Shuni Li
COMP 221 HWA - Experiments with Data Structure
Professor Shilad Sen
Jan 31, 2016

This program, HWA.java (https://github.com/Shunili/CS221Homework) intends to analyze the performance of add and delete operations on lists and dictionaries. For lists, I adopt ArrayList in order to compare the performance of adding and removing to the front, back, and middle. For dictionary, I adopt hashmap and timed these two methods: put(Key k, Value v) and remove(Key k). The following table shows all the data I get by runing HWA.java:

Average seconds per operation:

n	1.00	10.00	100.00	1k	10k	100k	1M	10M	100M
add to dict	0.00E+00	0.00E+00	0.00E+00	4.00E-07	2.59E-07	1.35E-07	2.23E-07	5.42E-07	Too Big
del from dict	0.00E+00	0.00E+00	2.00E-06	2.00E-07	1.80E-07	7.30E-08	9.96E-08	4.73E-07	Too Big
add to front of list	0.00E+00	0.00E+00	0.00E+00	6.00E-07	1.17E-06	7.73E-06	9.73E-06	Too Big	Too Big
add to middle of list	0.00E+00	0.00E+00	1.00E-06	3.00E-07	3.20E-07	3.38E-07	Too Big	Too Big	Too Big
add to end of list	0.00E+00	0.00E+00	0.00E+00	1.00E-07	7.00E-08	2.70E-08	1.73E-08	9.11E-08	Too Big
del from front of list	0.00E+00	0.00E+00	0.00E+00	2.00E-07	5.30E-07	7.53E-06	Too Big	Too Big	Too Big
del from middle of list	0.00E+00	0.00E+00	1.00E-06	1.00E-07	2.80E-07	3.25E-06	Too Big	Too Big	Too Big
del from end of list	0.00E+00	0.00E+00	1.00E-06	1.00E-07	1.00E-08	9.00E-09	4.40E-09	4.73E-09	Too Big

From the table, we can see that both the add and remove operations on Dictionary requires a relatively constant time (average is 3.12E-07 seconds) regardless of size of n. This resonates with the theory that Dictionary has a $\Theta(1)$ time complexity for adding and removing elements. The constant time complexity can be explained by the special structure of key-value pair, which is very efficient for simple adding and deleting operations.

As for ArrayList, things get more complicated.

The table shows that the time required to complete one operation is very small in general regardless of n for both operations -- adding to the end and deleting from the end of the list. This result of $\Theta(1)$ time complexity is to be expected since the feature of indexes of ArrayList makes adding to back and deleting from back very quick.

For the rest 4 operations on List, as n increases, the time required to complete a single operation increases as well. This overall trend is very reasonable, as we know that the complexity for these 4 operations are $\Theta(n)$. However, it is very difficult to predict the linear relationship just by these limited amount of data.

The two data that are colored in red are not consistent with my expectations. The first one is adding to middle of a list with size 100. In this case, one operation needs 1.00E-06 seconds, which is longer than the time needed for the same operation with bigger sizes. The same phenomenon happens to the delete from middle of the list operation. I'm not sure why this happens, but one possible explanation could be relatively high overhead cost of setting up that operation for a smaller n.