Hashing

In [ ]:

```

```

Question-1
[https://leetcode.com/problems/isomorphic-strings (https://leetcode.com/problems/isomorphic-strings)](https://leetcode.com/problems/isomorphic-strings)

~~~
**Pseudo code**
```C++
// TC: O(N)
// SC: O(N) / O(1) depending on how you wish to explain (since string only consists of ascii chars, which is a finite value)

class Solution {
public:
    bool isIsomorphic(string s, string t) {
        // map<char, char>
        // set<char> visited
        // for i 0... len(s)-1:
        //   c1 = s[i]
        //   c2 = t[i]
        //   if c1 not in map:
        //      if c2 in visited: // s = "badc" t="baba
        //            return false
        //       visited.add(c2)
        //      map[c1] = c2
        //   if map[c1] != c2:
        //       return false
        //
        // return true

            paper
            title
            01234
        c1  p a p e r p
        c2  t i t l e a
        map {p:t, a:i, e:l, r:e}
    }
};
```
~~~

```
public bool IsIsomorphic(string s, string t)
    {
        Dictionary<char, char> map = new Dictionary<char, char>();

        for(int i = 0; i < s.Length; i++)
        {
            if (map.ContainsKey(s[i]))
            {
                if(map[s[i]] != t[i]) return false;
            }
            else
            {
                if(map.ContainsValue(t[i])) return false;
                map.Add(s[i], t[i]);
            }
        }
        return true;
    }



    if(str1.length() != str2.length()){
            return false;
        }
    HashMap<Character, Character> charCount = new HashMap();
    char c;
     for (int i = 0; i < str1.length(); i++) {
            if (charCount.containsKey(str1.charAt(i))) {
                c = charCount.get(str1.charAt(i));
                if (c != str2.charAt(i))
                    return false;
            }else if (!charCount.containsValue(str2.charAt(i))) {
                charCount.put(str1.charAt(i),str2.charAt(i));
            }
            else {
                return false;
            }
        }
        return true;
    }
```

```
class Solution {
    public boolean isIsomorphic(String s, String t) {
        HashMap<Character, Character> map = new HashMap<Character,Cha
racter>();
        for(int i=0; i<t.length() ;i++){
            if(map.containsKey(t.charAt(i))){
                if(map.get(t.charAt(i)) == s.charAt(i))  continue;
                else return false;
            }else if(map.containsValue(s.charAt(i))){
                return false;
            }
            else {
```

```
class Solution {
    public boolean isIsomorphic(String s, String t) {
       Map<Character, Character> map = new HashMap<>();
        char c1;
        char c2;

        for (int i = 0; i<s.length(); i++) {
            c1 = s.charAt(i);
            c2 = t.charAt(i);

            if (!map.containsKey(c1)) {
                if (!map.containsValue(c2)) {
                    return false
                }
                map.put(c1, c2);
            }
                if (map.get(c1) != c2) {
                    return false;
                }
        }
        return true;
    }
}
```

Question-2
https://leetcode.com/problems/check-if-a-string-contains-all-binary-codes-of-size-k/ (https://leetcode.com/problems/check-if-a-string-contains-all-binary-codes-of-size-k/)

Pseudo Code

```
class Solution {
public:

    bool hasAllCodes(string s, int k) {
        // 0 ,1
        // 00 01 10 11
        // count of max options: 2^k

        //// Brute Force
        // for a binary string in  all possible string: // O(2^k)
        //   check if binary string is present in s or not. // O(n)
        // O(2^k * n)

        //// Optimal
        // count  = 2^k // O(1)
        // set<>
        // for i = k.. len(s):
        //    substr = s.substring(i-k, i)
        //    set.add(substr)
        //    if set.size() == count:
        //       return true
        //
        // return false


        //// Dry Run
        // n = len(s)
        // TC: O(n)
        // SC: O(2^k)

        // 00110110
        // 01234567
        // k = 2
        // i = 2 3  ... 8
        // i-k 0 1      6

        // 0000
        // 0123
        // k=1
        //
    }
};
```

C++

```
bool hasAllCodes(string s, int k) {

        set<string> subStrSet;
        int subStrCnt = 1 << k;
        for (int i=k; i<s.size(); i++) {
            subStrSet.insert(s.substr(i-k, i-1));
            if (subStrSet.size() == subStrCnt) {
                return true;
```

C#

```
{
    var total = Math.Pow(2, k);
      HashSet<string> visited = new HashSet<string>();
      for (int i = k; i <= s.Length; i++)
      {
        string subStr = s.Substring(i - k, k);
        if (!visited.Contains(subStr))
        {
          visited.Add(subStr);
          if (visited.Count == total) return true;
        }
      }

      return false;
}
```

```
public boolean hasAllCodes(String s, int k) {
        Set<String> seen = new HashSet<>();
        int count = 1 << k;
        for (int i = k; i <= s.length() && seen.size() < count; ++i)
{
            seen.add(s.substring(i - k, i));
        }
        return seen.size() == 1 << k;
    }
```

```
class Solution:
    def hasAllCodes(self, s: str, k: int) -> bool:
        count = 2**k
        hashset = set()
        for i in range(k,len(s)):
            substr = s[i-k:i]
            hashset.add(substr)
            if len(hashset) ==count:
                return True
        return False
```

Question-3
https://leetcode.com/problems/longest-substring-with-at-least-k-repeating-characters/ (https://leetcode.com/problems/longest-substring-with-at-least-k-repeating-characters/)

Python

```
class Solution(object):
    def longestSubstring(self, s, k):
        """
        :type s: str
        :type k: int
        :rtype: int
        """

        return self.util(s, k, 0, len(s))

        # freq = {}
        # for c in s: # iterate each character
        #     freq[c] = freq.get(c,0) + 1

        # for i in range(len(s)): # 0 .. len - 1
```

In [ ]:

```
Find peak element
```

Question-4 (DIY)
[https://leetcode.com/problems/longest-consecutive-sequence (https://leetcode.com/problems/longest-consecutive-sequence)](https://leetcode.com/problems/longest-consecutive-sequence)

- Sorting
- Set

In [ ]:

```

```