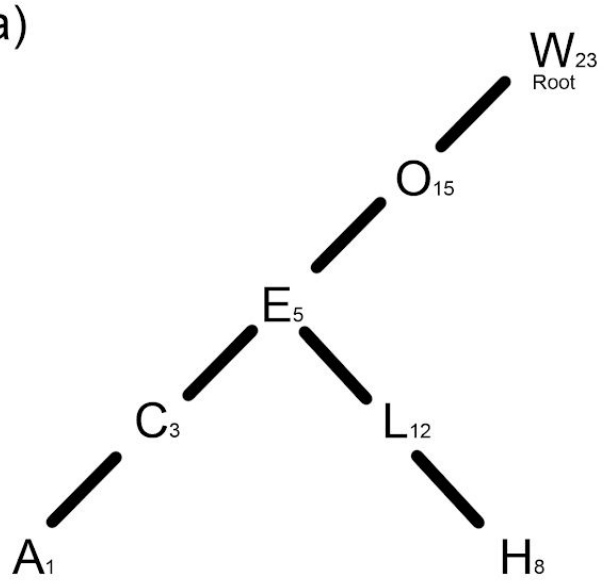
Input: W O E C A L H

Letter Number	Letter
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L
13	M
14	N
15	O
16	P
17	Q
18	R
19	S
20	T
21	U
22	V
23	W
24	X
25	Y
26	Z

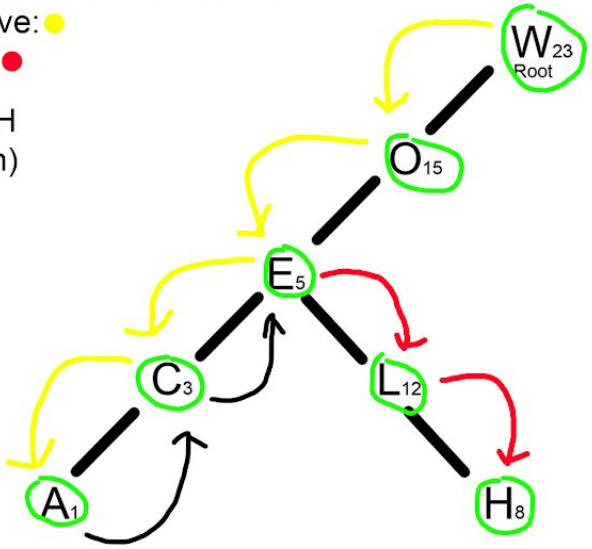
a)



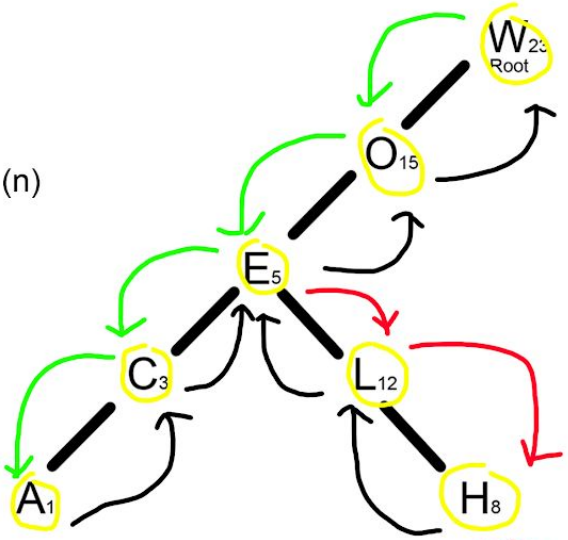
b) Output:

First move: ● Done all moves: ●
Second move: ●
Last move: ●

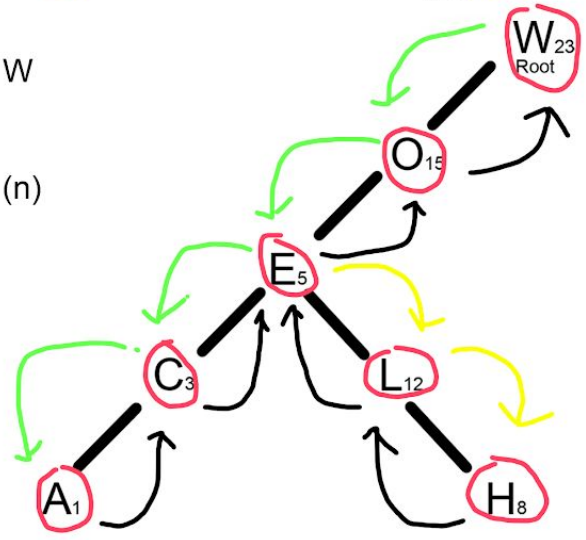
1. Preorder(n l r): W O E C A L H
First print the node we are on (n)
Second go to the left (l)
Lastly go to the right (r)



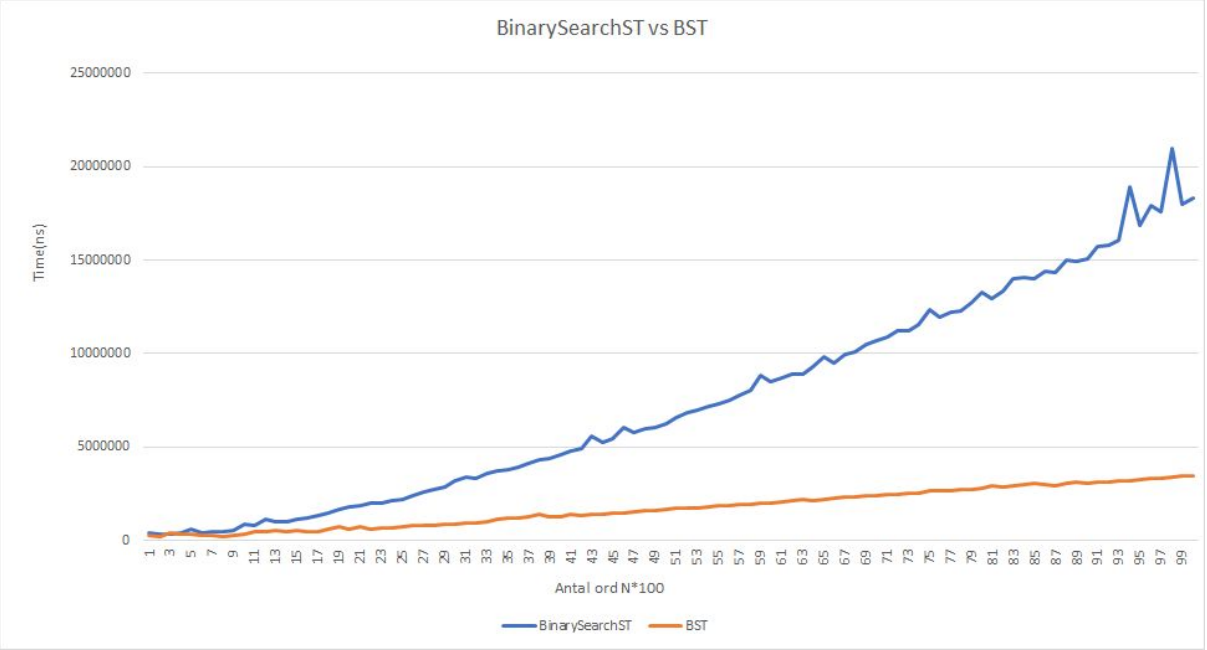
2. Inorder(l n r): A C E H L O W
First to the left (l)
Second print the node we are on (n)
Lastly go to the right (r)



3. Postorder(l r n): A C H L E O W
First go to the left (l)
Second go to the right (r)
Lastly print the node we are on (n)



2.



I measure the time with System.nanoTime() and increase the number of words by 100 at a time. And I also take average of 15 tries for each increase in words.

The reason why binarySearchST is slower is because it uses two arrays which in turn gives it a time complexity of N^2 while initiating the arrays.