

Advanced Topics: Searching and Sorting



Colorado State University

Department of Computer Science

Slides Originally Created by Albert Lionelle (Albert.Lionelle@colostate.edu),
updated by Marcia Moraes (marcia.moraes@colostate.edu)

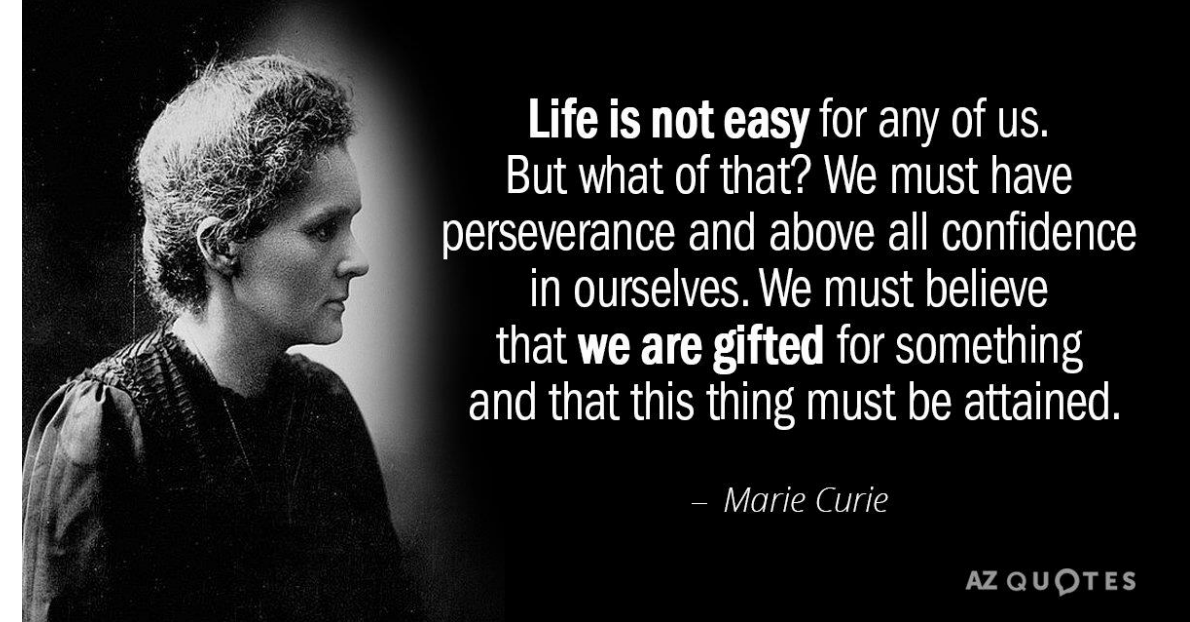
Announcements

TODO Reminders:

Readings are due **before** lecture

- Reading 25 (zybooks) – you should have already done that 😊
- Lab 16
- Reading 26 (zyBooks) you should have already done that 😊
- Lab 17
- RPA 12

Keep practicing your RPAs in a spaced and mixed manner 😊



Help Desk

Day	Time : Room
Monday	2 PM - 5 PM : CSB 120
Tuesday	6 PM - 8 PM : Teams
Wednesday	3 PM - 5 PM : CSB 120
Thursday	6 PM - 8 PM : Teams
Friday	3 PM - 5 PM : CSB 120

Linear Search

k
i
n
n
i
k
i
n
n
i
k



stops searching,
returns 1
stops searching,
returns 2

Is there a quicker way?

- You have already done it!
- Searches an array
 - If item exists – return location
 - else return -1

```
public static int linearSearch(char key, char[] array) {  
    for(int i = 0; i < array.length; i++) {  
        if(key == array[i]) return i;  
    }  
    return -1;  
}
```

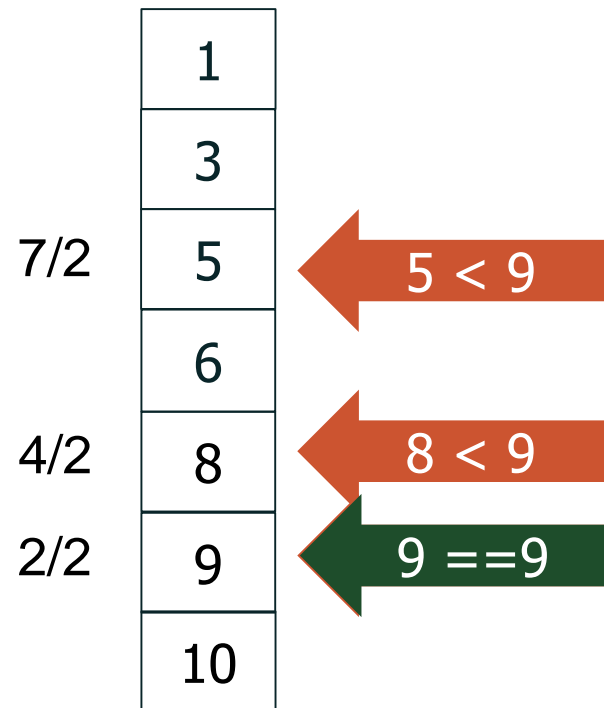
```
public static void main(String[] args) {  
    char[] array = new char[]{'k', 'i', 'n', 'n', 'i', 'k', 'i', 'n', 'n', 'i', 'k'};  
    System.out.println(linearSearch('i', array));  
    System.out.println(linearSearch('n', array));  
}
```

Binary Search

- Similar to guess number game
 - I will choose a number between 0 and 100, with 0 and 100 included
 - Now you try to guess, what is the number that I am thinking 😊
 - What strategy you should use?
- Guess the middle number
 - If it is the middle – you found!
 - If the number is less than the middle – divide again in the middle but just for the first half part (0-49)
 - If the number is greater than the middle – divide again in the middle but just for the second half part (51-100)
 - Repeat the process until you find the number!

Binary Search

- if the elements are **in order / sorted**
- why go in order?
 - start in the middle!
 - example: looking for 9



Very Fast – if Sorted!

We can quickly search by 'dividing' the array into two parts each time

We ask ourselves, is the key 'larger' or 'smaller' each time, until we find it!

Binary Search

- Lets implement a recursive binary search!
 - Our method will receive an ordered array of ints, a number that we are looking for, index for the beginning of the search, index for the end of the search
 - Method should return the index where that element is in our array
 - Or -1 if the number is not in our array
 - We will need to adapt our strategy and transform it into a program 😊!
- Guess the middle number
 - If it is the middle – you found!
 - If the number is less than the middle – divide again in the middle but just for the first half part (0-49)
 - If the number is greater than the middle – divide again in the middle but just for the second half part (51-100)
 - Repeat the process until you find the number!

Bubble Sort (Sorting by Exchange)



Kenneth Inversion

- "Bubbles" up the largest numbers
- Takes the first number
 - if the next is less, swap it so the larger moves up
 - if the next is more, shift to that number, and continue
- Pass 1
 - the largest number ends up at the end
- Pass 2
 - the second largest number ends up at the end
- and so on

Look at 3 [3, 2, 1, 5, 8]

[2, 3, 1, 5, 8]

[2, 1, 3, 5, 8]

[2, 1, 3, 5, 8]

[2, 1, 3, 5, 8]

Start again [2, 1, 3, 5, 8]

[1, 2, 3, 5, 8]

Should never use Bubble Sort

Bubble Sort (Sorting by Exchange)

- <https://visualgo.net/en/sorting>
- Now let's implement the Bubble Sort!



Kenneth Inverson

Selection Sort

- Searches array for **lowest** value
- Moves that to the start index
 - repeats
 - incrementing index

<https://visualgo.net/en/sorting>

Let's implement the selection sort!

[3, 2, 1, 5, 8]

[**1**, 2, **3**, 5, 8]

Then checks 2-8

Example 2:

[3, 5, 1, 9, 8]

[**1**, 5, **3**, 9, 8]

[1, **3**, **5**, 9, 8]

[1, 3, 5, **8**, **9**]

[1, 3, 5, 8, 9]

The Big-O

- Both selection and bubble sort
 - 10 elements, it can look at all 10 ten times!
 - N elements N times
 - We call this - $O(n^2)$
- Linear Search
 - we only look at each element once
 - we call this $O(n)$
- Binary Search
 - we only look at reducing halves of elements
 - we call this $O(\log n)$
- You will learn this more later in 165, 220 and 250
 - Why? Knowing the most efficient algorithm for different situations matter
 - CS 320 really dives into how to speed up programs by knowing algorithms

Practice - Attendance

- Given the following list write each 'step' (array) of the bubble sort

[3, 2, 5, 1, 8]

Practice - Attendance

- Given the following list write each 'step' (array) of the bubble sort

[3, 2, 5, 1, 8]

[2, 3, 5, 1, 8]

[2, 3, 1, 5, 8]

[2, 1, 3, 5, 8]

[1, 2, 3, 5, 8]