Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по «Алгоритмам и структурам данных»

Базовые задачи

Выполнил:

Студент группы P3230

Асташин С.С.

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2022

Задача E «Коровы в стойла»

Пояснение к примененному алгоритму:

Главная идея решения данной задачи – двоичный поиск по расстоянию. Сначала сортируем стойла (`vector<int> stalls`) по возрастанию (можно и в другом порядке, сути не меняет, просто потом в цикле придётся идти с другой стороны). Затем ищем наибольшее возможное допустимое расстояние между коровами в промежутке от 1 до `stalls.back() - stalls.front()`:

1. Для каждого `mid = l + (r - l + 1) / 2` идём в цикле от 1 до `n` по стойлам;
2. Пытаемся расставить по стойлам `k` коров (в случае выполнения условия `stalls[i] - stalls[last_stall] >= mid`, где `last_stall` – индекс последнего стойла, в которое поставлена корова, увеличиваем счётчик коров);
3. После цикла, если `cows < k`, то крутим дальше поиск по левой части промежутка, иначе – по правой;
4. Поиск заканчивается, когда `r <= l`. В качестве ответа берём `l`.

Оценка сложности: O(n*log(n)) + O(log(x)) * O(n) = O(n*log(n)) + O(n*log(x)), где x – наибольшее расстояние между двумя стойлами.

Задача F «Число»

Пояснение к примененному алгоритму:

Пожалуй, самая простая задача блока, тут особо комментировать нечего. Достаточно было написать компоратор, который принимает на свой вход строки `a, b` и возвращает `a + b > b + a`: смотрим, какая из двух возможных склеенных строк больше (тут уже используется дефолтный компоратор для строк). Далее, сортируем `list<string> chunks.`

Оценка сложности: O(n*log(n)).

Задача G «Кошмар в замке»

Пояснение к примененному алгоритму:

Главная идея – то, что за вес всего слова отвечают только дублирующиеся буквы с ненулевыми весами, поэтому слово максимального веса будет похоже на что-то вроде “abccdefcba”, причём нам абсолютно не важно, в каком порядке находятся буквы в середине слова (на вес это не влияет).

Для начала, считаем, сколько раз в слове встречается каждая буква. После этого, в цикле ввода весов букв, проверяем, что вес i-ой буквы ненулевой, и она встречается больше 1 раза – если это условие выполнилось, то добавим эту букву в `new_word`, строку для хранения результата. После этого отсортируем `new_word` по весам букв в нём. Кроме того, уменьшим счётчик каждой из его букв на 2, поскольку из изначального слова нужно убрать 2 экземпляра данной буквы. Это надо потому, что из изначального слова мы будем формировать «середину» нового слова, и из этой середины один экземпляр парной буквы будет перемещён в начало, другой – в конец нового слова. После добавления середины, в конец нового слова нужно будет “отзеркалить” его начало до середины. Наверно, проще объяснить на примере (будем считать, что ненулевой вес имеют буквы a, b и c, причём вес a > вес b > вес c):

1. badecfcabc (изначальное слово после ввода)
2. abc (новое слова после ввода весов и сортировки)
3. abccdef (новое слово после добавления середины cdef, порядок букв в середине – алфавитный (так удобнее лично мне))
4. abccdefcba (новое слово после “отзеркаливания” его начала)

Оценка сложности: O(x*log(x)) + O(n), где x – число дублирующихся букв с ненулевыми весами.

Задача H «Магазин»

Пояснение к примененному алгоритму:

Тут всё тоже просто: надо догадаться, что исключение из чека одного дорогого товара всегда выгоднее исключения нескольких дешёвых. Сортируем цены по убыванию, потом идём по вектору и берём в итоговую сумму все товары, кроме каждого k-ого (для этого используем счётчик).

Оценка сложности: O(n*log(n)) + O(n) = O(n*log(n)).