struct Item {
    long long Weight;
    long long index;
};

bool operator < (const Item& a, const Item& b) {
    return abs(a.Weight - x) < abs(b.Weight - x);
}

int main() {

    //ios::sync_with_stdio(false);
    //cin.tie(nullptr);
    long long n, t;
    cin >> n >> x >> t;

    vector<Item> items(n);
    for (long long i = 0; i < n; i++) {
        cin >> items[i].Weight;
        items[i].index = i + 1;
    }

    sort(items.begin(), items.end());
    long long i = 0;
    while (t >= abs(x - items[i].Weight) && i < n) {
        t -= abs(x - items[i].Weight);
        i++;
    }

    cout << i << "\n";
```

Отправить

Предыдущая

Следующая