This is CS50
input $\rightarrow$ 

$\rightarrow$ output
searching
algorithms
running times
size of problem vs. time to solve
size of problem \rightarrow time to solve

- $O(n)$
- $O(n/2)$
- $O(\log_2 n)$
size of problem

time to solve

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

$O(\log_2 n)$
The table shows the time to solve problems as a function of the size of the problem. The graphs represent:

- **Red line**: $O(n)$, linear time complexity.
- **Yellow line**: $O(n)$, linear time complexity.
- **Green line**: $O(\log_2 n)$, logarithmic time complexity.

As the size of the problem increases, the time to solve the problem increases as well, with the logarithmic time complexity growing much more slowly than the linear time complexity.
The size of the problem affects the time to solve:

- \(O(n)\) and \(O(n)\) are linear time complexities.
- \(O(\log n)\) is a logarithmic time complexity.

Graph showing the relationship between time to solve and size of problem.
The graph shows the relationship between the size of the problem and the time to solve it. The time to solve is compared to two different complexities: $O(n)$ and $O(\log n)$. As the size of the problem increases, the time to solve for $O(n)$ grows linearly, while for $O(\log n)$, it grows much slower, approaching the origin more slowly.

- $O(n)$: Red line
- $O(\log n)$: Green line

The red line represents a linear relationship, while the green line represents a logarithmic relationship. The red line starts from the origin and moves upwards, whereas the green line starts from a lower point and grows more slowly as it moves upwards.
\(O(n^2)\)
\(O(n \log n)\)
\(O(n)\)
\(O(\log n)\)
\(O(1)\)
$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

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

$\Theta(n \log n)$

$\Theta(n)$

$\Theta(\log n)$

$\Theta(1)$
linear search
For each door from left to right
    If number is behind door
        Return true
Return false
For i from 0 to n-1
    If number behind doors[i]
        Return true
Return false
$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

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

$O(n \log n)$

$O(n)$  linear search

$O(\log n)$

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

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

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

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$ linear search
binary search
If number behind middle door
   Return true
Else if number < middle door
   Search left half
Else if number > middle door
   Search right half
If no doors

If number behind middle door
    Return true
Else if number < middle door
    Search left half
Else if number > middle door
    Search right half
If no doors
    Return false
If number behind middle door
    Return true
Else if number < middle door
    Search left half
Else if number > middle door
    Search right half
If no doors
    Return false
If number behind doors[middle]
    Return true
Else if number < doors[middle]
    Search doors[0] through doors[middle - 1]
Else if number > doors[middle]
    Search doors [middle + 1] through doors[n - 1]
$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

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

$O(n \log n)$

$O(n)$

$O(\log n)$  binary search

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

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$

binary search
int numbers[]
string names[]
string names[]
string numbers[]
data structures
person people[]
string name;
string number;
typedef struct
{
    string name;
    string number;
}
person;
sorting
input → output
unsorted → [square] → output
unsorted → □ → sorted
6 3 8 5 2 7 4 1 → sorted
selection sort
For i from 0 to n-1
   Find smallest number between numbers[i] and numbers[n-1]
   Swap smallest number with numbers[i]
\( n + (n - 1) \)
\[ n + (n - 1) + (n - 2) \]
\[ n + (n - 1) + (n - 2) + ... + 1 \]
\[ n + (n - 1) + (n - 2) + \ldots + 1 \]

\[ n(n + 1)/2 \]
\[ n + (n - 1) + (n - 2) + \ldots + 1 \]

\[ n(n + 1)/2 \]

\[ (n^2 + n)/2 \]
\[
\begin{align*}
n + (n - 1) + (n - 2) + ... + 1 \\
n(n + 1)/2 \\
(n^2 + n)/2 \\
n^2/2 + n/2
\end{align*}
\]
\[ n + (n - 1) + (n - 2) + \ldots + 1 \]
\[ n(n + 1)/2 \]
\[ (n^2 + n)/2 \]
\[ n^2/2 + n/2 \]
\[ O(n^2) \]
$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

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

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$
For i from 0 to n-1
  Find smallest number between numbers[i] and numbers[n-1]
  Swap smallest number with numbers[i]
\Omega(n^2)
\Omega(n \log n)
\Omega(n)
\Omega(\log n)
\Omega(1)
Ω(n^2) selection sort
Ω(n \log n)
Ω(n)
Ω(\log n)
Ω(1)
\Theta(n^2)
\Theta(n \log n)
\Theta(n)
\Theta(n)
$\Theta(n^2)$

$\Theta(n \log n)$

$\Theta(n)$

$\Theta(n)$

$\Theta(\log n)$

$\Theta(1)$

selection sort
bubble sort
Repeat n-1 times
  For i from 0 to n-2
    If numbers[i] and numbers[i+1] out of order
      Swap them
\((n - 1) \times (n - 1)\)
\((n - 1) \times (n - 1)\)

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

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

$O(n \log n)$

$O(n)$

$O(\log n)$

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

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$
Repeat n-1 times
    For i from 0 to n-2
        If numbers[i] and numbers[i+1] out of order
            Swap them
    If no swaps
        Quit
$\Omega(n^2)$
$\Omega(n \log n)$
$\Omega(n)$
$\Omega(\log n)$
$\Omega(1)$
\(\Omega(n^2)\)
\(\Omega(n \log n)\)
\(\Omega(n)\)  
\(\Omega(\log n)\)
\(\Omega(1)\)

bubble sort
recursion
If no doors
   Return false
If number behind middle door
   Return true
Else if number < middle door
   Search left half
Else if number > middle door
   Search right half
If no doors
    Return false
If number behind middle door
    Return true
Else if number < middle door
    Search left half
Else if number > middle door
    Search right half
1  Pick up phone book
2  Open to middle of phone book
3  Look at page
4  If person is on page
5     Call person
6  Else if person is earlier in book
7     Open to middle of left half of book
8     Go back to line 3
9  Else if person 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 person is on page
5      Call person
6  Else if person is earlier in book
7      Open to middle of left half of book
8      Go back to line 3
9  Else if person 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 person is on page
5       Call person
6 Else if person is earlier in book
7       Open to middle of left half of book
8       Go back to line 3
9 Else if person 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 person is on page
5       Call person
6   Else if person is earlier in book
7       Search left half of book
8   Else if person is later in book
9       Search right half of book
10  Else
11     Quit
1 Pick up phone book
2 Open to middle of phone book
3 Look at page
4 If person is on page
5 Call person
6 Else if person is earlier in book
7 Search left half of book
8 Else if person is later in book
9 Search right half of book
10 Else
11 Quit
merge sort
Sort left half of numbers
Sort right half of numbers
Merge sorted halves
If only one number
   Quit
Else
   Sort left half of numbers
   Sort right half of numbers
   Merge sorted halves
If only one number
   Quit
Else
   Sort left half of numbers
   Sort right half of numbers
   Merge sorted halves
If only one number
   Quit
Else
   Sort left half of numbers
   Sort right half of numbers
   Merge sorted halves
\(O(n^2)\)

\(O(n \log n)\)

\(O(n)\)

\(O(\log n)\)

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

\(O(n \log n)\)  
merge sort

\(O(n)\)

\(O(\log n)\)

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

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

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

\( \Omega(n \log n) \)  merge sort

\( \Omega(n) \)

\( \Omega(\log n) \)

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

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

$\Theta(n)$

$\Theta(\log n)$

$\Theta(1)$
This is CS50