Codenza

Home About Learning Interview Big O Cheat Sheet

String

How to reverse String in Java without “reverse”?

```
for(int i = str.length() - 1; i &gt;= 0; i--) {
      reverse = reverse + str.charAt(i);
}
for(int i = str.length() - 1; i &gt;= 0; i--){
      sb.append(str.charAt(i)); //String Builder
}
```

Step 1: Start

Step 2: Take two variable I and j.

Step 3: j is positioned on last character (Technically we can do this by length(string)-1)

Step 4: I is positioned on first Character (we can do this by string[0])

Step 5: String[i] is interchanged String[j] Step 6: Increment I by 1

Step 7: Increment J by 1

Step 8: If ‘I’ > ‘j’ then go to step 3

Step 9: Stop

The efficient solution is to use a character as an index in a count array. Traverse the given string and store index of the first occurrence of every character, also store count of occurrences. Then traverse the count array and find the smallest index with count as 1.

How do you print duplicate characters from a string?

The standard way to solve this problem is to get the character array from String, iterate through that and build a Map with character and their count. Then iterate through that Map and print characters which have appeared more than once. So you actually need two loops to do the job, the first loop to build the map and second loop to print characters and counts.

How do you check if two strings are anagrams of each other?

Classical way is getting character array of each String, and then comparing them, if both char array is equal then Strings are an anagram. But before comparing, make sure that both Strings are in the same case e.g. lowercase or uppercase and character arrays are sorted,

How do you print the first non-repeated character from a string?

One way to solve this problem is creating a table to store count of each character, and then picking the first entry which is not repeated. The key thing to remember is order, your code must return first non-repeated letter.

– The first solution uses LinkedHashMap to store character count since LinkedHashMap maintains insertion order and we are inserting a character in the order they appear in String, once we scanned String, we just need to iterate through LinkedHashMap and choose the entry with value 1. this solution requires one LinkedHashMap and two for loops.
(It first gets character array from given String and loop through it to build a hash table with the character as key and their count as value. In next step, It loop through LinkedHashMap to find an entry with value 1, that's your first non-repeated character, because LinkedHashMap maintains insertion order, and we iterate through character array from beginning to end. Bad part is it requires two iterations, first one is proportional to number of character in String, and second is proportional to number of duplicate characters in String. In worst case, where String contains non-repeated character at end, it will take 2*N time to solve this problem.)

– Second solution is a trade-off between time and space, to find first non repeated character in one pass. This time, we have used one Set and one List to keep repeating and non-repeating character separately. Once we finish scanning through String, which is O(n), we can get the magic character by accessing List which is O(1) operator. Since List is an ordered collection get(0) returns first element.

Codenza

Home About Learning Interview Big O Cheat Sheet

How can a given string be reversed using recursion?

```
1  public static String reverse(String str) {
2  StringBuilder strBuilder = new StringBuilder();
3  char[] strChars = str.toCharArray();
4
5  for (int i = strChars.length - 1; i >= 0; i--) {
6  strBuilder.append(strChars[i]);
7  }
8
9  return strBuilder.toString();
10 }
11
12 public static String reverseRecursively(String str) {
13
14 if (str.length() < 2) {
15 return str;
16 }
17
18 return reverseRecursively(str.substring(1)) + str.charAt(0);
19 }
```

How do you check if a string contains only digits?

Use Regex: Pattern pattern = Pattern.compile(".*[^0-9].*");

How do you count a number of vowels and consonants in a given string?

```
case 'u' :
count++;
break;
default: // no count increment
}
}
```

How do you count the occurrence of a given character in a string?

If a String contains multiple characters and you need to store count of each character, consider using HashMap for storing character as key and number of occurrence as value.

How do you find all permutations of a string?

```
private static void permutation(String perm, String word) {
if (word.isEmpty()) {
System.err.println(perm + word);
} else {
for (int i = 0; i &lt; word.length(); i++) {
permutation(perm + word.charAt(i), word.substring(0, i) + word.subs
}
```

How do you check if two strings are a rotation of each other?

1) check length of two strings, if length is not same then return false

2) concatenate given string to itself

3) check if rotated version of String exists in this concatenated string

4) if yes, then second String is rotated version of first string

How do you check if a given string is a palindrome?

1) Reverse the given String

2) Check if reverse of String is equal to itself, if yes then given String is palindrome.

We have a for loop to go through each character of String e.g. for input “123” this loop will run three times. In each iteration, we are making a recursive call to function itself i.e. permutation(String perm, String word) method, where the first parameter is used to store the result.

After 1st iteration perm (first parameter of permutation() method) will be “” + 1 as we are doing word.charAt(i) and i is zero. Next, we take out that character and pass the remaining characters to permutation method again e.g. “23” in the first iteration. Recursive call ends when it reaches to base case i.e. when remaining word becomes empty, at that point “perm” parameter contains a valid permutation to be printed. You can also store it into a List if you want to.

Codenza

Home About Learning Interview Big O Cheat Sheet