1. 题目
2. 题干

输入一个字符串，打印出该字符串中字符的所有排列。

你可以以任意顺序返回这个字符串数组，但里面不能有重复元素。

1. 示例

**示例:**

**输入：**s = "abc"

**输出：[**"abc","acb","bac","bca","cab","cba"**]**

1. 题解
2. 思路

怎么求一个字符串的所有排列呢？我们通过简单的“abc”来模拟，首先以a为第一个字符，第二个字符可以选择b或者c，第三个字符是自动确定的；接着回到第一个字符，以b为第一个字符，第二个字符选择a或者c；同理，以c为第一个字符。这样，就枚举出了字符串的所有排列。

有这个简单的模拟，可以提示我们使用dfs的思路来递归求解，由于最终的字符串排列是不能有重复的字符的，因此在每一层中每个字符之能使用一次，这里联想到使用set集合来排除重复字符。每一层设置一个set集合，用“abc”来举例，第一层循环3次，用于三次第一层字符变换，第二层交换2次，最后一层自动确定，每次交换使用一个swap()函数交换字符数组中的位置，在下一层递归回溯到这一层后，再用swap函数将两个位置交换回来。

1. 代码

Java：

class Solution {

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

    char[] c;

    public String[] permutation(String s) {

        // 不能重复

        // 字符串的所有排列？

        c = s.toCharArray();

        dfs(0);

        return list.toArray(new String[list.size()]);

    }

    public void dfs(int x){

        if(x == c.length - 1){

            list.add(String.valueOf(c));

            return;

        }

        // 每一级的set防止重复

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

        for(int i = x; i < c.length; ++i){

            if(set.contains(c[i])){

                continue;

            }

            set.add(c[i]);

            swap(i, x);

            dfs(x+1);

            // 换回来，恢复原样，回溯到上级中

            swap(x, i);

        }

    }

    public void swap(int i , int x){

        char temp = c[i];

        c[i] = c[x];

        c[x] = temp;

    }

}