

# Linear Search vs Binary Search



Computer Science Department  
University of Central Florida

*COP 3502 – Computer Science I*



# More Announcements

---

- Additional Announcements:
  - Best way to get in touch with me:
    - Again, for anything Program/Assignment related, contact the TA
    - but when you need to contact me, email is the best way
      - [jcazalas@cs.ucf.edu](mailto:jcazalas@cs.ucf.edu)
      - I check email often and respond quickly
    - If you want a quick response, do NOT email through webcourses, as I check that weekly, at best



# More Announcements

---

- Additional Announcements:
  - Webcourses is up!
    - What is webcourses for?
    - Only used for submitting assignments, taking quizzes, and used to store your grades
  - The course website is where ALL content for the course will be stored



# More Announcements

---

- Additional Announcements:
  - Program 1 is now assigned
  - Main purpose of Program 1:
    - WAKE UP call to those who have forgotten C
    - It is not a difficult program.
    - May take only 5 hours for some of you
    - May take upwards of 20+ for others
  
- Silver Lining:
  - If you put in the time investment to correctly do Program 1, you will be prepared for any program that is assigned during the semester



# More Announcements

---

- Additional Announcements:
  - Class attendance:
    - The reality is that those who come to class
      - And are MENTALLY present as well as physically
    - Do end up with a better grasp of the material
    - Therefore, to encourage attendance, I will randomly pick seven days throughout the semester and will take attendance on those days.
      - You will simply print and sign your name on a piece of paper
    - To avoid excuses for missing days, only five of the days will count...allowing you two free misses
    - We will announce how this will be applied to your grades later in the semester



# More Announcements

---

- Additional Announcements:
  - PPT slides:
    - A quick comment on these PPT slides
    - I hate when professors talk for 50 minutes and then when you go to study their slides for the exam, you find 8 slides in the form of an outline.
    - So unless you took notes like crazy that day, or have a great memory, you have no idea what was talked about
    - So you will notice these slides are the opposite
    - What could be done with 20 slides, I may use 60.
      - Simply because I want to give you a good study aid when it comes to the exams.



# Linear Search

---

- Searching from C-Programming class
  - In COP 3223, we studied how to find a value in an array
    - Look at each value in the array
      - Compare it to what we're looking for
    - If we see the value we are searching for,
      - Return that we've found it!
    - Otherwise, if we've iterated through the entire array and haven't located the value,
      - Return that the value isn't in the array



# Linear Search

---

- Searching from C-Programming class
  - Your code should look something like this:

```
int search(int array[], int len, int value) {  
  
    int i;  
    for (i=0; i<len; i++) {  
        if (array[i] == value)  
            return 1;  
    }  
    return 0;  
}
```





# Linear Search

- Searching from C-Programming class
  - Analyze code:
    - Clearly, if the array is unsorted, this algorithm is optimal
      - They ONLY way to be sure that a value isn't in the array is to look at every single spot of the array
      - Just like you can't be sure that you DON'T have some piece of paper or form unless you look through ALL of your pieces of paper
  - But we ask a question:
    - Could we find an item in an array faster if it were already sorted?



# Binary Search

---

- Number Guessing Game from childhood
  - Remember the game you most likely played as a child
    - I have a secret number between 1 and 100.
    - Make a guess and I'll tell you whether your guess is too high or too low.
    - Then you guess again. The process continues until you guess the correct number.
    - Your job is to MINIMIZE the number of guesses you make.



# Binary Search

---

- Number Guessing Game from childhood
  - What is the first guess of most people?
    - 50.
  - Why?
    - No matter the response (too high or too low), the most number of possible values for your remaining search is 50 (either from 1-49 or 51-100)
    - Any other first guess results in the risk that the possible remaining values is greater than 50.
      - Example: you guess 75
      - I respond: too high
      - So now you have to guess between 1 and 74
        - 74 values to guess from instead of 50



# Binary Search

---

- Number Guessing Game from childhood
  - Basic Winning Strategy
    - Always guess the number that is halfway between the lowest possible value in your search range and the highest possible value in your search range
  
- Can we now adapt this idea to work for searching for a given value in an array?



# Binary Search

## ■ Array Search

- We are given the following sorted array:

<b>index</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>
<b>value</b>	<b>2</b>	<b>6</b>	<b>19</b>	<b>27</b>	<b>33</b>	<b>37</b>	<b>38</b>	<b>41</b>	<b>118</b>

- We are searching for the value, 19
- So where is halfway between?
  - One guess would be to look at 2 and 118 and take their average (60).
  - But 60 isn't even in the list
  - And if we look at the number closest to 60
    - It is almost at the end of the array



# Binary Search

---

- Array Search

- We quickly realize that if we want to adapt the number guessing game strategy to searching an array, we **MUST** search in the middle INDEX of the array.
- In this case:
  - The lowest index is 0
  - The highest index is 8
  - So the middle index is 4



# Binary Search

## ■ Array Search

### ■ Correct Strategy

- We would ask, “is the number I am searching for, 19, greater or less than the number stored in index 4?”
  - Index 4 stores 33
- The answer would be “less than”
- So we would modify our search range to in between index 0 and index 3
  - Note that index 4 is no longer in the search space
- We then continue this process
  - The second index we’d look at is index 1, since  $(0+3)/2=1$
  - Then we’d finally get to index 2, since  $(2+3)/2 = 2$
  - And at index 2, we would find the value, 19, in the array



# Binary Search

## ■ Binary Search code:

```
int binsearch(int a[], int len, int value) {  
  
    int low = 0, high = len-1;  
    while (low <= high) {  
        int mid = (low+high)/2;  
        if (value < a[mid])  
            high = mid-1;  
        else if (value > a[mid])  
            low = mid+1;  
        else  
            return 1;  
    }  
  
    return 0;  
}
```





# Binary Search

---

- Binary Search code:
  - At the end of each array iteration, all we do is update either low or high
  - This modifies our search region
    - Essentially halving it



# Binary Search

## ■ Efficiency of Binary Search

### ■ Analysis:

- Let's analyze how many comparisons (guesses) are necessary when running this algorithm on an array of  $n$  items

First, let's try  $n = 100$

- After 1 guess, we have 50 items left,
- After 2 guesses, we have 25 items left,
- After 3 guesses, we have 12 items left,
- After 4 guesses, we have 6 items left,
- After 5 guesses, we have 3 items left,
- After 6 guesses, we have 1 item left
- After 7 guesses, we have 0 items left.



# Binary Search

---

## ■ Efficiency of Binary Search

### ■ Analysis:

#### ■ Notes:

- The reason for the last iteration is because the number of items left represent the number of other possible values to search
  - We need to reduce this to 0.
- Also, when  $n$  is odd, such as when  $n=25$ 
  - We search the middle element, # 13
  - There are 12 elements smaller than 13
  - And 12 elements bigger than 13
  - This is why the number of items is slightly less than  $\frac{1}{2}$  in those cases



# Binary Search

- Efficiency of Binary Search
  - Analysis:
    - General case:
      - After 1 guess, we have  $n/2$  items left
      - After 2 guesses, we have  $n/4$  items left
      - After 3 guesses, we have  $n/8$  items left
      - After 4 guesses, we have  $n/16$  items left
      - ...
      - After  $k$  guesses, we have  $n/2^k$  items left



# Binary Search

---

## ■ Efficiency of Binary Search

### ■ Analysis:

- General case:
- So, after  $k$  guesses, we have  $n/2^k$  items left
- The question is:
  - How many  $k$  guesses do we need to make in order to find our answer?
  - Or until we have one and only one guess left to make?
- So we want to get only 1 item left
- If we can find the value that makes the above fraction equal to 1, then we know that in one more guess, we'll narrow down the item



# Binary Search

## ■ Efficiency of Binary Search

### ■ Analysis:

- General case:
- So, after  $k$  guesses, we have  $n/2^k$  items left
  - Again, we want only 1 item left
  - So set this equal to 1 and solve for  $k$

$$\frac{n}{2^k} = 1$$

$$n = 2^k$$

$$k = \log_2 n$$

- This means that a binary search roughly takes  $\log_2 n$  comparisons when searching in a sorted array of  $n$  items



# Binary Search

## ■ Efficiency of Binary Search

### ■ Analysis:

- Runs in logarithmic ( $\log n$ ) time
- This is MUCH faster than searching linearly
- Consider the following chart:

<u>n</u>	<u>log n</u>
8	3
1024	10
65536	16
1048576	20
33554432	25
1073741824	30

- Basically, any  $\log n$  algorithm is SUPER FAST.

# Linear Search vs Binary Search



Computer Science Department  
University of Central Florida

*COP 3502 – Computer Science I*