	sorted array	unsorted array	linked list
find	$O(\lg n)$	$O(n)$	$O(n)$
insert	$O(n)$	$O(1)$	$O(1)$
delete	$O(n)$	$O(n)$ shifting to fill spot	$O(1)$
add/put	$O(\lg n) + O(n) = O(n)$	$O(n) + O(1) = O(n)$	$O(n) + O(1) = O(n)$
remove	$O(\lg n) + O(n) = O(n)$	$O(n) + O(n) = O(n)$	$O(n) + O(1) = O(n)$

find in unsorted array

- linear search

- for each item
  until found
  - check if item
    is the value ← 1-3 operations
    we want

BC

in first
element
1 iteration
$O(1)$

WC

last
element
n iterations
$\approx 3n$
$O(n)$

AC

middle
element
$\frac{n}{2}$ iterations
$\approx 3/2n$
$O(n)$

find in sorted array

- binary search

- let mid be size/2
- let start be 0
- let end be size-1

} 3 operations

- while not found and end-start > 0 ← $\log_2 n$ iterations
  - check mid
    - if < value
      - end = mid - 1
      - mid = (start + end)/2
    - if > value
      - start = mid + 1
      - mid = (start + end)/2
    - else
      - found

} ~2 operations

$3 + 2\log_2 n$

$\approx 2\log_2 n$

$= O(\lg n)$

http://host/path/file?a=b&c&d=bob