2.3-4
In order to insert an element to a $(n-1)$ sorted array, we need to identify the place to insert and then move the element there. $T(n) = T(n-1) + (n-1)$
$T(n) = \Theta(n^2)$
2.3-5
refer to $http://en.wikipedia.org/wiki/Binary\_search\_algorithm$
2.3-6
We can use binary search to improve the time of identifying the place to insert, but it will still take $O(n)$ to move the elements to that place. Therefore, using binay search cannot improve the worst-case running time of insert sort to $\Theta(nlgn)$
7.2-1
Assume $T(n) = \Theta(n^2)$, do induction as follows:
For base case, it is obviously true;
Suppose it is true for all $m < n$, we have
$T(n-1) \leq c(n-1)^2$, so
$T(n) \leq c(n-1)^2 + c_1 n = cn^2 + (c_1 - 2c)n + c \leq cn^2$, when $c_1 - 2c < 0$.
7.2-2
The case yields the worst-case partitioning, and recursion is
$T(n) = T(n-1) + T(0) + \Theta(n)$
so we get $T(n) = \Theta(n^2)$
7.2-5
In the partition step, one part, say A, is reduced to α of the original size, and the other part, say B, is reduced to α of the original size, until the size reaches 1. Take A for example, suppose its depth is k, then we have $\alpha^k n = 1$, which yields $k = log_\alpha 1/n = -lgn/lg\alpha$
Similar for B, we have $(1-\alpha)^k n = 1$, which yields $k = -lgn/lg(1-\alpha)$