5/16/17

Ch 1: Arrays and Strings

I.Q. 1.1: "Is Unique" Implement an algorithm to determine if a string has all unique characters. What if you can not use additional data structures.

Example:

"apple" → NO ;

"Peach" → yes ;

Solution:

Pseudo code:

With additional data structure: Put values in hash table and see if value is in hash table before putting. If so, duplicate, and return false. If not, add to hash table and continue down string. If string completely searched then success, is unique.

O(n) time and O(n) space.

without additional data structure: Sort the string and then check if two values next to each other are the same.

$O(n \log n) \overset{\text{time}}{\downarrow}$ and O(1) additional space

Brute Force: For each value in the string, traverse the string and see if it appears twice.

$O(n^2)$ time and O(1) space

EX: "apple" → NO
"Peach" → yes

Code

w/ other data:

```
Public boolean isUnique(String s) {
 HashMap<Character, Character> h = new HashMap<>();
 for( char c : s.toCharArray() ) {
      if( h.get(c) == null)
           h.put(c, c);

      else // value already in hash Map
           return false;
 }
    // for loop finishes so Success
    return true;

}
```

without other data structures

```
Public boolean isUnique(String s) {
  char [] sArray = s.toCharArray;
   Arrays.sort(sArray);
   for(int i=0 ; i < sArray.length-1; i++) {
        if (sArray[i] == sArray[i+1])
             return false;
   }
   return true;
}
```