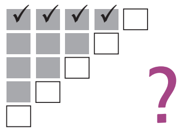
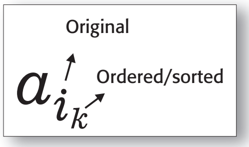
Analysis of Recursive Algorithms
Quickselect



Can the worst-case **efficiency sequence** (op count terms at different input sizes) be derived from the recursive pseudocode? Write summation. *(Fig. assumes Lomuto.)*



Stated in textbook in terms of desired order (k^{th}), here the index (k–1 in textbook).

Quiz
Which step depends on n? *(Note nested loops pattern, here a loop inside recursion body.)*

Exercise
Write the worst-case recurrence with Lomuto partition, investigate in *WolframAlpha*.

⋆Exercise
Compare to case when a basic op is performed a constant number of times in recursion body, give examples.

Algorithm *Quickselect*

Input Subarray $A[l..r]$ where $0 \le l \le r \le n-1$, a valid index $l \le k \le r$

Output $A[k]$, the $(k+1)$th order statistic

```
s ← partition(A[l..r])   ↻n
if s = k then  return A[s]
[else] if s > k then
      r ← s − 1
else
      l ← s + 1
return  Quickselect(A[l..r], k)   ↻
```

Basic operation?

Efficiency recurrence

A pattern?

Beyond Brute Force

Exercise
Compare divide-conquer to decrease-conquer addition and exponentiation. *Hint: check the recurrences.*

➪ Divide-and-conquer

- ✎ Divide problem into smaller instances
- ✎ Apply solution independently to smaller instances (repeatedly, typically recursive)
- ✎ Construct problem solution from solutions to smaller instances

➪ Familiar examples

Divide-and-Conquer
Mergesort

Exercise
Trace in your mind lists of 2 and 3 keys.

- ✎ <u>Divide</u> problem into smaller instances
- ✎ Apply solution independently to smaller instances
- ✎ Construct problem solution from solutions to smaller instances

Algorithm *Mergesort*
Input … $A[0 .. n-1]$ …
Output …

```
if n > 1 then
    copy A[0 .. ⌊n/2⌋ − 1] to B[0 .. ⌊n/2⌋ − 1]
    copy A[⌊n/2⌋ .. n − 1] to C[0 .. ⌊n/2⌋ − 1]
    Mergesort(B[0 .. ⌊n/2⌋ − 1])
    Mergesort(C[0 .. ⌊n/2⌋ − 1])
    Merge(B, C, A)
```

Divide-Conquer: Mergesort Example

➪ Backtracking phase

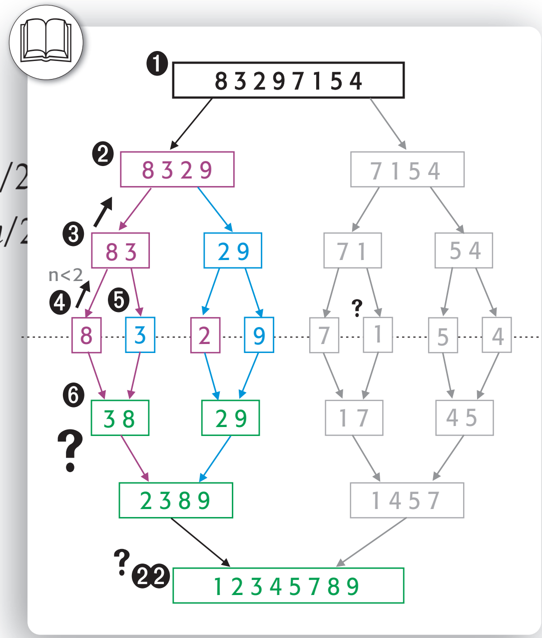
Exercise
List calls in steps 4-6, show input arrays for each (i.e, **serialize** figure), note *list reduction sequence*. **Hint**: print an operation log to study (note recursive call to n<2 instance triggers backtracking phase).

To visualize backtracking (black arrows), fold figure along the dotted line to overlay the green box #6 on the magenta box #3.

Quiz
How many extra array elements were allocated by *mergesort* during the expansion phase of the recursion? What if n=16?

Algorithm *Mergesort*

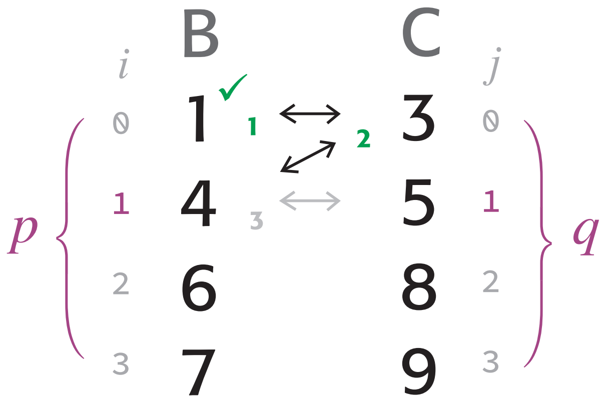
1: **if** $n > 1$ **then**
2: copy $A[0 .. \lfloor n/2 \rfloor - 1]$ to $B[0 .. \lfloor n/2$
3: copy $A[\lfloor n/2 \rfloor .. n-1]$ to $C[0 .. \lfloor n/2$
4: $Mergesort(B[0 .. \lfloor n/2 \rfloor - 1])$
5: $Mergesort(C[0 .. \lfloor n/2 \rfloor - 1])$
6: $Merge(B, C, A)$



Divide-Conquer: Mergesort
Merge Sorted Lists

Quiz
How many key comps were performed in this case? (Assume B ≤ C merge condition.) Compare to instance from textbook (Slide 3).

⇨ Examine small instance



Quiz
Give examples for best and worst case instances.

⇨ Merge performance

Divide-Conquer: Mergesort

📖 A Merge Procedure

Exercise
Trace the small instance from previous slide.

Quiz
How many key comps are performed? How many times the next element is picked from C?

★Exercise
Write pseudocode for a 3-way merge.

Algorithm *Merge*
Input Sorted arrays $B[0 .. p-1]$, $C[0 .. q-1]$, original array A
Output ...

```
i ← 0, j ← 0, k ← 0
while i < p and j < q do
    if B[i] ≤ C[j] then
        A[k] ← B[i], i ← i + 1
    else
        A[k] ← C[j], j ← j + 1
    k ← k + 1
if i = p then
    copy C[j .. q − 1] to A[k .. p + q − 1]      ▷ note 2 ≤ p + q ≤ n
else
    copy B[i .. p − 1] to A[k .. p + q − 1]
```

Efficiency

Divide-Conquer: Mergesort Performance

Quiz
Which steps in *merge-sort* depend on *n*?

```
if n > 1 then
   copy A[0 .. ⌊n/2⌋ − 1] to B[0 .. ⌊n/2⌋ − 1]
   copy A[⌊n/2⌋ .. n − 1] to C[0 .. ⌊n/2⌋ − 1]
   Mergesort(B[0 .. ⌊n/2⌋ − 1])
   Mergesort(C[0 .. ⌊n/2⌋ − 1])
   Merge(B, C, A)   n
```

- ➪ **Choice of basic operation**
- ➪ **Write recurrences**
- ➪ **Investigate worst-case**

General Divide-Conquer Recurrence

$$T(n) = aT(n/b) + f(n) \text{ where } a \geq 1, b \geq 2$$

Check comment in Appendix B.

For

$$n = b^k, k = 1, 2, \cdots$$

Divide-and-Conquer
Master Theorem

If $f(n) \in \Theta(n^d)$ with $d \geq 0$ in recurrence

$$T(n) = aT(n/b) + f(n), a \geq 1, b > 1$$

$$n = b^k, k = 1, 2, \cdots$$

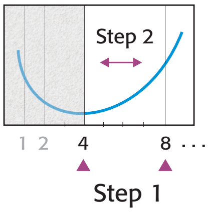
Determine growth class $g(n)$ from Theorem for the special case n powers of b, then use **Smoothness Rule** to generalize result for all n if $g(n)$ is smooth (figure next slide).

Quiz
Does it matter if we specify a base in the log efficiency class? Justify.

$$T(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(\underbrace{n^d \log n}_{g(n)}) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Divide-Conquer: Mergesort
Conclusions

➪ **Smoothness Rule**



Exercise
Solve the worst-case recurrence of *mergesort* for $n=2^k$ using backward substitution.

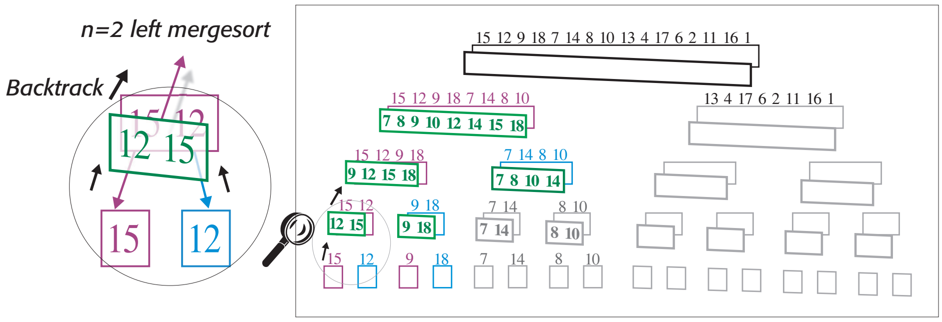
Quiz
Write the best-case recurrence, investigate in *WolframAlpha*.

➪ Time efficiency

- ✎ Worst-case (2-step solution)
- ✎ Average-case 🕮
- ✎ Class?

➪ Space efficiency (?) 🕮

Divide-Conquer: Mergesort Practice



Exercise
Modify the pseudocode in *Mergesort* for a 3-way merge.