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 1

Выполнил:
Бен Шамех Абделазиз
Группа: K3239

Проверила:
Ромакина Оксана Михайловна

Санкт-Петербург
2025 г.

Содержание

1 Задача 1. Сортировка слиянием

1.1 Условие

Используя псевдокод процедур Merge и Merge-sort, напишите программу сортировки слиянием на Python. Перепишите Merge так, чтобы в ней не использовались сигнальные значения.

1.2 Листинг кода

```python
def merge_sort(arr, p, r):
    if p < r:
        q = (p + r) // 2
        merge_sort(arr, p, q)
        merge_sort(arr, q + 1, r)
        merge(arr, p, q, r)

def merge(arr, p, q, r):
    arr1 = arr[p:q+1]
    arr2 = arr[q+1:r+1]
    i = j = 0
    while i < len(arr1) and j < len(arr2):
        if arr1[i] < arr2[j]:
            arr[p+i+j] = arr1[i]
            i += 1
        else:
            arr[p+i+j] = arr2[j]
            j += 1
    while i < len(arr1):
        arr[p+i+j] = arr1[i]; i += 1
    while j < len(arr2):
        arr[p+i+j] = arr2[j]; j += 1
```

1.3 Текстовое объяснение

Метод разделяй и властвуй: рекурсивно делим массив пополам до единичных элементов и выполняем на возврате слияние двух отсортированных массивов в один, сравнивая элементы по очереди.

1.4 Результаты выполнения

Ввод: `6 \n 31 41 59 26 41 58`. **Вывод:** `26 31 41 41 58 59`.

Метрики эффективности:

Сценарий	Время выполнения	Затраты памяти
Нижняя граница (n=1)	0.00022 sec	15.54 Mb
Пример из задачи	0.00016 sec	15.54 Mb
Верхняя граница	0.02904 sec	16.51 Mb

2 Задача 2. Сортировка слиянием+

2.1 Условие

Отсортировать массив, выводя после каждого слияния индексы начала и конца, а также значения первого и последнего элементов отсортированного участка.

2.2 Листинг кода

```
def merge_sort_with_logging(arr, left, right, logs):
    if left < right:
        mid = (left + right) // 2
        merge_sort_with_logging(arr, left, mid, logs)
        merge_sort_with_logging(arr, mid + 1, right, logs)

        # Merge logic
        L = arr[left:mid+1]; R = arr[mid+1:right+1]
        i = j = 0; k = left
        while i < len(L) and j < len(R):
            if L[i] <= R[j]:
                arr[k] = L[i]; i += 1
            else:
                arr[k] = R[j]; j += 1
            k += 1
        while i < len(L): arr[k] = L[i]; i += 1; k += 1
        while j < len(R): arr[k] = R[j]; j += 1; k += 1

        logs.append(f"{left+1} {right+1} {arr[left]} {arr[right]}")
```

2.3 Результаты выполнения

Ввод: 10 \n 1 8 2 1 4 7 3 2 3 6.
Вывод (логи слияний + массив):
1 2 1 8
4 5 1 4
...
1 10 1 8
1 1 2 2 3 3 4 6 7 8

3 Задача 3. Число инверсий

3.1 Условие

Подсчитать количество пар (i, j) таких, что $i < j$ и $A[i] > A[j]$.

3.2 Текстовое объяснение

Во время процесса слияния, если элемент выбирается из правого подмассива, он образует инверсию со всеми оставшимися элементами в левом подмассиве. Суммируем эти значения для получения итога.

3.3 Результаты выполнения

Ввод: `10 \n 1 8 2 1 4 7 3 2 3 6`. **Вывод:** `17`.

Метрики эффективности:

Сценарий	**Время выполнения**	**Затраты памяти**
Нижняя граница	0.0001 sec	15.55 Mb
Верхняя граница (n=10^5)	0.18916 sec	16.18 Mb

4 Задача 4. Бинарный поиск

4.1 Условие

Реализовать бинарный поиск для нахождения индексов элементов в отсортированном массиве.

4.2 Листинг кода

```
def bs(arr, key):
    lt = -1
    rt = len(arr)
    while rt - lt > 1:
        m = (rt + lt) // 2
        if key > arr[m]: lt = m
        else: rt = m
    if rt < len(arr) and arr[rt] == key: return rt
    else: return -1
```

4.3 Результаты выполнения

Ввод: `5 1 5 8 12 13 \n 5 8 1 23 1 11`. **Вывод:** `2 0 -1 0 -1`.

5 Задача 5. Представитель большинства

5.1 Условие

Проверить наличие элемента, встречающегося более $n/2$ раз, методом «Разделяй и властвуй» за $O(n \log n)$.

5.2 Листинг кода

```python
def majority(arr, lt, rt):
    if lt == rt: return arr[lt]
    m = (lt + rt) // 2
    lt_item = majority(arr, lt, m)
    rt_item = majority(arr, m + 1, rt)

    if lt_item == rt_item: return lt_item

    cnt_lt = arr[lt:rt+1].count(lt_item)
    cnt_rt = arr[lt:rt+1].count(rt_item)
    return lt_item if cnt_lt > cnt_rt else rt_item
```

5.3 Текстовое объяснение

Рекурсивно разбиваем массив. На этапе слияния проверяем, какой из двух кандидатов (из левой и правой половин) встречается чаще в текущем диапазоне. Побеждает тот, кто набирает больше голосов.

5.4 Метрики эффективности:

Сценарий	**Время выполнения**	**Затраты памяти**
Пример из задачи	0.00014 sec	15.54 Mb
Верхняя граница	0.03461 sec	16.92 Mb

6 Задача 6. Поиск максимальной прибыли

6.1 Условие

Найти максимальный подмассив (период наибольшей прибыли акций) методом декомпозиции.

6.2 Листинг кода

```python
def max_subarray(arr, low, high):
    if low == high: return (low, high, arr[low])
    m = (low + high) // 2
    l_l, l_h, l_s = max_subarray(arr, low, m)
    r_l, r_h, r_s = max_subarray(arr, m + 1, high)
    c_l, c_h, c_s = max_cross_subarray(arr, low, m, high)
    if l_s >= r_s and l_s >= c_s: return (l_l, l_h, l_s)
    elif r_s >= l_s and r_s >= c_s: return (r_l, r_h, r_s)
    else: return (c_l, c_h, c_s)

def max_cross_subarray(arr, low, m, high):
    lt_sum = rt_sum = float('-inf'); tmp = 0
    for i in range(m, low - 1, -1):
        tmp += arr[i]
        if tmp > lt_sum: lt_sum = tmp; mx_l = i
    tmp = 0
    for i in range(m + 1, high + 1):
        tmp += arr[i]
        if tmp > rt_sum: rt_sum = tmp; mx_r = i
    return (mx_l, mx_r, lt_sum + rt_sum)
```

6.3 Результаты выполнения

Вывод:

```
Фирма: ПАО Газпром
Дата покупки: 08.01.2024
Дата продажи: 09.01.2024
Прибыль: 38
```

7 Задача 7. Поиск максимального подмассива (Линейно)

7.1 Листинг кода

```python
def max_subarray_linear(arr):
    min_pref = min(arr[0], 0)
    max_sum = cur_sum = arr[0]
    for i in range(1, len(arr)):
        cur_sum += arr[i]
        max_sum = max(max_sum, cur_sum - min_pref)
        min_pref = min(cur_sum, min_pref)
    return max_sum
```

7.2 Результаты выполнения

Ввод: `6 \n -2 1 -3 4 -1 2`. **Вывод:** `5`.

Метрики эффективности:

Сценарий	Время выполнения	Затраты памяти
Нижняя граница	0.00808 sec	29.70 Mb
Верхняя граница	0.10914 sec	31.95 Mb

8 Вывод

1. Реализована и протестирована сортировка слиянием (*Merge Sort*) со сложностью $O(n \log n)$.

2. Было практически подтверждено, что метод «Разделяй и властвуй» позволяет решать сложные задачи поиска и анализа за логарифмическое или линейно-логарифмическое время.

3. Линейный алгоритм Кадана (задача 7) продемонстрировал наибольшую скорость выполнения при поиске максимального подмассива на больших данных.