**Отчёт**

**1.Сравнение MyLinkedList и LinkedList.**

(Проводилось при числе элементов, равном 500000)

|  |  |  |
| --- | --- | --- |
|  | MyLinkedList | LinkedList |
| Add(Element) | 10.85\*10^7 | 10.17\*10^7 |
| Add(index, Element) | 2.62\*10^6 | 2.98\*10^6 |
| IndexOf(Element) | 3.36\*10^6 | 3.51\*10^6 |
| remove(index) | 11.69\*10^8 | 6.31\*10^8 |

Таким образом, на основании представленной таблицы можно увидеть, что MyLinkedList выполняет основные операции за время того же порядка, что и LinkedList.

**2.Сравнение LinkedList и ArrayList.**

Добавление.

Было 500000 элементов, добавлялось 100000.

|  |  |  |
| --- | --- | --- |
|  | ArrayList | LinkedList |
| В начало | 71,8\*10^8 | 3.72\*10^6 |
| В середину | 34.4\*10^8 | 49.26\*10^9 |
| В конец | 17.77\*10^5 | 24.46\*10^5 |

Удаление.

Было 600000 элементов, удалялось 100000.

|  |  |  |
| --- | --- | --- |
|  | ArrayList | LinkedList |
| Из начала | 72.23\*10^8 | 2.97\*10^6 |
| Из середины | 45.96\*10^8 | 39.68\*10^9 |
| Из конца | 46.84\*10^5 | 34.43\*10^5 |

Выводы:

1)Вставлять и удалять элементы из начала выгоднее с помощью LinkedList.

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

3) Операции вставки и удаления из конца являются операциями примерно одного порядка. Явного превосходства во времени выполнения ни у одной из них нет. Однако, нельзя не отметить, что данные операции для конца списков значительно быстрее аналогичных операций для начала и середины.

Полученные результаты можно объяснить тем, что ArrayList - список на основе массива, а LinkedList - связный список на основе элементов и связи между ними. Он имеет более быстрый доступ к первому и последнему элементам. Поэтому при работе с началом и концом списка выгоднее использовать 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.

**3.Сравнение HashSet, LinkedHashSet и TreeSet.**

Было 500000 элементов. Каждая операция выполнялась для 100000.

|  |  |  |  |
| --- | --- | --- | --- |
|  | HashSet | LinkedHashSet | TreeSet |
| Add(String) | 95.98\*10^6 | 18.7\*10^6 | 35.38\*10^6 |
| Contains(String) | 31.90 \*10^6 | 20.35 \*10^6 | 35.87 \*10^6 |
| Remove(String) | 30.22 \*10^6 | 22.78 \*10^6 | 59.75 \*10^6 |

Выводы: LinkedHashSet имеет лучшие результаты. TreeSet лучше HashSet в добавлении элементов, но хуже в удалении.

**4.Сравнение HashMap, LinkedHashMap и TreeMap.**

Было 500000 элементов. Каждая операция выполнялась для 100000.

|  |  |  |  |
| --- | --- | --- | --- |
|  | HashMap | LinkedHashMap | TreeMap |
| Put(String key,String) | 36.49\*10^6 | 14.20 \*10^6 | 59.72 \*10^6 |
| Get(String key) | 22.52 \*10^6 | 15.36 \*10^6 | 48.70 \*10^6 |
| Remove(String key) | 23.49 \*10^6 | 14.65 \*10^6 | 38.07 \*10^6 |

Выводы: LinkedHashMap имеет лучшие результаты во всех операциях. Следом идет HashMap. TreeMap заметно уступает.