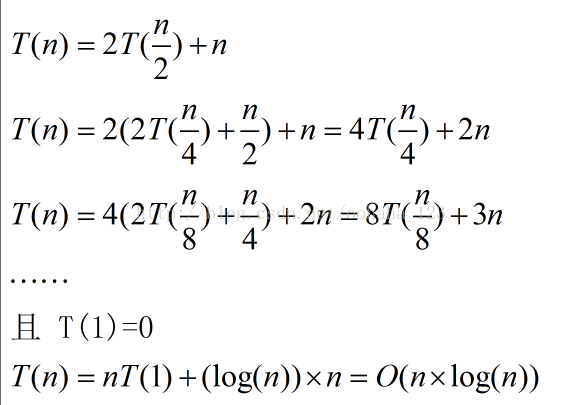
1.

a 首先使用将所有的points元素去重，然后使用快速排序算法进行排序，排序之后，重新遍历每个point，获取该point在排序之后列表的索引位置，即为对应的imp。

b. 因为排序之后，指定point前面的元素均比它小，所以对应前面元素的个数即为imp

c． Partition每次都划分得很均匀，如果排序n个关键字，其递归树的深度就为 [log2n]+1（ [x] 表示不大于 x 的最大整数），即仅需递归 log2n 次，需要时间为T（n）的话，第一次Partiation应该是需要对整个数组扫描一遍，做n次比较。然后，获得的枢轴将数组一分为二，那么各自还需要T（n/2）的时间（注意是最好情况，所以平分两半）。于是不断地划分下去，就有了下面的不等式推断：



2.

A．同时遍历两个列表A,B，当A中的元素p1 inferior 于B中的元素p2时，这时遍历B后面的元素，如果后面的元素superior p2时， 将对应的imp加1，当p1 superior p2时，遍历A后面的元素，如果后面的元素superior p1时，对应imp+1。 当遍历结束之后，将A和B中未遍历的元素添加到结果列表中。

B. 每次添加元素到结果列表的时候，都是按照先x2 后x1的顺序添加，并更新所有比superior于该元素的imp。所以最后得到的结果是排好序，对应的imp也是正确的。

C.T(n) = T(n\*n) = T(n^2)

3.

a.使用归并排序，分而治之的算法，不断拆分，最后变成合并两个单元素列表的问题。

b. 将已有序的子序列合并，得到完全有序的序列；即先使每个子序列有序，再使子序列段间有序。子序列有序，则最终的序列肯定也是有序的。

c. ，把一个规模为n的问题分成两个规模分别为n/2的子问题，时间为2T(n/2).合并时间复杂度为o（n）。总时间T(n)=2T(n/2)+o(n),不断递归，得到最终时间复杂度为o(nlogn)