cheonhyangzhang.gitbooks.io

187 Repeated DNA Sequences – Medium · LeetCode solutions

Tianxiang Zhang

3 minutes

Problem:

All DNA is composed of a series of nucleotides abbreviated as A, C, G, and T, for example: “ACGAATTCCG”. When studying DNA, it is sometimes useful to identify repeated sequences within the DNA.

Write a function to find all the 10-letter-long sequences (substrings) that occur more than once in a DNA molecule.

For example,

```
Given s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT",

Return:
["AAAAACCCCC", "CCCCCAAAAA"].
```

Thoughts:

This is a very basic and easy problem if space is not a problem.

What makes it a hard one is because if you have a HashSet of String to keep all appearing DNA Sequences, it needs too much

storing memory.

The trick to handle this is to use Set of Integer instead of String. Because each character has only 4 possible values: A, C, G, T. So that it could be converted to 2 bit integer: 0, 1, 2, 3. In order to represent 10-letter-long sequence, we only need 10 * 2 = 20 bit. Since Integer is 32 bit long, so a Integer is already good enough to represent a 10 letter long DNA sequence.

Having this “ACGT” to “0123” converting system, the problem could be solved easily and efficiently.

Solutions:

```
public class Solution {
    public List<String>
findRepeatedDnaSequences(String s) {
        List<String> result = new
LinkedList<String>();
        if (s == null || s.length() < 10)
            return result;
        HashMap<Character, Integer> map = new
HashMap<Character, Integer>();
        map.put('A',0);
        map.put('C',1);
        map.put('T',2);
        map.put('G',3);
        Set<Integer> appear = new HashSet<Integer>();
        Set<Integer> added = new HashSet<Integer>();
        int tmp = 0;
        for (int i = 0; i < s.length(); i ++){
            tmp = (tmp << 2) + map.get(s.charAt(i));
```

```
            if (i >= 9){
                tmp = tmp & ((1 << 20) - 1);
                if (appear.contains(tmp) &&
!added.contains(tmp)){
                    added.add(tmp);
                    result.add(s.substring(i - 9, i +
1));
                }
                else{
                    appear.add(tmp);
                }
            }//if i >= 9
        }//for i
        return result;
    }
}
```

results matching ""

No results matching ""