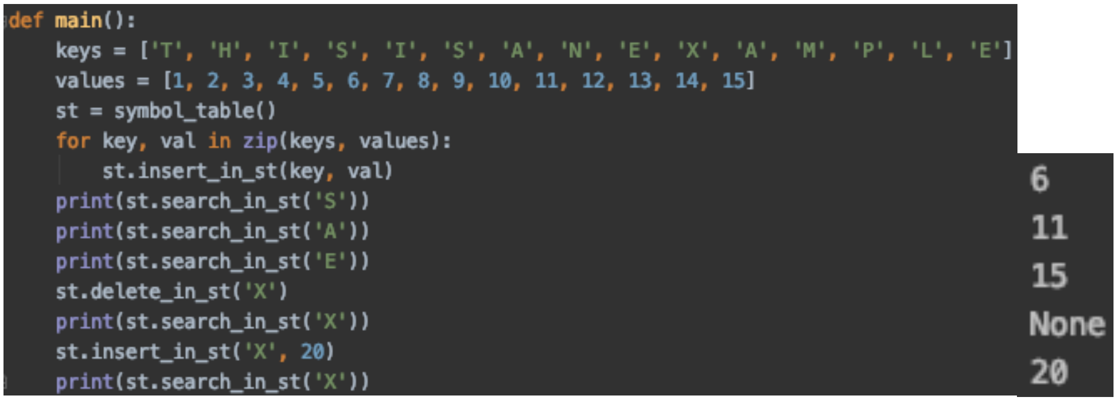
Homework 3 Report

Haocong Wang

mw814

Q1

I implemented a left-lean red black tree in this question. All the resources I took advantage of to finish the assignment are listed in the code. The keys and values I used in the main function can be changed to verify the left-lean red black tree.
Also, you can change the operations I used in my code to verify the tree, such as insert and delete.



```
def main():
    keys = ['T', 'H', 'I', 'S', 'I', 'S', 'A', 'N', 'E', 'X', 'A', 'M', 'P', 'L', 'E']
    values = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
    st = symbol_table()
    for key, val in zip(keys, values):
        st.insert_in_st(key, val)
    print(st.search_in_st('S'))
    print(st.search_in_st('A'))
    print(st.search_in_st('E'))
    st.delete_in_st('X')
    print(st.search_in_st('X'))
    st.insert_in_st('X', 20)
    print(st.search_in_st('X'))
```

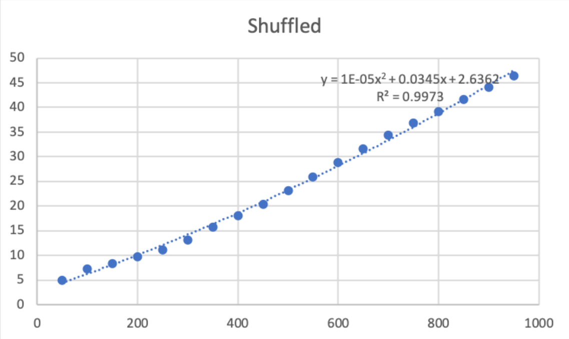
```
6
11
15
None
20
```

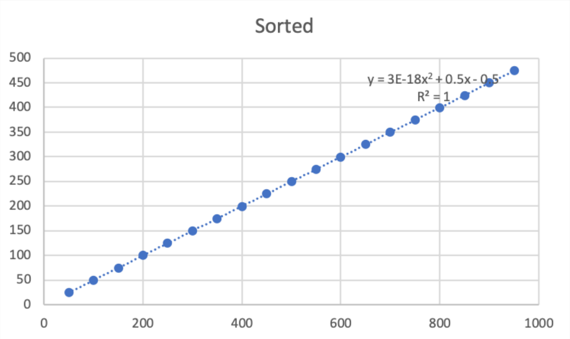
Q2

In this question, I implemented a binary search tree. All the resources I used to finish the task are listed in the code. The results are shown below. I used the curve fitting tool in the Excel to estimate the value of the number of path length when inserting N values, random or sorted, in the binary search tree.

Insertion	Shuffled	Sorted
50	4.98	24.5
100	7.23	49.5
150	8.29333333	74.5
200	9.77	99.5
250	11.132	124.5
300	13.1033333	149.5
350	15.76	174.5
400	18.065	199.5
450	20.3644444	224.5
500	23.092	249.5
550	25.88	274.5
600	28.8016667	299.5

650	31.56	324.5
700	34.3171429	349.5
750	36.7893333	374.5
800	39.175	399.5
850	41.6070588	424.5
900	44.08	449.5
950	46.3284211	474.5





For each value of N, I used 10 trials to calculate the average path length. As we can see, for both random and sorted insertion, the curves of the average path length are almost linear. The function of two path lengths are listed in the picture.

For shuffled insertion, the function is:

The average path length = 10^{-5}*N^2+0.0345*N+2.6362

For sorted insertion, the function is:

The average path length = 3*10^{-18}*N^2+0.5*N-0.5

Q3

The results are:

Result for 10000 is: 0.330952647

Result for 100000 is: 0.331594752

Result for 1000000 is: 0.329688456

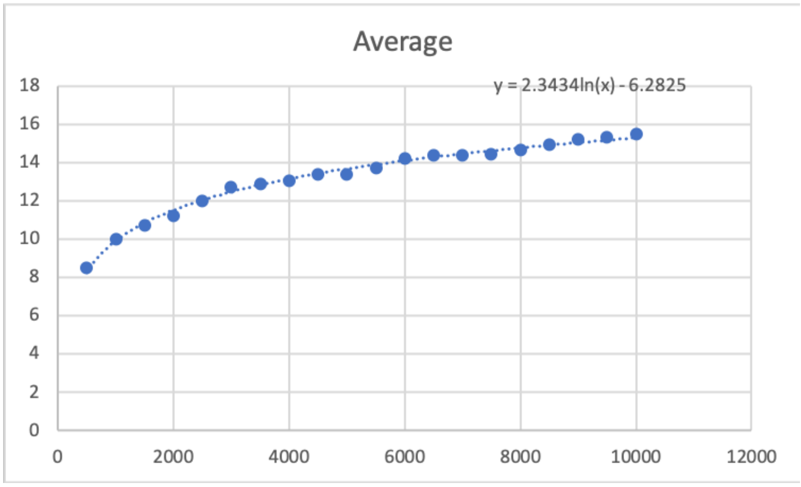
From what I have learned from the course, I understand the percentage of red node in a red black tree is about 25%. However, no matter how I changed my code, I can only get the result of 33%. I guess I need to work harder to find out the problem of my code.

Q4

In this question, I used the same code in Q1 for implementing a red black tree. I ran experiments on trees of different insertion ranging from 500 to 10000. The results are shown below.

Insertion	Average	std_dev
500	8.482164	0.70401313
1000	9.988392	0.80560911

1500	10.708172	0.86924886
2000	11.24503	0.95149804
2500	12.0230024	0.95260704
3000	12.7312773	0.93090966
3500	12.8622446	0.97776102
4000	13.060161	1.05445409
4500	13.3783356	1.0741206
5000	13.4160912	1.13569504
5500	13.7490916	1.08687538
6000	14.2424767	1.07352661
6500	14.4069215	1.09713845
7000	14.3963883	1.11685474
7500	14.420704	1.15256101
8000	14.6893903	1.19007029
8500	14.9605445	1.16297103
9000	15.2423151	1.19501527
9500	15.33398	1.12078241
10000	15.4839772	1.18132269



For each value of N, I used 1000 trials to calculate the average path length and the standard deviation.

The curve function of the path length of N = 2.3434ln(x)-6.2825

The standard deviation is small, with each lower than 1.20.

Q5

The value of select(7) is 8.

The value of rank(7) is 6.