下面这个例子是一个典型的通过排列树来生成排列，通过这个例子， 我们能够理解排列树的实现的细节：

package com.dengkai.divideAndConquer;

public class Perm {

static int i=0;

public static void perm(int [] a,int start,int end){

if(start==end){

i++;

System.out.println("第"+i+"个排列为：");

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

System.out.println(a[i]);

return;

}

for(int i=start;i<=end;i++){

swap(a,i,start);

perm(a,start+1,end);

//swap(a[i],a[start]);

swap(a,i,start);

}

}

public static void swap(int [] a,int i,int j){

int temp;

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

int [] a={1,2,3,4,5};

perm(a,0,4);

}

}

/\*通过i值我也能够发现算法的复杂度为n!

\* \*/

# 总结：排列树的实现

1. 排列树的实现是基于交换和恢复的
2. 排列树的排列每次从剩余部分选择一个元素和当前的第k个元素交换，显然随着递归深度的增加，可供选择的元素在减少。

# 总结：对排列的算法复杂度的计算

算法复杂度的计算，主要依据for循环执行的次数来计算得到，从程序可以看出，随着递归深度的增加，K值递减，并且这是嵌套循环，因此复杂度为n\*(n-1)\*(n-2)…..\*1