**ArrayList & LinkedList**

Используем ArrayList:

Когда часто нужно получать данные по индексу

Когда добавление/удаление происходит в серидину, в конец/с середины, с конца

Используем LinkedList:

Когда добавление/удаление происходит в начало, в конец / с начала, с конца

**HashSet, LinkedHashSet, TreeSet**

Набор входных данных от (количество элементов - 1) до нуля.

Данные для удаления от нуля до (количество элементов - 1)

Используем TreeSet:

Нужно отсортированное в порядке возрастания множество при каждом новом добавлении

Используем LinkedHashSet:

Необходим порядок записи данных

В остальных HashSet

LinkedHashSet & HashSet: если нужно константное время записи, удаления элемента

**HashMap, LinkedHashMap, TreeMap**

Используем TreeMap:

Нужны отсортированные по ключу входные записи при каждой новой записи

HashMap:

Топ. Когда нужна производительность хорошая, неважно в каком порядке ключи и в каком порядке поступали данные

LinkedHashMap:

Когда нужна производительность, а также когда важен порядок записи данных в map.

Вообще, по замерам, если нужен лучший перфоманс, то структуры с Tree… работают медленнее, так как при каждой новой записи/удаления записи они балансируются или проверяются на сбалансированность.