**Сравнение производительности:**

**1.ArrayList и LinkedList**

-Добавление

Изначально было 500.000 элементов, добавлялось 100.000

|  |  |  |
| --- | --- | --- |
|  | ArrayList | LinkedList |
| В начало | 6,9\* | 6,3\* |
| В середину | 2,9\* | 44,7\* |
| В конец | 3.3\* | 4,7\* |

-Удаление

Изначально было 600.000 элементов,удалялось 100.000

|  |  |  |
| --- | --- | --- |
|  | ArrayList | LinkedList |
| В начало | 6,2\* | 2,6\* |
| В середину | 4\* | 28\* |
| В конец | 4,6\* | 4,8\* |

Выводы:

1)Вставлять/удалять элементы из начала выгоднее с помощью LinkedList(в случае удаления выигрыш во времени огромен)

2) Вставлять/удалять элементы из середины выгоднее с помощью ArrayList.

3) Результаты данных операций для конца списка примерно одинаково и гораздо меньше аналогичных операций для начала и середины списка.

Полученные результаты можно объяснить тем, что ArrayList – список на основе массива, а LinkedList – связный список на основе элементов и связи между ними, у него быстрый доступ к первому и последнему элементу. Поэтому при работе с началом и концом списка используют, как правило LinkedList. ArrayList, в свою очередь, имеет лучшую производительность при работе с элементами по индексу (этим и объясняются лучшие результаты в середине списка), что делает его более универсальным

**2.HashSet, LinkedHashSet and TreeSet**

Каждая операция выполнялась для 100.000 элементов(Начальный размер - 500.000 элементов)

|  |  |  |  |
| --- | --- | --- | --- |
|  | HashSet | LinkedHashSet | TreeSet |
| Add(String) | 55\* | 28\* | 70\* |
| Contains(String) | 53\* | 105\* | 60\* |
| Remove(String) | 35\* | 32\* | 82\* |

Выводы: HashSet имеет лучший результат при поиске элемента, в остальных функциях лидирует LinkedHashSet. TreeSet по сумме показателей заметно уступает первым двум коллекциям.

**3.HashMap, LinkedHashMap and TreeMap**

Каждая операция выполнялась для 100.000 элементов(Начальный размер - 500.000 элементов)

|  |  |  |  |
| --- | --- | --- | --- |
|  | HashMap | LinkedHashMap | TreeMap |
| Put(String key,String) | 22\* | 18\* | 45\* |
| Get(String key) | 25\* | 18\* | 32\* |
| Remove(String key) | 26\* | 16\* | 50\* |

Выводы: Результы HashMap и LinkedHashMap примерно одинаковы и оптимальны(но все-таки LinkedHashMap немного лидирует). TreeMap заметно уступает, за исключением метода Get(key)(можно объяснить тем, что элементы хранятся в отсортированном порядке по ключу, на что и уходит дополнительное время).