```java
package com.fiveamsolutions.interview;
import java.util.HashMap;

/**
 * Created by nnala on 10/5/16.
 * Implementation Class for AnagramFinder.java
 */
public class AnagramFinderImpl implements AnagramFinder {

    public boolean areAnagrams(String s1, String s2) {
        // Checking if both the strings are null
        if (s1 != null && s2 != null) {
            //Proceed further only if the length of two strings are equal
            if (!(s1.length() == s2.length())) {
                return false;
            } else {
                //Best Case Scenario where two strings are equal and this runs in O(1)
complexity
                if(s1.equalsIgnoreCase(s2)){
                    return true;
                } else{
                    //now that we know strings length are same but don't know if they
are anagrams are not.
                    //Below algorithm runs the O(n) complexity
                    //Remove Whitespaces and Convert to Lowercase
                     return
performNOrderAnagramAlgorithm(s1.toLowerCase().replaceAll("\\s",
""),s2.toLowerCase().replaceAll("\\s", ""));
                }

            }
        }
        return false;
    }

    /**
     * Input that takes two strings as input and returns whether they are Anagrams or
not*/

    protected boolean performNOrderAnagramAlgorithm(String s1, String s2) {
        char[]s1Arr = s1.toCharArray();
        char[]s2Arr = s2.toCharArray();
        HashMap<Character, Integer> charCountMap = new HashMap<>();
        for (int i =0; i< s1Arr.length;i++) {
            int count = 0;
            if(charCountMap.get(s1Arr[i]) == null) {
                count++;
                charCountMap.put(s1Arr[i], new Integer(count));
            } else {
                charCountMap.put(s1Arr[i], new
Integer(charCountMap.get(s1Arr[i]).intValue()+1));
            }
        }
        for(int j = 0 ; j<s2Arr.length;j++) {
            if(charCountMap.get(s2Arr[j]) != null) {
                charCountMap.put(s2Arr[j], new
Integer(charCountMap.get(s2Arr[j]).intValue()-1));
            } else{
                return false;
            }
        }
```

```
            for(int value : charCountMap.values()) {
                if(value !=0){
                    return false;
                }
            }

            return true;
        }

    }
```

n

The three for loops over size "n"

Here the Time complexity

$$T(n) = 3n + 5$$

By just taking only cycles into count and assuming const time can be discarded in Computing Big-oh

$$T(n) = O(n)$$

So my algorithm runs in $O(n)$ Complexity.