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

Выполнила:
Пантюхова В.А.
К3126

Отчет по лабораторной работе № 2
по курсу «Алгоритмы и структуры данных»
Тема: Сортировка слиянием. Метод декомпозиции.
Вариант 18

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

Содержание отчета

Задачи по варианту

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).

Листинг кода.

```
import time
import os

t_start = time.perf_counter()
def memory_usage_psutil():
    import psutil
    process = psutil.Process(os.getpid())
    mem = process.memory_info()[0] / float(2 ** 20)
    return mem

def merge(A, list, left, mid, right):
    count = i = left
    j = mid + 1
    inversCount = 0
    while i <= mid and j <= right:
        if A[i] <= A[j]:
            list[count] = A[i]
            i = i + 1
        else:
            list[count] = A[j]
            j = j + 1
            inversCount += (mid - i + 1)
        count = count + 1
    while i <= mid:
        list[count] = A[i]
        count = count + 1
        i = i + 1
    for i in range(left, right + 1):
        A[i] = list[i]
    return inversCount
def mergesort(A, list, left, right):
    if right <= left:
        return 0
    mid = left + ((right - left) // 2)
    inversCount = 0
    inversCount += mergesort(A, list, left, mid)
    inversCount += mergesort(A, list, mid + 1, right)
    inversCount += merge(A, list, left, mid, right)
    return inversCount
```

```
f = open("input.txt")
m = int(f.readline())
A = list(map(int, f.readline().split( )))
list = A.copy()

result = mergesort(A, list, 0, len(A) - 1)
mf = open("output.txt", "w+")
mf.write(f'{result} ')
mf.close()

print("Время:", time.perf_counter() - t_start)
print("Память:", memory_usage_psutil(), "мб")
f.close()
```

Текстовое объяснение решения.
Это алгоритм для упорядочивания элементов в массиве

Результат работы кода:(скрины input, output файлов).

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи		
Пример из задачи	0.004090799999999999	
Пример из задачи		
Верхняя граница диапазона значений входных данных из текста задачи		

Вывод по задаче: научились сортировке.

Задача №6. Поиск максимальной прибыли

Используя псевдокод процедур Find Maximum Subarray и Find Max Crossing Subarray из презентации к Лекции 2 (страницы 25-26), напишите программу поиска максимального подмассива. Примените ваш алгоритм для ответа на следующий вопрос. Допустим, у нас есть данные по акциям какой-либо фирмы за последний месяц (год, или иной срок). Проанализируйте этот срок и выдайте ответ, в какой из дней при покупке единицы акции данной фирмы, и в какой из дней продажи, вы бы получили максимальную прибыль? Выдайте дату покупки, дату продажи и максимальную прибыль. Вы можете использовать любые данные для своего анализа. Например, я набрала в Google "акции" и мне поиск выдал акции Газпрома, тут - можно скачать информацию по стоимости акций за любой период. (Перейдя по ссылке, нажмите на вкладку "Настройки"→ "Скачать") Соответственно, вам нужно только выбрать данные, посчитать изменение цены и применить алгоритм поиска максимального подмассива.

Листинг кода.

```
 import time
import os

t_start = time.perf_counter()
def memory_usage_psutil():
    import psutil
    process = psutil.Process(os.getpid())
    mem = process.memory_info()[0] / float(2 ** 20)
    return mem

f = open("1.csv")
a = []
for i in f.readlines():
    b = i.strip().split(";")
    a.append((b[0], float(b[1])))
c = [0]
for i in range(1, len(a)):
    c.append(a[i][1] - a[i - 1][1])

def sdfsf(a, low, high):
    if low == high:
        return (a[low], low, high)
    mid = (low + high) // 2
    left = sdfsf(a, low, mid)
    right = sdfsf(a, mid + 1, high)
    m = ghghgh(a, low, mid, high)
    if left[0] >= right[0] and left[0] >= m[0]:
        return left
    elif right[0] >= left[0] and right[0] >= m[0]:
        return right
    else:
        return m
```

```
def ghghgh(a, low, mid, high):
    l_max = -10**1001
    l_sum = 0
    l_index = 0
    for i in range(mid, low - 1, -1):
        l_sum += a[i]
        if l_sum > l_max:
            l_max = l_sum
            l_index = i
    r_max = -10**1001
    r_sum = 0
    r_index = 0
    for i in range(mid + 1, high + 1):
        r_sum += a[i]
        if r_sum > r_max:
            r_max = r_sum
            r_index = i
    return (l_max + r_max, l_index, r_index)

t = sdfsf(c, 0, len(c) - 1)

mf = open("output.txt", "w+")
mf.write(f'{a[t[1]][0]}, {a[t[2]][0]}, {t[0]}')
mf.close()

print("Время:", time.perf_counter() - t_start)
print("Память:", memory_usage_psutil(), "мб")
f.close()
```

Текстовое объяснение решения.
Мы выбираем элемент из определённого количества и помещаем его на нужную позицию массива.

Результат работы кода на примерах из текста задачи:(скрины input output файлов)

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи		
Пример из задачи		
Пример из задачи		
Верхняя граница		

диапазона значений входных данных из текста задачи		

Вывод по задаче: здесь мы изучили .

Задача №10 *

Реализуйте сортировку слиянием, учитывая, что можно сэкономить на отсортированных массивах, которые не нужно объединять. Проверьте A[q], меньше он или равен A[q + 1], и объедините их, только если A[q] > A[q + 1], где q - середина при делении в Merge_Sort.

Листинг кода:

```
def merge_sort(n):
    if len(n) == 1:
        return n
    mid = len(n) // 2
    left = merge_sort(n[:mid])
    right = merge_sort(n[mid:])
    if left[-1] <= right[0]:
        return left + right
    return merge_sort_unity(left, right)

def merge_sort_unity(list1, list2):
    unity_list = []
    count1 = 0
    count2 = 0
    while count1 < len(list1) and count2 < len(list2):
        if list1[count1] < list2[count2]:
            unity_list.append(list1[count1])
            count1 += 1
        else:
            unity_list.append(list2[count2])
            count2 += 1
    if count1 < len(list1):
        unity_list += list1[count1:]
    if count2 < len(list2):
        unity_list += list2[count2:]
    return unity_list

f = open("input.txt")
m = int(f.readline())
n = list(map(int, f.readline().split( )))

mf = open("output.txt", "w+")
mf.write(f'{merge_sort(n)} ')
mf.close()
```

Текстовое объяснение решения.
Алгоритм, который упорядочивает списки в определённом порядке.

Вывод:

В этой лабораторной работе я рассмотрела и узнала много нового и познавательного.