САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе № 2

по курсу «Алгоритмы и структуры данных»

Тема: Сортировка слиянием. Метод декомпозиции

Вариант 2

Выполнила:

Азизова Наида Элимизовна

К3140

Проверил(а):

Афанасьев А. В.

Санкт-Петербург

2024 г.

Содержание отчёта

[Задания 3](#_Toc185368519)

[Задача №1. Сортировка слиянием 3](#_Toc185368520)

[Задача №3. Число инверсий 5](#_Toc185368521)

[Задача №4. Бинарный поиск 8](#_Toc185368525)

[Задача №5. Представитель большинства 10](#_Toc185368529)

[Задача №7. Поиск максимального подмассива за линейное время 12](#_Toc185368533)

[Вывод 14](#_Toc185368534)

# Задания

## Задача №1. Сортировка слиянием

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Решение:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Текстовое объяснение решения:

1. **Функция merge(A, p, q, r)** — сливает два отсортированных подмассива в один отсортированный массив. Подмассивы находятся в интервалах [p..q] и [q+1..r].
2. **Функция merge\_sort(A, p, r)** — рекурсивно разделяет массив на более мелкие части (пока не останется одноэлементных массивов), затем сливает их обратно, используя функцию merge.
3. **Основной блок (if \_\_name\_\_ == '\_\_main\_\_')**:

* Определяется путь к файлу с входными данными.
* Читаются данные из файла с помощью функции utils.read\_input.
* Массив данных сортируется с помощью merge\_sort.
* Выводится результат с помощью utils.print\_task\_data

Результат работы кода на примерах из текста задачи:

Ввод:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Вывод:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Вывод по задаче: Я научилась осуществлять сортировку слиянием.

## Задача №3. Число инверсий

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Решение:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Текстовое объяснение решения:

### **Функция** merge(A, p, q, r, inversion\_count)

* Сливает два отсортированных подмассива в один.
* Дополнительно считает инверсии:
* Когда элемент из правого подмассива R[j]R[j]R[j] меньше элемента из левого подмассива L[i]L[i]L[i], то все оставшиеся элементы в левом подмассиве (от iii до конца) будут больше R[j]R[j]R[j]. Это создаёт инверсии, и количество таких инверсий добавляется к счётчику.

### 2. **Функция** merge\_sort(A, p, r, inversion\_count)

* Рекурсивно делит массив пополам и сортирует каждую половину.
* Каждый раз, когда выполняется слияние двух отсортированных частей, обновляется счётчик инверсий.

### 3. **Основной блок (**if \_\_name\_\_ == '\_\_main\_\_'**)**

* Определяет путь к файлу с входными данными.
* Читает данные с помощью utils.read\_input.
* Вызывает функцию merge\_sort, чтобы отсортировать массив и подсчитать количество инверсий.
* Выводит результат с помощью utils.print\_task\_data, передавая количество инверсий как результат.

Результат работы кода на примерах задачи:

Ввод:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Вывод:

Изображение выглядит как снимок экрана, Шрифт, черный, текст

Автоматически созданное описание

Вывод по задаче: Я научилась считать количество инверсий в массиве при его сортировке.

## Задача №4. Бинарный поиск

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Решение:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Текстовое объяснение решения:

### **Функция** bin\_search(A, target, l, r)

* Это стандартная функция бинарного поиска, которая ищет элемент target в отсортированном массиве A в пределах индексов от l до r.
* Если элемент найден, возвращается его индекс.
* Если элемент не найден, возвращается -1.

### **Функция** find\_index(A, B, n)

* Функция принимает два массива: A (отсортированный) и B.
* Для каждого элемента из массива B выполняется бинарный поиск в массиве A с помощью функции bin\_search.
* Результаты (индексы элементов из массива B в массиве A) добавляются в список res.

### **Основной блок (**if \_\_name\_\_ == '\_\_main\_\_'**)**

* Определяется путь к файлу с входными данными.
* Данные считываются с помощью функции utils.read\_input.
* Затем вызывается функция find\_index, которая находит индексы элементов из массива B в массиве A и возвращает список индексов.
* Результаты выводятся с помощью функции utils.print\_task\_data.

Результат работы кода на примерах задачи:

Ввод:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Вывод:

Изображение выглядит как текст, часы, Шрифт, снимок экрана

Автоматически созданное описание

Вывод по задаче: Я научилась осуществлять бинарный поиск

## Задача №5. Представитель большинства

Изображение выглядит как текст, Шрифт

Автоматически созданное описание

**Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание**

Решение:  
Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Текстовое объяснение решения:

### **1.Функция** count\_majority(A, target, l, r)

* Эта функция подсчитывает, сколько раз элемент target встречается в подмассиве A[l..r]. Для этого она перебирает элементы массива в указанном диапазоне и увеличивает счётчик, если элемент равен target.

### 2. **Функция** majority(A, l, r)

* Функция рекурсивно находит кандидат на элемент большинства в подмассиве A[l..r].
* Алгоритм использует метод "разделяй и властвуй":
  + Если диапазон содержит только один элемент (то есть l==r), то этот элемент — кандидат.
  + Если диапазон больше, то массив делится на два подмассива и рекурсивно ищется большинство в каждом из них.
  + Если большинство в обеих частях совпадает, то оно считается кандидатом.
  + Если кандидат в левой и правой частях не совпадает, то проводится подсчёт, сколько раз каждый кандидат встречается в диапазоне [l..r]. Если один из кандидатов встречается больше половины раз, он становится кандидатом на большинство.

### 3. **Функция** has\_majority(A, n)

* Эта функция вызывает функцию majority для поиска кандидата на большинство в массиве.
* Затем она проверяет, встречается ли этот элемент более половины раз в массиве. Если да, возвращает 1 (есть большинство), иначе — 0 (большинства нет).

Результат работы кода на примерах задачи:

Ввод:

Изображение выглядит как снимок экрана, Шрифт, число, часы

Автоматически созданное описание

Вывод:

Изображение выглядит как снимок экрана, дизайн

Автоматически созданное описание

Вывод по задаче: Научилась осуществлять поиск большинства в массиве, используя алгоритм, похожий на разделяй и властвуй.

## Задача №7. Поиск максимального подмассива за линейное время

Изображение выглядит как текст, Шрифт, снимок экрана, черно-белый

Автоматически созданное описание

Решение:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Текстовое объяснение решения:

* Инициализируем две переменные:

current\_sum — текущая сумма элементов подмассива, который мы рассматриваем в данный момент.

max\_sum — максимальная сумма среди всех рассмотренных подмассивов.

* Итерируем по всем элементам массива:

Для каждого элемента проверяем, стоит ли начинать новый подмассив с этого элемента или продолжать текущий.

Если продолжение текущего подмассива с этим элементом даёт большую сумму, продолжаем, иначе начинаем новый подмассив с текущего элемента.

* На каждом шаге обновляем значение max\_sum, если текущая сумма подмассива (current\_sum) больше, чем ранее найденная максимальная сумма.

Результат работы кода на примерах задачи:

Ввод:

Изображение выглядит как текст, Шрифт, часы, снимок экрана

Автоматически созданное описание

Вывод:

Изображение выглядит как часы, снимок экрана, Шрифт, текст

Автоматически созданное описание

Вывод по задаче: Я научилась осуществлять поиск максимального подмассива за линейное время.

# Вывод

В ходе лабораторной работы я поняла, что рекурсивные алгоритмы, основанные на принципе разделяй и властвуй можно использовать во многих задачах, которые легко задаются правилом упрощения.