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: Очереди с приоритетом. Параллельная обработка

Студент гр. 1303 _____ Коренев Д. А.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург

2022

Цель работы.

Научиться работать со структурой данных «Куча» и применить её для решения алгоритмической задачи.

Задание.

Параллельная обработка.

На вход программе подается число процессоров n и последовательность чисел t0 , . . . , tm−1 , где ti — 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: в работе запрещено использовать библиотечные реализации алгоритмов и структур.

Формат входа Первая строка входа содержит числа n и m. Вторая содержит числа t0 , . . . , tm−1 , где ti — 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.

Выполнение работы.

Для реализации задачи создан класс processor. Метод инициализации (присваивает номер каждому объекту), перегруженый метод __str__ - для удобства вывода, метод сравнения __lt__ - для сравнения двух объектов данного класса: первый процессор меньше второго, если его загруженность меньше, либо (при равной загруженности) если его номер меньше второго, геттеры для полей __load__, __index__, метод добавления нагрузки на процессор.

Также создан класс queuePriority, реализующий абстрактную структуру данных — очередь с приоритетом. В методе инициализации принимает массив объектов. Метод increaseMaxPriorityElement увеличивает нагрузку процессора, самого приоритетного для этой задачи, вызывает метод move, т.к. данный

процессор может не являться самым приоритетным для следующего увеличения нагрузки. Так как поле __queue__ удовлетворяет условиям мин-кучи, можно использовать ее свойства. Метод move принимает индекс элемента, вычисляет какой из элементов (родитель и его дети) имеет наименьший приоритет и, если это не родитель, меняет его и родителя местами в массиве, рекурсивно вызывает себя по индексу, на котором раньше стоял ребенок.

Функция runTime принимает две входные строки, преобразует их в данные для работы, создает массив экземпляров класса processor, для каждой нагрузки из входных данных применяет метод increaseMaxPriorityElement с текущей нагрузкой.

Тестирование программы.

Чтобы удостовериться в правильности работы программы, она была протестирована на следующих случаях:

- Подается несколько задач для одного процессора
- Подается несколько задач для нескольких процессоров, но нагрузка всех задач равна 1
- Для двух процессоров подается одна большая задача и много маленьких, таких что сумма их нагрузки не превышает большую
- Количество задач равно количеству процессоров

Тестирование проводится с применением фреймворка pytest.

Выводы.

В ходе выполнения лабораторной работы была изучена структура данных «Куча» и реализована в виде минимальной кучи, хранящей объекты определенного класса.