Asymptotics & Induction
Asymptotic notation

Big-O and friends
Is $10n$ or $20n$ bigger?
Is $10n$ or $20n$ bigger?

• What about if $20n$ is run on a faster machine?
• What if I wait until next processor generation?
Is $n^2$ or $10n+11$ faster?
n^2 vs. 10n+11 revisited

Cross at n=11
10n vs. 20n revisited (with $n^2$)
Want to ignore
a) multiplicative constants
b) behavior at “small” n
Want to ignore
a) multiplicative constants
b) behavior at “small” n

Big-O notation:
f(n) = O(g(n)) if there exist constants \( n_0 \) and \( c \) such that \( f(n) \leq cg(n) \) for all \( n \geq n_0 \)
Show that $23n^2 + 4n + 85$ is $O(n^2)$

$23n^2 + 4n + 85$
Show that $23n^2 + 4n + 85$ is $O(n^2)$

$23n^2 + 4n + 85 \leq 23n^2 + 89n$ for $n \geq 1$
Show that $23n^2 + 4n + 85$ is $O(n^2)$

$23n^2 + 4n + 85 \leq 23n^2 + 89n$ for $n \geq 1$

$\leq 112n^2$ for $n \geq 1$
Show that $23n^2 + 4n + 85$ is $O(n^2)$

$23n^2 + 4n + 85 \leq 23n^2 + 89n$ \quad \text{for } n \geq 1$
$\leq 112n^2$ \quad \text{for } n \geq 1

i.e. $23n^2 + 4n + 85 = O(n^2)$ using $c = 112$ and $n_0 = 1$
Many ways to compare functions

\[ f(n) = \mathcal{O}(g(n)) \quad \text{Big-O} \]
\[ f(n) = \Omega(g(n)) \quad \text{Omega} \]
\[ f(n) = \Theta(g(n)) \quad \text{Theta} \]
\[ f(n) = o(g(n)) \quad \text{little-o} \]
Many ways to compare functions

\[ f(n) = O(g(n)) \quad \text{Big-O} \quad \text{“≤”} \]
\[ f(n) = \Omega(g(n)) \quad \text{Omega} \]
\[ f(n) = \Theta(g(n)) \quad \text{Theta} \]
\[ f(n) = o(g(n)) \quad \text{little-o} \]
Many ways to compare functions

\[
\begin{align*}
  f(n) &= O(g(n)) & \text{Big-O} & \text{“≤”} \\
  f(n) &= \Omega(g(n)) & \text{Omega} & \text{“≥”} \\
  f(n) &= \Theta(g(n)) & \text{Theta} \\
  f(n) &= o(g(n)) & \text{little-o}
\end{align*}
\]
Many ways to compare functions

\( f(n) = O(g(n)) \) \quad \text{Big-O} \quad “\leq”

\( f(n) = \Omega(g(n)) \) \quad \text{Omega} \quad “\geq”

\( f(n) = \Theta(g(n)) \) \quad \text{Theta} \quad “=“ \quad \text{i.e. both O and \( \Omega \)}

\( f(n) = o(g(n)) \) \quad \text{little-o}
Many ways to compare functions

\[ f(n) = O(g(n)) \quad \text{Big-O} \quad \text{“} \leq \text{“} \]
\[ f(n) = \Omega(g(n)) \quad \text{Omega} \quad \text{“} \geq \text{“} \]
\[ f(n) = \Theta(g(n)) \quad \text{Theta} \quad \text{“} = \text{“} \quad \text{i.e. both O and \Omega} \]
\[ f(n) = o(g(n)) \quad \text{little-o} \quad \text{“} < \text{“} \]
Many ways to compare functions

\[ f(n) = O(g(n)) \quad \text{Big-O} \quad \text{“≤”} \]
\[ f(n) = \Omega(g(n)) \quad \text{Omega} \quad \text{“≥”} \]
\[ f(n) = \Theta(g(n)) \quad \text{Theta} \quad \text{“=”} \quad \text{i.e. both O and \ Ω} \]
\[ f(n) = o(g(n)) \quad \text{little-o} \quad \text{“<”} \]

Claim:

a) If \( f(n) = O(g(n)) \) and \( g(n) = O(h(n)) \), then \( f(n) = O(h(n)) \)
b) \( f(n) = O(f(n)) \)
c) If \( f(n) = O(g(n)) \), then \( g(n) = \Omega(f(n)) \)
Induction
Induction framework

• Base case(s):
  – claim is true for one or more small values

• Induction hypothesis (IHOP)
  – assume claim is true up to some point

• Induction step
  – show that claim holds for next value (using IHOP)
Proof that $1+2+3+...+n = n(n+1)/2$
Merge(A[1..n₁], B[1..n₂], R[1..(n₁+n₂)])
  //merge sorted arrays A and B into R
  i = j = 1;
  for k = 1 to (n₁+n₂)
    if(i ≤ n₁ and (j > n₂ or A[i] ≤ B[j]))
      R[k] = A[i]
      i++
    else
      R[k] = B[j]
      j++
Merge(A[1..n₁], B[1..n₂], R[1..(n₁+n₂)])

//merge sorted arrays A and B into R
i = j = 1;
for k = 1 to (n₁+n₂)
    if(i ≤ n₁ and (j > n₂ or A[i] ≤ B[j]))
        R[k] = A[i]
        i++
    else
        R[k] = B[j]
        j++
Invariant: R[1..(k-1)] has the k-1 smallest values in sorted order, plus A[i] and B[j] are each either out of bounds or have the smallest values not in R[1..(k-1)]
Exercise: Write a function to reverse a singly-linked list

```java
public class Node {
    int value;
    Node next;
}
```
Node reverse(Node list) {
   Node retVal = null;
   while(list != null) {
      Node temp = list;
      list = list.next;
      temp.next = retVal;
      retVal = temp;
   }
   return retVal;
}
Mergesort(A[1..n])
    if(n > 1)
        m = (n+1)/2
        copy 1\textsuperscript{st} m values of A into new array L
        copy rest into new array R
        Mergesort(L)
        Mergesort(R)
        Merge(L, R, A)
Mergesort(A[1..n])

if(n > 1)

m = (n+1)/2

copy 1st m values of A into new array L

copy rest into new array R

Mergesort(L)

Mergesort(R)

Merge(L, R, A)

IHOP: Mergesort sorts arrays of length < n