Comparing Time Complexities

Linear search

$f(n) = O(n) = \{t(n) \mid t(n) \leq c\, n$ for all $n \geq n_0$, n_0, c constants$\}$

Binary search

$f(n)= O(\log n) = \{t(n) \mid t(n) \leq c \log n$ for all $n \geq n_0$, n_0, c const$\}$

running time of EVERY implementation of binary search

Comparing Time Complexities

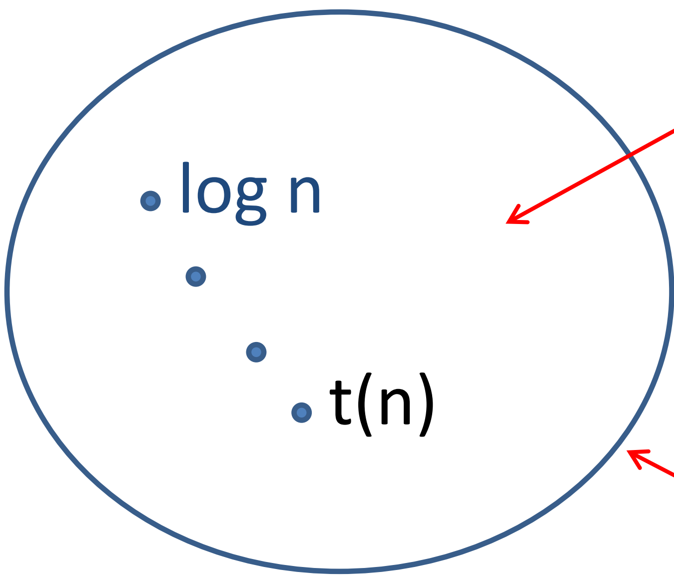
Linear search

$f(n) = O(n) = \{t(n) \mid t(n) \le c\, n \text{ for all } n \ge n_0,\ n_0,\ c \text{ constants}\}$

Binary search

$f(n) = O(\log n) = \{t(n) \mid t(n) \le c \log n \text{ for all } n \ge n_0,\ n_0,\ c \text{ const}\}$

running time of EVERY implementation of binary search

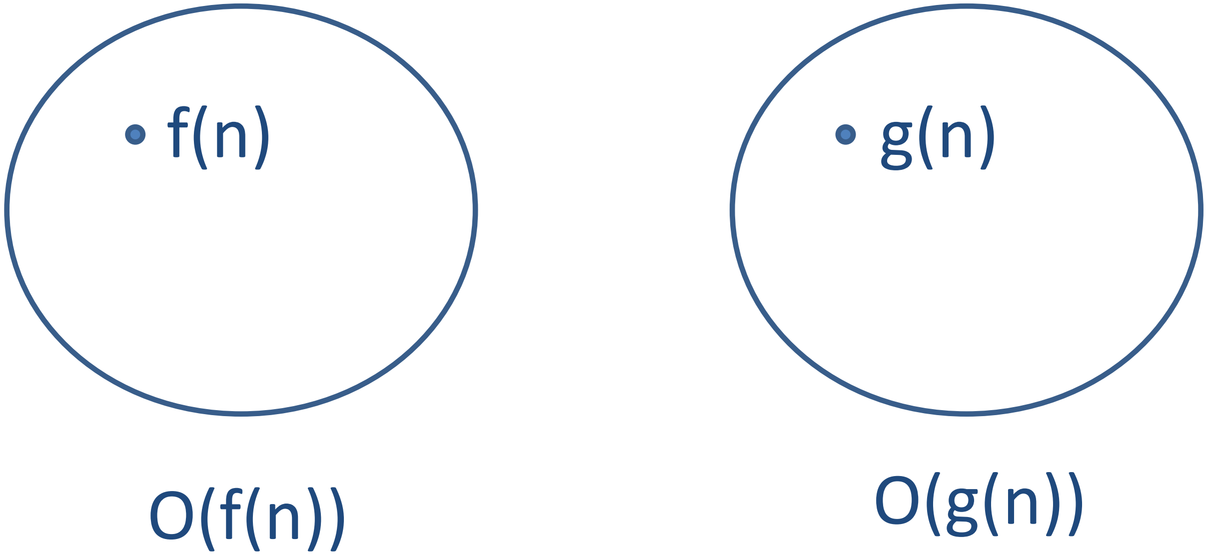


O(log n)

Comparing Orders

Algorithm A has complexity O(f(n))

Algorithm B has complexity O(g(n))



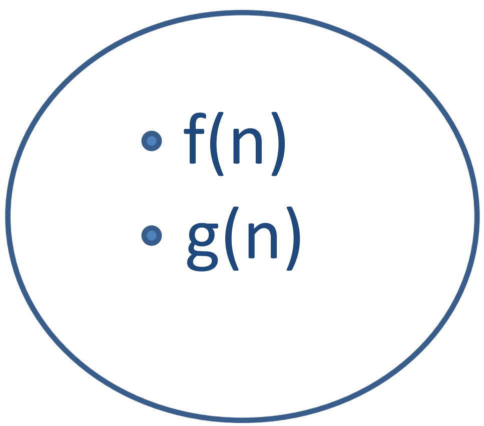
Comparing Orders

Algorithm A has complexity O(f(n))

Algorithm B has complexity O(g(n))

Two cases:

- f(n) is O(g(n)) and g(n) is O(f(n))



Both algorithms have the same set of possible running times

O(f(n)) = O(g(n))

Comparing Orders

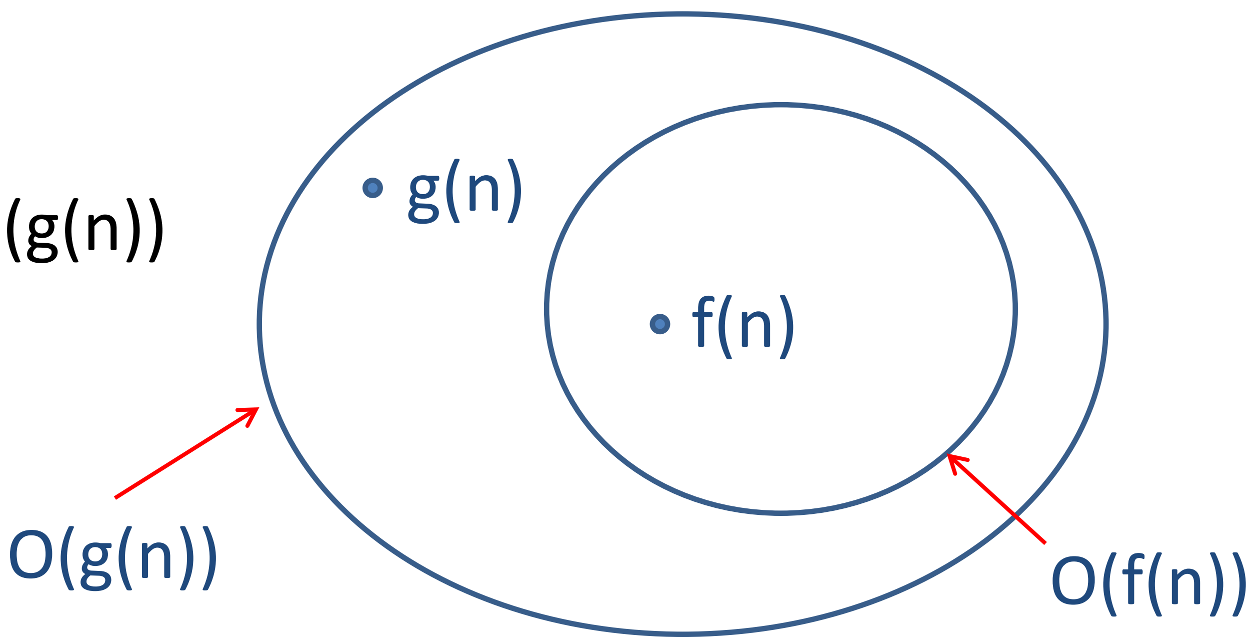
Algorithm A has complexity O(f(n))

Algorithm B has complexity O(g(n))

Two cases:

- f(n) is O(g(n)) and g(n) is **not** O(f(n))

$O(f(n)) \subset O(g(n))$



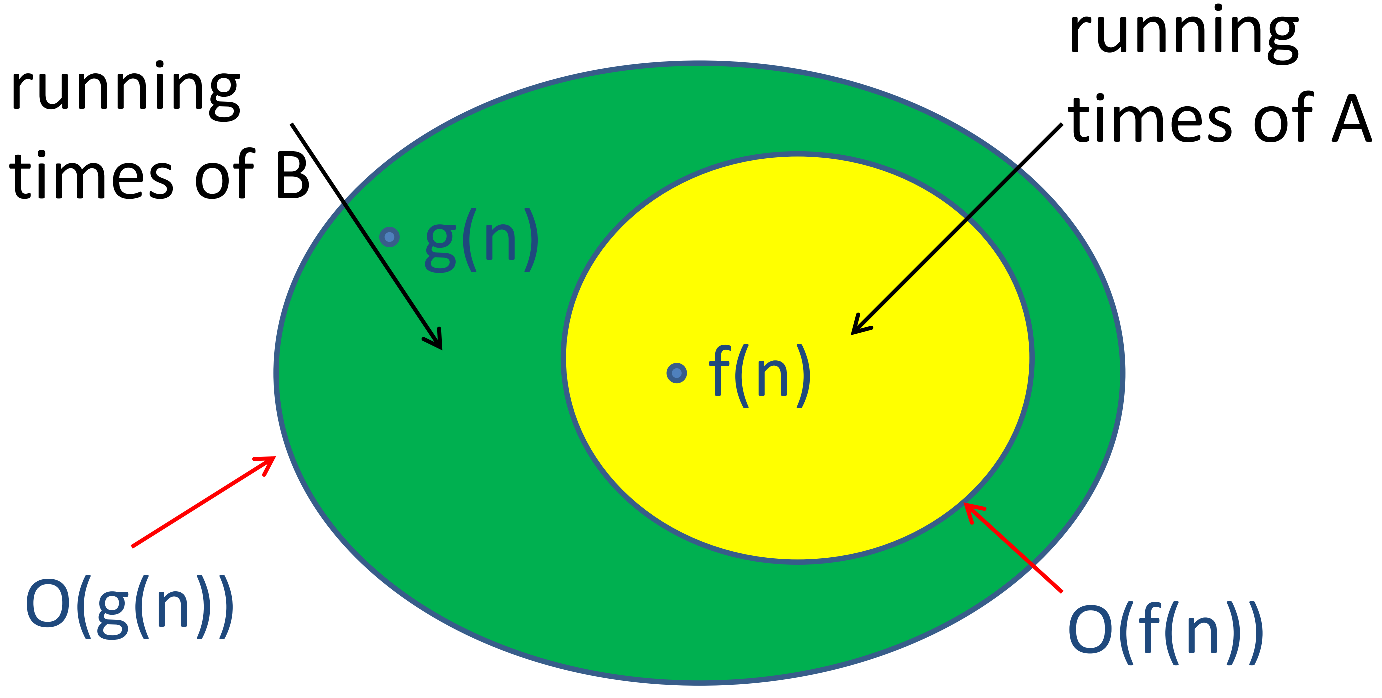
Comparing Orders

Algorithm A has complexity O(f(n))

Algorithm B has complexity O(g(n))

Two cases:

- f(n) is O(g(n)) and g(n) is **not** O(f(n))



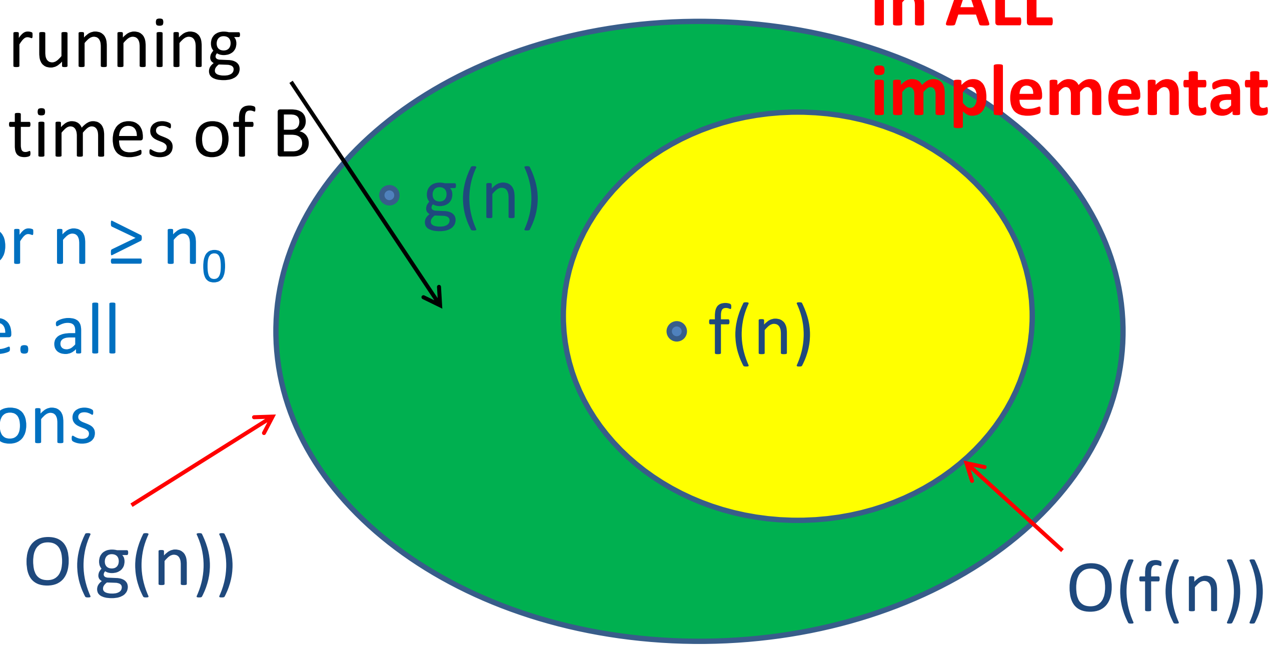
Comparing Orders

Algorithm A has complexity O(f(n))

Algorithm B has complexity O(g(n))

Two cases:

- f(n) is O(g(n)) and g(n) is **not** O(f(n)): **B is slower than A in ALL implementations**



$g(n)$ **>** $c\ f(n)$ for $n \geq n_0$ for all c, n_0, i.e. all implementations

Complexity Classes

$O(1) \subset O(\log n) \subset O(n) \subset O(n \log n)$

constant logarithmic linear

$\subset O(n^2) \subset O(n^a) \subset O(b^n)$

quadratic polynomial exponential

(constant a > 2) (b constant)

$\subset O(n!) \subset O(n^n) \ldots$

factorial

Efficient algorithms