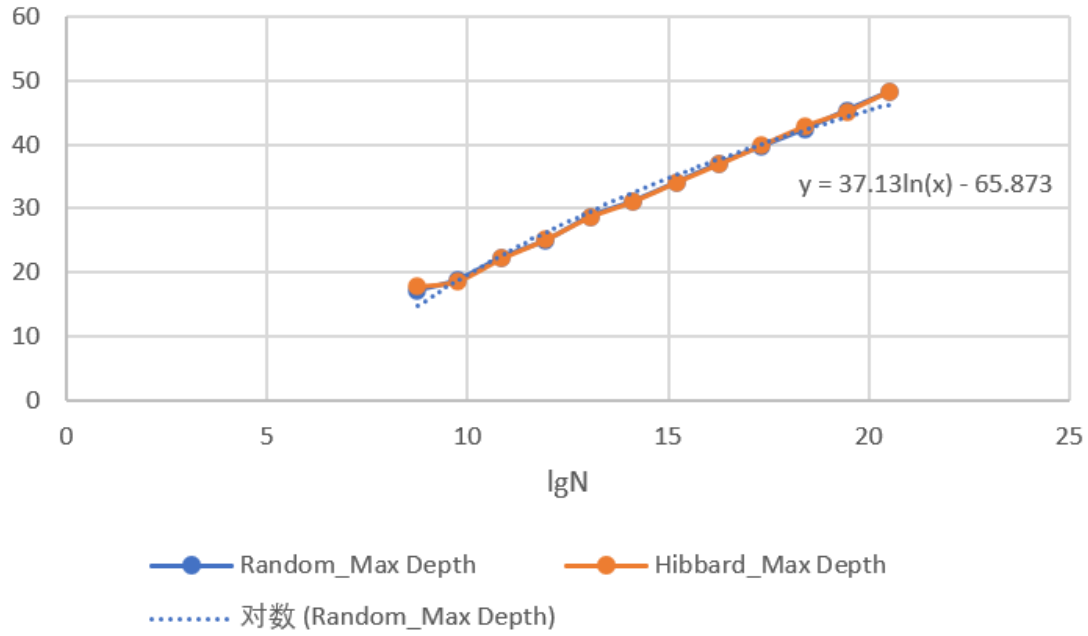
I implemented 2 approaches of deletion of nodes in a binary search tree: use the classics “Hibbard” deletion and randomly choose the right vs. left approach that follow the Arbitrary Substitution Principle. In the second approach, the binary tree will randomly pick the node with the smallest key in the right-handed subtree of the node to be deleted or the node with the largest key in the left-handed subtree of the node to be deleted for exchange. In each case, I mix different number of key additions randomly with half number of deletions from 512 insertions and 256 deletions up to 2^20 insertions and 2^19 deletion to build the BSTs and compare the maximum depth of the BST. For each group of specific number of insertion and deletion operation input, I run 50 times and get the average size N and the average maximum depth of the BSTs to ensure the accuracy. Since the average N is different between two cases, I import lgN column for better comparative analysis (lgNs are the same).

The result data are as follows:

lgN	Random_Max Depth	Hibbard_Max Depth
8.75	17.14	17.74
9.75	18.8	18.62
10.85	22.32	22.14
11.95	25	25.1
13.05	28.7	28.6
14.14	31.16	31.08
15.21	34.1	34.06
16.27	37.06	36.88
17.34	39.74	39.86
18.4	42.4	42.8
19.45	45.38	45.14
20.52	48.28	48.28

I also drew the line graphs for the two approaches with trend lines which roughly indicate the relationship between lgN and avg maximum depth.



From the data and the graphs, we can see that the maximum depth increases with the increase of N and present logarithmic relations and the average

maximum depth of two approaches is similar, the general formula is MaxDepth = 37.13ln(lgN) - 65.873. But more often, the maximum depth in “Random deletion” method is less than it in “Hibbard deletion” method. This means that the distribution of the binary tree obtained by the “Hibbard” deletion method is not symmetrical and the behavior is not good enough. However, the random deletion method almost guarantees logarithmic performance for all operations instead of power (the original deletion method performs as root n on average, which is 1/2 power of N). “Random deletion” improves the original method to a certain extent, more comprehensive and with better behavior. But the improvement degree is slight, which means that it is not the best and most fundamental way to improve the performance of BST. We need to find other methods to improve the performance of data structure.

- BSTTest.java screenshot