Time Complexity and Searching
Searching Algorithms

• Can be done one of two ways:
  1. Checking each item to see whether that specific item contained within a data structure (like an array)
  2. Using a key (unique value) to look up an item (like an ID number)

• Very common problem in CS

• For large data structures we need efficient ways to find things
Time Complexity Analysis

• We don’t consider the “speed” of an algorithm, but instead the number of steps it takes to accomplish the task.

• We need to know how many steps an algorithm will take based on the size of its input
  • In our cases, we are considering arrays
Searching for an element in an array

• Worst Case
  • The element is NOT in the array
  • Every element must be checked
  • Take n steps (where n is the size of the array)

Array

\{ 1, 2, -10, 40, 18 \}
Searching for an element in an array

• Worst Case
  • The element is NOT in the array
  • Every element must be checked
  • Take n steps (where n is the size of the array)

• The amount of time required to perform the search grows linearly with the size of the input array.

Array

\{ 1, 2, -10, 40, 18 \}
Searching for an element in an array

- **Best Case**
  - The element is ALWAYS the first element in the array
  - No elements beyond the first need to be checked

Array

\[ \{1, 2, -10, 40, 18\} \]
Searching for an element in an array

• **Best Case**
  • The element is ALWAYS the first element in the array
  • No elements beyond the first need to be checked

• The best-case time is constant no matter the size of the array

Array

\{ 1, 2, -10, 40 ,18 \}

Drew Guarnera
Binary Search

• Only works on a sorted array

• Algorithm:
  • If the array has 0 elements return false
  • Compare the middle element in the array to the target
    • If they are equal, return true
  • If the target is less than the middle element, repeat the search on the elements to the left of the middle
  • If the target is greater than the middle element, repeat the search on the element to the right of the middle
Binary Search Example

\{ -10, 1, 2, 10, 18, 40, 42 \}

Search for 42
Binary Search Example

\{ -10, 1, 2, 10, 18, 40, 42 \}

Search for 42

- Middle is 10, less than 42
- Repeat on \{ 18, 40, 42\}
Binary Search Example

\{ -10, 1, 2, 10, 18, 40, 42 \}

Search for 42

- Middle is 10, less than 42
- Repeat on \{ 18, 40, 42 \}
Binary Search Example

\{ -10, 1, 2, 10, 18, 40, 42 \}

Search for 42

- Middle is 10, less than 42
- Repeat on \{ 18, 40, 42 \}
- Middle is 40, less than 42
- Repeat on \{ 42 \}
Binary Search Example

\{ -10, 1, 2, 10, 18, 40, 42 \}

Search for 42
- Middle is 10, less than 42
- Repeat on \{ 18, 40, 42 \}
- Middle is 40, less than 42
- Repeat on \{ 42 \}
Binary Search Example

\{ -10, 1, 2, 10, 18, 40, 42 \}

Search for 42

- Middle is 10, less than 42
- Repeat on \{ 18, 40, 42 \}
- Middle is 40, less than 42
- Repeat on \{ 42 \}
- Middle is 42, return true
Binary Search Example

{ -10, 1, 2, 10, 18, 40, 42 }

Search for 42
• Middle is 10, less than 42
• Repeat on { 18, 40 ,42}
• Middle is 40, less than 42
• Repeat on { 42 }
• Middle is 42, return true

Search for -30
• Middle is 10, less than 42
Binary Search Example

\{ -10, 1, 2, 10, 18, 40, 42 \}

Search for 42
- Middle is 10, less than 42
- Repeat on \{ 18, 40, 42 \}
- Middle is 40, less than 42
- Repeat on \{ 42 \}
- Middle is 42, return true

Search for -30
- Middle is 10, greater than -30
- Repeat on \{ -10, 1, 2 \}
Binary Search Example

\{ -10, 1, 2, 10, 18, 40, 42 \}

Search for 42
• Middle is 10, less than 42
• Repeat on \{ 18, 40, 42 \}
• Middle is 40, less than 42
• Repeat on \{ 42 \}
• Middle is 42, return true

Search for -30
• Middle is 10, greater than -30
• Repeat on \{ -10, 1, 2 \}
Binary Search Example

\{ -10, 1, 2, 10, 18, 40, 42 \}

Search for 42
• Middle is 10, less than 42
• Repeat on \{ 18, 40, 42 \}
• Middle is 40, less than 42
• Repeat on \{ 42 \}
• Middle is 42, return true

Search for -30
• Middle is 10, greater than -30
• Repeat on \{ -10, 1, 2 \}
• Middle is 1, greater than -30
• Repeat on \{ -10 \}
Binary Search Example

\[
\{ -10, 1, 2, 10, 18, 40, 42 \}
\]

**Search for 42**
- Middle is 10, less than 42
- Repeat on \{18, 40, 42\}
- Middle is 40, less than 42
- Repeat on \{42\}
- Middle is 42, return true

**Search for -30**
- Middle is 10, greater than -30
- Repeat on \{-10, 1, 2\}
- Middle is 1, greater than -30
- Repeat on \{-10\}
Binary Search Example

{ -10, 1, 2, 10, 18, 40, 42 }

Search for 42
• Middle is 10, less than 42
• Repeat on { 18, 40, 42 }
• Middle is 40, less than 42
• Repeat on { 42 }
• Middle is 42, return true

Search for -30
• Middle is 10, greater than -30
• Repeat on { -10, 1, 2 }
• Middle is 1, greater than -30
• Repeat on { -10 }
• Middle is -10, greater than -30
• Repeat on { }
Binary Search Example

\{ -10, 1, 2, 10, 18, 40, 42 \}

Search for 42
• Middle is 10, less than 42
• Repeat on \{ 18, 40, 42 \}
• Middle is 40, less than 42
• Repeat on \{ 42 \}
• Middle is 42, return true

Search for -30
• Middle is 10, greater than -30
• Repeat on \{ -10, 1, 2 \}
• Middle is 1, greater than -30
• Repeat on \{ -10 \}
• Middle is -10, greater than -30
• Repeat on \{ \}
Binary Search Example

\{ -10, 1, 2, 10, 18, 40, 42 \}

**Search for 42**
- Middle is 10, less than 42
- Repeat on \{ 18, 40, 42 \}
- Middle is 40, less than 42
- Repeat on \{ 42 \}
- Middle is 42, return true

**Search for -30**
- Middle is 10, greater than -30
- Repeat on \{ -10, 1, 2 \}
- Middle is 1, greater than -30
- Repeat on \{ -10 \}
- Middle is -10, greater than -30
- Repeat on \{ \}
- Empty array, return false
Binary Search in Code

```c
bool binary_search(int target, int *array, size_t size)
```

- size = 9

- middle = size / 2 = 4
  - Odd array size give a true middle
  - Even gives a value slightly left of middle
Binary Search in Code

```c
bool binary_search(int target, int *array, size_t size)
```

• Searching for 5
  • Middle (index 4) is 27, so we need to search the left half.
  • return binary_search(target, array, middle)

• Next Search:
  ```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>-40</td>
<td>-23</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>
  ```
bool binary_search(int target, int *array, size_t size)

• Searching for 41
  • Middle (index 4) is 27, so we need to search the right half.
  • return binary_search(target, array + middle + 1, size - middle - 1)

• Next Search:
Binary Search Worst Case Analysis

<table>
<thead>
<tr>
<th>Size</th>
<th># of Calls to Binary Search (worst case)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^3$</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>
# Binary Search Worst Case Analysis

<table>
<thead>
<tr>
<th>Size</th>
<th># of Calls to Binary Search (worst case)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^3$</td>
<td>5</td>
</tr>
<tr>
<td>$2^4$</td>
<td>6</td>
</tr>
<tr>
<td>$2^5$</td>
<td>7</td>
</tr>
</tbody>
</table>
## Binary Search Worst Case Analysis

<table>
<thead>
<tr>
<th>Size</th>
<th># of Calls to Binary Search (worst case)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^3$</td>
<td>5</td>
</tr>
<tr>
<td>$2^4$</td>
<td>6</td>
</tr>
<tr>
<td>$2^5$</td>
<td>7</td>
</tr>
<tr>
<td>$2^{\log_2(n)}$</td>
<td>$\log_2(n) + 2$</td>
</tr>
<tr>
<td>$n$</td>
<td>$\log_2(n) + 2$</td>
</tr>
</tbody>
</table>
Binary Search Worst Case Analysis

<table>
<thead>
<tr>
<th>Size</th>
<th># of Calls to Binary Search (worst case)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^3$</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>$2^5$</td>
<td>32</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>$2^{\log_2(n)}$</td>
<td>$n$</td>
</tr>
<tr>
<td>$\log_2(n) + 2$</td>
<td></td>
</tr>
</tbody>
</table>

Logarithmic Time Complexity!!!!