МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ»

Федеральное государственное бюджетное образовательное учреждение
высшего образования
АДЫГЕЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Инженерно-физический факультет
Кафедра автоматизированных систем обработки информации и управления

Отчет по практике

« Реализация методов быстрой сортировки и сортировки слиянием.»

Вариант 4

2 курс, группа 2ИВТ2

Выполнил:
В.Д. Богомолов 2024 г

Руководитель:

С.В. Теплоухов 2024 г

Майкоп, 2024 г.

0.1. Введение

1. Текстовая формулировка задачи
2. Код данной задачи
3. Скриншот программы

0.2. Вариант 3

0.2.1. Задание

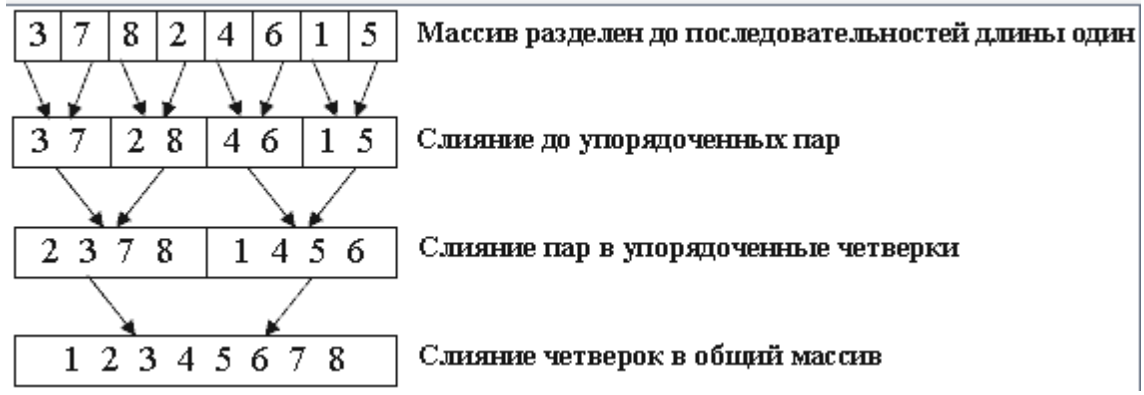
Сортировка массива: Быстрая и Слиянием.

0.2.2. Теория

"Быстрая сортировка хоть и была разработана более 40 лет назад, является наиболее широко применяемым и одним их самых эффективных алгоритмов. Метод основан на подходе "разделяй-и-властвуй". Общая схема такова: • из массива выбирается некоторый опорный элемент a[i], • запускается процедура разделения массива, которая перемещает все ключи, меньшие, либо равные a[i], влево от него, а все ключи, большие, либо равные a[i] - вправо, • теперь массив состоит из двух подмножеств, причем левое меньше, либо равно правого, • для обоих подмассивов: если в подмассиве более двух элементов, рекурсивно запускаем для него ту же процедуру.
Сортировка слиянием также построена на принципе "разделяй-и-властвуй однако реализует его несколько по-другому, нежели quickSort. А именно, вместо разделения по опорному элементу массив просто делится пополам. Функция merge на месте двух упорядоченных массивов a[lb]...a[split] и a[split+1]...a[ub] создает единый упорядоченный массив a[lb]...a[ub].
Рекурсивный алгоритм обходит получившееся дерево слияния в прямом порядке. Каждый уровень представляет собой проход сортировки слияния - операцию, полностью переписывающую массив. Обратим внимание, что деление происходит до массива из единственного элемента. Такой массив можно считать упорядоченным, а значит, задача сводится к написанию функции слияния merge. Один из способов состоит в слиянии двух упорядоченных последовательностей при помощи вспомогательного буфера, равного по размеру общему количеству имеющихся в них элементов.

Быстрая сортировка:

Сортировка слиянием:



0.3. Ход работы

0.3.1. Код программы

Реализовать метод быстрой сортировки и сортировки слиянием

```
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    else:
        pivot = arr[0]
        less = [x for x in arr[1:] if x <= pivot]
        greater = [x for x in arr[1:] if x > pivot]
        return quick_sort(less) + [pivot] + quick_sort(greater)

def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    left_half = merge_sort(left_half)
    right_half = merge_sort(right_half)

    return merge(left_half, right_half)

def merge(left, right):
    result = []
    i = j = 0
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1

    result += left[i:]
    result += right[j:]
```

```
    return result

arr = [5, 3, 8, 6, 2, 7, 1, 4]

print("Исходный массив:", arr)
print("Отсортированный массив с помощью быстрой сортировки:", quick_sort
print("Отсортированный массив с помощью сортировки слиянием:", merge_sor
```

0.4. Скриншот программы

Реализовать алгоритм сортировки массива(быстрой и слиянием).

```python
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    else:
        pivot = arr[0]
        less = [x for x in arr[1:] if x <= pivot]
        greater = [x for x in arr[1:] if x > pivot]
        return quick_sort(less) + [pivot] + quick_sort(greater)

def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    left_half = merge_sort(left_half)
    right_half = merge_sort(right_half)

    return merge(left_half, right_half)

def merge(left, right):
    result = []
    i = j = 0
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1

    result += left[i:]
    result += right[j:]

    return result

arr = [5, 3, 8, 6, 2, 7, 1, 4]

print("Исходный массив:", arr)
print("Отсортированный массив с помощью быстрой сортировки:", quick_sort(arr.copy()))
print("Отсортированный массив с помощью сортировки слиянием:", merge_sort(arr.copy()))
```

Рис. 0.1: Скриншот программы

0.5. Библиографические ссылки

Для изучения «внутренностей» TEX необходимо изучить [1], а для использования LATEX лучше почитать [2, 3].

Литература

[1] Кнут Д.Э. Всё про TEX. —- Москва: Изд. Вильямс, 2003 г. 550 с.

[2] Львовский С.М. Набор и верстка в системе LATEX. —- 3-е издание, исправленное и дополненное, 2003 г.

[3] Воронцов К.В. LATEX в примерах. 2005 г.