Отчет по домашней работе по Java NC

1. В результате сравнения MyLinkedList и LinkedList из Collections получил следующие результаты:

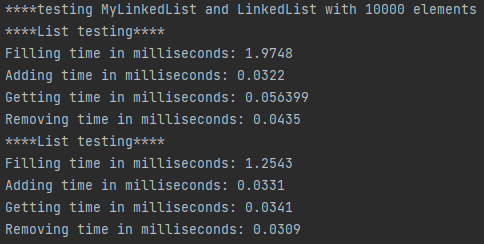


Рисунок . Тест с 10000 элементами

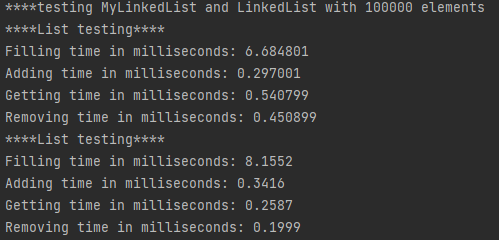


Рисунок . Тест с 100000 элементами

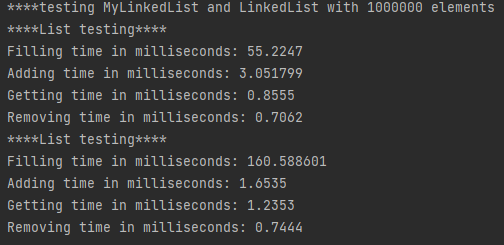


Рисунок . Тест с 1000000 элементами

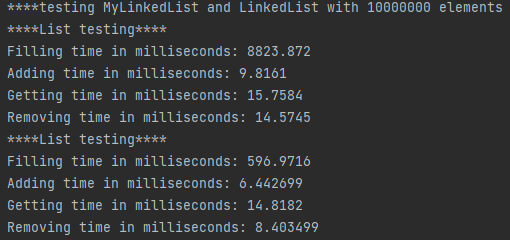


Рисунок . Тест с 10000000 элементами

Результаты замеров разнятся от запуска к запуску, поэтому нельзя сказать о преимуществе какой-либо реализации. Возможно, прироста скорости у самостоятельной реализации не происходит из-за того, что в ней я обращался к полям Узлов и Листа через геттеры, что в теории, может занимать немного больше времени, чем прямое обращение к этим полям.

Также, при каждом запуске происходит подвисание на заполнении MyLinkedList’а 10000000 элементов. С чем связано такое поведение, понять не получилось.

1. Сравнение работы структур данных из Collections:

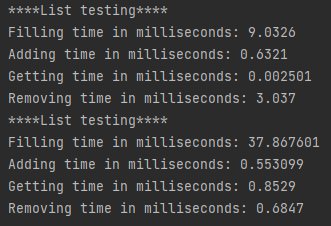


Рисунок . ArrayList & LinkedList, 1000000 элементов

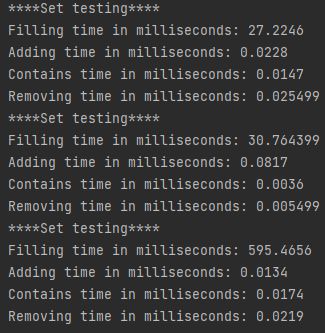


Рисунок 6. HashSet & LinkedHashSet & TreeSet, 1000000 элементов

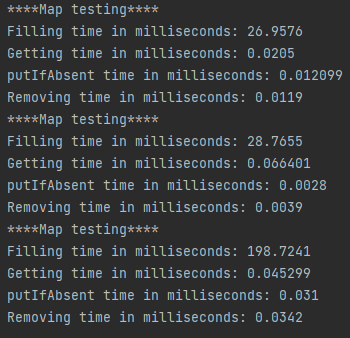


Рисунок . HashMap & LinkedHashMap & TreeMap, 1000000 элементов

Как видно из результатов, затраченное время на заполнение коллекции, в некоторых случаях, на порядок отличается от времени работы других коллекций, что достаточно странно. Объяснений этому найти не получилось.

Преимущества коллекций:

1. ArrayList – быстрый доступ по индексу, нет дополнительных затрат на хранение связок между элементами, но проигрывает в скорости вставки/удаления, т.к. все элементы правее сдвигаются на 1.

LinkedList – вставка и удаление работают быстрее, т.к. сдвигать ничего не надо, а только нужно «перепривязать» ссылки на соседние элементы.

1. HashSet – реализован с помощью хэш-таблицы, элементы не упорядочены, но операции добавления/удаления имеют константную сложность, что очень быстро

LinkedHashSet – реализован с помощью хэш-таблицы и связного списка, поэтому элементы сохраняются в порядке вставки, что может быть полезно в некоторых случаях.

TreeSet – реализован с помощью красно-черного дерева, поэтому очень быстро работает на поиск элементов, но более затратный по времени на добавление и удаление элементов.

1. HashMap – элементы не упорядочены, но вставки/удаления элементов работают быстро.

LinkedHashMap – элементы располагаются в порядке вставки, что может быть полезно в некоторых ситуациях.

TreeMap – отсортирован по ключам, что дает большую скорость в поиске элементов.