Detail: when do we not recurse anymore?
when subarray has size 1

Running time for input of size n is $T(n)$
p=1, r=n

Mergesort(A,p,r)

if $p < r$ then $q = \left\lfloor \frac{p + r}{2} \right\rfloor$

$\Theta(1)$

Mergesort(A,p,q) $T\left(\left\lfloor \frac{n}{2} \right\rfloor \right)$

Mergesort(A,q+1,r) $T\left(\left\lfloor \frac{n}{2} \right\rfloor \right)$

Merge(A,p,q,r) $\Theta(n)$

Running time for an input of size n is $T(n)$

$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + \Theta(n) + \Theta(1)$

we will see later that this is $2 \cdot T(n/2) + \Theta(n) = \Theta(n \log n)$
Designing Algorithms

Insertion sort uses incremental approach into a sorted $A[1..j-1]$, insert 1 element $A[j]

Today -- divide-and-conquer approach

Divide the problem into a number of subproblems
Conquer solve each subproblem recursively
Combine solutions to subproblems into a solution to original problem

Merge sort
- Divide: divide n-element array $A$ into 2 subarrays of $n/2$ elements each
- Conquer: recursively sort the two subarrays
- Combine: merge 2 sorted subarrays into 1 sorted array

Example: $A = 5 \ 2 \ 4 \ 6 \ 1 \ 3 \ 2 \ 6$

1:

2:

3: $A = \text{Merge} (2 \ 4 \ 5 \ 6) (1 \ 2 \ 3 \ 6) = 1 \ 2 \ 2 \ 3 \ 4 \ 5 \ 6 \ 6$
Intuition: look at the highest degree term, disregard other terms and the constant in front of the highest degree term

\[ \frac{1}{2} n^2 + 3n \text{ “becomes” } n^2 \]
\[ \frac{1}{2} n^2 - 3n \text{ “becomes” } n^2 \]
\[ \frac{1}{2} n^2 \lg n - 3n \text{ “becomes” } n^2 \lg n \]
\[ 5n^{2.5} \text{ “becomes” } n^{2.5} \text{ (not } n^2 \text{)} \]

Which of \( \Theta, \Omega \) should be put in place of the question mark?

\[ n^2 = ?(n) \quad n^2 = ?(n^3) \quad n^2 = ?(n^2) \]
\[ n^2 \lg n = ?(n) \quad n^2 \lg n = ?(n^2) \quad n^2 \lg n = ?(n^3) \]

A word on functions

for any constants \( a, b \) and \( c > 1 \), as \( n \) approaches \( \infty \)

\[ \lg^n a n \leq n^b \leq c^n \]
even when \( a = 10000 \) and \( b = 0.00001 \)
or \( b = 10000 \) and \( c = 1.00001 \)

Thus, \( \lg^a n = O\left(n^b\right) \) and \( n^b = O\left(c^n\right) \)

or equivalently, \( \Omega\left(\lg^a n\right) = n^b \) and \( \Omega\left(n^b\right) = c^n \)
Example (formal): want to show that \( \frac{1}{2} n^2 + 3n = \Theta(n^2) \)
\[
f(n) = \frac{1}{2} n^2 + 3n \quad g(n) = n^2
\]
to show desired result, need \( c_1, c_2 \) and \( n_0 \) such that
\[
0 \leq \frac{1}{2} n^2 + 3n \leq c_2 n^2
\]
try \( c=1 \)
\[
\frac{1}{2} n^2 + 3n \leq n^2 \quad 3n \leq \frac{1}{2} n^2 \quad 6 \leq n \quad \text{i.e. } n_0 = 6
\]

Example (formal): want to show that \( \frac{1}{2} n^2 - 3n = \Theta(n^2) \)
\[
f(n) = \frac{1}{2} n^2 - 3n \quad g(n) = n^2
\]
to show desired result, need \( c_1, c_2 \) and \( n_0 \) such that
\[
0 \leq c_1 n^2 \leq \frac{1}{2} n^2 - 3n \leq c_2 n^2
\]
dividing by \( n^2 \), we get
\[
0 \leq c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2
\]
holds for \( 10 \leq n \) and \( c_2 = 1 \)
holds for \( 10 \leq n \) and \( c_1 = 0.2 \)
\( f(n) = \Omega(g(n)) \) if there exist constants \( c \) and \( n_0 \) such that
\[
0 \leq c g(n) \leq f(n) \quad \text{for all } n_0 \leq n
\]

Think of \( \Omega \) as a lower bound function.

\( f(n) = \Theta(g(n)) \) if there exist constants \( c_1, c_2 \) and \( n_0 \) such that
\[
0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \text{for all } n_0 \leq n
\]

Think of \( \Theta \) as both upper and lower bound function.
Lecture 2: Growth of Functions

Want to determine the order of growth for the worst-case running time

\[ T(n) = \left( \frac{c_4}{2} + \frac{c_5}{2} + \frac{c_6}{2} \right) n^2 + \left( c_1 + c_2 + c_3 + \frac{c_4}{2} - \frac{c_5}{2} - \frac{c_6}{2} + c_7 \right) n \]
\[- (c_2 + c_3 + c_4 + c_7) \]

Growth of Functions (formal definitions)

\[ f(n) = O(g(n)) \] if there exist constants \( c \) and \( n_0 \) such that

\[ 0 \leq f(n) \leq c g(n) \quad \text{for all } n_0 \leq n \]

Think of \( O \) as upper bound function.