LeetCode 76

https://leetcode.com/problems/minimum-window-substring/description/

Yifeng Zeng

Description

76. Minimum Window Substring

S = "ADOBECODEBANC"

T = "ABC"

Minimum window is "BANC".

Idea Report

The primitive idea is to enumerate all the possible substrings from S. Where the enumeration is n^2 time, and check any substring with T is another n time, that is n^3 time. So how do we speed up?
We can scan the string S until a substring has all the characters in T. In our example "ADOBEC" has all the characters in T "ABC". We record this "ADOBEC" length. Then we have two pointers points to the begining (i) and the end (j) of this substring. As long as the substring not contains all the characers in T, we move j to the right to increase one character. Until it is a substring, we check its length and update the minimum length. Then we move i to right to decrease one character until the substring(i, j) no longer contains all the characters, then we move j to right again. We use two int[] arr = new int[256], int[] base as a hash to save all the characters in T. And while moving i and j, we update int[] hash to compare to int[] base to see if substring of S contains all the charaters in T(compare with int[] base).

Code

```
public class Solution {
```

```
    public String minWindow(String s, String t) {
        if (s == null || t == null || s.length() == 0 || t.length() == 0) {
            return "";
        }

        int[] hash = new int[256];
        int[] base = new int[256];
        int i = 0;
        int j = 0;
        int len = Integer.MAX_VALUE;
        String res = "";
        for (; i < t.length(); i++) {
            base[t.charAt(i)]++;
        }

        for (i = 0; i < s.length(); i++) {
            while (j < s.length() && !isSubstring(hash, base)) {
                hash[s.charAt(j)]++;
                j++;
            }
            if (isSubstring(hash,base) && len > j - i) {
                len = j - i;
                res = s.substring(i, j);
            }
            hash[s.charAt(i)]--;
        }

        return res;
    }

    private boolean isSubstring(int[] hash, int[] base) {
        for (int i = 0; i < hash.length; i++) {
            if (hash[i] < base[i]) {
                return false;
            }
        }
        return true;
    }
}
```

Summary

- Comparing a hash of string is faster, so use a hash to store the different charater count.