全排列

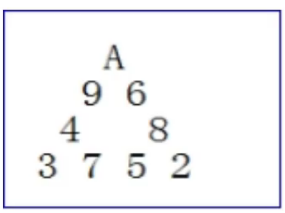
- 求全排列：全排列值得是：n个元素中取n个元素（全部元素）的所有排列组合情况。
- 求组合：n个元素，取m个元素(m<=n)的所有组合情况
- 求子集：n个元素的所有子集（所有的组合情况）

全排列常用解决办法：邻里交换法和回溯法

回溯法

> 题目：输入[1,2,3]
>
> 输出1,2,3所有的不重复的排列组合：
>
> 1 ,2,3　1,3,2
>
> 2,1,3　2,3,1
>
> 3,1,2　3,2,1

> 回溯：一般解决搜索问题，全排列也是一种搜索问题。
>
> - 回溯：就是类似枚举的搜索尝试过程。在搜索过程中寻找问题 的解。当发现不满足求解的条件时回，尝试别的路径。
>
> 全排列可以使用试探的办法列举所有的可能性。一个长度位n 的序列，所有的排列组合：n!
>
> > 1.从集合中选取一个元素（n种情况）,并标记该元素已经被使用。
> >
> > 2.在第一步的基础上递归到下一层，从剩余的n-1 个元素中，按照第一步的方法再找到一个元素，1)
> >
> > 3.依次类推，所有的元素都被标记，将元素存起来，取对比求解的情况。

> **2017年Java组c组第三题:**
>
> A,2,3,4,5,6,7,8,9共9张纸牌排成一个正三角形（A按1计算）。要求每个边的和相等。 下图就是一种排法。
>
> 
>

> 图片描述这样的排法可能会有很多。
>
> 如果考虑旋转、镜像后相同的算同一种，一共有多少种不同的排法呢？请你计算并提交该数字。

***回溯法:结果为： 144

```JAVA
public class Main {
    static int count = 0;//种类
    static int[] num = new int[10];
    static boolean[] bool = new boolean[10];
    public static void main(String[] args) {
       dfs(1);
        System.out.println(count/6);//除去旋转和镜像的清况(镜像2种，旋转3种)
    }
    public static void dfs(int step){//刚开始为1
        if(step == 10){//如果9位数都赋值完成，那么要开始判断是否满足各边和相同
            int a = num[1] + num[2] + num[4] + num[6];
            int b = num[6] + num[7] + num[8] + num[9];
            int c = num[1] + num[3] + num[5] + num[9];
            if ((a == b) && (b == c)) {//满足就次数加一
                count++;
            }
            return;
        }

        for(int i = 1; i < 10; i++){//
            if(!bool[i]){//如果当前位置没有表示
                bool[i] = true;//先把状态设为true
                num[step] = i;//把这位数字变为i
                dfs(step+1);//进行下一位的赋值
                bool[i] = false;//重新设置位false，为了可以把第1位的9种情况都可以存在，依次类推，第二位的8种...(体现回溯)
            }
        }
    }
}
```

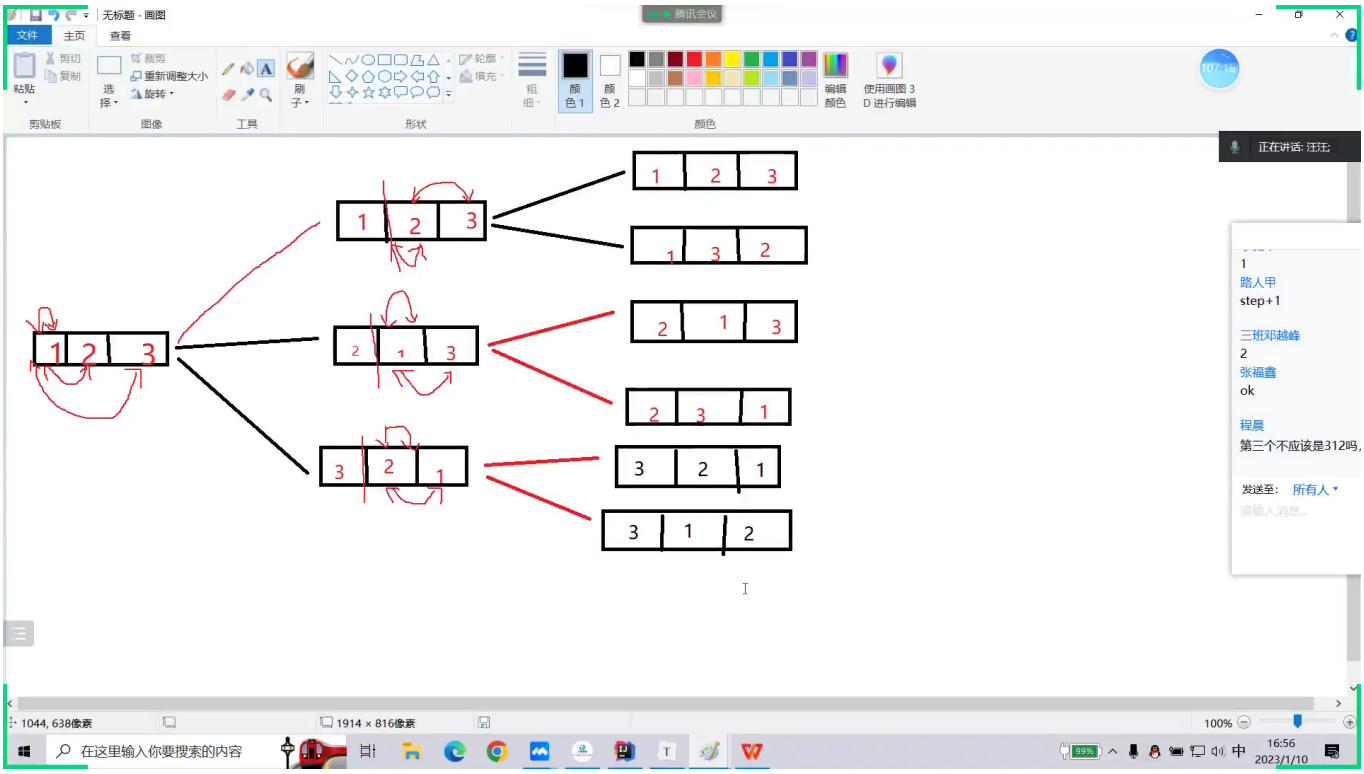
邻里交换法

> 回溯时试探性填充数据，给每个位置都试探性赋值。
>
> 邻里交换，也是通过递归实现，但是是一种基于交换的思路。

***邻里交换法

邻里交换法

> 回溯时试探性填充数据，给每个位置都试探性赋值。
>
> 邻里交换，也是通过递归实现，但是是一种基于交换的思路。
>
> 步骤：
>
> - 将数组分成2个部分：暂时确定部分和未确定部分。刚开始，都是未确定部分。
> - 在未确定部分中，让每一个数据都有机会和未确定部分中的第一位交换。然后第一位就变成暂时确定部分。
> - 以此类推：每个数据都和未确定部分中的第二位交换（第一位数据除外）.....直到确定所有数据
> - 将确定好的数据和条件对比，对比结束后，还原数据。



邻里交换法

```JAVA
public class Main {
    static int count = 0;
    static int[] a = {1,2,3,4,5,6,7,8,9};
    public static void main(String[] args) {
        f(a,0);
        System.out.println(count/6);
    }
    public static void f(int[] a, int step){
```

```
        if(step == a.length - 1){
            if(a[0] + a[1] + a[2] + a[3] == a[3] + a[4] + a[5] + a[6] &&
                    a[0] + a[1] + a[2] + a[3] == a[6] + a[7] + a[8] + a[0]){
                count++;
            }
            return;
        }
        for(int i = step; i < a.length; i++){
            {
                int tmp = a[i];//交换数据
                a[i] = a[step];
                a[step] = tmp;
            }
            f(a,step+1);//交换下一位
            {
                int tmp = a[i];//还原数据
                a[i] = a[step];
                a[step] = tmp;
            }
        }
    }
}
```

暴力破解法：9层循环

JAVA

```
public class Main {
    public static void main(String[] args) {
        int sum = 0;
        for(int a = 1 ; a < 10; a++){
            for(int b = 1 ; b < 10; b++){
                for(int c = 1 ; c < 10; c++){
                    for(int d = 1 ; d < 10; d++){
                        for(int e = 1 ; e < 10; e++){
                            for(int f = 1 ; f < 10; f++){
                                for(int g = 1 ; g < 10; g++){
                                    for(int h = 1 ; h < 10; h++){
                                        for(int i = 1 ; i < 10; i++){
                                            if(a==b || a==c || a==d || a==e || a==f || a==g ||
a==h || a==i
                                                    || b==c || b==d || b==e || b==f || b==g ||
b==h || b==i
                                                            || c==d || c==e || c==f || c==g ||
c==h || c==i
                                                                    || d==e || d==f || d==g ||
d==h || d==i
                                                                            || e==f || e==g ||
e==h || e==i
                                                                                    || f==g ||
f==h || f==i
                                                                                            ||
g==h || g==i
```

```
|| h==i
                                                ){
                                                    continue;//如果出现重复的两个数，就下一轮
                                                }
                                                if(a+b+c+d == d+e+f+g && d+e+f+g == g+h+i+a){//判断
满足条件吗
                                                    sum++;
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        System.out.println(sum/6);//除去镜像和旋转的情况
    }
}
```