Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №. 8

Выполнила студентка группы КС-36\_ Битарова Эмма Олеговна

Ссылка на репозиторий: https://github.com/MUCTR-IKT-CPP/

Приняли: Пысин Максим Дмитриевич

Дата сдачи 27.11.2022

Москва 2022

# Постановка задачи

В рамках лабораторной работы необходимо реализовать бинарную кучу (мин или макс), а так же:

1. Фибоначиеву кучу

Для реализованных куч выполнить следующие действия:

1. Наполнить кучу N кол-ва элементов (где N = 10 ^ i, i от 3 до 7).
2. После заполнения кучи необходимо провести следующие тесты:
   1. 1000 раз найти минимум/максимум
   2. 1000 раз удалить минимум/максимум
   3. 1000 раз добавить новый элемент в кучу  
      Для всех операция требуется замерить время на выполнения всей 1000 операций и рассчитать время на одну операцию, а так же запомнить максимальное время которое требуется на выполнение одной операции если язык позволяет его зафиксировать, если не позволяет воспользоваться хитростью и рассчитывать усредненное время на каждые 10,25,50,100 операций, и выбирать максимальное из полученных результатов, что бы поймать момент деградации структуры и ее перестройку.
3. По полученным в задании 2 данным построить графики времени выполнения операций для усреднения по 1000 операций, и для максимального времени на 1 операцию.

# Описание метода

Двоичная куча (binary heap) – просто реализуемая структура данных, позволяющая быстро (за логарифмическое время) добавлять элементы и извлекать элемент с максимальным приоритетом (например, максимальный по значению).

Основной особенностью двоичной кучи является то, что каждый из узлов кучи не может иметь более чем двух потомков.

Двоичная куча отлично представляется в виде одномерного массива, при этом: нулевой элемент массива всегда является вершиной кучи, а первый и второй потомок вершины с индексом i получают свои положения на основании формул: 2 \* i + 1 левый, 2 \* i + 2 правый.

# Фибоначчиева куча, это так же биноминальная куча, у которой прощаются все операции кроме операции удаления. За счет сознательного ухудшения состояния кучи внутри нее. Мы предполагаем, что наша куча может иметь в себе множество деревьев одинакового ранга. Т.е. у нас может быть хоть 20 деревьев первого нулевого ранга.

# В такой ситуации мы можем позволить себе при добавлении в дерево нового элемента просто добавлять в кучу дерево ранга 0 содержащего наше новое значение, перепроверяя только минимум/максимум в куче(в зависимости от того в какую сторону мы взращиваем кучу).

# Объединение двух куч так же позволяет нам совершать эту операцию просто выполняя объединение двух списков деревьев из которых состоят кучи, в конце лишь выбирая один из двух имеющихся минимумов.

# Данные действия сознательно ухудшат структуру и значит, что за это придется платить, основным взимателем этой платы является операция удаления минимума, которая проводит нормализацию дерева объединяя все созданные дубликаты деревьев.

# Выполнение задачи

Язык: Python.

Были написаны два класса для бинарной и фибоначиевой куч: класс Node и класс FibonacciHeap.

В классе бинарной кучи реализованы такие методы:

- вставка нового элемента

- получение минимума

- получение максимума

- удаление элемента

- существование элемента в куче

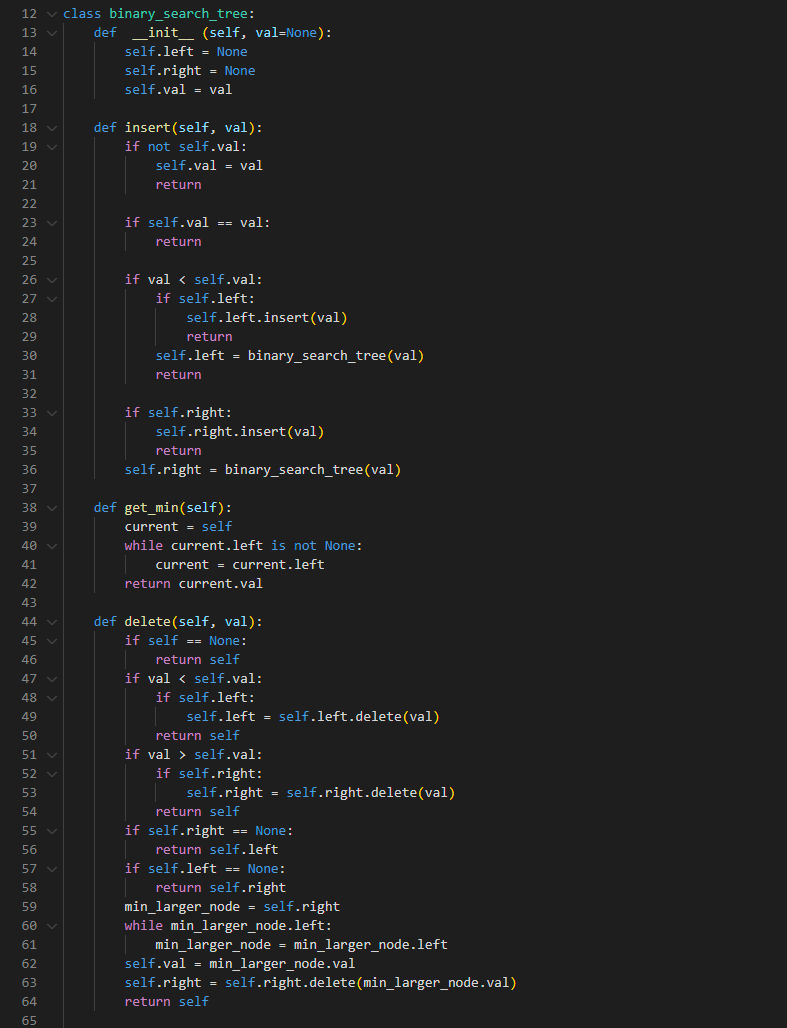
- вывод кучи до сортировки

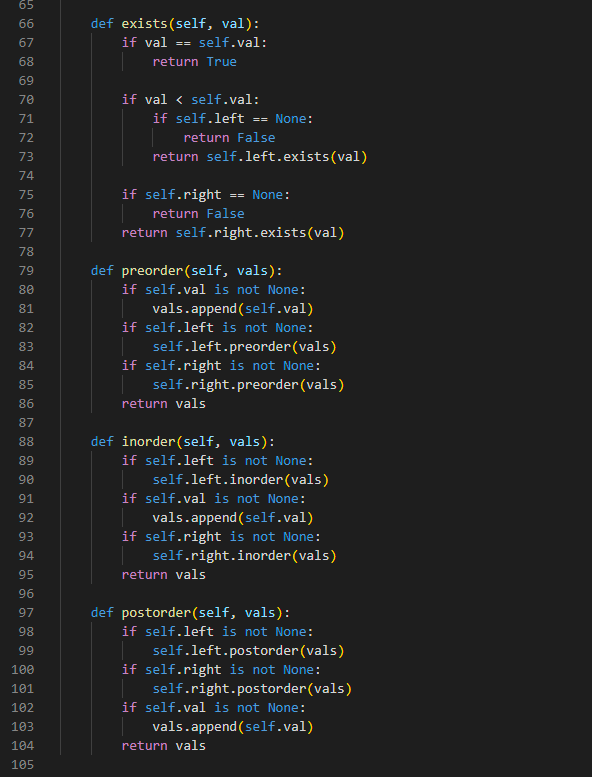
- вывод кучи после сортировки

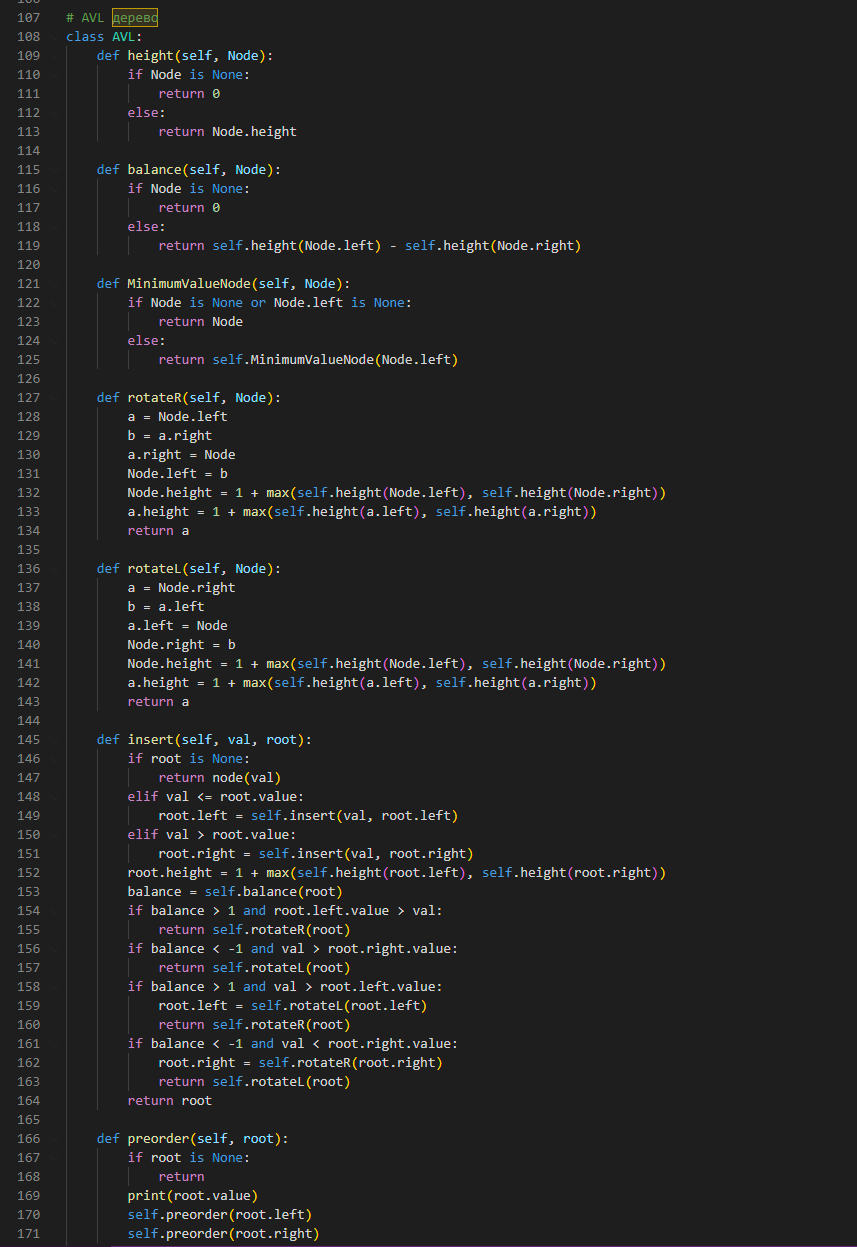
В классе фибоначиевой кучи реализованы такие методы:

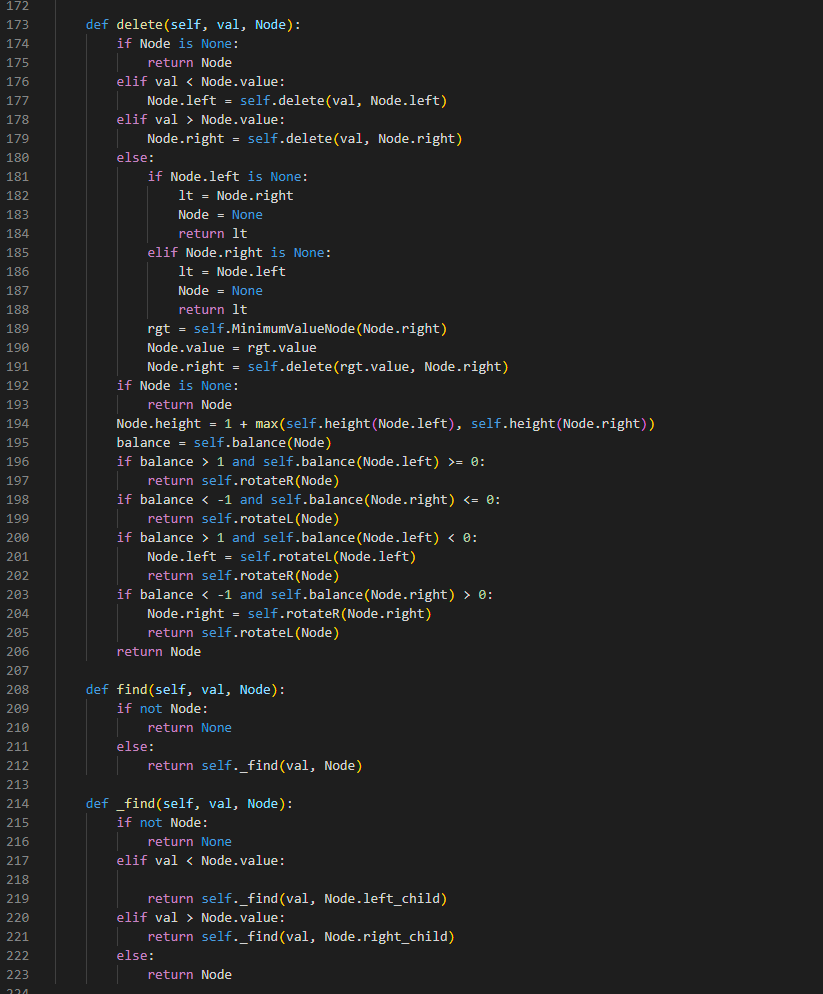
функция для перебора двухсвязного списка, нахождение минимума, объединение корневых узлов до одной степени вершины, связывание корневых узлов, объединение с корневым списком, объединение с дочерним списком корневого узла, удаление узла из корневого списка, удаление узла из дочернего списка, удаление минимума из кучи, вставка нового узла в кучу, узнать размер кучи

Также непосредственного для упорядочивания и проведения тестов, есть 3 функции:

функция нахождения минимума в обеих кучах и расчёт времени за N-итераций, расчёт времени за 1 итерацию; и максимальное время нахождения минимума за 1 итерацию, функция удаления минимума в обеих кучах и расчёт времени за N-итераций, расчёт времени за 1 итерацию; и максимальное время нахождения минимума за 1 итерацию, функция вставки нового элемента в обе кучи и расчёт времени за N-итераций, расчёт времени за 1 итерацию; и максимальное время нахождения минимума за 1 итерацию. 







Графики результатов:

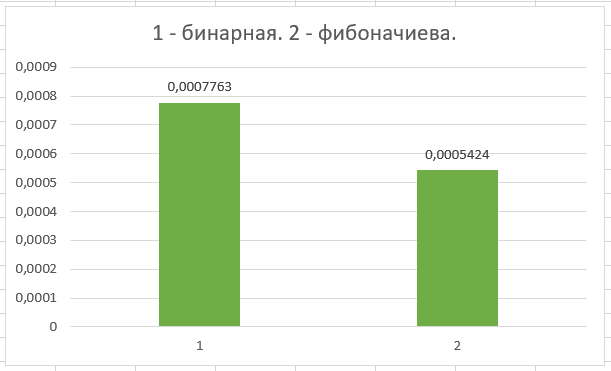


Рисунок 1. Поиск минимума за 1000 итераций

Расчитанное время за 1 итерацию бинарной кучи: 1.2302549818616283e-06

Расчитанное время за 1 итерацию фибоначиевой: 5.424022674560547e-07

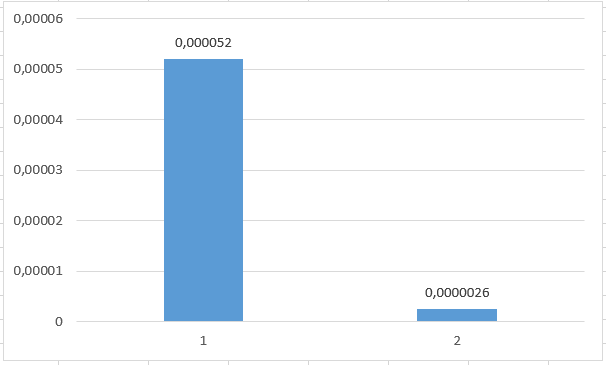


Рисунок 2. Максимальное время за 1 итерацию поиска



Рисунок 3. Время на удаление минимума

Расчитанное время за 1 итерацию бинарной кучи: 3.6034833421419994e-06

Расчитанное время за 1 итерацию фибоначиевой: 1.5566349029541015e-06

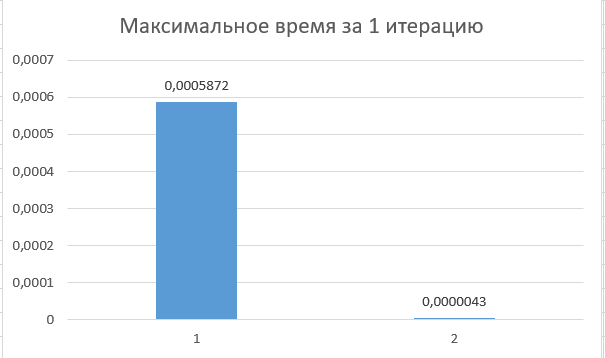


Рисунок 4. Максимальное время за 1 итерацию удаления

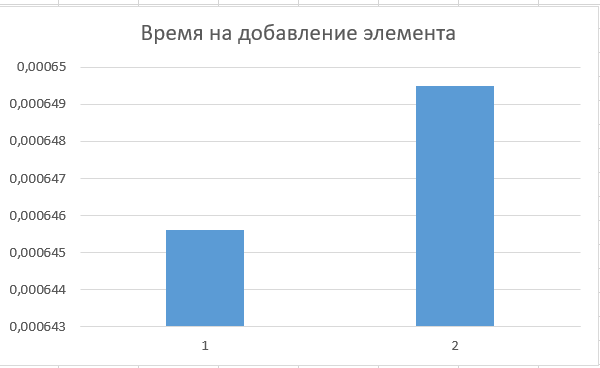


Рисунок 5. Время на добавление элемента

Расчитанное время за 1 итерацию бинарной кучи: 1.0231973252092412e-06

Расчитанное время за 1 итерацию фибоначиевой: 6.494522094726563e-07

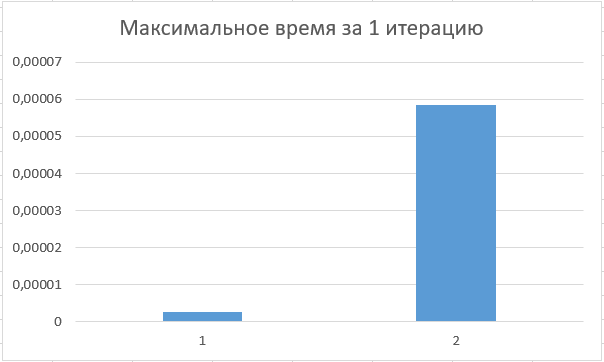


Рисунок 6. Максимальное время за 1 итерацию