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

Отчет по лабораторной работе №5
по курсу «Алгоритмы и структуры данных»
Тема: Деревья. Пирамида, пирамидальная сортировка. Очередь с приоритетами.

Выполнил:
Волжева М.И.
К3141

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

Содержание отчета

Задачи по варианту

Задача №1. Куча ли?

Текст задачи:
Структуру данных «куча», или, более конкретно, «неубывающая пирамида», можно реализовать на основе массива. Для этого должно выполнятся основное свойство неубывающей пирамиды, которое заключается в том, что для каждого $1 \leq i \leq n$ выполняются условия:
1) если $2i \leq n$, то $a_i \leq a_{2i}$
2) если $2i + 1 \leq n$, то $a_i \leq a_{2i+1}$.
Дан массив целых чисел. Определите, является ли он неубывающей пирамидой.

- Формат входного файла (input.txt Первая строка входного файла содержит целое число n ($1 \leq n \leq 10^6$). Вторая строка содержит n целых чисел, по модулю не превосходящих $2*10^9$.
- Формат выходного файла (output.txt) Выведите «YES», если массив является неубывающей пирамидой, и «NO» в противном случае.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб.
- Пример:

Input.txt	5 1 0 1 2 0	NO
Output.txt	5 1 3 2 5 4	YES

Листинг кода:

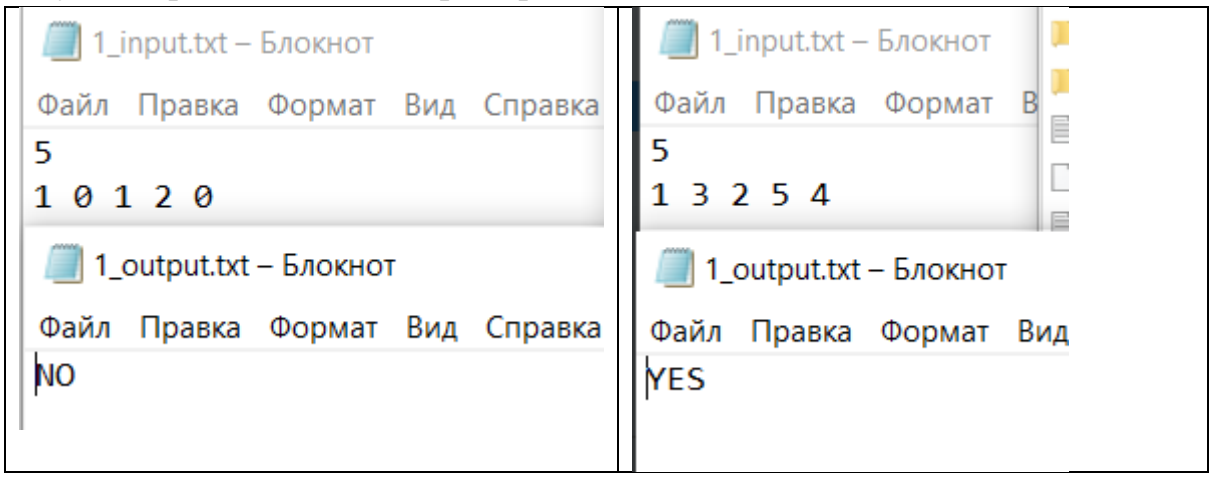
```
import time
import os, psutil


def is_heap(arr, size_arr):
    flag = False
    for i in range(1, size_arr//2):
        if (i*2 <= size_arr) and (arr[i] > arr[i*2]):
            flag = True
            break
        if (i*2 + 1 <= size_arr) and (arr[i] > arr[i*2 + 1]):
            flag = True
            break
    return flag
```

```
t_start = time.perf_counter()
process = psutil.Process(os.getpid())
f = open("1_input.txt")
m = open("1_output.txt", "w")

size = int(f.readline())
string = f.readline()
elements = list(map(str, string.split()))

if is_heap(elements, size):
    m.write("YES")
else:
    m.write("NO")


f.close()
m.close()

print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")
```

Текстовое объяснение решения:

Я прочла данные из файла, записала их в массив elements, размер массива записан в переменной size. Далее была написана функция is_heap, которая проверяет является ли массив представлением дерева по определению, то есть перебором всех значений и их “детей”. Функции передается массив и его размер, а получаем от функции TRUE или FALSE. Далее в файл записывается результат является ли массив представлением дерева.

Результат работы кода на примерах из текста задачи:



Тестирование алгоритма:

	Время выполнения	Затраты памяти
Пример 1 из задачи	0.007296599999999986 second	13.44921875 mb

Пример 2 из задачи	0.0037781000000000065 second	13.453125 mb

Вывод по задаче: Мы узнали, что такое дерево и научились определять является ли массив представлением дерева.

Задача №4. Построение пирамиды

Текст задачи:

Первым шагом алгоритма HeapSort является создание пирамиды (heap) из массива, который вы хотите отсортировать. Ваша задача - реализовать этот первый шаг и преобразовать заданный массив целых чисел в пирамиду. Вы сделаете это, применив к массиву определенное количество перестановок (swaps). Перестановка - это операция, как вы помните, при которой элементы a_i и a_j массива меняются местами для некоторых i и j. Вам нужно будет преобразовать массив в пирамиду, используя только O(n) перестановок.

- Формат входного файла (input.txt). Первая строка содержит целое число n ($1 \le n \le 10^5$), вторая содержит n целых чисел a_i входного массива, разделенных пробелом ($0 \le a_i \le 10^9$, все a_i - различны.)
- Формат выходного файла (output.txt). Первая строка ответа должна содержать целое число m - количество сделанных свопов. Число m должно удовлетворять условию $0 \le m \le 4n$. Следующие m строк должны содержать по 2 числа: индексы i и j сделанной перестановки двух элементов, индексы считаются с 0.
- Ограничение по времени. 3 сек.
- Ограничение по памяти. 512 мб.
- Пример:

Input.txt	5 5 4 3 2 1	3 1 4 0 1 1 3
Output.txt	5 1 2 3 4 5	0

-

Листинг кода:

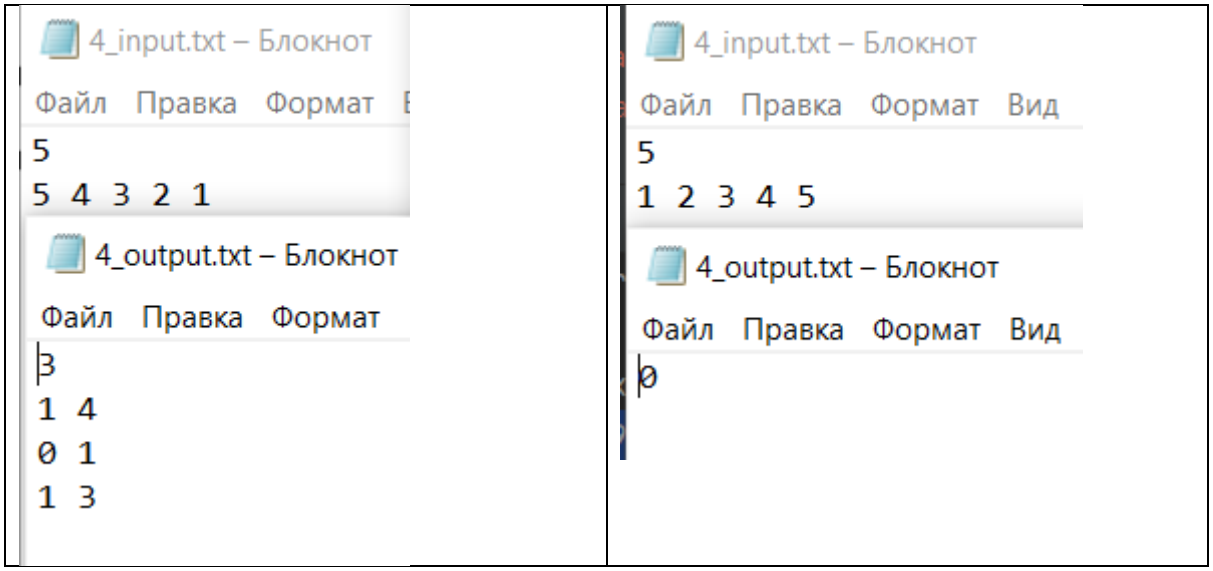
```
import time
import os, psutil


def swap(arr, a, i, answer_arr):
    n = i
    left = 2 * i + 1
    right = 2 * i + 2
    if left < a and arr[i] > arr[left]:
        n = left
    if right < a and arr[n] > arr[right]:
        n = right
    if n != i:
        arr[i], arr[n] = arr[n], arr[i]
```

```
        answer_arr.append([i, n])
        swap(arr, a, n, swaps)


def heapsort(arr, answer_arr):
    for i in range(len(arr) // 2 - 1, -1, -1):
        swap(arr, len(arr), i, answer_arr)


t_start = time.perf_counter()
process = psutil.Process(os.getpid())
f = open("4_input.txt")
m = open("4_output.txt", "w")

size = int(f.readline())
string = f.readline()
elements = list(map(str, string.split()))
swaps = []

heapsort(elements, swaps)

m.write(str(len(swaps)) + "\n")
for each in swaps:
    m.write(f"{each[0]} {each[1]} \n")


f.close()
m.close()

print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")
```

Текстовое объяснение решения:

Мы прочитали данные из файла, записали их в массив elements, размер массива записан в переменной – size. Далее была написана функция heap_sort, которая производит “протаскивание” каждого элемента. “Протаскивание” прописано в функции swap. В итоге у нас получается массив, который является представлением дерева. Это и есть первая часть и основа пирамидальной сортировки.

Результат работы кода на примерах из текста задачи:



Тестирование алгоритма:

	Время выполнения	Затраты памяти
Пример 1 из задачи	0.007296599999999986 second	13.44921875 mb
Пример 2 из задачи	0.019250900000000015 second	13.49609375 mb

Вывод по задаче:.Мы поподробнее разобрались в такой структуре, как дерево, ознакомились с алгоритмом пирамидальной сортировки и научились писать её первую часть – представление массива в виде дерева.

Вывод

Я узнала, что такое дерево и пирамида, научилась определять является ли массив представлением пирамиды, научилась преобразовывать произвольный массив до массива, который является представлением пирамиды, с помощью транспозиции элементов, ознакомилась с алгоритмом пирамидальной сортировки.