Федеральное государственное автономное образовательное
учреждение высшего образования
Университет ИТМО

Факультет инфокоммуникационных технологий

Алагритмы и структуры данных:

Отчёт по лабораторной работе №5: Деревья. Пирмамида, пирамидальная сортировка. Очередь с приоритетами.

Выполнил:
Бочкарь Артём Артёмович

Группа: **K32392**

Преподаватели:
Артамонова В. Е.

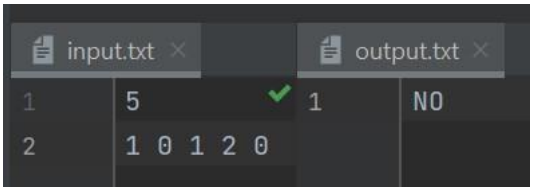
Санкт-Петербург 2023 г.

Задача №1: Куча ли?

В первом задании надо было реализовать структуру данных “куча” на основе массива. Первая строка входного файла содержит целое число n (1 ≤ n ≤ 10^6). Вторая строка содержит n целых чисел, по модулю не превосходящих 2 · 10^9. Нужно вывести “YES”, если массив является неубывающей пирамидой, и «NO» в противном случае:

```
input = open('input.txt')
output = open('output.txt', 'w')
n = int(input.readline())
mass= list(map(int, input.readline().split()))
mass = [0] + mass

for i in range(1, n + 1):
    if i * 2 <= n:
        if mass[i] > mass[i * 2]:
            output.write('NO')
            exit(0)
    if i * 2 + 1 <= n:
        if mass[i] > mass[i * 2 + 1]:
            output.write('NO')
            exit(0)
output.write('YES')
```

Тесты:

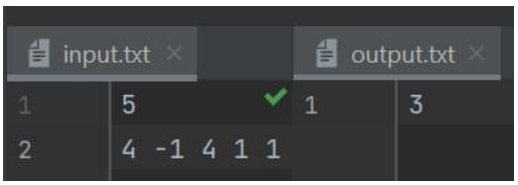


Задача №2: Высота дерева

Во втором задании нужно было реализовать структуру данных – дерево, сохранить его и вычислить его высоту. Первая строка содержит число узлов n ($1 \le n \le 10^5$). Вторая строка содержит n целых чисел от -1 до $n-1$ – указание на родительский узел. Если i-ое значение равно -1, значит, что узел i - корневой, иначе это число является обозначением индекса родительского узла этого i-го узла ($0 \le i \le n - 1$). Индексы считать с 0. Гарантируется, что дан только один корневой узел, и что входные данные представляют дерево. Нужно вывести высоту дерева:

```
input = open('input.txt')
output = open('output.txt', 'w')
n = int(input.readline())
mass = list(map(int, input.readline().split()))

g = {}
depth = list()
for i in range(0, n):
    depth.append(-1)

def dfs(v):
    if g.get(v) is None:
        return
    for to in g[v]:
        depth[to] = depth[v] + 1
        dfs(to)

root = -1
for i in range(0, n):
    if mass[i] == -1:
        root = i
        continue
    if g.get(mass[i]) is None:
        g[mass[i]] = list()
    g[mass[i]].append(i)

depth[root] = 1
dfs(root)
output.write(str(max(depth)))
```

Тесты:

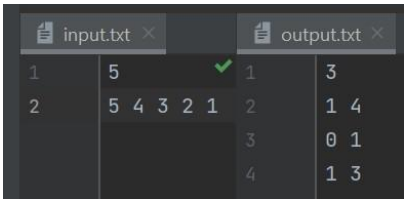


Задача №4: Построение пирамиды

В этой задаче надо было преобразовать массив, состоящий из целых чисел, в пирамиду. Первая строка входного файла содержит целое число n ($1 \le n \le 10^5$), вторая содержит n целых чисел ai входного массива, разделенных пробелом ($0 \le ai \le 10^9$, все ai - различны.). 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:

```
input = open("input.txt", "r").read().split()
output = open("output.txt", 'w+')
output.write("\n")
our_list = list(map(int, input[1:]))
list_index = []
def heapify(arr, i):
    left = 2*i + 1
    right = 2*i + 2
    largest = i
    heap_size = len(arr)
    if left < heap_size and arr[left] < arr[largest]:
        largest = left
    if right < heap_size and arr[right] < arr[largest]:
        largest = right
    if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]
        list_index.append(i)
        list_index.append(largest)
        output.write(str(i) + " ")
        output.write(str(largest))
        output.write("\n")
        heapify(arr, largest)
    return arr


def build_heap(arr):
    heap_size = len(arr)
    for i in range(heap_size//2 - 1, -1, -1):
        heapify(arr, i)

build_heap(our_list)
kol = str(len(list_index) // 2)
output.close()
output = open("output.txt", 'r')
a = output.read().split("\n")
a.pop(0)
a.insert(0, kol)
output.close()
output = open("output.txt", 'w')
for i in a:
    output.write(i + '\n')
```

Тесты:



Задача №7: Снова сортировка

В этом задании нужно было написать программу пирамидальной сортировки для последовательности в убывающем порядке. 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. В выходном файле должна быть одна строка с отсортированным по невозрастанию массивом. Между любыми двумя числами должен стоять ровно один пробел:

```
n = int(input())
mass = list(map(int, input().split()))

def heapify(nums, heap_s, root_ind):
    smallest = root_ind
    left = (2 * root_ind) + 1
    right = (2 * root_ind) + 2

    if left < heap_s and nums[left] < nums[smallest]:
        smallest = left

    if right < heap_s and nums[right] < nums[smallest]:
        smallest = right

    if smallest != root_ind:
        nums[root_ind], nums[smallest] = nums[smallest], nums[root_ind]
        heapify(nums, heap_s, smallest)

def heap_sort(nums):
    for i in range(n, -1, -1):
        heapify(nums, n, i)
    for i in range(n - 1, 0, -1):
        nums[i], nums[0] = nums[0], nums[i]
        heapify(nums, i, 0)

heap_sort(mass)
print(mass)
```

Тесты:

```
6
12 43 10 3 23 45
[45, 43, 23, 12, 10, 3]
```