МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Алгоритмы и структуры данных»
Тема: Очередь с приоритетом. Параллельная обработка.

Студентка гр. 1383 _____ Седова Э.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2022

Цель работы.

Ознакомиться и научиться применять для решения практических задач такую абстрактную структуру данных как очередь с приоритетом.

Задание.

Параллельная обработка.

На вход программе подается число процессоров n и последовательность чисел t_0 , . . . , t_{m-1} , где t_i — время, необходимое на обработку i-й задачи.

Требуется для каждой задачи определить, какой процессор и в какое время начнёт её обрабатывать, предполагая, что каждая задача поступает на обработку первому освободившемуся процессору.

Примечание #1: в работе необходимо использовать очередь с приоритетом (т.е. min или max-кучу)

Примечание #2: в работе запрещено использовать библиотечные реализации алгоритмов и структур.

Формат входа

Первая строка входа содержит числа n и m. Вторая содержит числа t_0 , . . . , t_{m-1} , где t_i — время, необходимое на обработку i-й задачи. Считаем, что и процессоры, и задачи нумеруются с нуля.

Формат выхода

Выход должен содержать ровно m строк: i-я (считая с нуля) строка должна содержать номер процессора, который получит i-ю задачу на обработку, и время, когда это произойдёт.

Ограничения

$1 \le n \le 10^5$; $1 \le m \le 10^5$; $0 \le t_i \le 10^9$.

Выполнение работы.

Программа принимает входные данные от пользователя. Создаётся двумерный массив *proc*, с помощью списков. Каждый элемент *proc* – это список из двух элементов: первый в самом начале инициализируется нулём, второй – номером процессора. Далее *proc*, и список *time*, хранящий время, необходимое

для выполнения каждой команды, передаются в функцию *process*. С помощью цикла *for* производится обход всех элементов списка *time*. К первому элементу *proc[0]* прибавляется текущее значение из списка *time*, вызывается функция *siftdown.*

Функция *siftdown* принимает индекс, находит индексы правого и левого потомков. Если хотя бы один из потомков существует и оказывается меньше текущего элемента, то они меняются местами. Функция вызывается рекурсивно и перестанет вызываться в двух случаях: если потомки будут отсутствовать или если текущее значение наименьшее из возможных.

Все пары “процессор время” записываются в строку *result*, которая возвращается методом *process*.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	[244, 7, 1234, 1, 7, 30], [[0, 0], [0, 1], [0, 2]]	[[0, 0], [1, 0], [2, 0], [1, 7], [1, 8], [1, 15]]
2.	[8, 7, 6, 5, 4, 3, 8], [[0, 0], [0, 1]]	[[0, 0], [1, 0], [1, 7], [0, 8], [0, 13], [1, 13], [1, 16]]
3.	[3, 7], [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5]]	[[0, 0], [1, 0]]
4.	[6, 4, 9], [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4]]	[[0, 0], [1, 0], [2, 0]]

Выводы.

Были получены навыки решения практических задач, используя такую абстрактную структуру данных как очередь с приоритетом.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
def siftdown(j, proc):
    maxj = j
    left = 2 * j + 1
    right = 2 * j + 2
    if left < len(proc) and proc[left] < proc[maxj]:
        maxj = left
    if right < len(proc) and proc[right] < proc[maxj]:
        maxj = right
    if j != maxj:
        proc[j], proc[maxj] = proc[maxj], proc[j]
        siftdown(maxj, proc)

def process(time, proc):
    result = []
    for i in time:
        result.append([proc[0][1], proc[0][0]])
        proc[0][0] += i
        siftdown(0, proc)
    return result

if __name__ == "__main__":
    n, m = map(int, input().split())
    time = list(map(int, input().split()))
    proc = [[0, i] for i in range(n)]
    print(proc,"proc")
    print(time,"time")
    result = process(time, proc)
    print(result,"result")
    for i in result:
        print(i[0], i[1])
```