Федеральное государственное
автономное учебное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Мегафакультет компьютеных технологий и управления
Факультет программной инженерии и компьютерной техники

Отчёт
по лабораторной работе №2
по дисциплине «Алгоритмы и структуры данных»

Базовые задачи (сортировка)

Группа: P3218

Студент: Богданова Мария Михайловна
(почта: Mariwolf.com@yandex.ru)

Преподаватели: Косяков М. С., Тараканов Д. С.

Санкт-Петербург
2024

Содержание

1 Задача E. Коровы в стойла (решена)

1.1 Исходный код решения

```
#include <algorithm>
#include <iostream>
#include <vector>


using namespace std;

int main() {

  int stalls, cows_cnt;
  cin >> stalls >> cows_cnt;

  vector<int> stalls_coords(stalls);
  for (int i = 0; i < stalls; ++i) {
    cin >> stalls_coords[i];
  }

  sort(stalls_coords.begin(), stalls_coords.end());

  int max_st_crd = stalls_coords.back();
  int min_st_crd = stalls_coords[0];

  int left_bound = 0;
  int right_bound = max_st_crd - min_st_crd + 1;

  while (right_bound - left_bound > 1) {
    int mid_stall = (left_bound + right_bound) / 2;
    int stall = 1;
    int last_stall = stalls_coords[0];
    for (int i = 0; i < stalls; ++i) {
      if (stalls_coords[i] - last_stall >= mid_stall) {
        stall++;
        last_stall = stalls_coords[i];
      }
    }
    if (stall >= cows_cnt) {
      left_bound = mid_stall;
    } else {
```

```
      right_bound = mid_stall;
    }
  }

  cout << left_bound << endl;

  return 0;
}
```

Листинг 1: Исходный код решения задачи E

1.2 Пояснения к алгоритму и оценка сложности

Задача на бинарный поиск, поэтому сначала мы определяем минимальную и максимальную координаты стойл, затем устанавливаем левую границу на минимально возможную координату - 0, а правую - на максимально возможную + 1, чтобы учесть все возможные расположения. На каждой итерации поиска вычисляем среднее расстояние между текущими границами и проверяем, можем ли мы разместить всех коров с таким расстоянием между ними, если да - увеличиваем расстояние и обновляем левую границу, если нет - уменьшаем расстояние и обновляем правую границу. Итерации прекращаем, когда левая граница не станет равна правой или меньше. В итоге выводим значение левой границы, которое представляет максимальное расстояние между коровами. Иными словами, мы ставим первую корову в самое левое стойло, затем идем в порядке возрастания по массиву координат стойл, храним координату последней поставленной коровы, и, либо пропускаем стойло, либо ставим в него следующую корову в зависимости от расстояния до предыдущей коровы.

Сложность алгоритма можно оценить следующим образом: в худшем случае: если бы стойла были не отсортированными, то сложность бинарного поиска составляла бы log^2(n), но так как координаты стойл подавались в виде отсортированного массива, сложность алгоритма - log(n). Сложность каждой проверки на возможность поставить корову в стойло - линейная O(n). Сложность самого бинарного поиска у нас составляет O(log(n)). Общая сложность O(nlogn)

2 Задача F. Число (решена)

2.1 Исходный код решения

```
#include <algorithm>
#include <iostream>
#include <vector>


bool cmp_obj(std::string a, std::string b) { return a + b > b + a; }

int main() {
  std::string str;
  std::vector<std::string> seq;
  while (std::cin >> str)
    seq.push_back(str);

  int n = seq.size();
  for (int i = 0; i < n; ++i) {
    for (int j = 0; j < n - i - 1; ++j) {
      if (!cmp_obj(seq[j], seq[j + 1])) {
        std::swap(seq[j], seq[j + 1]);
      }
    }
  }

  std::string result;
  for (int i = 0; i < seq.size(); ++i) {
    result += seq[i];
  }

  std::cout << result;

  return 0;
}
```

Листинг 2: Исходный код решения задачи 2

2.2 Пояснения к алгоритму и оценка сложности

Нам нужно конкатенацией получить максимальное значение из заданных строк, для этого нам нужно выполнить следующую сортировку: попарно склеивать между собой две соседние строки и сравнивать их склейки между собой . То есть,

если при конкатенации двух строк a и b результат a + b будет больше, чем b + a, то a должна расположиться в отсортированном списке перед b. После сортировки склеиваем строки и получаем результат. Сложность алгоритма: сортировка массива слиянием (функцией sort) имеет сложность O(nlog(n)), поэтому общая сложность составляет O(nlog(n)).)

3 Задача G. Кошмар в замке (не решена)

3.1 Исходный код программы

```
Не решена :(
```

Листинг 3: Исходный код решения задачи F

3.2 Пояснения к алгоритму и оценка сложности

Нет.

4 Задача H. Магазин (решена)

4.1 Исходный код решения

```
#include <algorithm>
#include <cmath>
#include <iostream>
#include <vector>


using namespace std;

int main() {
  int n, k;
  cin >> n >> k;

  vector<int> bill(n);
  for (int i = 0; i < n; ++i) {
    cin >> bill[i];
  }

  sort(bill.begin(), bill.end());

  int sale = 0;
  int cost = 0;
  for (int i = 0; i < bill.size(); ++i) {
    cost += bill[i];
  }

  for (int i = bill.size() - k; i >= 0; i -= k) {
    sale += bill[i];
  }

  cost -= sale;
  cout << cost << endl;

  return 0;
}
```

Листинг 4: Исходный код решения задачи H

4.2 Пояснения к алгоритму и оценка сложности

В данном алгоритме нам нужно найти минимальную стоимость товаров с учетом того, что каждый k-ый товар в чеке становится бесплатным. Для решения данной задачи сначала мы сортируем список стоимостей товаров в порядке убывания (чтобы k-ыми выпадали именно самые дорогие товары), считаем начальную общую стоимость товаров в чеке и следующим образом высчитываем скидку: проходя по отсортированному списку товаров с конца с шагом -k, мы суммируем стоимость "бесплатных"k-ых товаров, а затем вычитаем ее из первоначальной общей стоимости.

Сложность алгоритма: сложность сортировки списка - O(nlog(n)), остальные операции линейны и обходятся в O(n), поэтому общая сложность составляет O(nlog(n)).