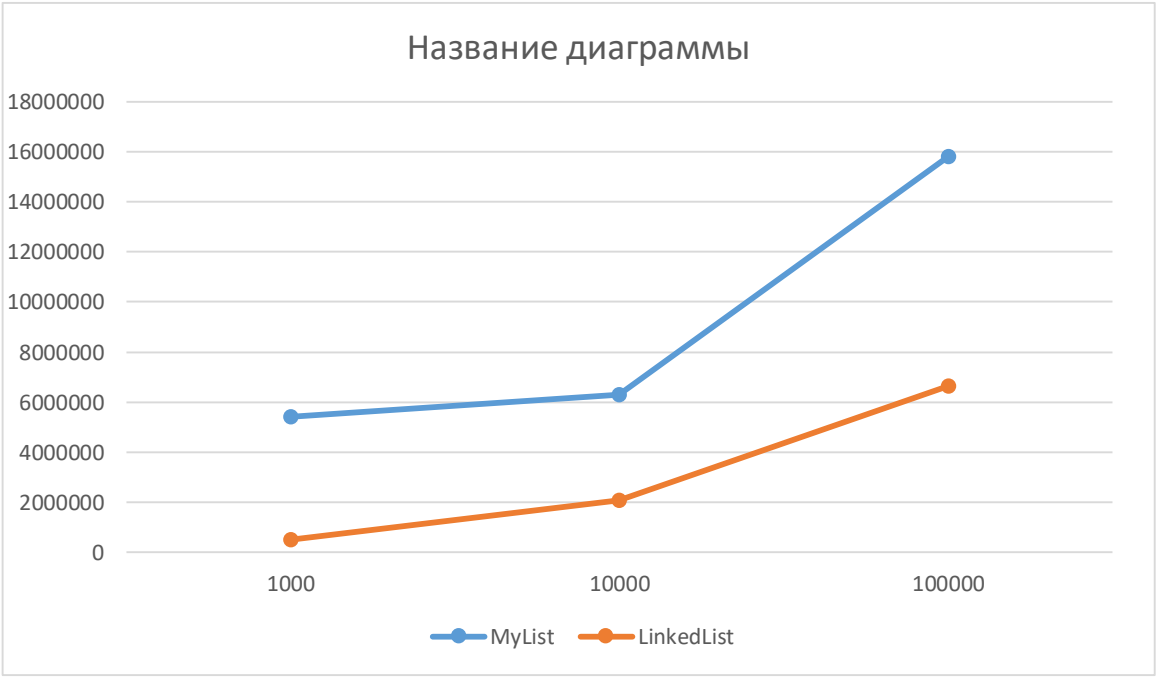
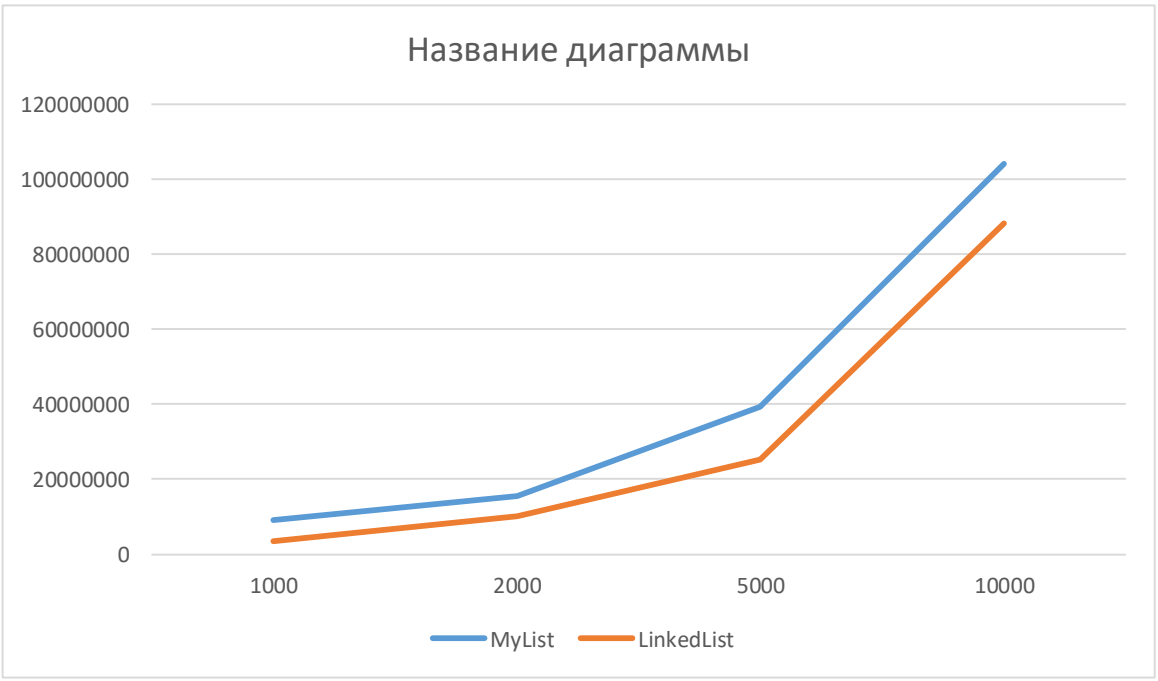
Результаты экспериментов с MyLinkedList.

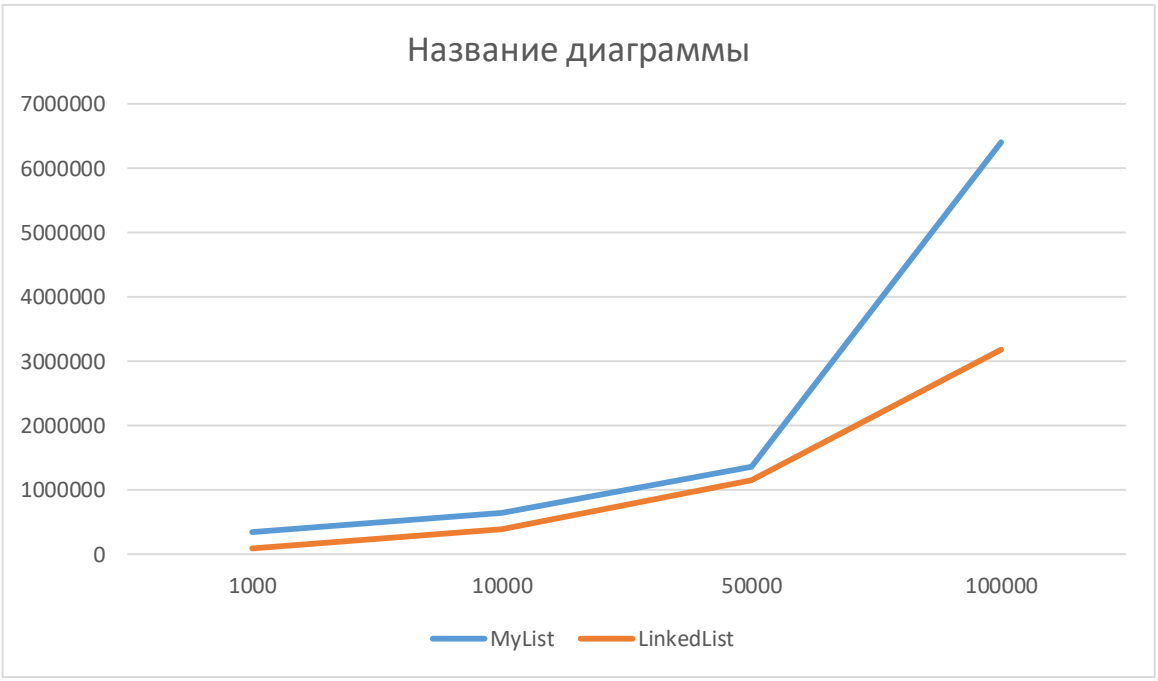
1. MyLinkedList vs Original LinkedList
   Ось x – количество элементов, ось y – время в наносекундах.
   - Добавление в конец



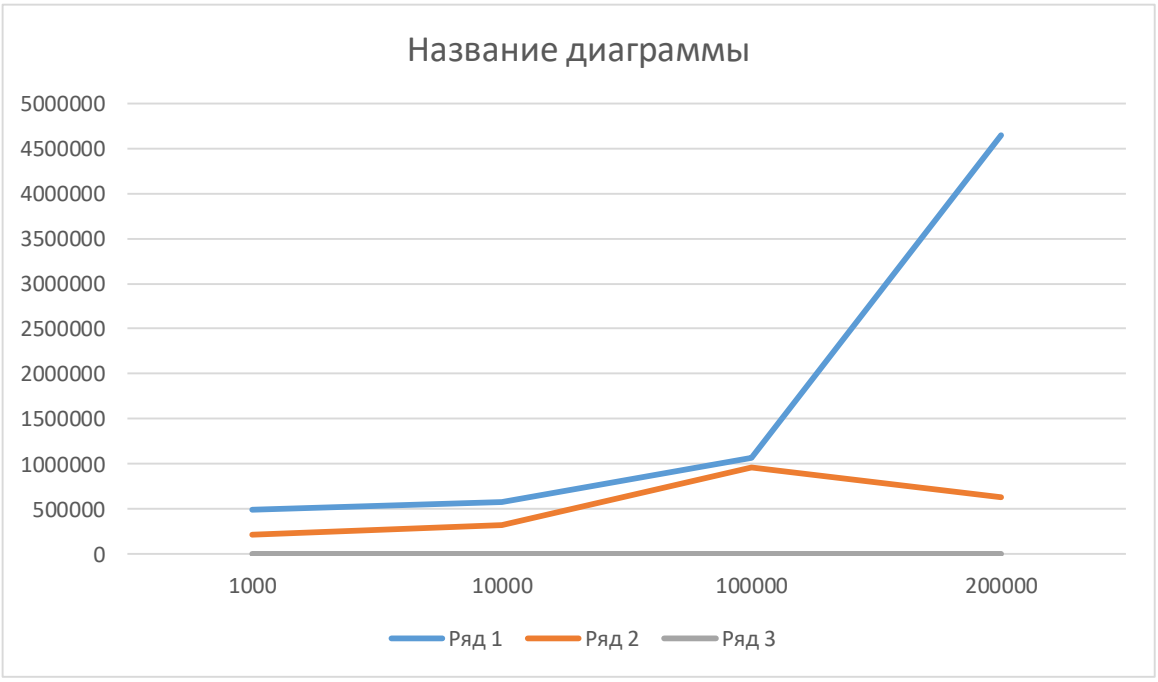
   - Добавление в середину



- Поиск элемента с середины

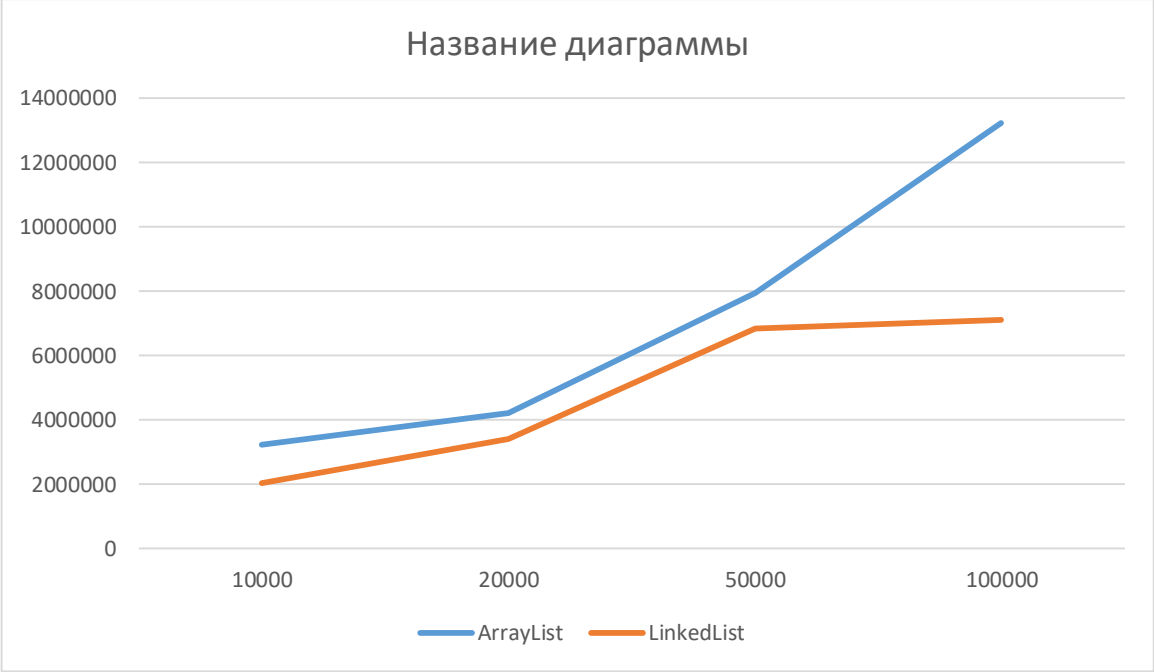


- Удаление элемента по индексу, заданному наугад



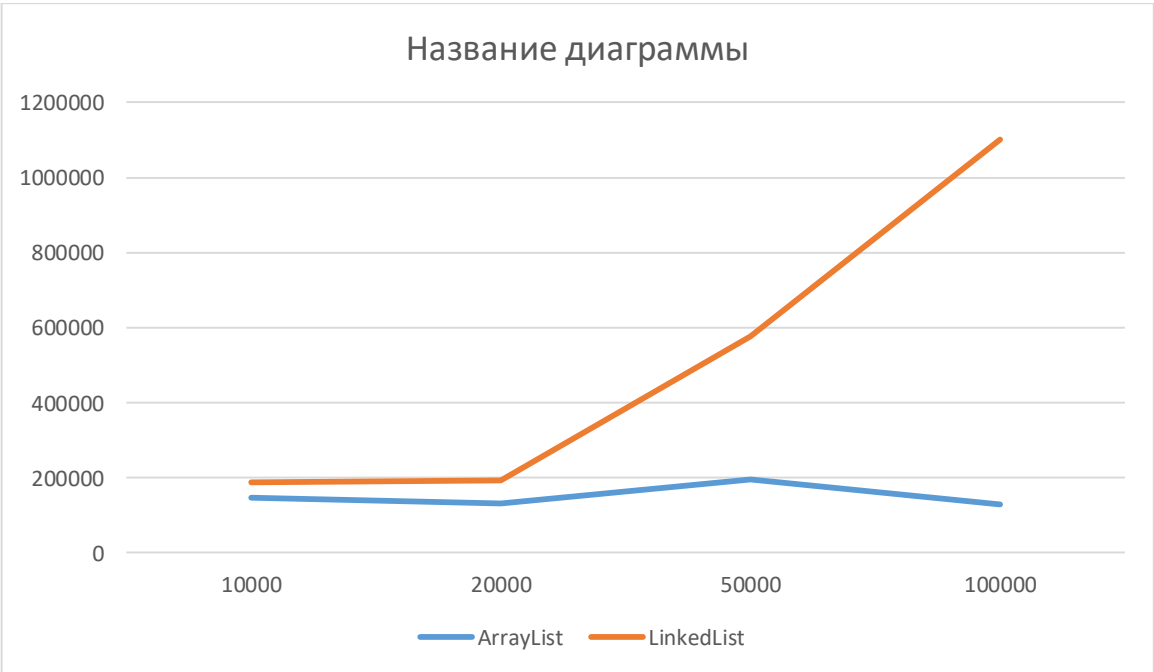
- Выводы: во всех тестах быстрее оказывается связный список из Java класса, скорее всего это связано с тем, что я в своем классе при переборе использовал обычный for, а в Java использовали какие-либо хитрости. Причем, чем больше количество элементов, тем больше видна разница во времени. Вывод: существуют методы быстрее, чем обычный for.

2. LinkedList vs ArrayList
   - Добавление элемента



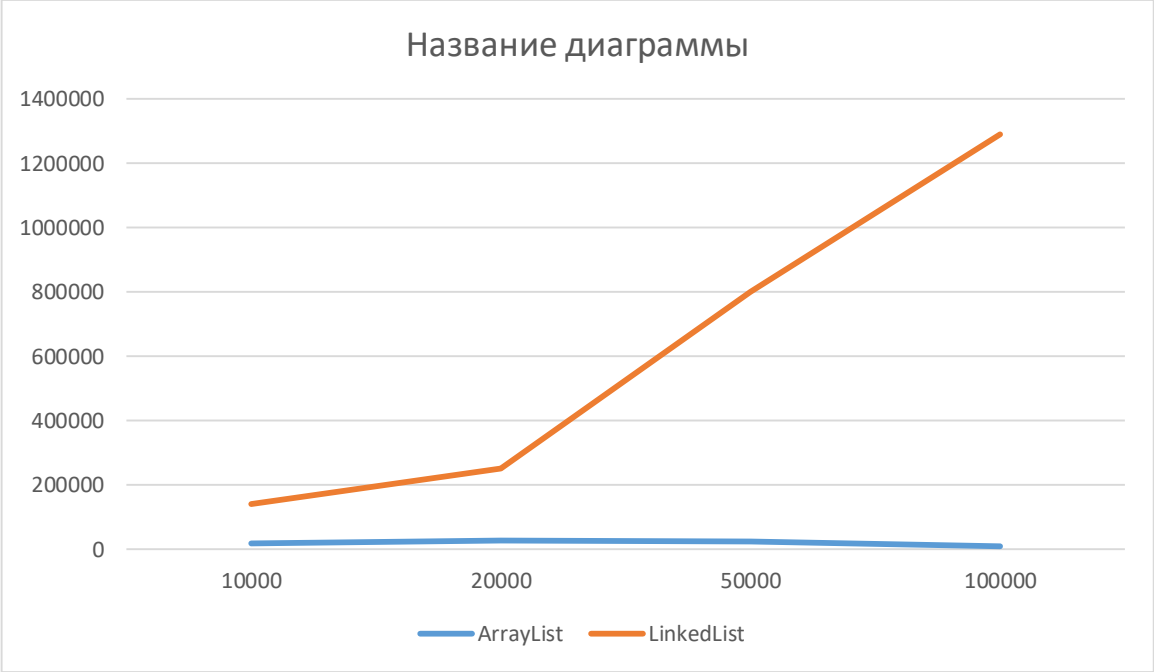
Выводы: операция добавление в LinkedList происходит быстрее, причем, чем больше количество элементов, тем больше он выигрывает, это связано с тем, что у LinkedList связная структура и при добавлении элемента просто создается новый узел и на него добавляется указатель, а в ArrayList существует ограниченная capacity, которая со временем вынуждает делать копирование всего массива.

   - Вставка



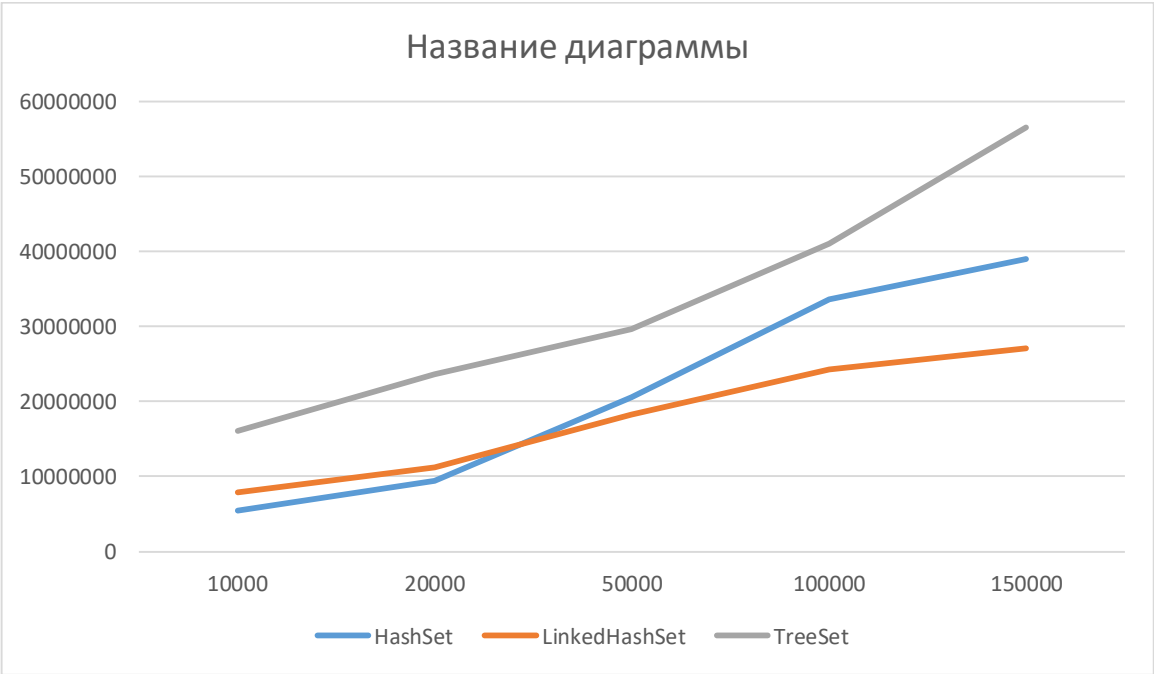
Выводы: вставка элемента производилась в середину списка, и по графику можно увидеть, что ArrayList лучше производит вставку в середину, и также, чем больше элементов, тем лучше это видно.

- Получение элемента



Выводы: Получение элемента в LinkedList постоянно увеличивается, тогда как в arrayList это происходит постоянно на одном уровне, те сложность получения элемента O(1). Если необходимо быстро получить элемент, нужно использовать ArrayList.

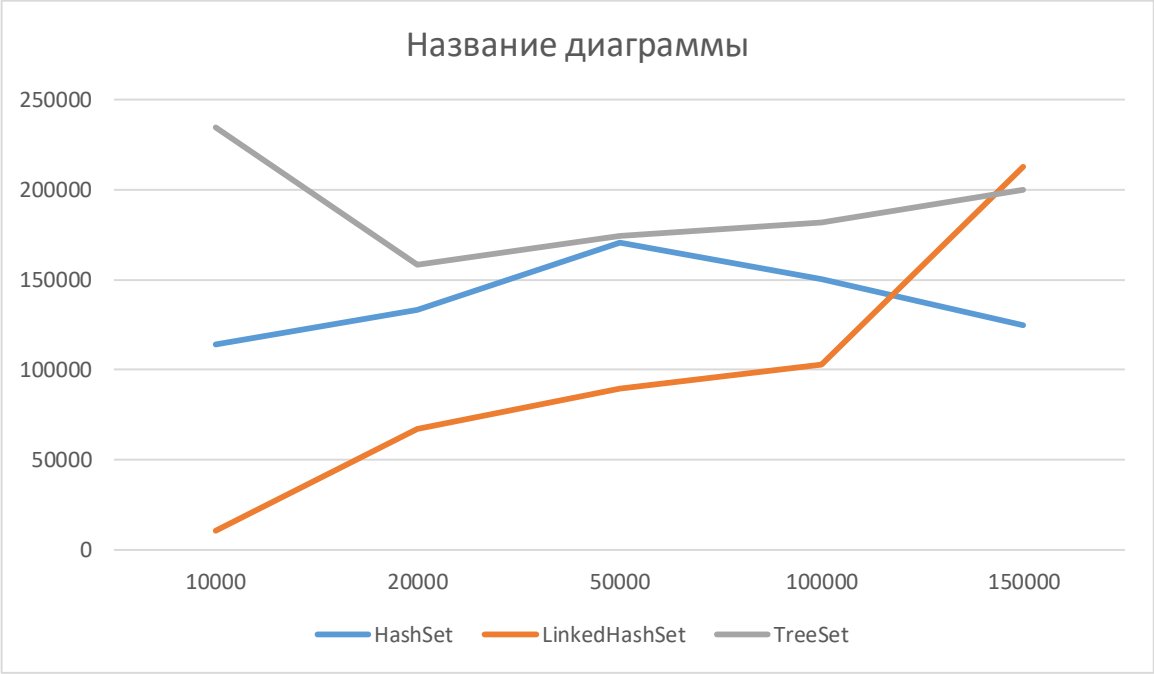
3. HashSet vs LinkedHashSet vs TreeSet
   - Добавление элемента



Выводы: Самым стабильным кажется Linked, тк добавление происходит просто элемент за элементом, Tree выполняется дольше остальных, скорее всего из-за того, что происходит дополнительная сортировка, а обычный Hash работает не стабильно и со временем добавляет дольше. Лучше

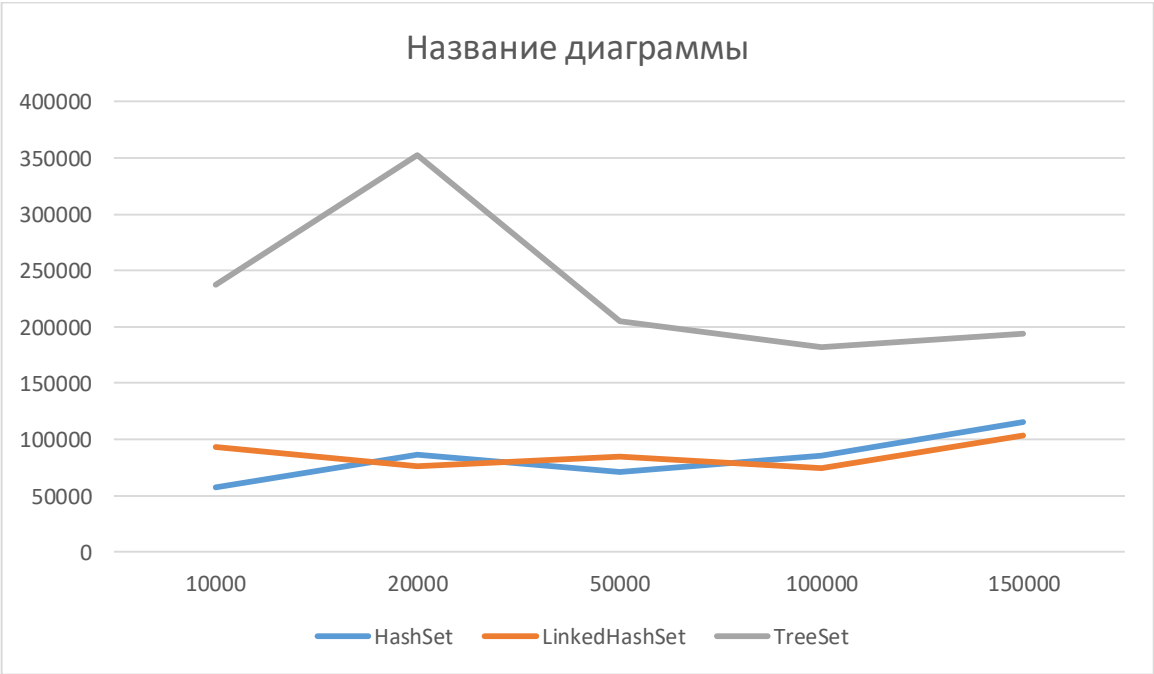
использовать Linked, тк у нас есть и порядок добавления элементов и работает быстрее всех.

- Вставка элемента в существующий сет. По оси x – число элементов в set



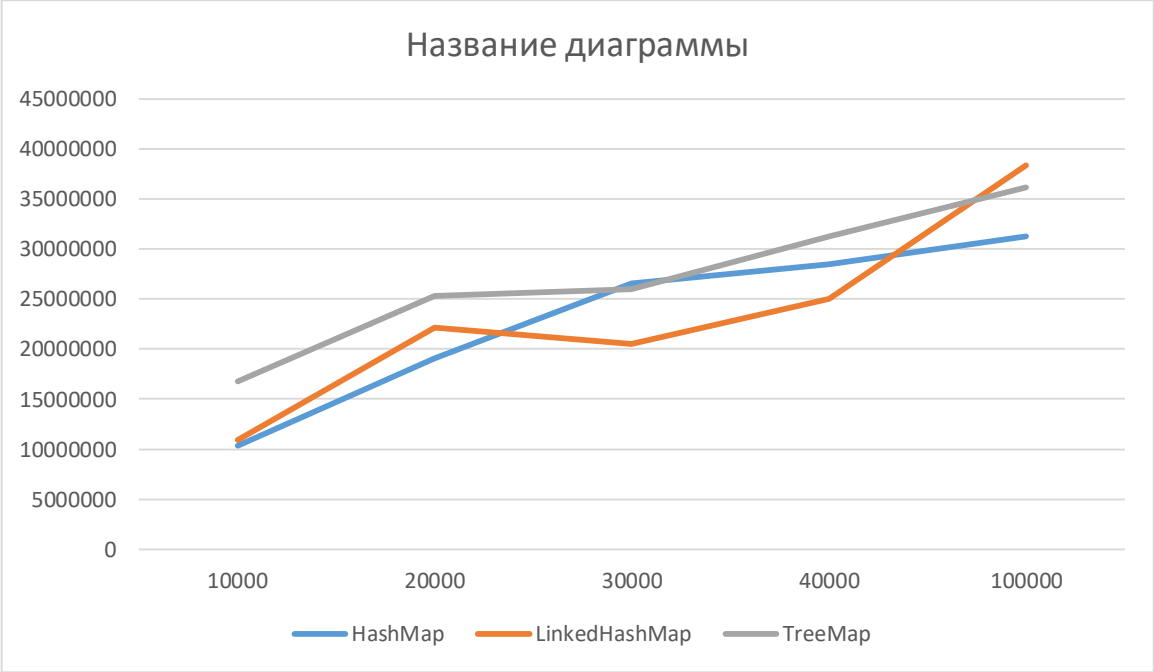
Выводы: при небольшом размере set выгоднее использовать Linked, тк другие set работают примерно за одно время всегда, но уже с большими размерами Linked использовать не выгодно, тк его время всегда растет с увеличением числа элементов.

- Удаление элемента



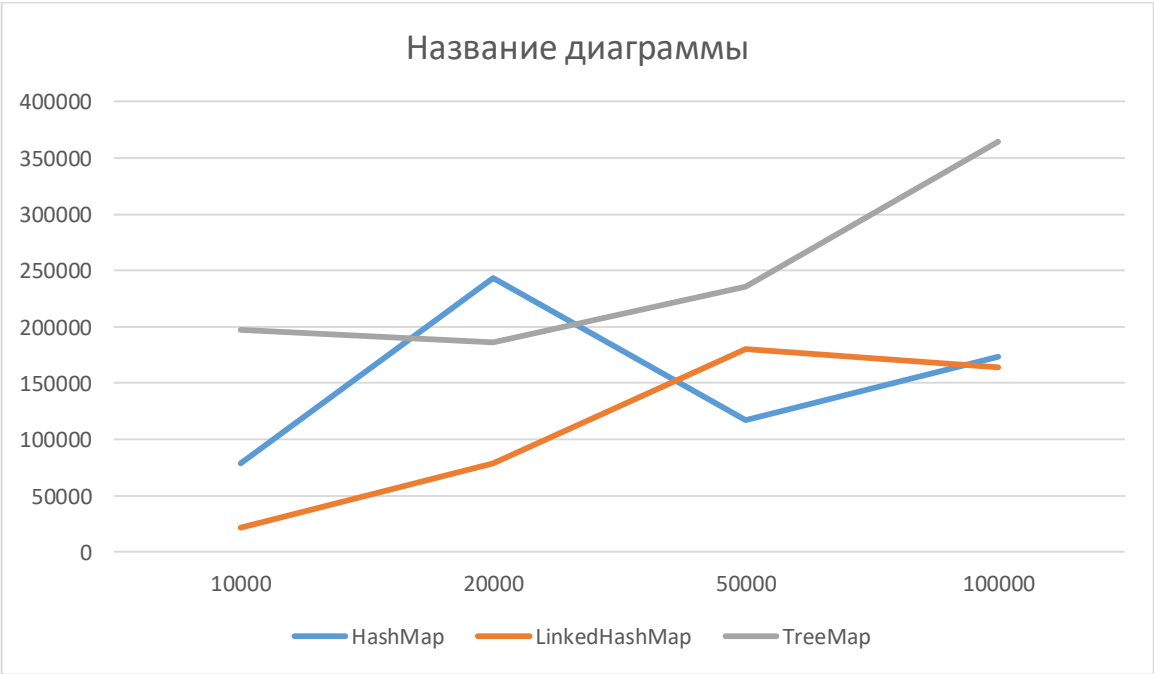
Выводы: удаление элемента у Hash и Linked идет примерно всегда на одном уровне и в раз эффективнее, чем TreeSet, поэтому выгоднее использовать их, даже на больших размерах set.

4. HashMap, LinkedHashMap, TreeMap.
   - Добавление элементов



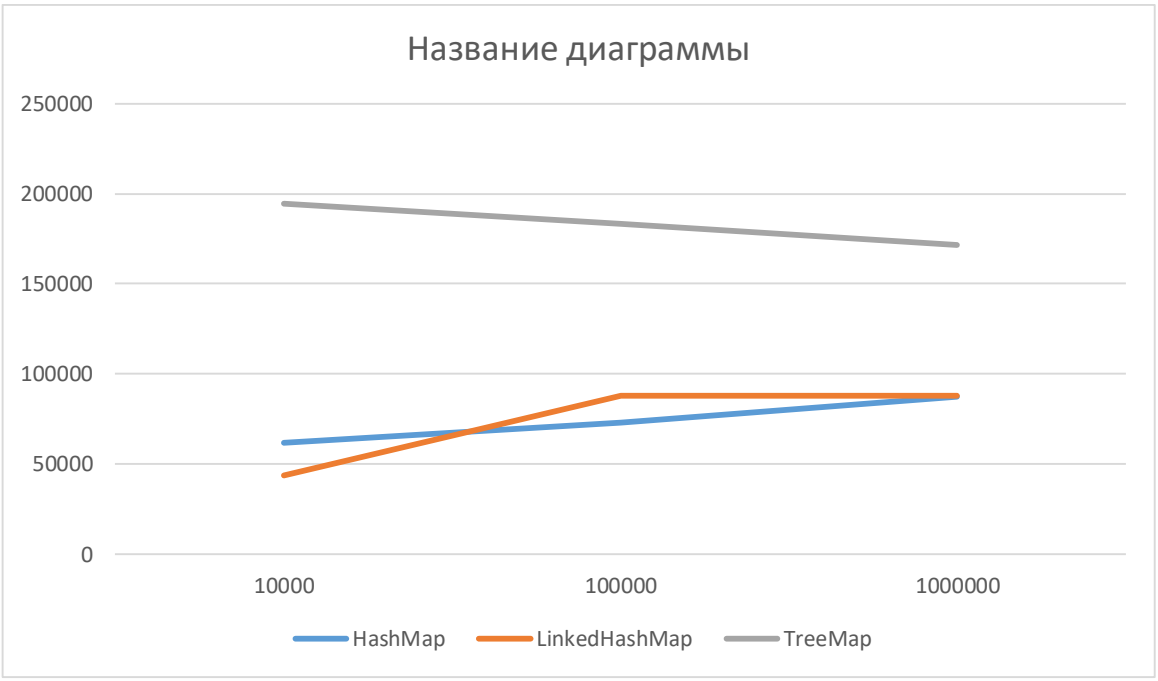
Выводы: для добавления элементов лучше использовать HashMap, тк если мы добавляем много элементов, то Linked растет быстрее, чем HashMap.

   - Вставка элемента



Выводы: Добавление элемента в уже существующий Map осуществляется примерно одинаково и примерно всегда в одно время, здесь уже нужно выбирать, что более предпочтительно, нужна сортировка или нет.

- Удаление элемента



Выводы: с удалением ситуация практически аналогичная, даже на очень больших размерах.

5. Общие выводы:
    - ArrayList – быстрое получение элемента по индексу.
    - LinkedList – быстрая вставка в начало или в конец списка.
    - HashMap – быстрый поиск, удаление, получение элемента по ключу, если сортировка не важна.
    - LinkedHashMap – тоже самое, если важен порядок элементов.
    - TreeMap – дополнительно сортирует значения.
    - Set – для хранения уникальных значений, остальные свойства как у Map.