HOMEWORK 3

Data Structure & Algorithm

Weijia Sun

Ws368 | 03/27/2018

HW3 Report
Weijia Sun (ws367)

Q1

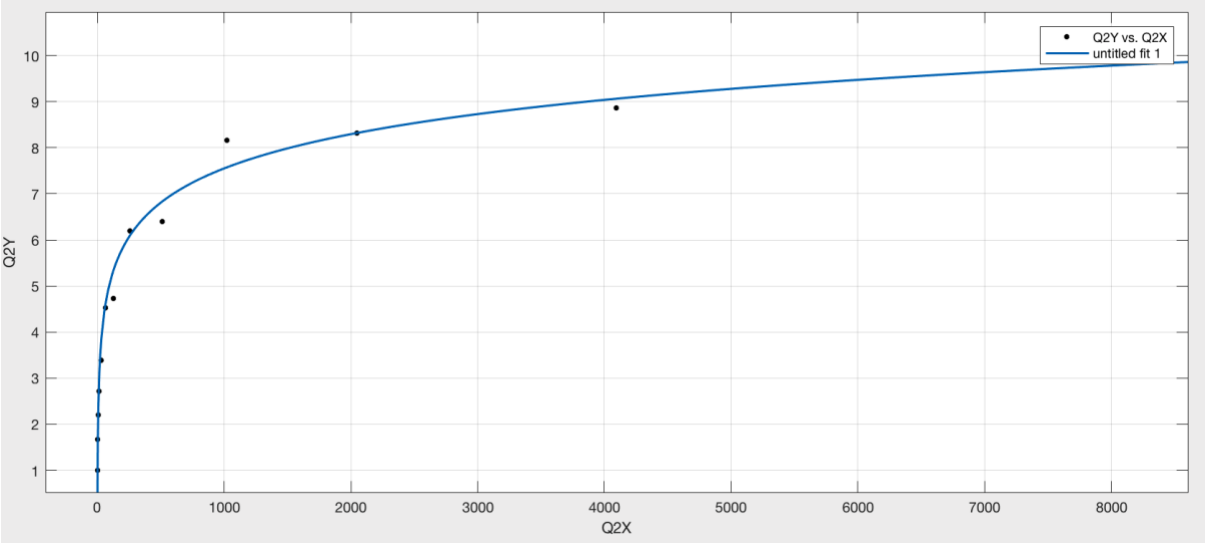
Basically, the 2-3 tree without balance restriction is a red black tree without rotateLeft, rotateRight, flip operation. because all 3-node can lean either way. But when hook the new node onto the bottom , you must link it with the parent node with a black link. So when you insert a new node, you have to check whether it is a 3-node. So by default, I insert a red node, if the node is a 3-node , I change it to a black node.
You can check the implementation in detail in my code.

Q2

For Q2, I think the question wants us to run experiments based upon the tree in Q1. Since the tree in Q1 is a 2-3 tree, so when calculate the path length, we have to ignore the red node, because the red node is a 3-node, if I calculate the red node, I will calculate the path length for 3-node twice. Also, after you get the total path length, you have to divide it by tree size minus number of red node. So I got the result as follows.

DataSet	Random	Sorted
1	1	1
2	1	1
4	1.67	1.5
8	2.2	2.5
16	2.72	4.5
32	3.39	8.5
64	4.53	16.5
128	4.73	32.5
256	6.19	64.5
512	6.4	128.5
1024	8.16	256.5
2048	8.32	512.5
4096	8.86	1024.5
8192	10.47	2048.5

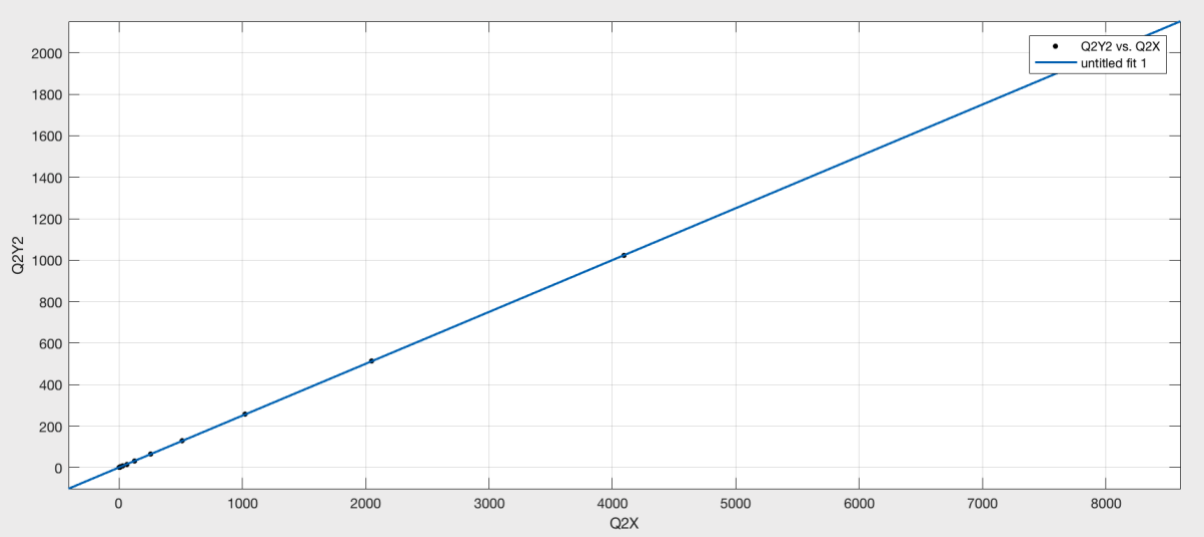
The curve fitting figure for random insert is as follow:



the curve fitting equation for random insert is :

$$y = 0.74 * log_2(x) + 0.1437$$

the curve fitting figure for sorted insertion is:

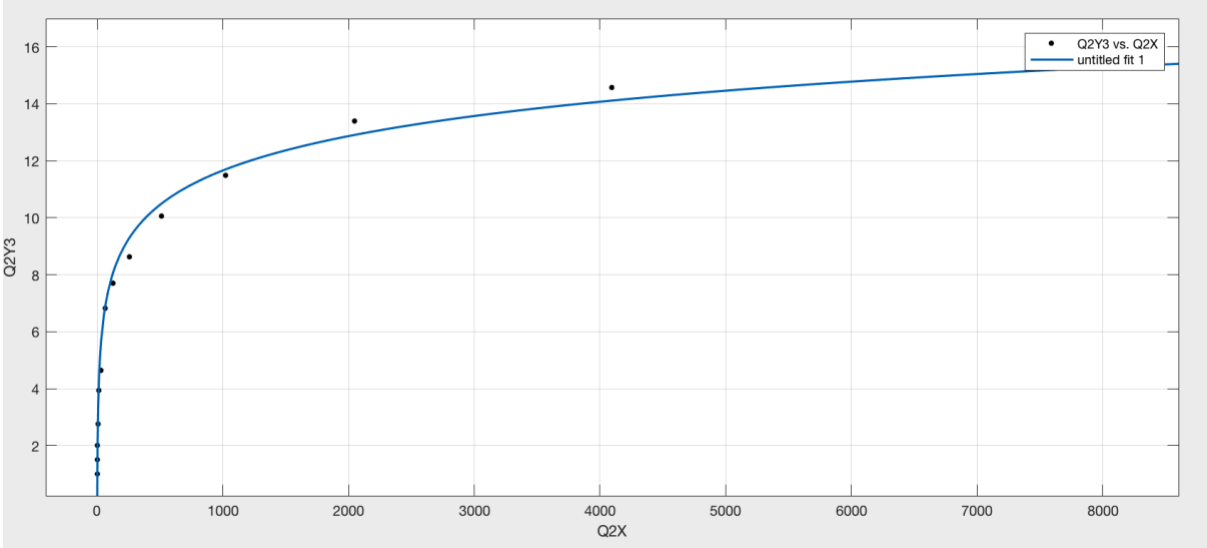


the curve fitting equation for random insert is :

$$y = 0.25 * x + 0.5$$

if we treat this tree as a red black tree, we need to calculate the internal path length divided by tree size. That is, add the path length from the root node to all the node in the tree. If we do so, we will get the result as follows:

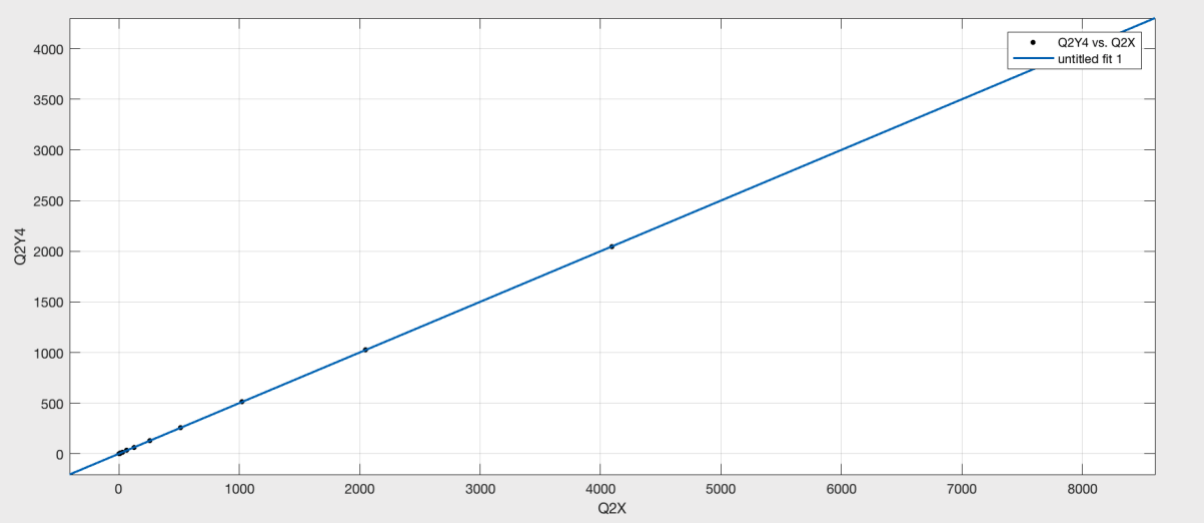
DataSet	Random	Sorted
1	1	1
2	1.5	1.5
4	2	2.5
8	2.75	4.5
16	3.93	8.5
32	4.63	16.5
64	6.81	32.5
128	7.71	64.5
256	8.62	128.5
512	10.06	256.5
1024	11.48	512.5
2048	13.4	1024.5
4096	14.56	2048.5
8192	16.23	4096.5



the curve fitting equation for random insert is :

$$y = 1.206 * log_2(x) - 0.364$$

the curve fitting figure for sorted insertion is:



the curve fitting equation for sorted insertion is:

$$y = 0.5 * x + 0.5$$

Q3

For this question, code for red black tree is referenced from Algorithm 4th by princeton official source code from Github. I implement the method to calculate the number of red node and calculate its percentage.

```
public double countRedPercentage() {
    return countRed(root);
}

private double countRed(Node x) {
    List<Integer> list = new ArrayList<>();
    countRedHelper(x, list);
    return (double) list.size()/size();
}

private void countRedHelper(Node x, List<Integer> list) {
    if (x == null)
        return;
    if (x.color == RED)
        list.add(1);
    countRedHelper(x.left,list);
    countRedHelper(x.right,list);
}
```

When I increase N random keys into an initially empty tree for N = 10^4, 10^5, 10^6, each set I test 100 times. I came up with the following result.

```
The percentage of red nodes after 10000-increasing insertions is: 25.406800000000004%
The percentage of red nodes after 100000-increasing insertions is: 25.382420000000007%
The percentage of red nodes after 1000000-increasing insertions is: 25.388215%
```

	10^4	10^5	10^6
Percentage	25.4068	25.3824	25.3882

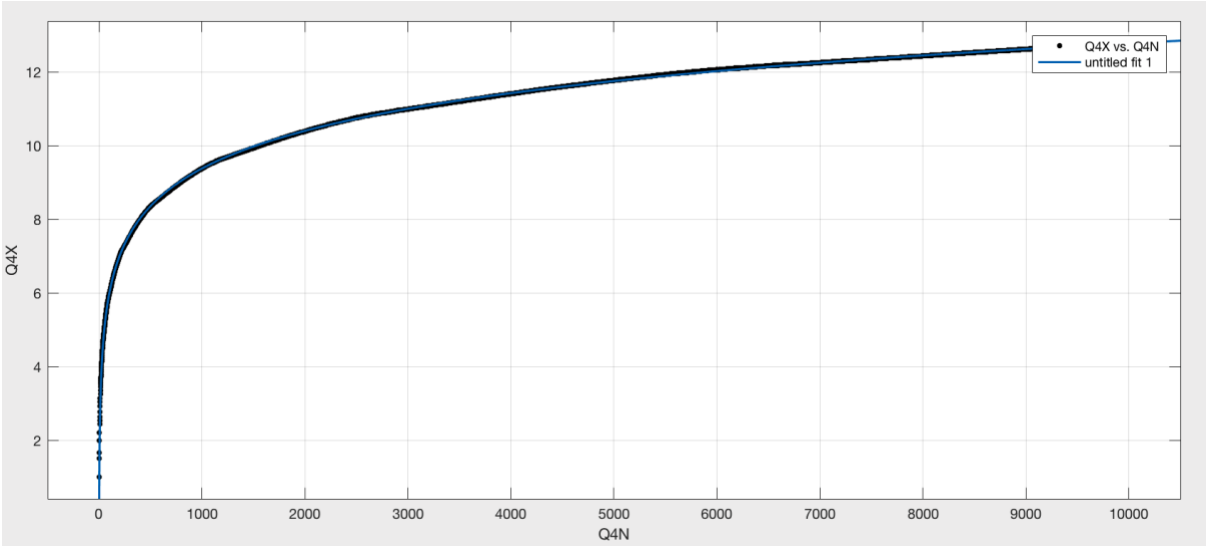
Q4.

For this question, the basic red black tree code is referenced from Algorithm 4th by princeton official source code from Github. I implement the method to count the internal path length

```
public double countInternalPathLength() {
    return countInternalPathLength(root);
}

private double countInternalPathLength(Node x) {
    return (double) countInternalPathLengthHelper(x)/size();
}

private int countInternalPathLengthHelper(Node x) {
    if (x == null)
        return 0;
    int pathLength = 0;
    pathLength = x.size + countInternalPathLengthHelper(x.left) + countInternalPathLengthHelper(x.right);

    return pathLength;
}
```

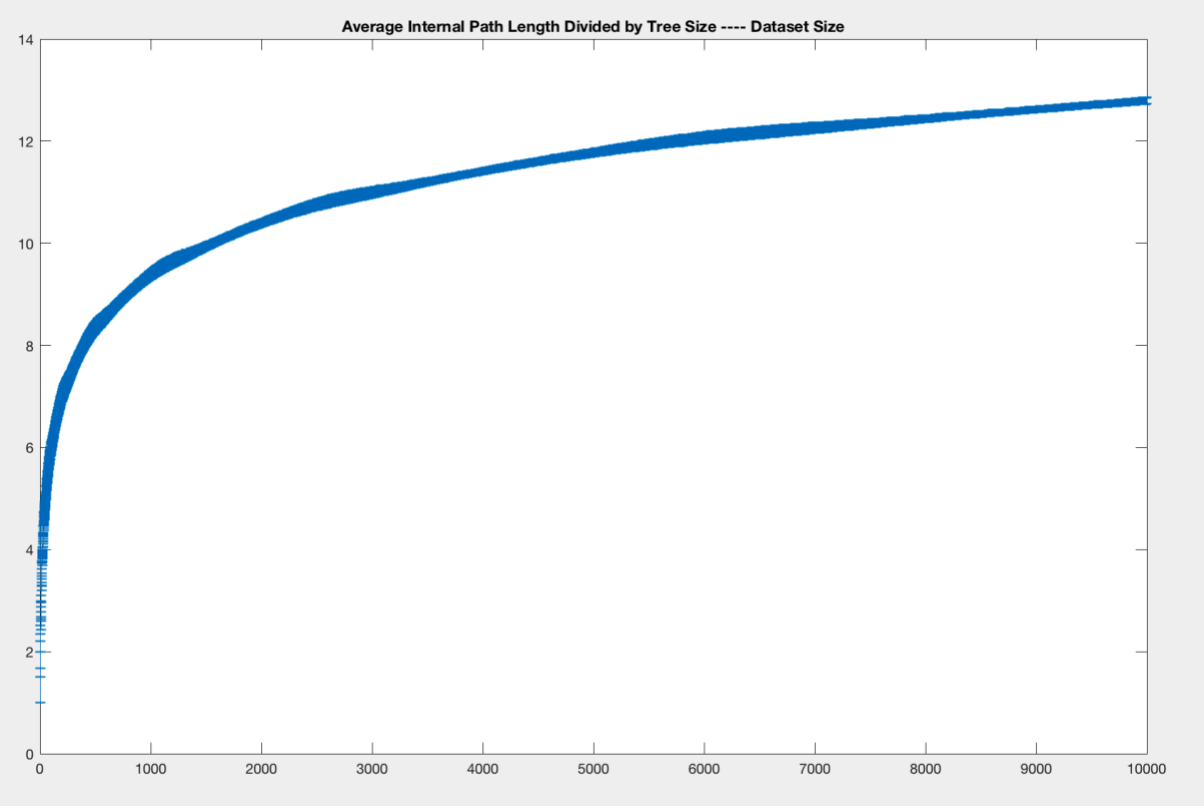
I save my result in a csv file, which you can find it in my submitted zip file. there are 3 columns in the csv file. The first column is the number of insertions, the second is the average internal path length divided by tree's size, the third one is the std deviation.

I do the curve fitting in the matlab



the equation is: y = 1.023 * $log_2(x)$- 0.8165

The figure with error bar is as follows:



Q5

For this question, I implement my BST data structure as MyBst class. you will get the result as follows:

```
the value of select(7) is: 8
the value of rank(7) is:  6
```