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 11

Выполнил:
Кузнецов А.Г.
К3140

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

Содержание отчета

Задачи по варианту

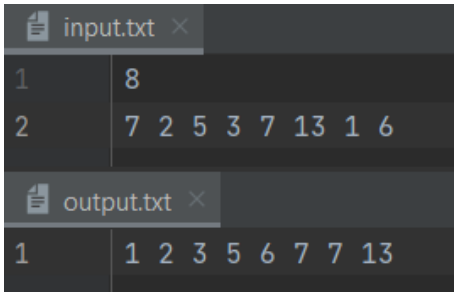
Задача №1. Сортировка слиянием

1. Используя псевдокод процедур Merge и Merge-sort из презентации к Лекции 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 ($1 \le n \le 2 \cdot 10^4$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10^9. • Формат выходного файла (output.txt). Одна строка выходного файла с отсортированным массивом. Между любыми двумя числами должен стоять ровно один пробел. • Ограничение по времени. 2сек. • Ограничение по памяти. 256 мб.
2. Для проверки можно выбрать наихудший случай, когда сортируется массив размера 1000, 10^4, 10^5 чисел порядка 10^9, отсортированных в обратном порядке; наилучший, когда массив уже отсортирован, и средний. Сравните, например, с сортировкой вставкой на этих же данных.
3. Перепишите процедуру Merge так, чтобы в ней не использовались сигнальные значения. Сигналом к остановке должен служить тот факт, что все элементы массива L или R скопированы обратно в массив A, после чего в этот массив копируются элементы, оставшиеся в непустом массиве. Или перепишите процедуру Merge (и, соответственно, Merge-sort) так, чтобы в ней не использовались значения границ и середины - p, r и q.

```
1  # Функция слияния массивов
2 def merge(A, l, m, r):
3         # Длины левого и правого массивов
4         n1 = m - l + 1
5         n2 = r - m
6 
7         L = [0] * n1
8         R = [0] * n2
9 
10        # Перенос значений в левый и правый массивы
11        for i in range(0, n1):
12                L[i] = A[l + i]
13        for j in range(0, n2):
14                R[j] = A[m + 1 + j]
15 
16        # Слияние двух массивов
17        while len(L) != 0 or len(R) != 0:
18                #Все элементы L[] скопированы в A[]
19                if len(L) == 0:
```

```
                        for l in range(l, len(R) + l):
                                A[l] = R[0]
                                R.pop(0)
                        break
                # Все элементы R[] скопированы в A[]
                elif len(R) == 0:
                        for l in range(l, len(L)+l):
                                A[l] = L[0]
                                L.pop(0)
                        break

                # Если L[] и R[] непустые массивы
                if L[0] <= R[0]:
                        A[l] = L[0]
                        L.pop(0)
                elif L[0] > R[0]:
                        A[l] = R[0]
                        R.pop(0)
                l += 1

# Функция деления массива и нахождения
def mergeSort(A, l, r):
        if l < r:
                # Нахождение медианы массива
                m = (l+r)//2

                # Деление массива на левую и правую части
                mergeSort(A, l, m)
                mergeSort(A, m+1, r)

                # Вызов функции слияния после конца деления массива
                merge(A, l, m, r)

with open('input.txt','r') as f:
        n = int(f.readline())
        A = [int(x) for x in f.readline().split(' ')]

if 1 <= n <= (2*10**4) and not(any([abs(x)>10**9 for x in A])):
        mergeSort(A, 0, n - 1)
        with open('output.txt','w') as f:
                f.write(' '.join(map(str,A)))
```

Решение задачи происходит следующим образом: сначала получаем данные из input.txt и проверяем подходят ли значения из файла заданному условию. Если все верно, то запускает функцию mergeSort, которая будет делить массив на несколько подмассивов, которые будут в функции merge сортироваться и возвращаться обратно в массив в отсортированном виде. Так происходит со всеми элементами, после чего отсортированный массив записываем в файл output.txt



	Наихудший случай при n=1000	Наихудший случай при n=10000
Сортировка слиянием	0.0040418999997200444 секунд	0.11021659999823896 секунд
Сортировка вставкой	0.04117889999997715 секунд	0.0028911999997944804 секунд

	Средний случай при n=1000	Средний случай при n=10000
Сортировка слиянием	0.003978599997935817 секунд	0.1060579000004509 секунд
Сортировка вставкой	0.03480669999953534 секунд	0.001504400000158057 секунд

	Наилучший случай при n=1000	Наилучший случай при n=10000
Сортировка слиянием	0.003977899999881629 секунд	0.10882459999993443 секунд
Сортировка вставкой	0.0019216000000596978 секунд	0.00030500000048050424 секунд

Вывод по задаче: В ходе выполнения первой задачи был разобран способ сортировки слиянием, а также была переписана процедура merge так, чтобы в ней не использовались сигнальные значения. Также было проведено сравнение сортировки слиянием и сортировки вставкой, результат которого говорит о том, что сортировка слиянием лучше сортировки вставкой при использовании небольших массивов, например,

при n=1000, где n – это длина массива. При n=10000 сортировка вставкой эффективнее, чем сортировка слиянием

Задача №3. Число инверсий

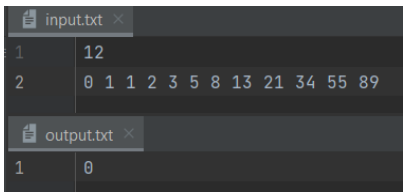
Инверсией в последовательности чисел A называется такая ситуация, когда i < j, а Ai > Aj . Количество инверсий в последовательности в некотором роде определяет, насколько близка данная последовательность к отсортированной. Например, в сортированном массиве число инверсий равно 0, а в массиве, сортированном наоборот - каждые два элемента будут составлять инверсию (всего n(n − 1)/2). Дан массив целых чисел. Ваша задача — подсчитать число инверсий в нем. Подсказка: чтобы сделать это быстрее, можно воспользоваться модификацией сортировки слиянием. • Формат входного файла (input.txt). В первой строке входного файла содержится число n ($1 \le n \le 10^5$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10^9 . • Формат выходного файла (output.txt). В выходной файл надо вывести число инверсий в массиве. • Ограничение по времени. 2сек. • Ограничение по памяти. 256 мб.

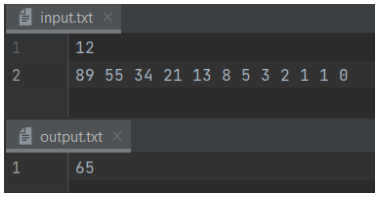
```
# Функция слияния массивов
def merge(A, l, m, r):
        global inversion
        # Длины левого и правого массивов
        n1 = m - l + 1
        n2 = r - m

        L = [0] * n1
        R = [0] * n2

        # Перенос значений в левый и правый массивы
        for i in range(0, n1):
                L[i] = A[l + i]
        for j in range(0, n2):
                R[j] = A[m + 1 + j]


        # Подсчёт числа инверсий
        i=0 # Индекс числа в L[]
        j=0 # Индекс числа в R[]
        while i<len(L) and j<len(R):
                if L[i]>R[j]:
                        j+=1
                        inversion+=len(L)-i
                else:
                        i+=1
```

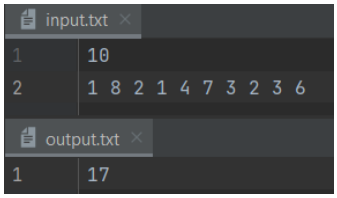
```
        # Слияние двух массивов
        while len(L) != 0 or len(R) != 0:
                #Все элементы L[] скопированы в A[]
                if len(L) == 0:
                        for l in range(l, len(R) + l):
                                A[l] = R[0]
                                R.pop(0)
                        break
                # Все элементы R[] скопированы в A[]
                elif len(R) == 0:
                        for l in range(l, len(L)+l):
                                A[l] = L[0]
                                L.pop(0)
                        break

                # Если L[] и R[] непустые массивы
                if L[0] <= R[0]:
                        A[l] = L[0]
                        L.pop(0)
                elif L[0] > R[0]:
                        A[l] = R[0]
                        R.pop(0)
                l += 1

# Функция деления массива и нахождения
def mergeSort(A, l, r):
        if l < r:
                # Нахождение медианы массива
                m = (l+r)//2

                # Деление массива на левую и правую части
                mergeSort(A, l, m)
                mergeSort(A, m+1, r)

                # Вызов функции слияния после конца деления массива
                merge(A, l, m, r)

with open('input.txt','r') as f:
        n = int(f.readline())
        A = [int(x) for x in f.readline().split(' ')]
inversion=0
if 1 <= n <= (2*10**4) and not(any([abs(x)>10**9 for x in A])):
        mergeSort(A, 0, n - 1)
        with open('output.txt','w') as f:
                f.write(str(inversion))
```

Используя метод сортировки слиянием, мы считаем количество инверсий по следующему принципу: получаем данные из файла input.txt, делим массив надвое и так до тех пор, пока каждый отдельный массив не будет состоять из одного элемента. Далее сравниваем значения левого и правого массива. Если есть инверсия, то добавляем к нашему счётчику следующее значение: длина левого массива – индекс элемента массива, при котором

возникла инверсия. Если же нет инверсии, то идёт дальше по правому массиву и снова сравниваем значения левого и правого массива. В конце работы записываем ответ в файл output.txt







Вывод по задаче: В ходе третьей задачи был использован метод сортировки слиянием для подсчёта количества инверсий в заданном массиве

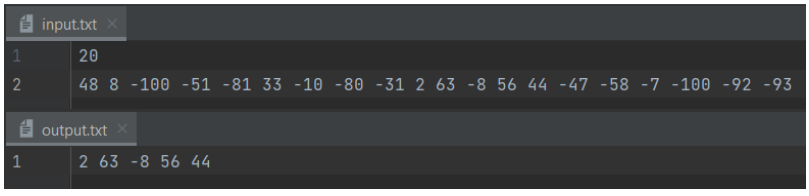
Задача №7. Поиск максимального подмассива за линейное время

Можно найти максимальный подмассив за линейное время, воспользовавшись следующими идеями. Начните с левого конца массива и двигайтесь вправо, отслеживая найденный к данному моменту максимальный подмассив. Зная максимальный 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 №1 (в этом случае формат входного и выходного файла смотрите там). Второй вариант - взять любые данные по акциям какой-либо компании, аналогично задаче №6.

```
# Функция нахождения максимального подмассива
def maxSubarray(A):
    start, end, sum = 0, 0, 0
    tempstart, tempend, tempsum = 0, 0, 0
    minus=-10**10
    for i in range(n):
        # переменная minus необходима, если все числа в массиве
отрицательные.
        # Если это так, то функция выдаст наибольшее отрицательное
значение, то есть максимальный подмассив
        minus=max(A[i],minus)
```

```
            if tempsum==0:
                tempstart=i
            tempsum += A[i]

            # Условие необходимое для нахождения максимальной суммы
            if sum < tempsum:
                sum=tempsum
                tempend=i

            # При отрицательной сумме мы заканчиваем с этим подмассивом и
переходим к следующему
            if tempsum<0:
                tempsum=0
                if start<tempstart and end<tempend:
                    start=tempstart
                    end=tempend

    # sum==0 означает, что массив состоит из отрицательных чисел
    if sum==0:
        return minus
    else:
        return A[start:end+1]
with open('input.txt','r') as f:
        n = int(f.readline())
        A = [int(x) for x in f.readline().split(' ')]

if 1 <= n <= (2*10**4) and not(any([abs(x)>10**9 for x in A])):
        with open('output.txt','w') as f:
                f.write(' '.join(map(str,maxSubarray(A))))
```

На вход получаем данные из файла input.txt. Если все значения подходят по условию, то запускаем функцию maxSubarray, где происходит следующее: мы проходим по всему массиву. Изначально берём первое значение и складываем его с временной суммой, далее проверяем является ли эта сумма больше максимальной, если это так, то присваиваем значение tempsum к sum и записываем временный конечный индекс максимального подмассива. После этого мы проверяем не является ли наша временная сумма отрицательной, если так оно и есть, то tempsum приравниваем к нулю и присваиваем значения tempstart и tempend к переменным start и end соответственно, если они подходят под условие, что start<tempstart и end<tempend. Так мы проходим по всему массиву, после чего записываем ответ в output.txt



Вывод по задаче: В ходе работы над седьмой задачей был реализован метод нахождения максимального подмассива за линейное время

Вывод

В ходе лабораторной работы были выполнены задачи с реализацией сортировки слиянием, нахождением инверсий, а также была выполнена задача, в которой было необходимо найти максимальный подмассив за линейное время