

https://www.hackerrank.com/challenges/rotate-string/problem?h_r=internal-search

Pseudocode:

Rotate(String s):

if s.length < 1:
return "Invalid String"

return RotateRec(s,1)

RotateRec(String s , int c):

if s.length == c:
first_L = s[0]
s1 = s.substring(1:) + first_L
return s1

first_L = s[0]
s1 = s.substring(1:) + first_L
return s1 + " " + RotateRec(s1,c+1)

Java Implementation:

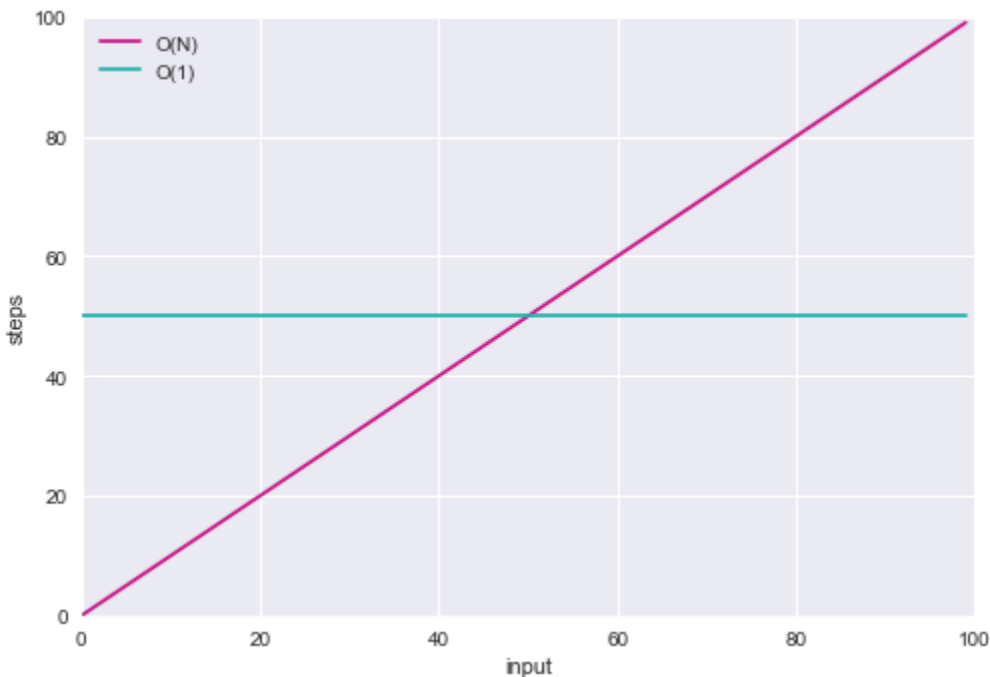
```
1
2 public class RotateString {
3
4     public static String rotate(String s) {
5         if(s.length() < 1) {
6             return "Input Valid String";
7         }
8         return rotateRec(s,1);
9     }
10
11     private static String rotateRec(String s, int c) {
12
13         if(c == s.length()) {
14
15             char first_L = s.charAt(0);
16
17             String s1 = s.substring(1) + first_L;
18
19             return s1;
20         }
21
22         char first_L = s.charAt(0);
23
24         String s1 = s.substring(1) + first_L;
25
26         return s1 + " " + rotateRec(s1,c+1);
27     }
28 }
29
```

```
1
2 public class Driver {
3
4     public static void main(String[] args) {
5
6         RotateString x = new RotateString();
7
8         System.out.println(x.rotate("h"));
9
10    }
11
12 }
13
```

While the above code uses a different Driver class to test the result, it still produces the desired output. Using recursion, the first character is isolated then added to the end of a substring not containing that same first character. The process is repeated until the string is fully rotated through, returning all the possible rotations separated by a space.

Time Complexity Analysis:

The time complexity of this program is **$O(n)$** . This is because we recursively iterate over a String which in and of itself is a collection of characters similar to an array. This means that the time taken is only dependent on the length of string s , being n . So ultimately we can say that the time complexity of the program is $O(n)$.



Github Repo:

<https://github.com/LukeSheldon19/Homework-2-Question-2-Algorithms>

