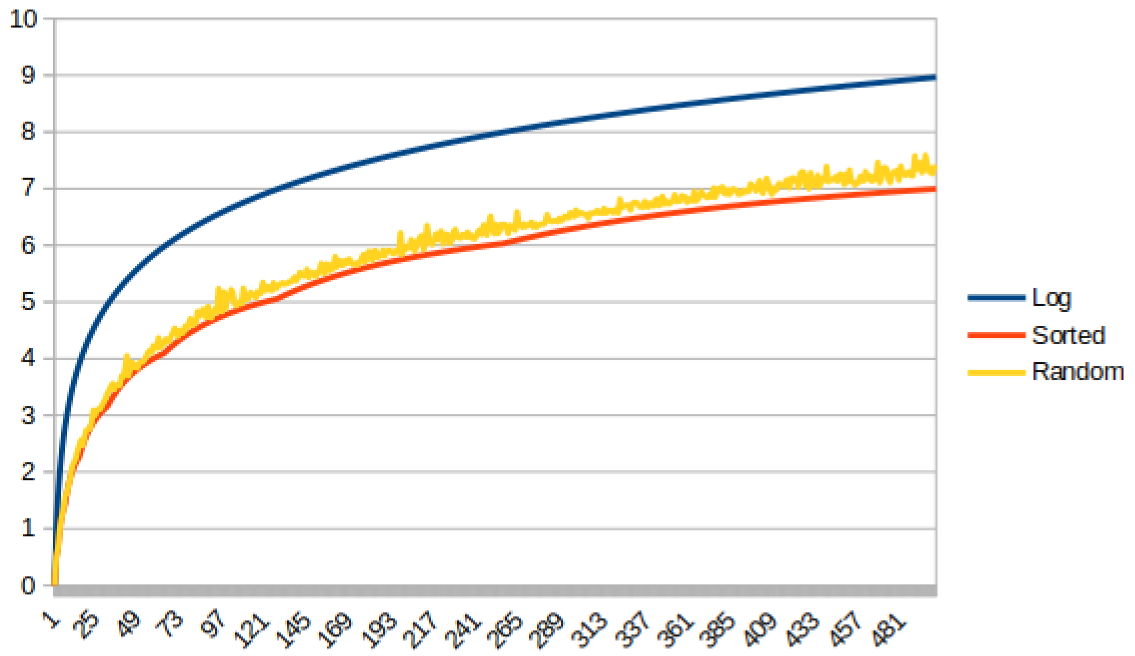
Xin Yang
xy213

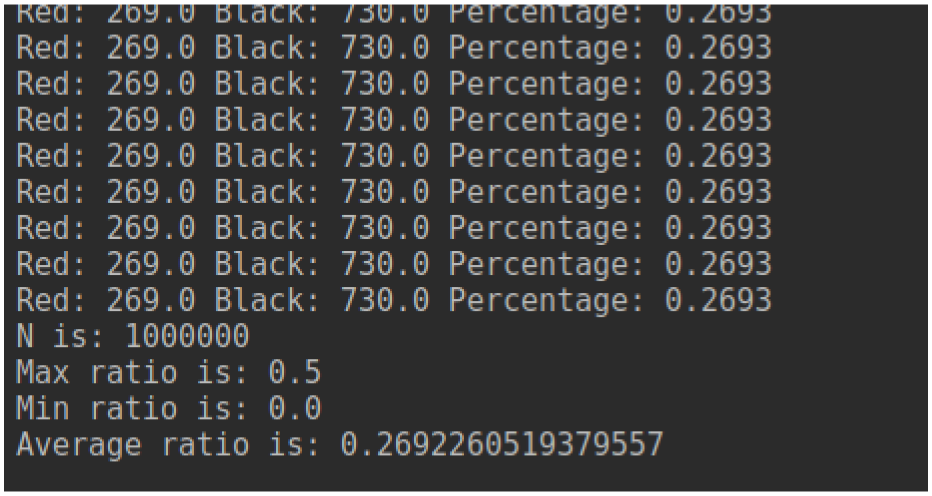
Report of Homework 3

Q1:
The program by default inserts nodes which have the same keys and values from 1 to 15, which is a 4 level binary search tree, and the results are printed by pre-order traversal, empty nodes are shown as “E”. To change the datasets, you need to change the “for” loop in the main function, which put the nodes into the tree.

Q2:
Since the detailed structure of the tree is not clarified, here I suppose the structure of the tree is a standard red-black tree, to make the results more accurate, the code of the red-black tree is referenced from the official source code of Algorithms 4th Edition, and I implemented the experiments based on this. As the result shows, the average length path will never exceed log(N), where N is the total number of nodes in the tree. This result is consistent with the feature of a red-black tree since it’s always perfectly balanced and the heights will always be within the range of log(N). Also, the results of N-sorted insertions are fixed for a certain N, but the results of N-random insertions might change due to the order of insertion, and the average path length will always greater than or equal to(relatively rare) N-sorted insertions.
My program will ask you to type the N and stores the result into an Output.txt file for the convenience of analysis and plot, the graph below is the result when N is 500.

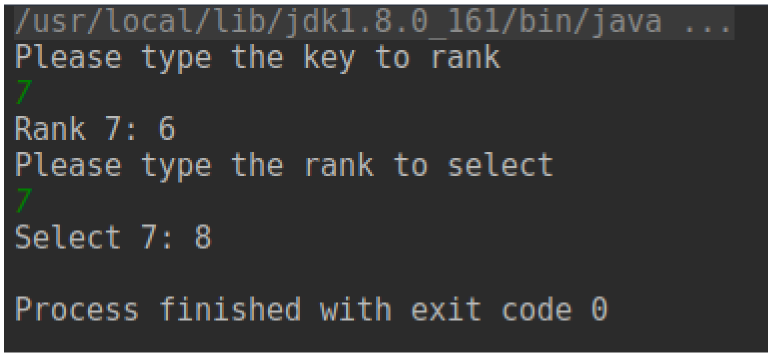


Q3:
In this part, code for red-black tree structure is also referenced from Algorithms 4th Edition’s source code to ensure the correctness. My implementation will ask you to input the number of N. In this part, for N from 10^4 to 10^6, the max ratio of red nodes is 0.5, and the min ratio is 0, the average ratio for 10^4 is around 26.494%, for 10^5 is around 26.883%, and for 10^6, the average is around 26.922%. No matter what the size of N, when comes to the end of execution, the ratio of the current tree is always 0.2693, which explains why when N grows, the mean ratio is getting closer to 26.93%.


```
Red: 269.0 Black: 730.0 Percentage: 0.2693
Red: 269.0 Black: 730.0 Percentage: 0.2693
Red: 269.0 Black: 730.0 Percentage: 0.2693
Red: 269.0 Black: 730.0 Percentage: 0.2693
Red: 269.0 Black: 730.0 Percentage: 0.2693
Red: 269.0 Black: 730.0 Percentage: 0.2693
Red: 269.0 Black: 730.0 Percentage: 0.2693
Red: 269.0 Black: 730.0 Percentage: 0.2693
Red: 269.0 Black: 730.0 Percentage: 0.2693
N is: 1000000
Max ratio is: 0.5
Min ratio is: 0.0
Average ratio is: 0.2692260519379557
```

Q4:
Also in this part, red-black tree code is referenced from Algorithms 4th Edition. The results are stored in the Output.txt file. The average path length is always less or equal to the log(N), which proves the properties of a red-black tree. Meanwhile the standard deviation is quite small, when the N is getting larger, standard deviation is around 0.06, which is quite small, it also proves that red-black tree can perfectly balance itself thus it can achieve such an excellent performance.

Q5:
The value of select(7) is 8, the value of rank(7) is 6. My program allows you to type the input number and show the result of select or rank. For this dataset, all keys range from 1 to 999, so my result can be verified to be correct in a Math way.


```
/usr/local/lib/jdk1.8.0_161/bin/java ...
Please type the key to rank
7
Rank 7: 6
Please type the rank to select
7
Select 7: 8

Process finished with exit code 0
```