This is CS50
Getting an education from MIT is like trying to get a drink from a firehose...
algorithms
linear search
For i from 0 to n-1
    If i'th element is 50
        Return true
Return false
binary search
If middle item is 50
   Return true
Else if 50 < middle item
   Search left half
Else if 50 > middle item
   Search right half
If no items

If middle item is 50
  
  Return true

Else if 50 < middle item
  
  Search left half

Else if 50 > middle item
  
  Search right half
If no items
    Return false
If middle item is 50
    Return true
Else if 50 < middle item
    Search left half
Else if 50 > middle item
    Search right half
The graph illustrates the relationship between the size of a problem and the time it takes to solve it. The time to solve increases with the size of the problem, with different lines representing different complexities or methods of solving.
Time to solve:
- $O(n)$
- $O(n/2)$
- $O(\log_2 n)$

Size of problem
size of problem

time to solve

$O(n)$

$O(n/2)$

$O(\log_2 n)$
time to solve

size of problem

$O(n)$  $O(n/2)$

$O(\log_2 n)$
The graph shows the relationship between the size of the problem and the time to solve it. The red line represents $O(n)$, the yellow line represents $O(n)$, and the green line represents $O(\log_2 n)$. The vertical axis represents the time to solve, and the horizontal axis represents the size of the problem.
size of problem

$O(n)$  $O(n)$

$O(\log n)$
The graph shows the relationship between the size of the problem and the time it takes to solve it. The red curve represents $O(n)$ complexity, where the time to solve increases linearly with the size of the problem. The yellow curve represents $O(\log n)$ complexity, where the time to solve increases logarithmically with the size of the problem. The green line compares these complexities, illustrating that $O(\log n)$ is more efficient than $O(n)$ as the size of the problem grows.
$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$
$O(n^2)$

$O(n \log n)$

$O(n)$  linear search

$O(\log n)$

$O(1)$
$O(n^2)$

$O(n \log n)$

$O(n)$  linear search

$O(\log n)$  binary search

$O(1)$
$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$
$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$  linear search
\( \Omega(n^2) \)

\( \Omega(n \log n) \)

\( \Omega(n) \)

\( \Omega(\log n) \)

\( \Omega(1) \)  linear search, binary search
int numbers[]
string names[]
person people[]
string name;
string number;
typedef struct
{
    string name;
    string number;
}
person;
unsorted → output
unsorted → sorted
7 2 1 6 3 4 50 → sorted
bubble sort
Repeat n-1 times
   For i from 0 to n-2
      If i'th and i+1'th elements out of order
         Swap them
Repeat n-1 times
  For i from 0 to n-2
    If i'th and i+1'th elements out of order
      Swap them
$(n - 1) \times (n - 1)$
\[(n - 1) \times (n - 1)\]

\[n^2 - 1n - 1n + 1\]
\((n - 1) \times (n - 1)\)

\(n^2 - 1n - 1n + 1\)

\(n^2 - 2n + 1\)
(n − 1) × (n − 1)

n^2 − 1n − 1n + 1

n^2 − 2n + 1

O(n^2)
$O(n^2)$

$O(n \log n)$

$O(n)$  linear search

$O(\log n)$  binary search

$O(1)$
$O(n^2)$  bubble sort  
$O(n \log n)$  
$O(n)$  linear search  
$O(\log n)$  binary search  
$O(1)$
Repeat n-1 times
  For i from 0 to n-2
    If i'th and i+1'th elements out of order
      Swap them
\( \Omega(n^2) \)

\( \Omega(n \log n) \)

\( \Omega(n) \)

\( \Omega(\log n) \)

\( \Omega(1) \)  linear search, binary search
\( \Omega(n^2) \)  

bubble sort

\( \Omega(n \log n) \)

\( \Omega(n) \)

\( \Omega(\log n) \)

\( \Omega(1) \)  

linear search, binary search
selection sort
6 3 8 5 2 7 4 1
For i from 0 to n-1
    Find smallest item between i'th item and last item
    Swap smallest item with i'th item
For $i$ from $0$ to $n-1$

Find smallest item between $i$'th item and last item

Swap smallest item with $i$'th item
\[ n + (n - 1) \]
\[ n + (n - 1) + (n - 2) \]
\[ n + (n - 1) + (n - 2) + ... + 1 \]
\[ n + (n - 1) + (n - 2) + \ldots + 1 \]

\[ n(n + 1)/2 \]
\[ n + (n - 1) + (n - 2) + \ldots + 1 \]
\[ n(n + 1)/2 \]
\[ (n^2 + n)/2 \]
\[ n + (n - 1) + (n - 2) + \ldots + 1 \]
\[ n(n + 1)/2 \]
\[ (n^2 + n)/2 \]
\[ n^2/2 + n/2 \]
\[ n + (n - 1) + (n - 2) + \ldots + 1 \]
\[ n(n + 1)/2 \]
\[ (n^2 + n)/2 \]
\[ n^2/2 + n/2 \]
\[ O(n^2) \]
$O(n^2)$  bubble sort

$O(n \log n)$

$O(n)$  linear search

$O(\log n)$  binary search

$O(1)$
$O(n^2)$    bubble sort, selection sort

$O(n \log n)$

$O(n)$    linear search

$O(\log n)$    binary search

$O(1)$
For i from 0 to n-1
    Find smallest item between i'th item and last item
    Swap smallest item with i'th item
\( \Omega(n^2) \)  
\( \Omega(n \log n) \)  
\( \Omega(n) \)  
\( \Omega(\log n) \)  
\( \Omega(1) \)  

bubble sort

linear search, binary search
\( \Omega(n^2) \)  
- bubble sort, selection sort

\( \Omega(n \log n) \)

\( \Omega(n) \)

\( \Omega(\log n) \)

\( \Omega(1) \)  
- linear search, binary search
bubble sort
Repeat n-1 times
   For i from 0 to n-2
      If i'th and i+1'th elements out of order
         Swap them
Repeat until no swaps
  For i from 0 to n-2
    If i'th and i+1'th elements out of order
      Swap them
\( \Omega(n^2) \)  
\( \Omega(n \log n) \)  
\( \Omega(n) \)  
\( \Omega(\log n) \)  
\( \Omega(1) \)
\( \Omega(n^2) \) selection sort

\( \Omega(n \log n) \)

\( \Omega(n) \) bubble sort

\( \Omega(\log n) \)

\( \Omega(1) \) linear search, binary search
elections
recursion
1. Pick up phone book
2. Open to middle of phone book
3. Look at page
4. If Smith is on page
5.   Call Mike
6. Else if Smith is earlier in book
7.   Open to middle of left half of book
8.   Go back to line 3
9. Else if Smith is later in book
10. Open to middle of right half of book
11. Go back to line 3
12. Else
13. Quit
1. Pick up phone book
2. Open to middle of phone book
3. Look at page
4. If Smith is on page
   5. Call Mike
5. Else if Smith is earlier in book
   6. Open to middle of left half of book
   7. Go back to line 3
6. Else if Smith is later in book
   7. Open to middle of right half of book
   8. Go back to line 3
12. Else
13. Quit
1. Pick up phone book
2. Open to middle of phone book
3. Look at page
4. If Smith is on page
5.   Call Mike
6. Else if Smith is earlier in book
7.   Open to middle of left half of book
8.
9. Else if Smith is later in book
10.  Open to middle of right half of book
11.
12. Else
13. Quit
1. Pick up phone book
2. Open to middle of phone book
3. Look at page
4. If Smith is on page
   5. Call Mike
5. Else if Smith is earlier in book
   6. Search left half of book
6. Else if Smith is later in book
   7. Search right half of book
7. Else
8. Quit
Pick up phone book
Open to middle of phone book
Look at page
If Smith is on page
    Call Mike
Else if Smith is earlier in book
    Search left half of book
Else if Smith is later in book
    Search right half of book
Else
    Quit
1 Pick up phone book
2 Open to middle of phone book
3 Look at page
4 If Smith is on page
  5 Call Mike
6 Else if Smith is earlier in book
  7 Search left half of book
7 Else if Smith is later in book
  9 Search right half of book
10 Else
11 Quit
merge sort
If only one item
    Return
Else
    Sort left half of items
    Sort right half of items
    Merge sorted halves
If only one item
   Return
Else
   Sort left half of items
   Sort right half of items
   Merge sorted halves
<table>
<thead>
<tr>
<th>7</th>
<th>5</th>
<th>2</th>
<th>6</th>
<th>3</th>
<th>8</th>
<th>1</th>
</tr>
</thead>
</table>
4 7 2 5

6 3 8 1
$O(n^2)$  bubble sort, selection sort
$O(n \log n)$
$O(n)$  linear search
$O(\log n)$  binary search
$O(1)$
$O(n^2)$   bubble sort, selection sort
$O(n \log n)$   merge sort
$O(n)$   linear search
$O(\log n)$   binary search
$O(1)$
\[ \Omega(n^2) \quad \text{selection sort} \]
\[ \Omega(n \log n) \]
\[ \Omega(n) \quad \text{bubble sort} \]
\[ \Omega(\log n) \]
\[ \Omega(1) \quad \text{linear search, binary search} \]
<table>
<thead>
<tr>
<th>Time Complexity</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Omega(n^2) )</td>
<td>selection sort</td>
</tr>
<tr>
<td>( \Omega(n \log n) )</td>
<td>merge sort</td>
</tr>
<tr>
<td>( \Omega(n) )</td>
<td>bubble sort</td>
</tr>
<tr>
<td>( \Omega(\log n) )</td>
<td></td>
</tr>
<tr>
<td>( \Omega(1) )</td>
<td>linear search, binary search</td>
</tr>
</tbody>
</table>
$\Theta(n^2)$

$\Theta(n \log n)$

$\Theta(n)$

$\Theta(\log n)$

$\Theta(1)$
$\Theta(n^2)$ selection sort

$\Theta(n \log n)$ merge sort

$\Theta(n)$

$\Theta(n)$

$\Theta(\log n)$

$\Theta(1)$
This is CS50