面试题38：字符串的排列

题目描述

输入一个字符串,按字典序打印出该字符串中字符的所有排列。例如输入字符串abc,则打印出由字符a,b,c所能排列出来的所有字符串abc,acb,bac,bca,cab和cba。

输入描述

输入一个字符串,长度不超过9(可能有字符重复),字符只包括大小写字母。

解题思路

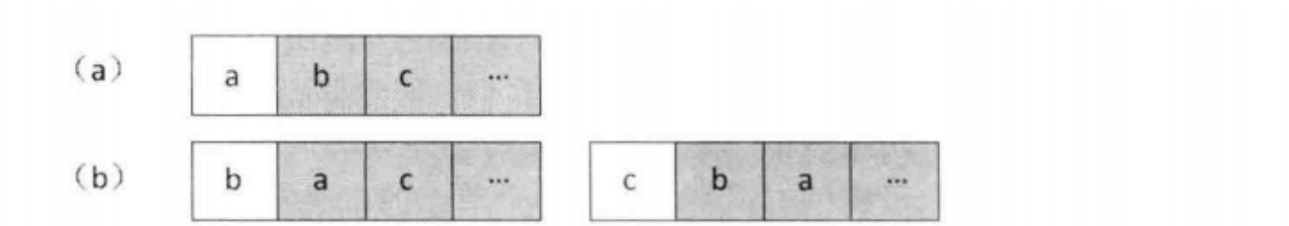
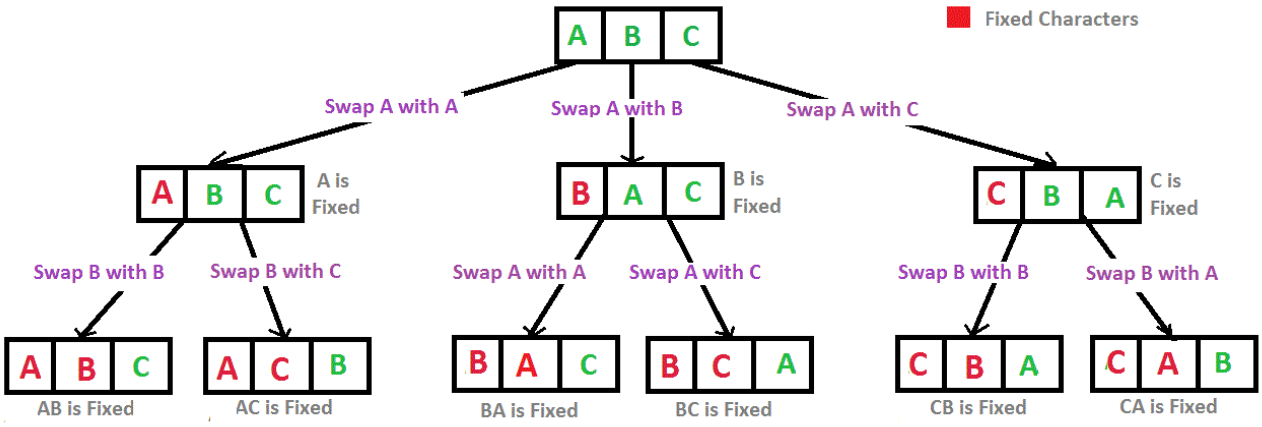


图 4.18　求字符串的排列的过程

注：（a）把字符串分为两部分：一部分是字符串的第一个字符；另一部分是第一个字符以后的所有字符（有阴影背景的区域）。接下来求阴影部分的字符串的排列。（b）拿第一个字符和它后面的字符逐个交换。



Recursion Tree for Permutations of String "ABC"

C++

```cpp
//思路：回溯法，排字典序。分而治之，k从1到最后一位，第k位开始字母落定，遍历到最后一位与第k位交换。
//要考虑到重复字母，用<set>容器记录已经出现过的字母。
class Solution {
public:
    vector<string> Permutation(string str) {
        Permutationmethod(0,str);
        return vec;
    }
    void Permutationmethod(int k,string str)
{
        if(k == str.size() - 1)
            vec.push_back(str);
        set<char>us;
        sort(str.begin() + k,str.end());
        for(int i = k;i < str.size();i++){
```

```
            if(us.find(str[i]) == us.end()){
                us.insert(str[i]);
                swap(str[i],str[k]);
                Permutationmethod(k + 1,str);
                swap(str[i],str[k]);
            }
        }
    }
private:
    vector<string>vec;
};
```

Java

```
import java.util.ArrayList;
public class Solution {
    public ArrayList<String> Permutation(String str) {
        ArrayList<String> list = new ArrayList<String>();

 PermutationS(str.toCharArray(),list,0,str.length()-1);
        list.sort(null);
        return list;
    }
```

```
9      public static void swap(char[] str, int
   i, int j) {
10         char t = str[i];
11         str[i] = str[j];
12         str[j] = t;
13     }
14     public static void PermutationS(char[]
   str,ArrayList<String> list, int begin, int
   end) {
15         if(begin==end) {
16
    if(!list.contains(String.valueOf(str))) {
17
    list.add(String.valueOf(str));
18             }
19             return;
20         }
21         for(int i=begin;i<=end;i++) {
22             swap(str, begin, i);
23             PermutationS(str, list, begin+1,
   end);
24             swap(str, begin, i);
25         }
26     }
27 }
```

Python 2.7.3

```python
# -*- coding:utf-8 -*-
class Solution:
    def Permutation(self, ss):
        # write code here
        res = []
        if len(ss) < 2:
            return ss.split()
        for i in range(len(ss)):
            for n in map(lambda x : x + ss[i],self.Permutation(ss[:i] + ss[i + 1:])):
                if n not in res:
                    res.append(n)
        return sorted(res)
```