**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технический университет имени Н. Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н. Э. Баумана)**

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

# РУБЕЖНЫЙ КОНТРОЛЬ № 1
# по дисциплине «Анализ алгоритмов»

**Студент** Ву Хай Данг

**Группа** ИУ7и-52Б

**Оценка (баллы)** ____________

**Преподаватель** Волкова Л. Л.

Москва — 2024 г.

# Содержание

# Задание

Что такое трудоёмкость в среднем?

Поиск 1-го вхождения такой пары чисел $x_1$ и $x_2$, что $x_1 < x_2$, $x_1$ и $x_2$ — вход алгоритма, как и массив целых чисел A из N элементов. Найти трудоёмкость в среднем, трудоёмкость в лучшем случае и в худшем случа1е.

# 1 Аналитическая часть

## 1.1 Трудоёмкость в худшем и лучшем случаях

Под худшим случаем будем понимать такой вход D длины n, на котором алгоритм A задает наибольшее количество элементарных операций, при этом максимум берется по всем $D \in D_n$. Трудоемкость алгоритма на этом входе будем называть трудоемкостью в худшем случае, и обозначать ее через $f_A^{\wedge}(n)$, тогда

$$f_A^{\wedge}(n) = \max_{D \in D_n} \{f_A(D)\} \tag{1.1}$$

по аналогии через $f_A^{\vee}(n)$ будем обозначать трудоемкость в лучшем случае, как трудоемкость с наименьшим количеством операций на всех входах длины n:

$$f_A^{\vee}(n) = \min_{D \in D_n} \{f_A(D)\} \tag{1.2}$$

## 1.2 Трудоёмкость в среднем

Трудоемкость алгоритма в среднем — это среднее количество операций, задаваемых алгоритмом A на входах длины n, где усреднение берется по всем $D \in D_n$. Введем для трудоемкости в среднем обозначение $\overline{f_A}(n)$, тогда

$$\overline{f_A}(n) = \sum_{D \in D_n} p(D) \cdot f_A(D), \tag{1.3}$$

где p(D) есть частотная встречаемость входа D для анализируемой области применения алгоритма.

## 1.3 Алгоритм полного перебора

Алгоритмом полного перебора называют метод решения задачи, при котором по очереди рассматриваются все возможные варианты. В случае реализации алгоритма в рамках данной работы будут последовательно перебираться 2 соседние элемента до тех пор, пока не будет найден нужный.

# 2 Конструкторская часть

В данном разделе будут представлены схемы алгоритма поиска полным перебором ищет соседние 2 элемента, требования к программному обеспечению.

## 2.1 Разработка алгоритмов

На рисунке 2.1 приведена схема алгоритма поиска полным перебором соседние 2 элемента.

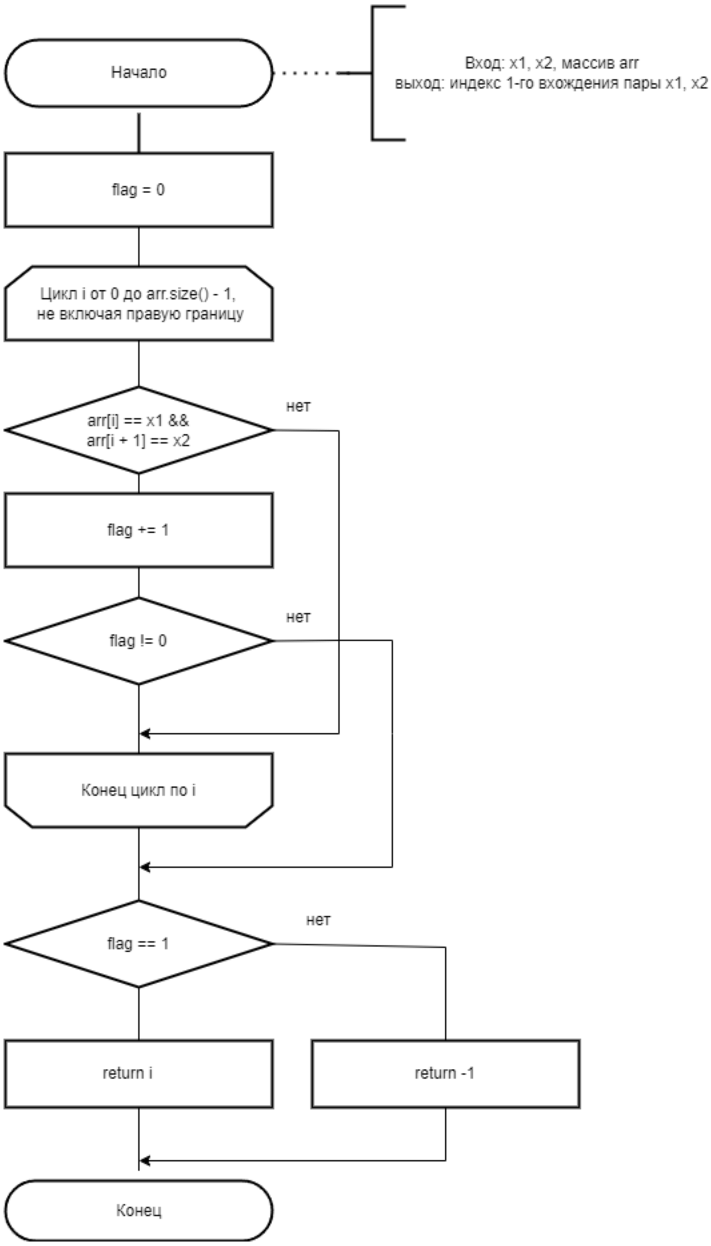


Рисунок 2.1 – Алгоритм полным перебором соседние 2 элемента

## 2.2 Модель вычислений для оценки трудоемкости алгоритмов

Для определения трудоемкости алгоритмов необходимо ввести модель вычислений [3]:

1) операции из списка (2.1) имеют трудоемкость равную 2;

$$*, /, \%, *=, /=, \%= \tag{2.1}$$

2) операции из списка (2.2) имеют трудоемкость равную 1;

$$+, -, +=, -=, =, ==, !=, <, >, <=, >=, [], ++, -- \tag{2.2}$$

3) трудоемкость оператора выбора `if условие then A else B` рассчитывается, как (2.3);

$$f_{if} = f_{\text{условия}} + \begin{cases} f_A, & \text{если условие выполняется,} \\ f_B, & \text{иначе.} \end{cases} \tag{2.3}$$

4) трудоемкость цикла рассчитывается, как (2.4);

$$f_{for} = f_{\text{инициализации}} + f_{\text{сравнения}} + N(f_{\text{тела}} + f_{\text{инкремент}} + f_{\text{сравнения}}) \tag{2.4}$$

5) трудоемкость вызова функции равна 0.

## 2.3 Трудоемкость алгоритмов

Трудоемкость данного алгоритма будет складываться из следующих:

— трудоемкости создания флаг, равной 1;

— трудоемкости инициализации и сравнения цикла, равной 4;

— трудоемкости тела, инкремента и сравнения цикла по $i$ в каждой итерации, равной 10;

— трудоемкости выходного условия для функции, равной 1;

В худшем случае — количество итераций в цикле будет максимально, это ситуация, когда элементы исходного массива различны и не нашел результат или результат находится в конце массива. В этом случае трудоемкость алгоритма равна:

$$f_A^{\wedge}(n) = 1 + 4 + (N-1) \cdot 10 + 1 = 10 \cdot N - 4 = O(N) \tag{2.5}$$

В лучшем случае — количество итераций в цикле будет минимально, это ситуация, когда элементы исходного массива различны и результат находится в начале массива. В этом случае трудоемкость алгоритма равна:

$$f_A^{\vee}(n) = 1 + 4 + 10 + 1 = 16 = (1) \tag{2.6}$$

В среднем случае. Пусть алгоритм поиска без выполнения цикла затрачивает $k_0 = 6$ операций, а при каждой итерации $k_1 = 10$ операций. алгоритм нашёл элементы на первом сравнении (лучший случай), тогда будет затрачено $k_0 + k_1$ операций, на втором — $k_0 + 2 \cdot k_1$, на последнем (худший случай) — $k_0 + (N-1) \cdot k_1$. Если такие элементы нет в массиве, то это, только перебрав все элементы, таким образом трудоёмкость такого случая равно трудоёмкости худшем случае.

Средняя трудоёмкость алгоритма может быть рассчитана как математическое ожидание по формуле 2.7 ($\Omega$ - множество всех возможных исходов).

$$\sum_{i\in\Omega} p_i \cdot f_i = (k_0 + k_1) \cdot \frac{1}{N+1} + (k_0 + 2 \cdot k_1) \cdot \frac{1}{N+1} + (k_0 + 3 \cdot k_1) \cdot \frac{1}{N+1} +$$
$$+ (k_0 + (N-1) \cdot k_1) \frac{1}{N+1} + (k_0 + (N-1) \cdot k_1) \cdot \frac{1}{N+1} =$$
$$= k_0 \frac{N+1}{N+1} + k_1 + \frac{1 + 2 + \cdots + N - 1 + N - 1}{N+1} = k_0 + k_1 \cdot \left( \frac{N-3}{N+1} + \frac{N}{2} \right) =$$
$$= k_0 + k_1 \cdot \left( 1 + \frac{N}{2} - \frac{4}{N+1} \right) = 6 + 10 \cdot \left( 1 + \frac{N}{2} - \frac{4}{N+1} \right) = O(N) \quad (2.7)$$

## 2.4 Требования к программному обеспечению

К программе предъявляются следующие требования:

- входные данные — массив целых чисел, 2 разные число $x_1$ и $x_2$, $x_1 < x_2$;
- выходные данные — индекс в массив, если найти результат, иначе -1.

# Вывод

Были представлены схемы алгоритмов поиска полным перебором ищет соседние 2 элемента и требования к программному обеспечению. Проведённая теоретическая оценка трудоемкости алгоритмов показала, что в трудоёмкость выполнения данного алгоритма равна $O(N)$ в худшем случае, $O(1)$ в лучшем случае и $O(N)$ в среднем случае.

# 3 Технологическая часть

В данном разделе будут указаны средства реализации, будут представлены реализация алгоритма, а также функциональные тесты.

## 3.1 Средства реализации

Данная программа разработана на языке C++, поддерживаемом многими операционными системами.

## 3.2 Реализация алгоритмов

Алгоритм поиска полным перебором ищет соседние 2 элемента приведены в листинге 3.1.

```
int solution(vector<int> arr, int x1, int x2)
{
  int flag = 0;
  int i = 0;
  for (; i < arr.size() - 1; i++)
  {
    if (arr[i] == x1 &&  arr[i + 1] == x2)
    flag = 1;
    if (flag == 1)
    break;
  }
  if (flag == 1)
  return i;
  else
  return -1;
}
```

Листинг 3.1 – Алгоритм поиска полным перебором

## 3.3 Функциональные тесты

В таблице 3.1 приведены функциональные тесты для функций, реализующих алгоритмы. Все тесты пройдены успешно.

Таблица 3.1 – Функциональные тесты

| № | Входные данные | Ожидаемый результат |
|---|---|---|
| 1 | 5 1 8 2 3 5 2 3 | 2 |
| 2 | 5 1 8 2 3 5 2 4 | -1 |
| 3 | 5 1 8 2 3 5 4 2 | Ошибка ввода |

## 3.4 Вывод

Были реализованы функции поиска полным перебором. Было проведено функциональное тестирование данного алгоритма.

# 4 Исследовательская часть

В данном разделе будут приведены примеры работы программы и предоставлена информация о технических характеристиках устройства.

## 4.1 Технические характеристики

Ниже представлены характеристики компьютера, на котором проводились замеры времени работы реализации алгоритмов:

— операционная система Windows 10 Домашняя 21H2;

— оперативная память 12 Гб;

— процессор Intel(R) Core(TM) i7-9750H CPU @ 2.6 ГГц.

## 4.2 Демонстрация работы программы

На рисунках 4.1 – 4.3 приведены примеры работы программы как показан в таблице 3.1.

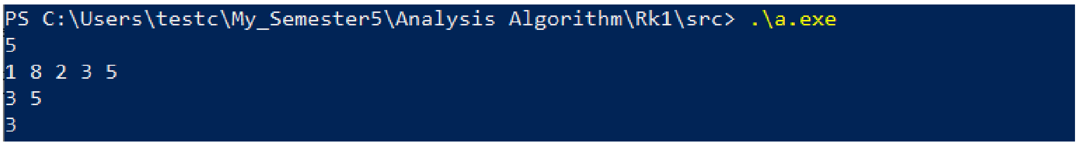


Рисунок 4.1 – Пример работы данного алгоритма при успешно поиске

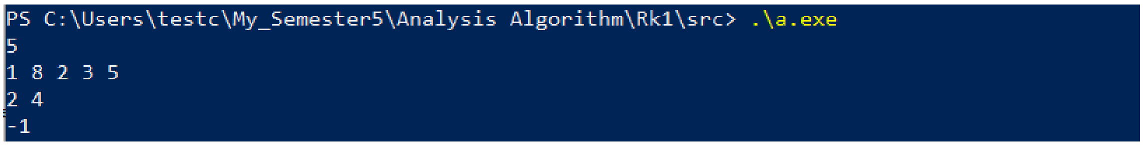


Рисунок 4.2 – Пример работы данного алгоритма при неудачном поиске

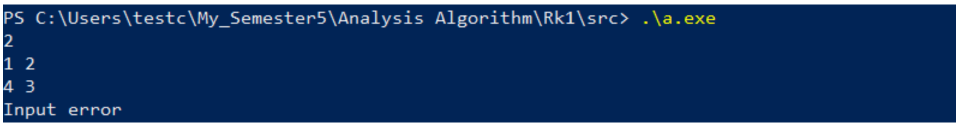


Рисунок 4.3 – Пример работы данного алгоритма при неправильном вводе

## 4.3 Вывод

В данном разделе были приведены примеры работы программы.

# Заключение

Проведённая теоретическая оценка трудоемкости алгоритмов показала, что в трудоёмкость выполнения данного алгоритма равна $O(N)$ в худшем случае, $O(1)$ в лучшем случае и $O(N)$ в среднем случае.

# Список использованных источников

1. Кнут Д. Э. Искусство программирования, том 1. Основные алгоритмы, 3-е изд.: Пер. с англ.— М.: Издательский дом «Вильямс», 2002. — 720 с.

2. Офман Ю. П. Об алгоритмической сложности дискретных функций // Доклады АН СССР. 1962. Т. 45. Вып. 1. С.48–51.

3. М. В. Ульянов Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ. —М.: Физматлит, 2007. — 304 с.