# 题目：

Given a string, find the length of the longest substring without repeating characters.

Examples:

Given "abcabcbb", the answer is "abc", which the length is 3.

Given "bbbbb", the answer is "b", with the length of 1.

Given "pwwkew", the answer is "wke", with the legth of3. Note that the answer must be a substring, "pwke" is a subsequence and not a substring.

# 分析：

这一题的目的是给定一个字符串，找到这个字符串的无重复字符的最长子串。也就是说，一个子串，它的最长子串肯定是不超过这个字符串的所有字符种类数量的。同时，这个字符串的子串不能跳过某些字符，也就是说“无重复字符的最长子串”，这个“子串”在原字符串里是连续不间断的！

# 需掌握的知识点

Java对字符的操作。

# 思路

1. 第一个字符和第二个字符比较；
2. 相等，则第二个和第三个字符比较；不相等，则第三个字符和第一个和第二个字符比较；
3. C/C++可以用指针，java用什么呢？java用封装好的类方法，但是不知道用什么方法好。

# 程序

# 不会的点

1. String字符串的两种初始化的方法：

第一种：像C/C++那样直接初始化： String str1 = “abc”

第二种：调用String的构造方法： String str1 = new String(“abc”);

1. String类常用的方法：
2. 求字符串长度：str.length()求得的长度为当前字符串的真正长度，而不是长度减一
3. 求字符串某一位置字符str.charAt(4),返回索引为4的字符。
4. 提取子串：str.substring(beginindex),返回从开始beginindex到末尾的字符串。
5. 字符串比较：str.compareTo(“待比较的字符串”)。
6. 字符串连接：”aa”.concat(“bb”).concat(“cc”);
7. 字符串是否含有…：”xxxx”.contains(“x”)
8. 获得字符串中的某一个特定的字符。

# 答案

我的答案：

import java.util.\*

class solution{

public String findSubString(String in){

int length = in.length(in)-1;

String tmp = new String();

for(int i = 0;i<=length;i++){

if(in.substring(0,i).contains(in.charAt(i+1))){

continue;

}

else{

tmp = in.substring(0,i+1);

}

}

return tmp;

}

}

public class Find\_SubString\_3{

public static void main(String[] args){

String result;

String in = new String("abcabcbb");

solution ex = new solution();

result = ex.findSubString(in);

System.out.println(result);

}

}

标准答案一，暴力法：

public class Solution {

public int lengthOfLongestSubstring(String s) {

int n = s.length();

int ans = 0;

for (int i = 0; i < n; i++)

for (int j = i + 1; j <= n; j++)

if (allUnique(s, i, j)) ans = Math.max(ans, j - i);

return ans;

}

public boolean allUnique(String s, int start, int end) {

Set<Character> set = new HashSet<>();

for (int i = start; i < end; i++) {

Character ch = s.charAt(i);

if (set.contains(ch)) return false;

set.add(ch);

}

return true;

}

}

标准答案二，滑动窗口法：

public class Solution {

public int lengthOfLongestSubstring(String s) {

int n = s.length();

Set<Character> set = new HashSet<>();

int ans = 0, i = 0, j = 0;

while (i < n && j < n) {

// try to extend the range [i, j]

if (!set.contains(s.charAt(j))){

set.add(s.charAt(j++));

ans = Math.max(ans, j - i);

}

else {

set.remove(s.charAt(i++));

}

}

return ans;

}

}

标准答案三，优化滑动窗口法：

public class Solution {

public int lengthOfLongestSubstring(String s) {

int n = s.length(), ans = 0;

Map<Character, Integer> map = new HashMap<>(); // current index of character

// try to extend the range [i, j]

for (int j = 0, i = 0; j < n; j++) {

if (map.containsKey(s.charAt(j))) {

i = Math.max(map.get(s.charAt(j)), i);

}

ans = Math.max(ans, j - i + 1);

map.put(s.charAt(j), j + 1);

}

return ans;

}

}

/\*

越是上班，越是发现能力好的人在哪都会出头。没上个好高中好大学都不重要，最后能有个轻松满意的工作才是最重要的，过程不重要，结果才重要。

\*/