ArrayList v LinkedList

**ArrayList** is implemented as a resizable array. As more elements are added to ArrayList, its *size* is increased dynamically. It's elements can be accessed directly by using the get and set methods, since ArrayList is essentially an array.

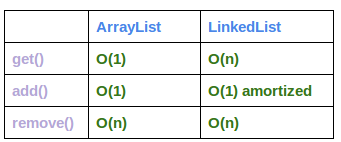
**LinkedList** is implemented as a double linked list. Its performance on add and remove is better than Arraylist, but worse on get and set methods.

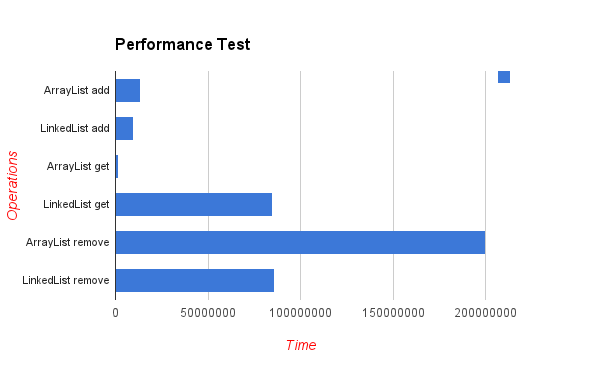
The difference of their performance is:

LinkedList is **faster** in add and remove, but slower in get. Based on the complexity table and testing results, we can figure out when to use ArrayList or LinkedList. In brief, LinkedList should be preferred if:

* there are no large number for random access of element
* there are a large number of add/remove operations

The time complexity comparison is as follows:





Sample runs (Comparisons)

|  |  |
| --- | --- |
| **Run 1** | **Run 2** |
| **ArrayList add: 0.005463203**  **LinkedList add: 0.0043239**  ArrayList get: 1.14699E-4  LinkedList get: 2.1598E-5  **ArrayList remove: 0.120621998**  **LinkedList remove: 0.096636284** | **ArrayList add: 0.006657748**  **LinkedList add: 0.006391073**  ArrayList get: 1.27034E-4  LinkedList get: 5.1482E-5  **ArrayList remove: 0.125247958**  **LinkedList remove: 0.111112507** |

Conclusion

