**DSA HW3 Report**

Zhongze Tang (zt67)

**Q1**

In Line 7, you can change the number of random input. Then you will see the output of in-order traversal of this tree.

**Q2**

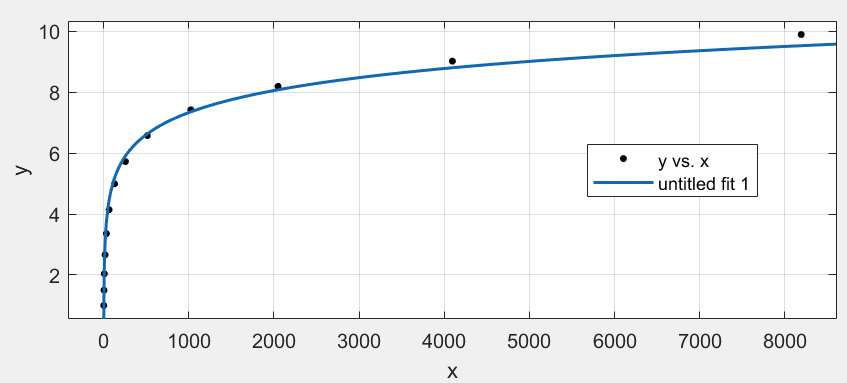
The experiment is based on the tree written in Q1.

If we consider the tree as a 2-3 tree (skip the red node when calculate the path length), the result is:

|  |  |  |
| --- | --- | --- |
| N | Ave path len(ordered) | Ave path len(random) |
| 1 | 1.0 | 1.0 |
| 2 | 1.0 | 1.0 |
| 4 | 1.5 | 1.5 |
| 8 | 2.5 | 2.03875 |
| 16 | 4.5 | 2.665 |
| 32 | 8.5 | 3.35625 |
| 64 | 16.5 | 4.14390625 |
| 128 | 32.5 | 4.995859375 |
| 256 | 64.5 | 5.723359375 |
| 512 | 128.5 | 6.57669921875 |
| 1024 | 256.5 | 7.422119140625 |
| 2048 | 512.5 | 8.19423828125 |
| 4096 | 1024.5 | 9.0209716796875 |
| 8192 | 2048.5 | 9.893995361328125 |

I use Curve Fitting Tool in MATLAB to fit the number and the average length (with random input), and the result is:

*f(x) = 0.7237\*log2(x) – 0.1194*



As for the ordered input,

*the average path length = 1 (N<4) or N / 4 + 0.5 (N>=4).*

At the same time, I also test the situation when we count the red nodes. I think this situation is the same as the BST’s. The result is shown in the table:

|  |  |  |
| --- | --- | --- |
| N | Ave path len(ordered) | Ave path len(random) |
| 1 | 1.0 | 1.0 |
| 2 | 1.5 | 1.5 |
| 4 | 2.5 | 2.225 |
| 8 | 4.5 | 3.165 |
| 16 | 8.5 | 4.188125 |
| 32 | 16.5 | 5.3846875 |
| 64 | 32.5 | 6.496875 |
| 128 | 64.5 | 8.089921875 |
| 256 | 128.5 | 9.2400390625 |
| 512 | 256.5 | 10.5942578125 |
| 1024 | 512.5 | 12.00736328125 |
| 2048 | 1024.5 | 13.3846533203125 |
| 4096 | 2048.5 | 14.75482177734375 |
| 8192 | 4096.5 | 16.236561279296875 |

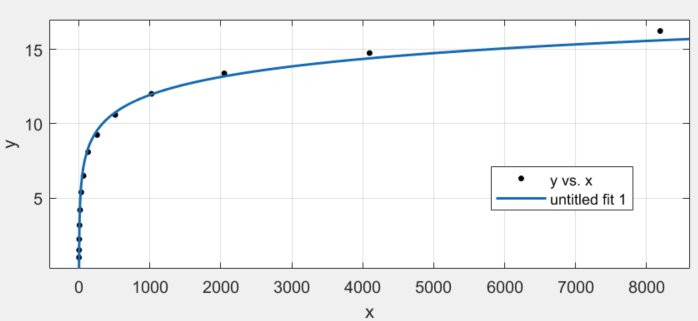
Adding the depths of all nodes, we get a quantity known as the internal path length of the tree. So, I use this formula to calculate the internal path length PN:

*PN = N + PL + PR*

where N is the size of Node P, and PL and PR are its two subtrees.

I use Curve Fitting Tool in MATLAB to fit the number and the average length (with random input), and the result is:

*f(x) = 1.212\*log2(x) – 0.1438*



As for the ordered input,

*the average path length = (N / 2 + 0.5).*

The curve fitting file is Q2/Q2withoudred.sfit (first one) and Q2/Q2curvefit.sfit (second one).

The program now is calculating the path length skipping the red node. If you want to check the result of the one with red node, please take place of *getLenSumWithoutRed()* in Line 17 and 19 to *getLenSum()*.

**Q3**

The results are:

Result of 10000 is: 0.25411100000000003

Result of 100000 is: 0.2539944

Result of 1000000 is: 0.25390362999999994

We can safely conclude that the average percentage of red nodes in a random-input red-black tree is 25.4%.

The code of Red-Black BST is from *Algorithms* book website.

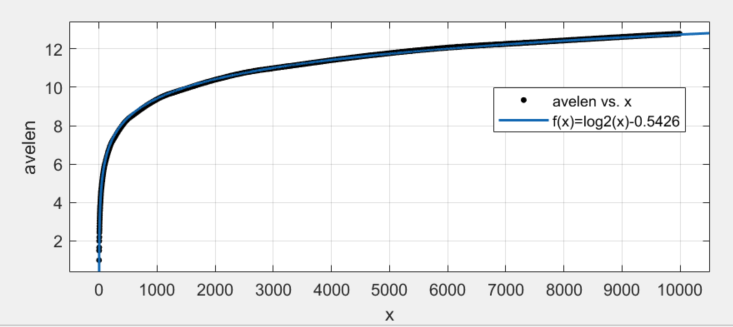
**Q4**

The calculation method of internal path length is the same in Q2:

*PN = N + PL + PR*

I use Curve Fitting Tool in MATLAB to fit the number and the average length, and the result is:

*f(x) = log2(x) – 0.5426*



At the same time, the standard deviation is quite small, and its average value is 0.077.

All the results are stored in Q4/result.csv, and the curve fitting file is Q4/curvefit.sfit. The code of Red-Black BST is from *Algorithms* book website.

Note that it took almost 3 hours on my computer (i7-4710MQ, 8G RAM) to work out all the results.

**Q5**

1. The value of *select(7)* is 8.
2. The value of *rank(7)* is 6.