Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по «Алгоритмам и структурам данных»

Яндекс.Контест

Выполнил:

Студент группы P3211

Кривоносов Е.Д.

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2021

Задача №E «Коровы в стойла»

Пояснение к примененному алгоритму:

Данную задачу мы решаем от обратного, проверяя можем ли мы расставить коров или нет. Первую корову мы всегда ставим в первое стойло, а всех остальных на обретённое расстояние (предположим X). При слишком маленьких значениях X, мы всегда можем расставить коров, а при слишком большом расстоянии – нет. Значит как раз границу нам и нужно найти по условию задачи. И благодаря бинарному поиску мы можем её найти сокращая каждый раз в 2 раза наш промежуток поиска.

Сложность алгоритма:
O(n*log(n))

Код:

```
#include <iostream>


using namespace std;

void solve(){
    int n, k;
    int left, mid, right;
    cin >> n >> k;
    int coords[n];

    for (int i = 0; i < n; i++){
        cin >> coords[i];
    }
    // Используя бинарный поиск найдем макс расстояния
    // присвоим левую и правую границу нашего поиска
    left = 0;
    right = coords[n - 1] - coords[0] + 1;

    // Мы будем делать поиск пока не придем к одному числу на нашем промежутке
(решению)
    while (left != right){
        mid = (left + right) / 2;
        // Первую корову мы всегда можем поставить на 1 координату,
        // а остальных уже распределить
        int cows = 1;
        int cow_coord = coords[0];
        // Здесь мы осуществляем поиск количества коров, которое можем
расположить
        // При нашем выбранном расстоянии mid.
        for (int i = 1; i < n; i++){
            if (coords[i] - cow_coord > mid){
                cow_coord = coords[i];
                cows++;
            }
        }
        // Если мы расположим всех коров на нашем промежутке, то меняем левую
границу
        // Если нет, то меняем правую границу. Потому что при слишком маленьком
значении
        // mid - мы сможем расставить всех, а при слишком большом - нет
        if (cows >= k){
            left = mid + 1;
```

```
        } else {
            right = mid;
        }
    }
    cout << left;
}

int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    solve();
    return 0;
}
```

Задача №F «Число»

Пояснение к примененному алгоритму:

Мы просто меняем местами строки, если сумма чисел одной комбинации больше другой a[i]+a[i+1] < a[i+1]+a[i] внутренний цикл с каждой внешней итерацией пробегает на 1 меньше, чем в прошлый раз ведь мы в первый раз пробежали весь массив и уже переставили последнее число в нужное нам место, зачем его ещё раз трогать? Поэтому нам не нужно каждый раз делать так, чтобы внутренний цикл пробегал весь массив. В итоге мы получаем сортировку пузырьком.

swap() - меняет местами

Сложность алгоритма:
O(n^2)

Код:

```
#include <iostream>
#include <string.h>

using namespace std;

void solve(){
    string arr_num[100];
    int i, j, count = 0;
    for (int i = 0; i < 100; i++){
        cin >> arr_num[i];
        if (arr_num[i].size() == 0){
            break;
        }
        count++;
    }
    for(i = count - 1; i > 0; i--){
        for (j = 0; j < i; j++){
            if (arr_num[j] + arr_num[j + 1] < arr_num[j + 1] + arr_num[j]){
                swap(arr_num[j], arr_num[j + 1]);
            }
        }
    }
```

```
    for(int i = 0; i < count; i++){
        cout << arr_num[i];
    }
}

int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    solve();
    return 0;
}
```

Задача №G «Кошмар в замке»

Пояснение к примененному алгоритму:

Самое сложное при решении задачи было понять, а что делать если у нас 3 или более одинаковых букв. Но прочитав условие более внимательно, я понял, что их можно, как угодно, расставить в “середину” строки. Веса мы должны отсортировать по убыванию. Поэтому буквы с наибольшим весом мы ставим по бокам строки, а с помощью переменной, которая отвечает за количество встретившихся букв более 1 раза, мы знаем позицию всех остальных букв куда мы должны ставить. Таким образом мы и получаем строку с наибольшим весом.

Сложность алгоритма:
O(n*log(n))

Код:

```
#include <iostream>
#include <string.h>
#include <algorithm>
#include <vector>
#include <map>

using namespace std;

bool sortbysecond(const pair<char, int> &a, const pair<char, int> &b){
    return a.second > b.second;
}

void solve(){
    vector <pair<char, int>> weight;
    map<char, int> count_char;
    string line;
    int count = 0;
    cin >> line;

    // Заполняем вектор пар для каждой буквы: буква-вес и сортим по возрастанию
    for (int i = 0; i < 26; i++){
        int wght;
        cin >> wght;
        weight.push_back(make_pair(char(i + 97), wght));
```

```
    }
    sort(weight.begin(), weight.end(), sortbysecond);

    // Считаем количество каждого символа и позицию всех остальных символов
    for (int i = 0; i < line.length(); i++){
        count_char[line[i]]++;
        if (count_char[line[i]] == 2) count++;
    }

    char mas[line.length()];
    int start = 0;
    int end = line.length() - 1;
    for (auto p: weight){
        if (count_char[p.first] > 1){
            mas[start] = p.first;
            mas[end] = p.first;
            start++;
            end--;
            count_char[p.first] -= 2;
        }
        while (count_char[p.first] != 0){
            mas[count] = p.first;
            count_char[p.first]--;
            count++;
        }
    }

    for (auto p: mas){
        cout << p;
    }
}

int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    solve();
    return 0;
}
```

Задача №H «Магазин»

Пояснение к примененному алгоритму:

Во время считывания цен мы одновременно и подсчитываем сумму всех наших товаров. Когда мы отсортируем наши цены по убыванию, для того чтобы больше всего сэкономить, мы должны будем из нашей суммы вычитать каждый k товар, на который скидка (потому что товар бесплатный с самой низкой ценой).

Сложность алгоритма:
O(n*log(n))

Код:

```
#include <iostream>
#include <vector>     // vector
```

```cpp
#include <algorithm>  // sort

using namespace std;

void solve(){
    int n, k;
    vector <int> price;
    cin >> n >> k;
    int num;
    int sum = 0;
    // Просто заполняем цены.
    for (int i = 0; i < n; i++){
        cin >> num;
        sum += num;
        price.push_back(num);
    }
    // Производим сортировку и переворачиваем, чтобы цены были
    // В порядке убывания т.к. по условию сказано, что персонаж захотел разбить
товары
    // По цене, чтобы добиться большей скидки. Логичнее идти от большего к
меньшему.
    sort(price.begin(), price.end());
    reverse(price.begin(), price.end());
    // Уменьшаем суммы покупки с учетом скидочного товара
    for (int i = 1; i <= n; i++){
        if (i % k == 0){
            sum -= price[i - 1];
        }
    }
    cout << sum;

}

int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    solve();
    return 0;
}
```