Analyzing runtimes

We seek to understand worst-case running times of operations.

Recall:
get(int rank) in an array: $C_1$
get(int rank) in an list: $C_1 + \frac{n}{2}C_2$

why?
These are constant time:
- assignment
- addition
- indexing
- comparing
- branching

Removing an elt, if it has been found
adding in a particular position

Growth rate matters
Examples of Functions

\[ f(n) = 4n^2 \]
\[ f(n) = 4n^2 + 1000 \]
\[ f(n) = \begin{cases} 4n^3 & \text{if } n \text{ is even} \\ 5n^2 & \text{if } n \text{ is odd} \end{cases} \]

Lower bounds

\[ \Omega(g(n)) = \{ f(n) \mid \exists n_0, c \geq 0, \forall n \geq n_0, f(n) \geq c \cdot g(n) \} \]

\[ f(n) \in \Theta(g(n)) \text{ if both } O(g(n)) \text{ and } \Omega(g(n)) \]

\[ \lim_{n \to \infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & f(n) \in O(g(n)) \\ c > 0 & f(n) \in \Theta(g(n)) \\ \infty & f(n) \in \Omega(g(n)) \end{cases} \]

More code examples:
- nested loops
- selection sort
Suppose \( a \) is a sorted array.

Suppose we want to know if an array or list contains a particular value. How do we do it. How long?

Suppose the array or list is sorted?

\[
\text{int find (int a[], int x, int lo, int hi) } \{ \\
\text{ if (hi < lo) return -1; } \\
\text{ int mid = (lo + hi) / 2; } \\
\text{ if (a[mid] == x) return mid; } \\
\text{ else if (x > a[mid]) } \\
\text{ return find(a, mid+1, hi); } \\
\text{ else } \\
\text{ return find(a, lo, mid-1); } \\
\}
\]

3

Forgetting recursion, how long does this take?

Resize