**Divide and Conquer Algorithm Summary**:

**Standard Correctness Proof:** Prove by induction

**Divide and Conquer in three steps**:

**Divide**->Break up problem into several parts, usually into two equal parts.

**Conquer**->Solve each part recursively.

**Combine**->Combine solutions to sub-problem into overall solution. (Usually costs linear time.)

**Binary Search**: The array is pre-sorted. 从中间切. Runs in O(logn). Keep comparing X with the middle element of the array. T(n) = T(n/2) + 1 = O(logn).

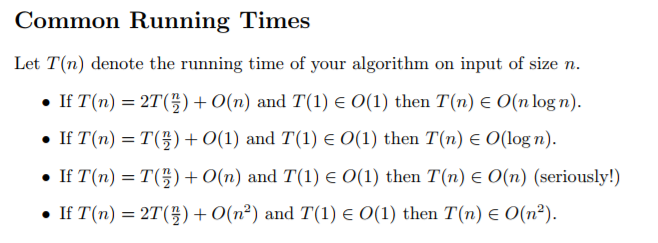
**Merge Sort Algorithm**:

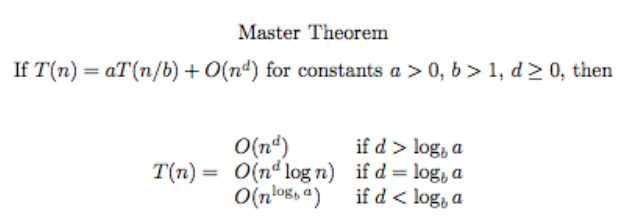
1. Divide: Divide the array into two halves.
2. Conquer: Recursively sort each half.
3. Combine: Merge two halves to make a sorted whole, merge done in linear time, O(n), since linear number of comparisons (n comparisons).

**Merging**: Keep track of the smallest element in each sorted half, and add it into the auxiliary (result) array. Keep repeating until done. One of the half may exhausted in the halfway. This takes linear time, since linear number of comparisons (n comparisons).

**Merge Sort Running Time**: O(nlog(n)).

**Time Complexity of Divide and Conquer Algorithm**: Usually done by master theorem.



****

**General Proof Correctness for Divide and Conquer**: Usually done using Proof by Induction.

**Counting Inversions Algorithm:**

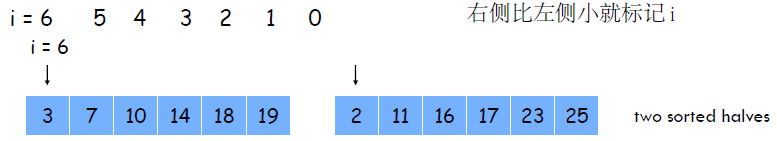
**Definition:** Inversion: Item i and k are inverted if i<k but ai > ak .

**Steps**: Divide: divide the list into two pieces.

Conquer: Recursively count inversions in each half.

Combine: Assume each half is sorted, count inversions where ai and aj are in different halves, and return sum of all three quantities.

**Running time** T(n) = 2T(n/2) + O(n) = O(nlog(n))

****

**Closest Pair of Points**

**Assumption:** No two points have the same x coordinate.

Divide: Using a vertical line to divide points into two sides, each side has roughly ½ n points.

Conquer: Find closest pair in each side recursively.

Combine: Find closest pair with one point in each side.

Return best of 3 solutions.

Running time: O(nlog(n))

一般方法：对半分，之后两边分别求，之后求两边衔接处的值，最后三者相加。

For the middle part, only need to consider 2sigma-strip, where sigma = min(dist(L), dist(R))

**Running time analysis for Divide-and-Conquer Algorithm**: Solve recursion by Prove by Unrolling or Master Theorem.