Comparing Time Complexities

Linear search

\[ f(n) \text{ is } O(n) = \{t(n) | t(n) \leq c \ n \text{ for all } n \geq n_0, \ n_0, \ c \text{ constants} \} \]

Binary search

\[ f(n) \text{ is } O(\log n) = \{t(n) | t(n) \leq c \log n \text{ for all } n \geq n_0, \ n_0, \ c \text{ const} \} \]

running time of EVERY implementation of binary search
Comparing Time Complexities

Linear search

\[ f(n) = O(n) = \{ t(n) | t(n) \leq c \cdot n \text{ for all } n \geq n_0, n_0, c \text{ constants} \} \]

Binary search

\[ f(n) = O(\log n) = \{ t(n) | t(n) \leq c \cdot \log n \text{ for all } n \geq n_0, n_0, c \text{ constants} \} \]

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 $O(f(n))$

\[ O(f(n)) = O(g(n)) \]

Both algorithms have the same set of possible running times
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 $O(f(n))$
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

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

In ALL implementations
Complexity Classes

\[
\begin{align*}
O(1) & \subseteq O(\log n) & \subseteq O(n) & \subseteq O(n \log n) \\
\text{constant} & \subseteq \text{logarithmic} & \subseteq \text{linear} & \\
\subseteq O(n^2) & \subseteq O(n^a) & \subseteq O(b^n) \\
\text{quadratic} & \subseteq \text{polynomial} & \text{exponential} & (\text{constant } a > 2) & (b \text{ constant}) \\
\subseteq O(n!) & \subseteq O(n^n) & \ldots \\
\text{factorial} & \\
\end{align*}
\]

Efficient algorithms