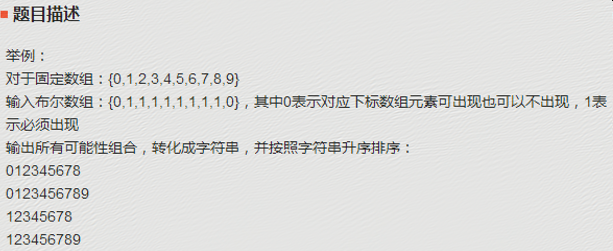
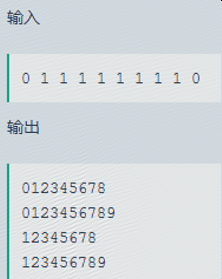
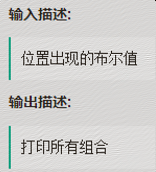
输出数组的所有可能组合-回溯法的应用-好未来笔试题3

# 输出数值的所有可能组合

## 题目描述





## 思路分析

\* 思路分析：利用回溯法；当bit为1时，只需要继续递归；

\* 若bit为0时，添加该元素递归，然后删除该元素，再进行递归。

## Java代码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*20180828好未来笔试题目\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

\* 回溯法的应用

public ArrayList<String> **findAllCombinationsOfNumbers**(int[] nums,int[] bits){

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

if(nums == null || nums.length == 0||bits == null||bits.length == 0|| nums.length!= bits.length) return list;

StringBuilder sb = new StringBuilder();//存放临时结果

TreeSet<String> result = new TreeSet<String>();//保证字典顺序

helper(nums,bits,sb,result,new int[]{0});

list.addAll(result);//转换为ArrayList

return list;

}

/\*\*

\* 递归方法

\*/

public void helper(int[] ns,int[] bits,StringBuilder sb,TreeSet<String> result,int[] index){

if(index[0] == ns.length){//递归终止条件

result.add(sb.toString());

return;

}

if(bits[index[0]] == 1){

sb.append(ns[index[0]++]);

helper(ns,bits,sb,result,index);

index[0]--;

sb.delete(sb.length()-1,sb.length());//删除最后一个

}else{//bit为0，可以分为添加或不添加

//先添加

**sb.append(ns[index[0]++]);**

**helper(ns,bits,sb,result,index);**

// index[0]--;

**sb.delete(sb.length()-1,sb.length());//删除最后一个**

// index[0]++;

**helper(ns,bits,sb,result,index);**

**index[0]--;**

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

笔试时，提交的代码

import java.util.Scanner;

import java.util.TreeSet;

public class Main {

public static void main(String[] args) {

int[] ns = {0,1,2,3,4,5,6,7,8,9};

Scanner sc = new Scanner(System.in);

String line = sc.nextLine();

String[] nums = line.split(" ");

int[] bits = new int[nums.length];

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

bits[i] = Integer.parseInt(nums[i]);

TreeSet<String> result = new TreeSet<String>();

StringBuilder sb = new StringBuilder();

helper(ns,bits,sb,result,new int[]{0});

for(String str:result)

System.out.println(str);

}

public static void helper(int[] ns,int[] bits,StringBuilder sb,TreeSet<String> result,int[] index){

if(index[0] == ns.length){

result.add(sb.toString());

return;

}

if(bits[index[0]] == 1){

sb.append(ns[index[0]++]);

helper(ns,bits,sb,result,index);

index[0]--;

sb.delete(sb.length()-1,sb.length());//删除最后一个

}else{

//先添加

sb.append(ns[index[0]++]);

helper(ns,bits,sb,result,index);

index[0]--;

sb.delete(sb.length()-1,sb.length());//删除最后一个

index[0]++;

helper(ns,bits,sb,result,index);

index[0]--;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/