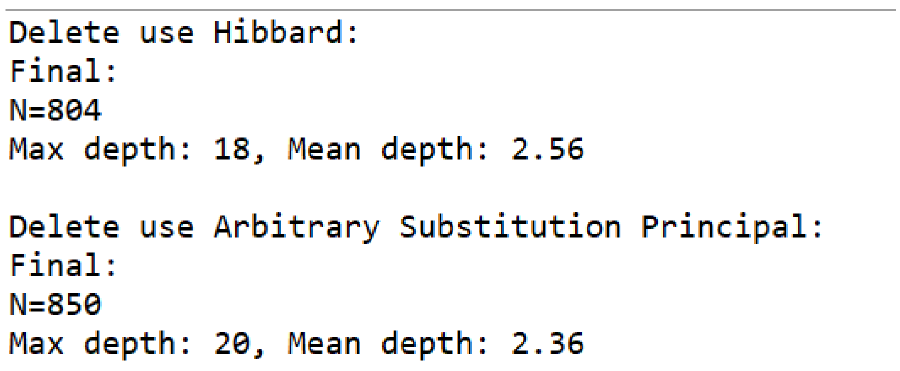
I implemented 2 approaches of deletion of nodes in a binary search tree: use the classics "Hibbard" deletion and randomly choose the right vs. left approach that follow the Arbitrary Substitution Principle. In the second approach, the binary tree will randomly pick the node with the smallest key in the 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 1024 key additions randomly with 512 deletions to build the BST and compare the maximum depth and average depth of the BST.

One of the outputs after 1024 key additions and 512 deletions is as follows:



```
Delete use Hibbard:
Final:
N=804
Max depth: 18, Mean depth: 2.56

Delete use Arbitrary Substitution Principal:
Final:
N=850
Max depth: 20, Mean depth: 2.36
```

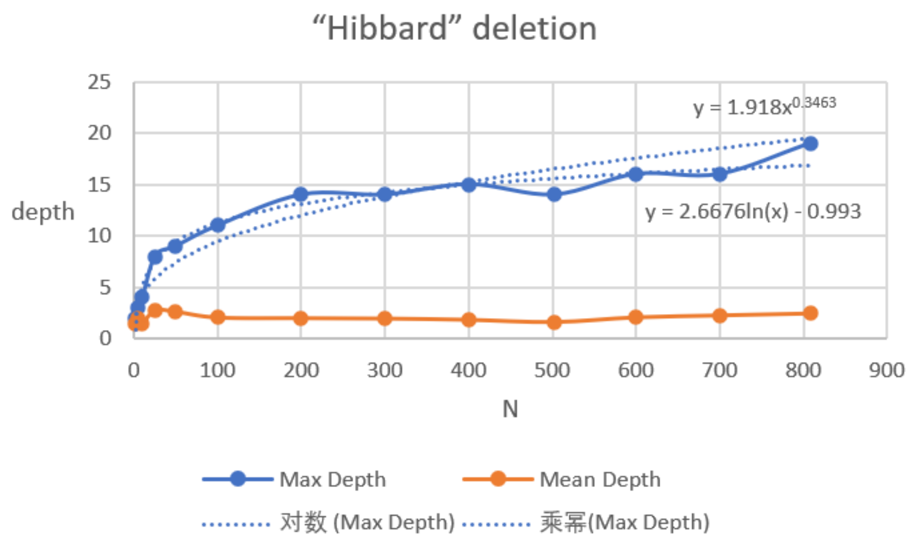
After running for many times, I got similar results, and found that the maximum depth of the two approaches of deleting nodes is similar, about 18 to 20, but the average depth of nodes in the random selection deletion method is always less than "Hibbard" deletion method. This means that the distribution of the binary tree obtained by the "Hibbard" deletion method is not symmetrical and the behavior is not good enough. So random selection deletion method is better, more comprehensive, and get more balanced tree.

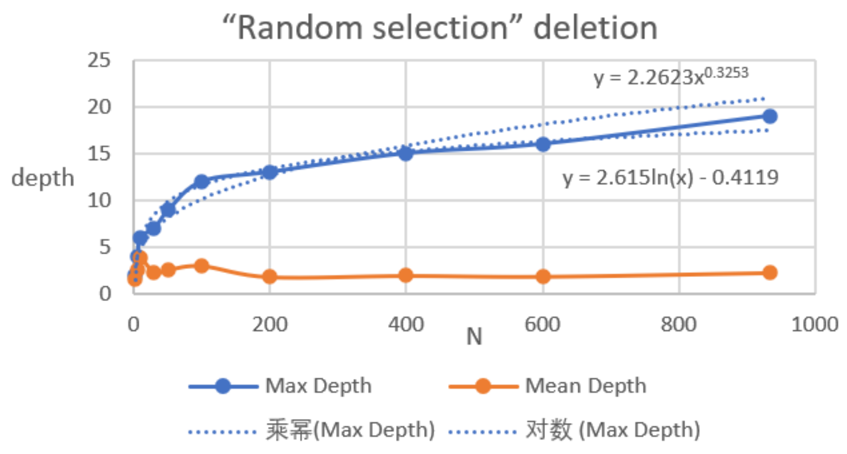
In each operation of insertion and deletion of BST, I record the size of BST N to find the relationship between N and max depth and mean depth.

Some of the result data are as follows:

	N	Max Depth	Mean Depth
"Random selection" deletion	933	19	2.23
	600	16	1.83
	400	15	1.92
	200	13	1.78
	100	12	2.98
	51	9	2.59
	30	7	2.23
	9	6	3.89
	5	4	2.6
	2	2	1.5
"Hibbard" deletion	808	19	2.45
	700	16	2.26
	600	16	2.07
	502	14	1.61
	400	15	1.83
	300	14	1.95
	200	14	1.99
	100	11	2.05
	50	9	2.62
	25	8	2.76

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

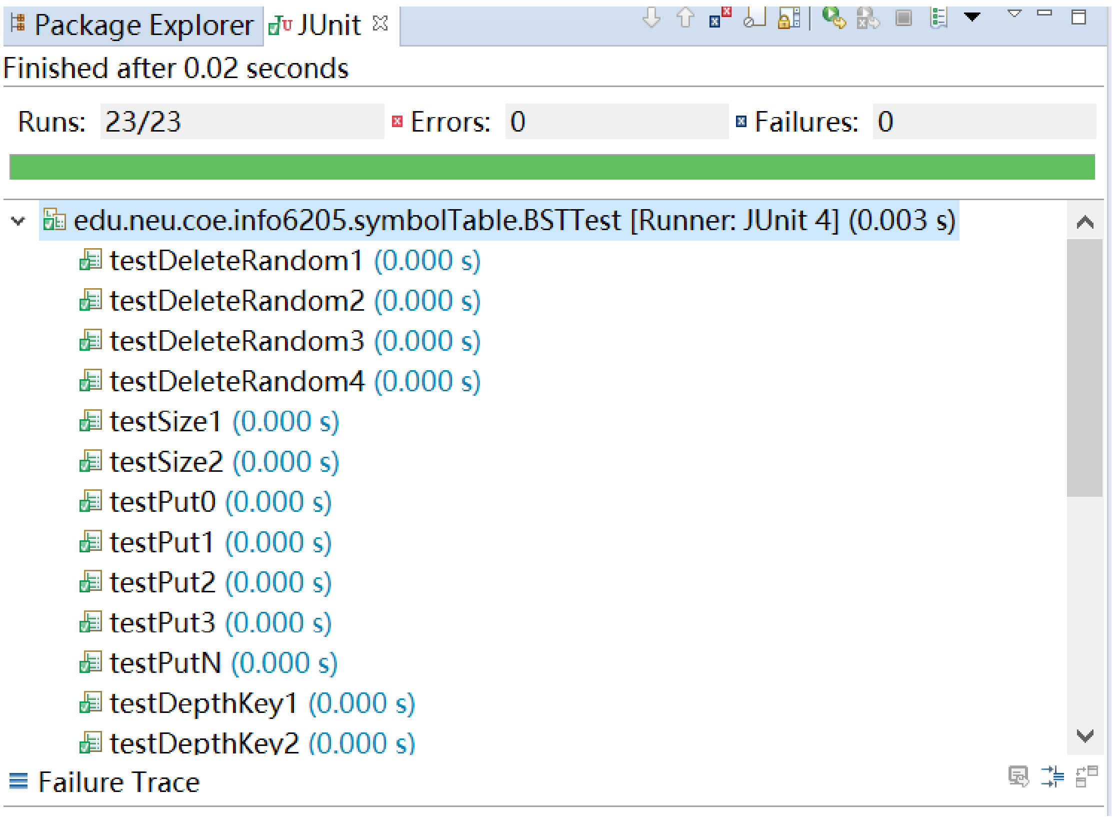




From the data and the graphs, we can see that the maximum depth increases with the increase of N present logarithmic or power relations. There is little difference between the two approaches, but in the randomly selected deletion approach, the relationship between N and the maximum depth is more logarithmic, with the formula y = 2.615ln(x) - 0.4119, while in the Hibbard deletion approach, the relationship between N and the maximum depth tends to be 1/2 power, with the formula y = 1.918x0.3463.

Therefore, the random deletion method guarantees logarithmic performance for all operations and improves the original method to a certain extent, with better behavior. But the improvement degree is not big, 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



```
Finished after 0.02 seconds
Runs: 23/23    Errors: 0    Failures: 0

edu.neu.coe.info6205.symbolTable.BSTTest [Runner: JUnit 4] (0.003 s)
    testDeleteRandom1 (0.000 s)
    testDeleteRandom2 (0.000 s)
    testDeleteRandom3 (0.000 s)
    testDeleteRandom4 (0.000 s)
    testSize1 (0.000 s)
    testSize2 (0.000 s)
    testPut0 (0.000 s)
    testPut1 (0.000 s)
    testPut2 (0.000 s)
    testPut3 (0.000 s)
    testPutN (0.000 s)
    testDepthKey1 (0.000 s)
    testDepthKey2 (0.000 s)
Failure Trace
```