Palindrome

[Walmart] - Number of different Palindrome substrings

Give a string s and its length n, some of its substring could be palindrome. Return the number of distinct substrings that are palindrome.

Note: single character such as“a”is considered as palindrome.

**S1. Two pointers**

需要考虑单数的状态和双数的状态。time O(n^2)

**但是这个方法还是不够快！**

public static int numberOfPalindromeSubstring2(String s, int n) {

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

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

findPalindrome(s, set, i, i + 1);

findPalindrome(s, set, i, i);

}

return set.size();

}

private static void findPalindrome(String input, Set<String> set, int i, int j) {

int n = input.length();

while (i >= 0 && j < n && input.charAt(i) == input.charAt(j)) {

set.add(input.substring(i, j + 1));

i--;

j++;

}

}

387. First Unique Character in a String

Given a string, find the first non-repeating character in it and return it's index. If it doesn't exist, return -1.

s = "leetcode" return 0.

s = "loveleetcode", return 2.

这种判断是否unique，或者对char计数的，如果是26个字母，直接用int[26] count即可，

如果无法确定字母范围，就用256，比map快。

public int firstUniqChar(String s) {

if (s == null || s.length() == 0) return -1;

int[] count = new int[26];

for (char c:s.toCharArray()) {

count[c - 'a']++;

}

int len = s.length();

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

if (count[s.charAt(i) - 'a'] == 1) return i;

}

return -1;

}

68. Text Justification

Given an array of words and a length L, format the text such that each line has exactly L characters and is fully (left and right) justified.

You should pack your words in a greedy approach; that is, pack as many words as you can in each line. Pad extra spaces ' ' when necessary so that each line has exactly L characters.

Extra spaces between words should be distributed as evenly as possible. If the number of spaces on a line do not divide evenly between words, the empty slots on the left will be assigned more spaces than the slots on the right.

For the last line of text, it should be left justified and no extra space is inserted between words.

[ "This is an",

"example of text",

"justification. "]

题解：

1.for loop，确定单词数，sum[i, j] = m 刚好 < len

2. number of spaces between each word = (len - m) / (# of words - 1),

check if extra spaces are needed, if needed, assign from left to right

3.when reached last line, just put the left words with 1 space in between.

public List<String> fullJustify(String[] words, int maxWidth) {

List<String> res = new ArrayList<>();

*// w is previous index, [i, w - 1] are included.*

for (int i = 0, w;i < words.length; i = w) {

int len = -1; // length of all characters in words

for (w = i;w < words.length && len + words[w].length() + 1 <= maxWidth ;w++) {

len += words[w].length() + 1;

}

StringBuilder sb = new StringBuilder(words[i]);

int space = 1, extra = 0;

if (w != i + 1 && w != words.length) { *// not last line, not 1 char*

space += (maxWidth - len) / (w - i - 1);

extra = (maxWidth - len) % (w - i - 1);

}

for (int j = i + 1; j < w; j++) {

for (int s = 0; s < space;s++) sb.append(' '); *// add enough spaces*

if (extra-- > 0) sb.append(' ');

sb.append(words[j]);

}

*// for last line, should append enough spaces in the end:*

int leftLen = maxWidth - sb.length();

while (leftLen-- > 0) sb.append(' ');

res.add(sb.toString());

}

return res;

}

序列化压缩

271. Encode and Decode Strings

Design an algorithm to encode a list of strings to a string. The encoded string is then sent over the network and is decoded back to the original list of strings.

就是设计一种方法，把一个list的string压缩成一个string，重点是怎么划分string，用什么符号？这个符号本身可能在string中存在。

并且写出解压缩的方法。

**S1. 长度：original string**

压缩方法：每个string变成：string的长度+冒号+原来的string

解压缩方法：每次从i = 0开始，找冒号的位置，取得数字确定长度，然后取出string，更新i的值。

*// Encodes a list of strings to a single string.*

public String encode(List<String> strs) {

StringBuilder sb = new StringBuilder();

for (String str:strs) {

sb.append(str.length());

sb.append(':');

sb.append(str);

}

return sb.toString();

}

*// Decodes a single string to a list of strings.*

public List<String> decode(String s) {

List<String> res = new ArrayList<>();

int i = 0;

while (i < s.length()) {

int delimiter = s.indexOf(':', i);

int len = Integer.valueOf(s.substring(i, delimiter));

i = delimiter + len + 1;

res.add(s.substring(delimiter + 1, i));

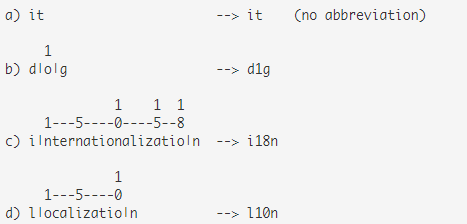
}

return res;

}

288. Unique Word Abbreviation

An abbreviation of a word follows the form <first letter><number><last letter>. Below are some examples of word abbreviations:



Assume you have a dictionary and given a word, find whether its abbreviation is unique in the dictionary. A word's abbreviation is unique if no other word from the dictionary has the same abbreviation.

**S1.Map**

通过map来实现，map中保存的是<abbreviation, original string>.

遍历处理dictionary中的每个string，如果发现key已经存在，说明这两个单词都不是unique，那就把key对应的value设成一个两者都无法匹配的值 – “” 设成空。

public class ValidWordAbbr {

Map<String, String> map; *// <abbreviation, original word>*

public ValidWordAbbr(String[] dictionary) {

map = new HashMap<>();

for (String s:dictionary) {

String abbr = getAbbr(s);

if(map.containsKey(abbr)){

if(!map.get(abbr).equals(s)){

map.put(abbr, "");

}

} else {

map.put(abbr, s);

}

}

}

private String getAbbr(String s) {

if (s.length() < 3) return s;

else return s.charAt(0) + String.valueOf(s.length() - 2) + s.charAt(s.length() - 1);

}

public boolean isUnique(String word) {

String abbr = getAbbr(word);

return !map.containsKey(abbr) || map.get(abbr).equals(word);

}

}

320. Generalized Abbreviation

Write a function to generate the generalized abbreviations of a word.

Given word = "word", return the following list (order does not matter):

["word", "1ord", "w1rd", "wo1d", "wor1", "2rd", "w2d", "wo2", "1o1d", "1or1", "w1r1", "1o2", "2r1", "3d", "w3", "4"]

**S1. Backtrack**

这道题虽然放在string里，但用的是典型的backtrack的方法，通过index来判断是否到leaf condition，

多出来的是，参数还需要传入一个current Number, 表示当前有几个字母会被用数字表示。

而且，在DFS中，每次都需要考虑两种情况：当前index对应的字母，是应该用数字还是用字母来表示。

所以用两个dfs表示。

public List<String> generateAbbreviations(String word) {

List<String> res = new ArrayList<>();

dfs(res, "", word, 0, 0);

return res;

}

private void dfs(List<String> res, String cur, String word,

int idx, int curNum) {

if (idx == word.length()) {

if (curNum != 0) cur += curNum;

res.add(cur);

} else {

dfs(res, cur, word, idx + 1, curNum + 1); *// "4"*

if (curNum != 0) cur += curNum;

dfs(res, cur + word.charAt(idx), word, idx + 1, 0); *// "word"*

}

}

**String中的DFS, Backtrack**

301. Remove Invalid Parentheses

Remove the minimum number of invalid parentheses in order to make the input string valid. Return all possible results.

Note: The input string may contain letters other than the parentheses ( and ).



**S1. DFS**

这道题其实需要和另外一种情况区分开。

也就是：返回所有valid parenthesis这种情况：



在这种题设的情况下，left, right记录的是左括号和右括号的实际数量，返回条件是：

idx=s.length() && left == right && open = 0. open在这里表示的是左括号数量与右括号数量之间的差值。

如果是0，代表左右括号数量相等，如果>0, 左括号数量>右括号。

但在这道题，只返回“刚刚好好把多余的括号删掉”的情况。

所以left, right保存的不是实际值，而是多出来的值。如果左括号多n个，left = n, 如果右括号多n个，right = n。

返回条件把left == right改成 left == 0 && right == 0

在这里，用open >= 0的条件去除 ")(" 类似的情况，make sure # of '(' >= # of ')' till now.

public List<String> removeInvalidParentheses(String s) {

List<String> res = new ArrayList<>();

int left = 0;

int right = 0;

int len = s.length();

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

if (s.charAt(i) == '(') left++;

else if (s.charAt(i) == ')') {

if (left > 0) left--;

else right++;

}

}

dfs(res, s, new StringBuilder(), 0, left, right, 0);

return res;

}

private void dfs(List<String> res, String s, StringBuilder sb, int idx, int left, int right, int open) {

if (idx == s.length() && left == 0 && right == 0 && open == 0 && !res.contains(sb.toString())) {

res.add(sb.toString());

return;

}

if (idx == s.length() || left < 0 || right < 0 || open < 0) return;

char c = s.charAt(idx);

int len = sb.length();

idx++; *// update idx*

if (c == '(') {

dfs(res, s, sb, idx, left - 1, right, open); *// remove it*

dfs(res, s, sb.append(c), idx, left, right, open + 1); *// keep it*

} else if (c == ')') {

dfs(res, s, sb, idx, left, right - 1, open); *// remove it*

dfs(res, s, sb.append(c), idx, left, right, open - 1); *// keep it*

} else { *// skip*

dfs(res, s, sb.append(c), idx, left, right, open);

}

sb.setLength(len);

}

282. Expression Add Operators

Given a string that contains only digits 0-9 and a target value, return all possibilities to add binary operators (not unary) +, -, or \* between the digits so they evaluate to the target value.

"123", 6 -> ["1+2+3", "1\*2\*3"]

"232", 8 -> ["2\*3+2", "2+3\*2"]

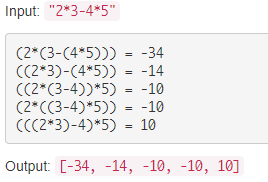
"105", 5 -> ["1\*0+5","10-5"]

"00", 0 -> ["0+0", "0-0", "0\*0"]

"3456237490", 9191 -> []

241. Different Ways to Add Parentheses

Given a string of numbers and operators, return all possible results from computing all the different possible ways to group numbers and operators. The valid operators are +, - and \*.



大数运算

字符串匹配

28. Implement strStr()

Implement strStr(). Returns the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.

Follow up: given two string, find how many length = 2 substring is a match. e.x. string a = apple, string b = app, return 2, because ap, pp are matches

S1 用java的函数

public int strStr(String haystack, String needle) {

return haystack.indexOf(needle);

}

如果纯靠自己写：

public int strStr(String haystack, String needle) {

if (haystack == null || needle == null) return -1;

int m = haystack.length(), n = needle.length();

if (m < n) return -1;

if (n == 0) return 0;

int threthold = m - n;

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

if (haystack.substring(i, i + n).equals(needle)) return i;

}

return -1;

}

151. Reverse Words in a String

Given an input string, reverse the string word by word.

For example, Given s = "the sky is blue", return "blue is sky the".

这道题要做的不仅是reverse words, 而且在string前面，后面，单词中间有多个空格的情况下，要把多余的空格删除 – 也就是前面后面都不留空格，words之间留一个空格。

**S1 如果可以用split**

直接用spaces来split，然后insert

public String reverseWords(String s) {

String[] words = s.split("\\s+");

StringBuilder sb = new StringBuilder();

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

sb.insert(0, words[i] + " ");

}

return sb.toString().trim();

}

**S2. 如果不可以用split:**

* 先reverse 整个string
* 再reverse每个单词

注意空格，如果string head, tail, 以及单词中间可能存在多个空格的话会麻烦一点，需要在最后clean spaces

public String reverseWords(String s) {

if (s == null || s.length() < 3) return s.trim();

int n = s.length();

char[] res = s.toCharArray();

reverse(res, 0, n-1); // index: both included.

int i = 0, j = 0; *// start, end of a word*

while (i < n) {

while (i < n && res[i] == ' ') i++;

j = i;

while (j < n && res[j] != ' ') j++;

reverse(res, i, j-1);

}

return removeExtraSpaces(res);

}

private String removeExtraSpaces(char[] res) {

int i = 0, j = 0, n = res.length; *// start/end of string*

while (j < n && res[j] == ' ') j++;

while (j < n) {

while (j < n && res[j] != ' ') res[i++] = res[j++];

while (j < n && res[j] == ' ') j++;

if (j < n) res[i++] = ' ';

}

return new String(res).substring(0, i);

}

public void reverse(char[] res, int i, int j) {

while (i < j) {

char tmp = res[i];

res[i++] = res[j];

res[j--] = tmp;

}

}

186. Reverse Words in a String II

Given an input string, reverse the string word by word. A word is defined as a sequence of non-space characters.

The input string does not contain leading or trailing spaces and the words are always separated by a single space.

For example,

Given s = "the sky is blue", return "blue is sky the".

Could you do it in-place without allocating extra space?

题解：

和上一道的思路一样：先reverse all, 再reverse每个单词。

注意在while循环中，每个循环的最后更新i，j的值。

public void reverseWords(char[] s) {

int n = s.length;

reverse(s, 0, n-1);

int i = 0, j = 0;

while (j < n) {

while (j < n && s[j] != ' ') j++;

reverse(s, i, j-1);

j++;

i = j;

}

}

private void reverse(char[] s, int i, int j) {

while (i < j) {

char tmp = s[i];

s[i++] = s[j];

s[j--] = tmp;

}

}

557. Reverse Words in a String III

Given a string, you need to reverse the order of characters in each word within a sentence while still preserving whitespace and initial word order.

**S1. 2 pointers**

只用reverse每个word，是上道题的简化版本。

public String reverseWords(String s) {

if (s == null || s.length() == 0) return s;

int i = 0, j = 0;

int n = s.length();

char[] chars = s.toCharArray();

while (j < n) {

while (j < n && chars[j] != ' ') j++;

reverse(chars, i, j-1);

i = ++j;

}

return new String(chars);

}

private void reverse(char[] s, int i, int j) {

while (i < j) {

char c = s[i];

s[i++] = s[j];

s[j--] = c;

}

}

344. Reverse String

Write a function that takes a string as input and returns the string reversed.

Example: Given s = "hello", return "olleh".

**S1. Array**

需要额外空间，O(n)

public String reverseString(String s) {

char[] words = new char[s.length()];

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

words[i] = s.charAt(s.length() - i - 1);

}

return String.valueOf(words);

}

**S2. 直接用stringBuilder自带的方法**

public String reverseString(String s) {

return new StringBuilder(s).reverse().toString();

}

541. Reverse String II

93. Restore IP Addresses

Given a string containing only digits, restore it by returning all possible valid IP address combinations.

For example: Given "25525511135",

return ["255.255.11.135", "255.255.111.35"]. (Order does not matter)

**S1. 遍历**

重点其实是三个点的位置，直接遍历一遍所有的可能性，并且check四个部分，每个部分的数字是否有效。

*// using 3 "." to divide s into 4 parts, key point is the position of ".",*

*// be i, j, k here*

public List<String> restoreIpAddresses(String s) {

List<String> list = new ArrayList<>();

int len = s.length();

if (len > 12 || len < 4) return list; // 剪枝

for (int i = 1;i < 4 && i < len - 2;i++) {

for (int j = i + 1;j <= i + 4 && j < len - 1;j++) {

for (int k = j + 1;k <= j + 4 && k < len;k++) {

String s1 = s.substring(0, i), s2 = s.substring(i, j);

String s3 = s.substring(j, k), s4 = s.substring(k, len);

if (isValidIP(s1) && isValidIP(s2) && isValidIP(s3) && isValidIP(s4))

list.add(s1 + "." + s2 + "." + s3 + "." + s4);

}

}

}

return list;

}

private boolean isValidIP(String s) {

if ((s.charAt(0) == '0' && s.length() != 1) || Integer.parseInt(s) > 255)

return false;

return true;

}

Apple - remove substring in set

input a string and a set of substring ("ab", "cd") -> output a new string with all substring removed, e.g. "ccabd" -> "ccd" -> "c"

暴力解的话其实很简单的

public static String removeSubstringInSet(String s, Set<String> set) {

if (s == null || s.length() == 0 || set.isEmpty()) return s;

int n = s.length();

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

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

if (set.contains(s.substring(i, j))) {

if (i == 0) {

s = s.substring(j);

} else if (j == n) {

s = s.substring(0, i);

} else {

s = s.substring(0, i) + s.substring(j);

}

return removeSubstringInSet(s, set);

}

}

}

return s;

}

**Anagrams**

49. Group Anagrams

Given an array of strings, group anagrams together. Note: All inputs will be in lower-case.

For example, given: ["eat", "tea", "tan", "ate", "nat", "bat"], Return:

[ ["ate", "eat","tea"],

["nat","tan"],

["bat"] ]

**S HashMap**

通过charArray sort所得相同string的方法寻找anagrams

public List<List<String>> groupAnagrams(String[] strs) {

if (strs == null || strs.length == 0) return new ArrayList<List<String>>();

Map<String, List<String>> map = new HashMap<>();

for (String s : strs) {

char[] ca = s.toCharArray();

Arrays.sort(ca);

String keyStr = String.valueOf(ca);

if (!map.containsKey(keyStr)) map.put(keyStr, new ArrayList<String>());

map.get(keyStr).add(s);

}

return new ArrayList<List<String>>(map.values());

}

242. Valid Anagram

Given two strings s and t, write a function to determine if t is an anagram of s.

For example, s = "anagram", t = "nagaram", return true. s = "rat", t = "car", return false.

You may assume the string contains only lowercase alphabets.

**S Array**

只要用int[] 存下每个字母出现的次数，再做比对就可以

follow up: 如果有非正常的字符，可以用map, <Character, Integer>

public boolean isAnagram(String s, String t) {

if (t == null && t == null) return true;

if (s.length() != t.length()) return false;

int[] chars = new int[26];

*// 1. store # of each char of s in map*

for (char c:s.toCharArray()) {

chars[c - 'a']++;

}

*// 2. check each char of t*

for (char c:t.toCharArray()) {

if (chars[c - 'a'] < 1) return false;

chars[c - 'a']--;

}

for (int count:chars) {

if (count != 0) return false;

}

return true;

}