递归与回溯

1. 递归程序的两个条件：

   a. 递归的出口，可能有多个跳出条件；

   b. 问题的单步处理+原问题向更小规模问题的转化处理（问题的相似性）；

2. 编写递归程序的要点：

   a. 递归在不同层递归调用时，参数的值是变化的；

   b. 递归可以通过添加参数的数量在不同递归层之间传递更多的信息；

2. 典型例题

<1>简单基本问题：

①打印从0-N。

```java
public class P01_PrintN {
    public static void main(String [] args){
        Print1( n: 10);
        System.out.println();
        Print2( n: 10, k: 0);
    }
    public static void Print1(int n){
        if(n > 0) Print1( n: n-1);
        System.out.print(n+"\t");
    }
    public static void Print2(int n,int k ){
        if(k > n) return ;
        System.out.print(k+"\t");
        Print2(n, k: k+1);
    }
}
```

②数组求和。

```java
public class P02_ArraySum {
    public static void main(String [] args){
        int [] nums = {1,2,3,4,5,6,7,8,9,10};
        int res1 = arrSum1(nums, index: 0);
        int res2 = arrSum2(nums, index: 0, res: 0);
        int res3 = arrSumBin(nums, start: 0, end: nums.length-1);
        System.out.println(res1+"\t"+res2+"\t"+res3);
    }
    public static int arrSum1(int [] arr,int index){
        if (index == arr.length){
            return 0;
        }
        return arr[index]+arrSum1(arr, index: index+1);
    }
    public static int arrSum2(int [] arr,int index,int res){
        if (index == arr.length){
            return res;
        }
        return arrSum2(arr, index: index+1, res: res+arr[index]);
    }
    public static int arrSumBin(int [] arr,int start,int end){
        if (start == end){
            return arr[start];
        }
        int mid = start + (end - start)/2;
        return arrSumBin(arr,start,mid)+arrSumBin(arr, start: mid+1,end);
    }
}
```

③判断两个字符串是否相等；

```
public class P03_isSameString {
    public static void main(String [] args){
        String s1 = "abc";
        String s2 = "bcd";
        System.out.println(isSameString(s1,s2));

    }
    public static boolean isSameString(String s1,String s2){
        if(s1.length() != s2.length())      return false;
        //执行到此处表明s1.length() = s2.length(),此时若
        //s1.length() = 0，则s1.length() = s2.length() =0，返回true;
        if(s1.length() == 0) return true;
        if(s1.charAt(0) == s2.charAt(0))
            return isSameString(s1.substring(1),s2.substring(1));
        else
            return false;

    }
}
```

<2>排列组合问题（递归回溯）

①从 N 个球中，取出 M 个球（不放回），有多少种不同的取法。

```
public class P01_GetBallMethod {
    public static void main(String [ ]args){
        System.out.println(GetBall( n: 10, m: 3));
    }
    public static int GetBall(int n ,int m){
        if( n < m ) return 0;
        if( n==m || m== 0) return 1;
        //想象一个特殊球c,取法划分位两种:
        // ①取的M个球中包含c:再从剩下的n-1个球中取m-1个球;
        // ②取的M个球中不包含c: 从剩下的n-1个球中取m个球。
        return GetBall( n: n-1, m: m-1)+GetBall( n: n-1,m);
    }
}
```

②全排列问题：

a. N 个元素中没有重复元素，求这 N 个元素的全排列；（交换法）

```
public static void permutation(char [] arr,ArrayList<String> res,int index){
        if(index == arr.length){
            res.add(new String(arr));
            return ;
        }
        for(int i = index ;i < arr.length;i++){
            swap(arr,index,i);          //递归试探
            permutation(arr,res, index: index+1);
            swap(arr,index,i);          //回溯
        }
}
```

b. 求 N 个元素的全排列，有重复元素；

①交换法

```java
public static void permutation(char [] arr,ArrayList<String> res,int index){
    if(index == arr.length){
        res.add(new String(arr));
        return ;
    }
    Set<Character> set = new HashSet<>();
    for(int i = index ;i < arr.length;i++){
        //如果 Set 集合中不包含要添加的对象，则添加对象并返回 true，否则返回 false.
        if(set.add(arr[i])){
            swap(arr,index,i);          //递归试探
            permutation(arr,res, index: index+1);
            swap(arr,index,i);          //回溯
        }

    }
}
```

②统计次数法

```java
public static ArrayList<String> permutation2(char [] arr){
    //统计字符出现次数
    Map<Character,Integer> map = new HashMap<>();
    for(int i = 0 ;i < arr.length;i++){
        map.put(arr[i],map.getOrDefault(arr[i], defaultValue: 0)+1);
    }
    char [] ch = new char[map.size()];
    int [] count = new int[map.size()];
    int i = 0;
    for(char key:map.keySet()){
        ch[i] = key;
        count[i++] = map.get(key);
    }
    ArrayList<String> res = new ArrayList<>();
    char [] temp = new char[arr.length];
    //递归回溯
    permutation2R(ch,count,temp, index: 0,res);
    return res;
}
```

```java
public static void permutation2R(char [] ch,int [] count,char [] temp,int index, ArrayList<String> res){
    if(index == temp.length){
        res.add(new String(temp));
        return ;

    }
    for(int i = 0; i < ch.length;i++){
        if(count[i]!=0){
            temp[index]=ch[i];
            count[i]--;

            permutation2R(ch,count,temp, index: index+1,res);

            count[i]++;
        }
    }
}
```

<3>其他问题

a. 整数划分

```java
//对一个整数进行加法划分
public class AdditiveDivision {
    public static void main(String [] args){
        int num = 6;
        int [] arr = new int [num];
        AddDivision(num,arr, pos: 0);
    }
    public static void AddDivision(int num,int [] arr,int pos){

        if(num <= 0) {
            for(int i = 0; i < pos;i++)
                System.out.print(arr[i]+" ");
            System.out.println();
        }
        for(int i =num; i > 0;i--){
            if( pos > 0 && i > arr[pos-1]) continue;
            arr[pos] = i;
            AddDivision( num: num - i,arr, pos: pos+1); //试探
        }

    }
}
```

b. 最长公共子序列

```java
//动态规划题，用递归只是展示解题思路
public class P02_LongestCommonSequence {
    public static void main(String [] args){
        String s1 = "abec";
        String s2 = "abcd";
        int res = LCS(s1,s2);
        System.out.println(res);
    }
    public static int LCS(String s1,String s2){
        if(s1.length() == 0 || s2.length() == 0) return 0;
        if(s1.charAt(0) == s2.charAt(0))
            return 1+ LCS(s1.substring(1),s2.substring(1));
        else
            return Math.max(LCS(s1.substring(1),s2),LCS(s1,s2.substring(1)));
    }
}
```

c. 最长公共子串

```java
public class P03_LongestCommonSubString {
    private  static  int res = 0;
    public static void main(String [] args){
        String s1 = "aefc";
        String s2 = "abedc";
        LCString(s1,s2, len: 0);
        System.out.println(res);
    }
```

```
    public static void LCString(String s1,String s2,int len){
        if(s1.length() == 0 || s2.length() == 0) return ;
        if(s1.charAt(0) == s2.charAt(0)){
            len+=1;
            LCString(s1.substring(1),s2.substring(1),len);

        }
        else{
            //当前位不相等，则比较res与len的大小，并赋值后将len置为0；
            res = res < len ? len:res;
            len = 0;
            LCString(s1.substring(1),s2,len);
            LCString(s1,s2.substring(1),len);
        }
    }
}
```