2/8/16

**Big-oh**: let f(n) be the running time of an algorithm with n input values.

O(n) (with capital O) means <=. o(n) (with small o) means it is true for all constants and means strictly <.

This is the lower bound (best time).

We only care about worst time in this class.

The following is used as **proof** what complexity an algorithm is (give best time).

F(n) <= c\*g(n) -> f(n) = O(g(n))

C is any positive constant.

n0 is any n where it holds true.

Prove 2n+10 is an O(n) algorithm.

F(n) = 2n+10. G(n) = n.

2n+10 <= c\*n

10 <= n(c – 2).

n >= 10/(c – 2).

If c = 3, then this is true for any n >= 10.

2n+10 <= 3n, which is true for any n >= 10.

Thus, f(n) = 2n+10 is O(n). This is also O(n^2) because O(n) < O(n^2).

Always give best time unless problem states “is also O(n^2)”.

Big-Omega: worst time.

Big-Theta: when the upper bound is the same as the lower bound.

O(n) and Omega(n), then it is also Theta(n).