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 №1
по дисциплине «Алгоритмы и структуры данных»

Базовые задачи

Группа: P3218

Студент: Богданова Мария Михайловна

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

Санкт-Петербург
2024

Содержание

1 Задача 1 – Агроном-любитель (решена)

1.1 Исходный код решения

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

int main() {
  int n;
  std::cin >> n;

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

  int duplicate_cnt = 0;
  int current_longest_segment = 1;
  int longest_segment = 1;
  int current_start_content = 0;
  int start_content = 0;
  int i = 1;

  while (i < n) {
    if (current_longest_segment > longest_segment) {
      longest_segment = current_longest_segment;
      start_content = current_start_content;
    }

    if (i + 1 <= n) {
      if (flwr_type[i - 1] == flwr_type[i]) {
        duplicate_cnt += 1;
      } else {
        duplicate_cnt = 0;
      }
    }

    if (duplicate_cnt == 2) {
      current_longest_segment = 1;
      current_start_content = i - 1;
      duplicate_cnt = 0;
      i -= 1;
```

```
    } else {
      current_longest_segment += 1;
    }

    i += 1;
  }

  if (current_longest_segment > longest_segment) {
    start_content = current_start_content;
    longest_segment = current_longest_segment;
  }

  std::cout << start_content + 1 << " " << start_content +
    longest_segment
            << std::endl;

  return 0;
}
```

Листинг 1: Исходный код решения задачи 1

1.2 Пояснения к алгоритму и оценка сложности

Для решения данной задачи необходимо было найти максимальную длину линейного сектора, содержащего менее 3-х повторяющихся цветков, а для этого нам необходимо пройтись по всем цветкам и ввести указатель начала подходящего сектора, а также счетчики дубликатов и длины этого сектора. Как только на нашем счетчике набирается 3 одинаковых цветка (2 сходства текущего с последующим), мы сохраняем полученный отрезок вне цикла (если он строго больше полученных на предыдущих итерациях, причем строгое неравенство обеспечит тот факт, что из нескольких одинаковых секторов будет выбран самый первый) и уже внутри цикла обнуляем счетчик дубликатов, длину подходящего сектора и переносим указатель на предыдущий элемент относительно счетчика итераций(i), чтобы захватить максимальное количество подходящих цветов, и уже от него продолжаем заново отсчитывать длину искомого сектора, пока массив цветов не закончится.

Сложность алгоритма можно оценить следующим образом: так как программа должна один раз пройтись по массиву, в худшем случае придется перебрать все элементы и тогда сложность будет зависеть от длины массива - это O(n), внутри цикла все операции элементарны и их можно оценить как O(1), поэтому общая сложность алгоритма - линейная O(n).

2 Задача 2 – Зоопарк Глеба (не решена)

2.1 Исходный код решения

```
Не решена :(
```

Листинг 2: Исходный код решения задачи 2

2.2 Пояснения к алгоритму и оценка сложности

3 Задача 3 – Конфигурационный файл (не решена)

3.1 Исходный код программы

```
Не решена :(
```

Листинг 3: Исходный код решения задачи 3

3.2 Пояснения к алгоритму и оценка сложности

4 Задача 4 – Профессор Хаос (решена)

4.1 Исходный код решения

```
#include <iostream>
using namespace std;

int main() {
  int a, b, c, d;
  long k;
  cin >> a >> b >> c >> d >> k;
  int begin_day;

  long bacteria_number = a, current_day = 0;

  while (current_day < k) {
    current_day++;
    begin_day = bacteria_number;
    bacteria_number = bacteria_number * b;
    if (bacteria_number <= c) {
      cout << 0 << endl;
      return 0;
    }
    bacteria_number = bacteria_number - c;
    if (bacteria_number > d) {
      bacteria_number = d;
    }
    if (begin_day == bacteria_number) {
      break;
    }
  }

  cout << bacteria_number << endl;

  return 0;
}
```

Листинг 4: Исходный код решения задачи 4

4.2 Пояснения к алгоритму и оценка сложности

Для решения данной задачи нам необходимо вычислять количество оставшихся бактерий в конце каждого дня, пока не наступит k-ый день окончания эксперимента: мы берем полученные в начале эксперимента a бактерий, умножаем их на b (столько должно получаться бактерий после деления в инкубаторе) и вычитаем из них c бактерий, которые уничтожаются после окончания очередного опыта. После этих операций над бактериями, мы должны определить, хватает ли нам на очередной эксперимент с бактерий, если нет, то наш эксперимент вместе с программой завершается (1). Причем в конце каждого дня бактерии помещаются обратно в контейнер объемом d, не влезающие в него бактерии уничтожаются, поэтому в коде есть проверка на то, что если по окончанию первого дня бактерий уже больше, чем положено помещать в этот контейнер и количество бактерий в конце дня не меняется (2), итерации также прекращаются, ибо дальше их число не изменится и в последующих вычислениях отсутствует необходимость (1) и (2) - причины завершения работы программы до наступления дня k.

Сложность алгоритма: все операции внутри цикла простые и имеют сложность O(1), но так как в худшем случае мы должны считать количество бактерий, пока не наступит последний день n, сложность алгоритма будет линейной O(n).