САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1 по курсу «Алгоритмы и структуры данных»

Тема: сортировка вставками, выбором, пузырьковая

Вариант 2

Выполнил:
Шаповалов
С.К
К3141

Проверила:
Артамонова В.Е.

Санкт-Петербург
2023 г.

Содержание отчета

1 задача. Сортировка слиянием

1. Используя *псевдокод* процедур Merge и Merge-sort из презентации к Лек- ции 2 (страницы 6-7), напишите программу сортировки слиянием на Python и проверьте сортировку, создав несколько рандомных массивов, подходящих под параметры:

- **Формат входного файла (input.txt).** В первой строке входного файла содержится число n ($1 \leq n \leq 2 \cdot 104$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 109.
- **Форматвыходногофайла(output.txt).**Однастрокавыходногофайла с отсортированным массивом. Между любыми двумя числами должен стоять ровно один пробел.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб.

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

    mid = len(arr) // 2
    left = arr[:mid]
    right = arr[mid:]

    left = merge_sort(left)
    right = merge_sort(right)
```

```python
    return merge(left, right)

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.extend(left[i:])
    result.extend(right[j:])
    return result


with open('input_1.txt', 'r') as f:
    n = int(f.readline())
    array = list(map(int, f.readline().split()))


sorted_array = merge_sort(array)

with open('output_1.txt', 'w') as f1:
    f1.write(str(sorted_array))
```

input_1.txt

```
1    10
2    1 8 2 1 4 7 3 2 3 6
```

output_1.txt

```
1    [1, 1, 2, 2, 3, 3, 4, 6, 7, 8]
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.00043750001350417733 секунд	14.390625 МБ
Пример из задачи	0.00039187504444271326	14.203125 МБ
Верхняя граница диапазона значений входных данных из текста задачи	0.00045708403922617435 секунд	14.46875 МБ

3 задача. Число инверсий

Инверсией в последовательности чисел A называется такая ситуация, когда i < j, а Ai > Aj. Количество инверсий в последовательности в некотором ро- де определяет, насколько близка данная последовательность к отсортированной. Например, в сортированном массиве число инверсий равно 0, а в массиве, сортированном наоборот - каждые два элемента будут составлять инверсию (всего n(n − 1)/2).

Дан массив целых чисел. Ваша задача — подсчитать число инверсий в нем.

```
total=0

def merge_sort(arr):

   if len(arr) <= 1:

       return arr
```

```
    mid = len(arr) // 2

    left = arr[:mid]

    right = arr[mid:]



    left = merge_sort(left)

    right = merge_sort(right)



    return merge(left, right)



def merge(left, right):

    global total  # Используем глобальную переменную total для подсчета инверсий

    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

            total += len(left) - i



    result.extend(left[i:])

    result.extend(right[j:])

    return result
```

```
with open('input_1.txt', 'r') as f:

   n = int(f.readline())

   array = list(map(int, f.readline().split()))



sorted_array = merge_sort(array)


with open('output_3.txt', 'w') as f1:

   f1.write(str(total))
```

input_1.txt

```
1    10
2    1 8 2 1 4 7 3 2 3 6
```

output_3.txt

```
1    20
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.00023750001350417733 секунд	13.390625 МБ
Пример из задачи	0.00039187504444271326	14.003125 МБ

Верхняя граница диапазона значений входных данных из текста задачи	0.0006765839643776417	14.46875 МБ

Вывод по задаче: я просто добавил тотал)

4 задача. Бинарный поиск

В этой задаче вы реализуете алгоритм бинарного поиска, который позволяет очень эффективно искать (даже в огромных) списках при условии, что список отсортирован. Цель - реализация алгоритма двоичного (бинарного) поиска.

```
def binary_search(arr, target):
    left, right = 0, len(arr) - 1

    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1

    return -1



with open('input_4.txt', 'r') as f:
    n = int(f.readline())
    array = list(map(int, f.readline().split()))
    k = int(f.readline())
    targets = list(map(int, f.readline().split()))

indices = []
for target in targets:
    index = binary_search(array, target)
    indices.append(index)
```

```
with open('output_4.txt', 'w') as f1:
    for index in indices:
        f1.write(str(index) + ' ')
```

input_4.txt

```
5
1 5 8 12 13
5
8 1 23 1 11
```

output_4.txt

```
2 0 -1 0 -1
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000293958000838756 56 секунд	14.203125 МБ
Пример из задачи	0.000260082990280352 53 секунд	13.890625 МБ
Пример из задачи	0.04503050001221709 секунд	14.0 МБ
Верхняя граница диапазона значений входных данных из текста задачи	1.3461720830091508 секунд	14.1875 МБ

Вывод по задаче: эта задача показалась мне наиболее интересной из всех

Дополнительные задачи:

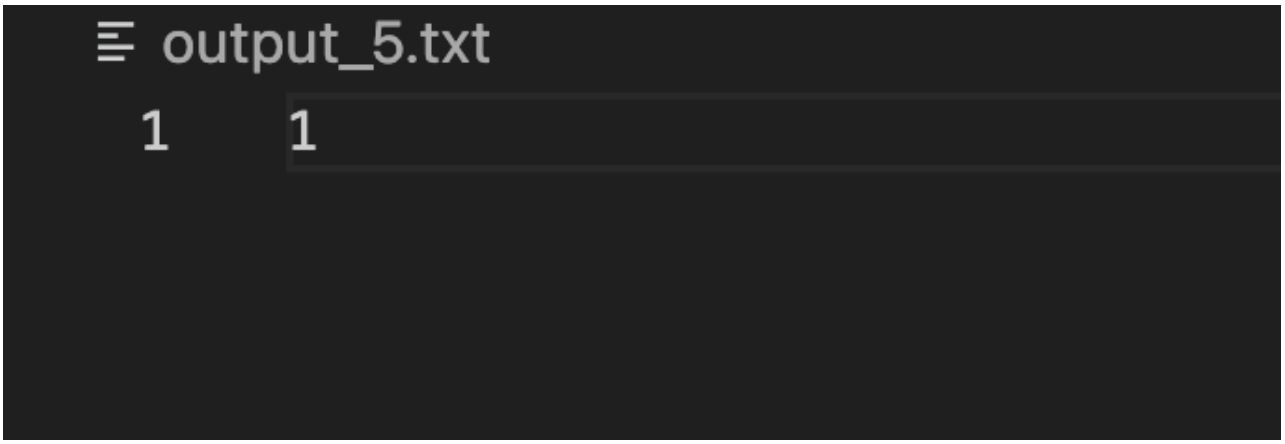
5 задача. Представитель большинства

Правило большинства - это когда выбирается элемент, имеющий больше половины голосов. Допустим, есть последовательность A элементов a1, a2, ...an, и нужно проверить, содержит ли она элемент, который появляется больше, чем n/2 раз. Наивный метод это сделать:

Очевидно, время выполнения этого алгоритма квадратично. Ваша цель - использовать метод "Разделяй и властвуй"для разработки алгоритма проверки, содержится ли во входной последовательности элемент, который встречается боль- ше половины раз, за время O(n log n).

```
            def fme(arr):
    can = None
    total = 0
```

```
    for e in arr:
        if total == 0:
            can = e
            total = 1
        elif can == e:
            total += 1
        else:
            total -= 1

    total = 0
    for e in arr:
        if e == can:
            total += 1

    if total > n // 2:
        return 1
    else:
        return 0

with open('input_5.txt', 'r') as f:
    n = int(f.readline())
    arr = list(map(int, f.readline().split()))

result = fme(arr)

with open('output_5.txt', 'w') as f1:
    f1.write(str(result))
```



7 задача. Поиск максимального подмассива за линейное время

Можно найти максимальный подмассив за линейное время, воспользовавшись следующими идеями. Начните с левого конца массива и двигайтесь вправо, отсле- живая найденный к данному моменту максимальный подмассив. Зная максималь- ный подмассив массива A[1..j], распространите ответ на поиск максимального подмассива, заканчивающегося индексом j + 1, воспользовавшись следующим наблюдением: максимальный подмассив массива A[1..j + 1] представляет собой либо максимальный подмассив массива A[1..j], либо подмассив A[i..j + 1] для некоторого $1 \le i \le j + 1$. Определите максимальный подмассив вида A[i..j + 1] за константное время, зная максимальный подмассив, заканчивающийся индексом j.

В этом случае у вас возможны 2 варианта тестирования: первый предполагает создание рандомного массива чисел, аналогично **задаче No1** (в этом случае фор- мат входного и выходного файла смотрите там). Второй вариант - взять любые данные по акциям какой-либо компании, аналогично **задаче No6**.

```
with open('input_7.txt', 'r') as f:
    n = f.readline() # кол-во элементов в массиве(по условию)
    l = [int(x) for x in f.readline().split()] # считывание чисел по условию.


def ex7(A):
    s = 0 # переменная для хранения текущей суммы подмассива
    maxi = 0  # переменная для хранения максимальной суммы подмассива
    for i in A: # проходим по каждому элементу списка A
        s = max(s + i, i) # обновляем текущую сумму, выбирая максимум из суммы текущего элемента и предыдущей суммы
        maxi = max(s, maxi)   # обновляем максимальную сумму, выбирая максимум из текущей суммы и предыдущего значения максимальной суммы
    return maxi

with open('output_7.txt', 'w') as f1:
    f1.write(f"{ex7(l)}")
```

input_7.txt

```
1    10
2    31 -41 59 -26 41 58 76 54 32 12
```

output_7.txt

```
1    306
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000594457989791408 секунд	14.390625 МБ
Пример из задачи	0.000162708995048888 секунд	14.203125 МБ
Пример из задачи	0.000113249989226460 46 секунд	14.046875 МБ
Верхняя граница диапазона значений	0.000136125003336928	14.46875 МБ

входных данных из текста задачи		