1.【实验内容】

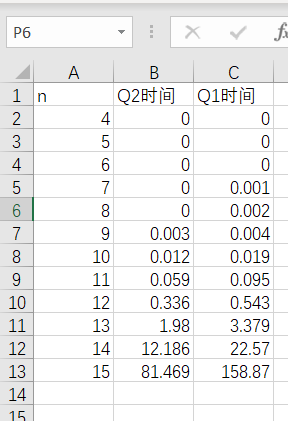
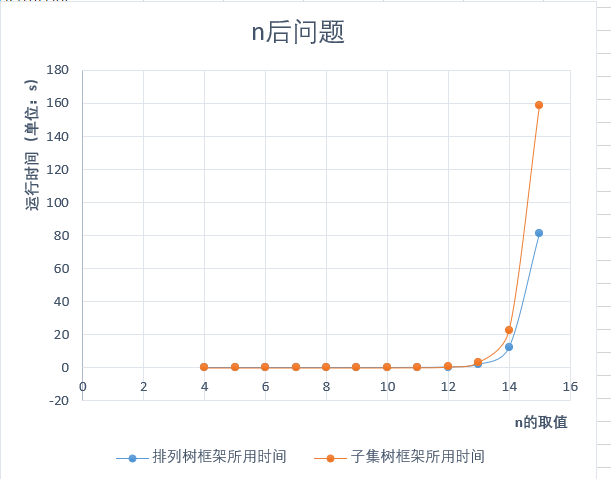
回溯法III

教材第5.5节n后问题，采用了子集树的算法框架，它还能用排列树框架求解。这2种算法的代码见附件。请你分别用不同的n值，测量它们的执行时间。然后用EXCEL的“散点图”，画出它们的对比曲线，并说明造成差别的原因。

2.【实验过程以及结果】

**2.1 测试两种方法的执行时间和n的散点图如下：**

其中，Q1：子集树的框架 Q2：排列树框架

可见，使用排列树框架的算法对比于子集树明显更优，用时更少。

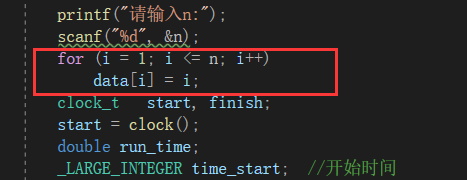
n越大，曲线越抖，增长的幅度越大，说明两种算法的时间复杂度都很高。

**2.2 分析**

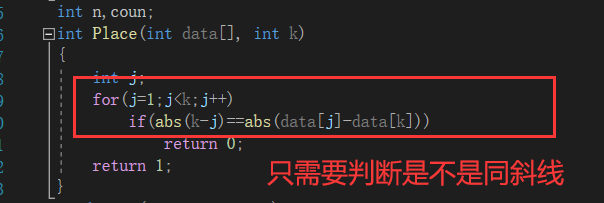
**2.2.1 【实验原理】**

**子集树想法：**用一个数组存储每行中的皇后的纵坐标，例如n=8时X=(4,6,8,2,7,1,3,5)表示第一行的皇后的横坐标为4，第二行的皇后的横坐标为6。从第一行开始放皇后，然后第二行继续放，直到放到第n行，一，放置的规则需要用一个约束条件来规范（约束条件用来决定下一列是取还是不取），而这个约束条件为abs(k-j)！=abs(x[j]-x[k]) 【不在同一斜线上】并且 x[j]！=x[k]【不在同一行】；二，当某一行没有了可以放置皇后的位子，要回溯至上一列重新摆放。直到第一行所有列都探索过，才算是算法结束。

**排列数想法：**与上相似，也用一个数组存储每行中的皇后的纵坐标。排列树与子集树的区别在于子集树不需要初始化而排列树需要,（先假设一个排列），此初始化内容为众多可能解集合中的一个。在这一次的实验代码里面，我们选择的是所有皇后不同行不同列却都在同一个斜线上的初始化序列。



然后不断的层层交换去更新数组，更新后判断该位置是否与已经存在的皇后的位置冲突(同斜线) ，【我们的数组必定不会有同行同列的冲突，因为已经我们一开始初始化假定的序列数组已经去除掉同行同列的可能】，**所以，这个问题也可以描述为n个数字的排列问题，约束条件为abs(k-j)！=abs(x[j]-x[k])，这样该问题就可以用全排列的算法求解。**



如果合法,则进入n+1;重复讨论n+2个皇后的位置 /如果不合法,交换回之前位置(只有合法之后才能占该列值) i++;当i++到for循环结束,也就是说该皇后在这一行所有列中都没有找到合适自己的位置,回退,即该方法执行结束,重新讨论之前上个皇后的位置

**2.2.2 时间复杂度分析**

**子集树：**算法中每个皇后都要试探n列，共n个皇后，试探了所有的n\*n的可能，其解空间是一棵子集树，不同于前面一般的二叉树子集树，这里每个结点有n棵子树。

所以，对应的算法时间复杂度为O(n^n)。

**排列树：**从2.2.1的分析，可以知道不同于子集树，我们求解的结果是集合1~n不重复的n个元素的某一种排列的时候，其对应的解空间即是排列树。这样的算法一开始我们就不需要去考虑存在同行同列的情况，只需要考虑同斜线的问题，自然相比于子集树更加简单。所以排列数算法就是转换成了一个n个元素全排列的问题，时间复杂度为O(n！)。

O(n！)< O(n^n),所以在散点图上，所以，子集树算法的坡度更抖更耗时间。