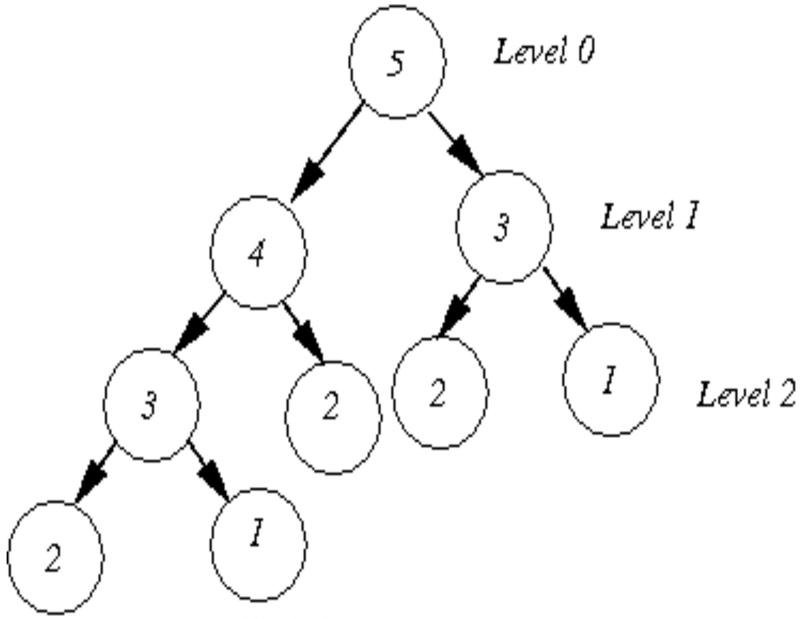
# Divide-n-Conquer and Recursion

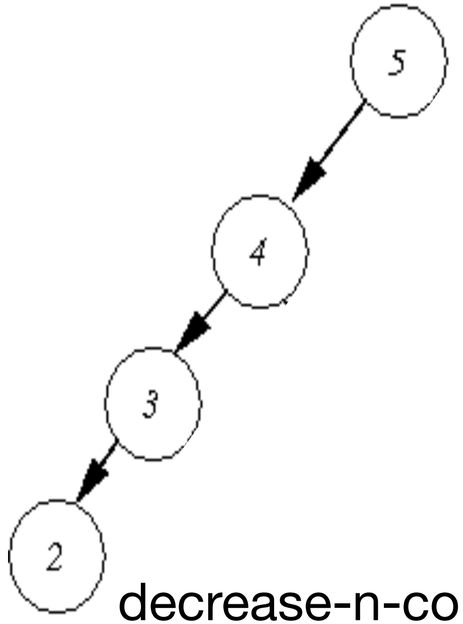
- divide (or reduce) a big problem to one or more smaller subproblems
- conquer: recursively solve each subproblem
- combine the results of subproblems to form the result of the big problem

$$f(n) = f(n-1) + f(n-2)$$




divide-n-conquer
branching recursion

$$f(n) = n \cdot f(n-1)$$




tail recursion
== for loop

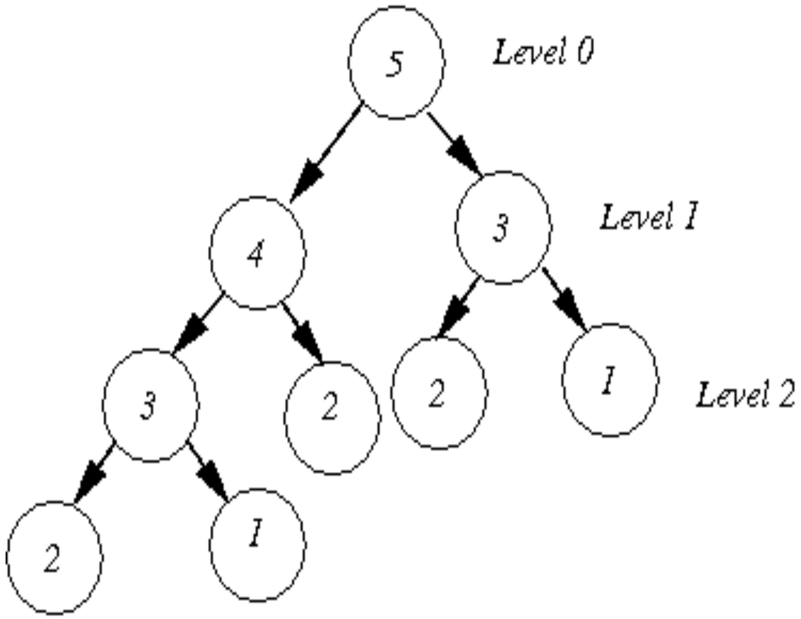
decrease-n-conquer
tail recursion == loop

# Divide-n-Conquer vs. Decrease-n-Conquer

Fibonacci (naive)

fast sorting: quicksort, mergesort

tree traversals (balanced)



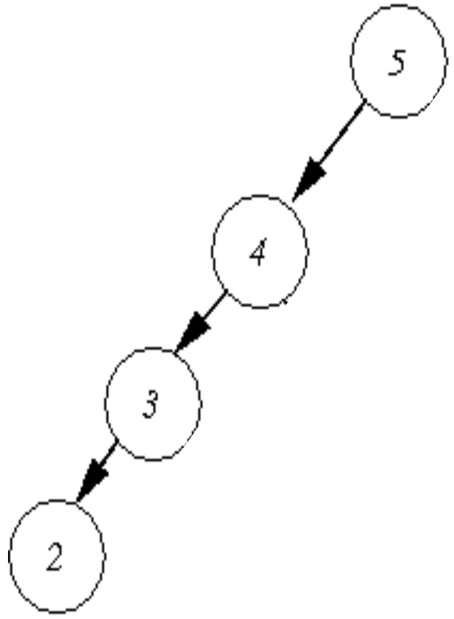

factorial

slow sorting: selection, insertion

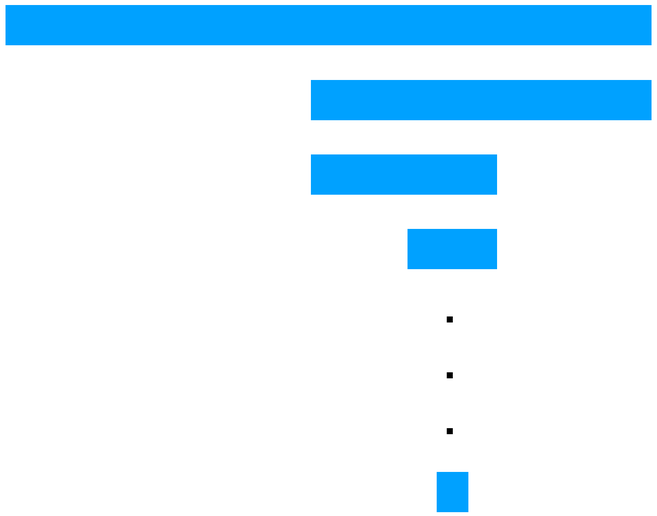
binary search

quickselect

tree traversals (linear-chain)



# Binary Search vs. Quickselect



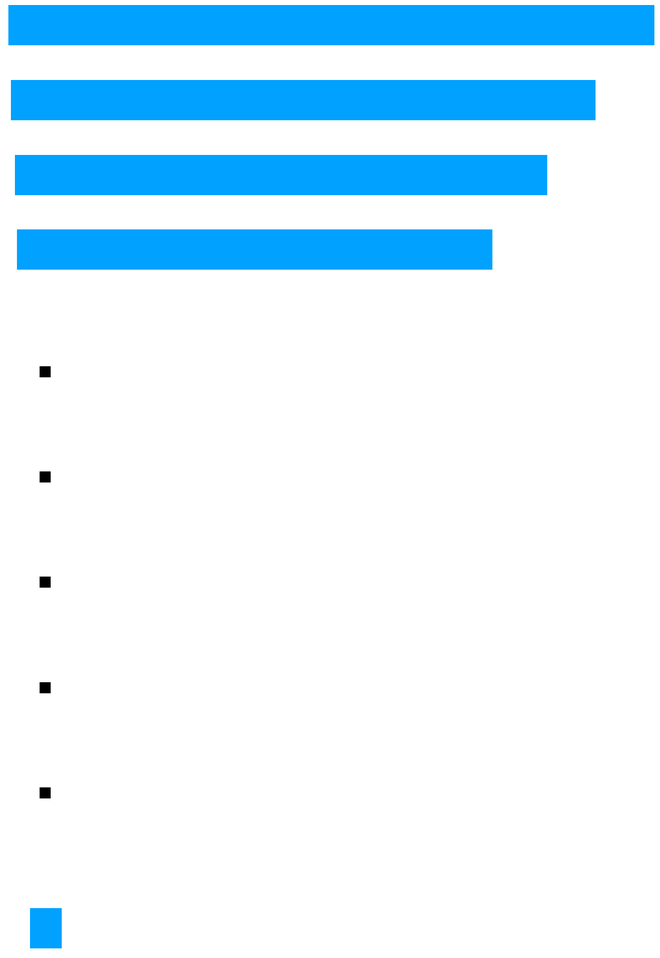
| size | binary search | quickselect (best-case) |
|---|---|---|
| $n$ | $O(1)$ | $n$ |
| $n/2$ | $O(1)$ | $n/2$ |
| $n/4$ | $O(1)$ | $n/4$ |
| $n/8$ | $O(1)$ | $n/8$ |
| . | . | . |
| . | . | . |
| . | . | . |
| 1 | $O(1)$ | 1 |

$$T(n) = T(n/2) + O(1)$$

$$T(n) = T(n/2) + O(n)$$

$$n(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots) = n \cdot 2 = O(n)$$

quickselect (worst-case)




$$n + (n-1) + (n-2) + \cdots + 1 = O(n^2)$$

# Recurrence Relations

- $T(n) = \ldots T(\ldots) + O(\ldots) = ?$
  - recursive part (conquer): $2T(n/2)$, $T(n-1)$, or $T(n/2)$
  - non-recursive part (divide+combine): $O(1)$ or $O(n)$

the $\log n$ factor only shows up if each level has the same amount of work

otherwise, either dominated by the top-level or bottom-level

| | O(1) | O(n) | |
|---|---|---|---|
| T(n/2) | binary search<br>O(log n) | **quickselect-best**<br>**O(n)** | decrease-n-conquer<br>(tail recursion) |
| 2T(n/2) | balanced tree traversal<br>O(n) | mergesort, quicksort-best<br>O(n log n) | divide-n-conquer<br>(branching recursion) |
| T(n-1) | linear scan, unbalanced tree traversal<br>O(n) | select-sort, insert-sort, quicksort-worst, **quickselect-worst**<br>**O(n^2)** | decrease-n-conquer<br>(tail recursion) |