



# CS 113 – Computer Science I

## Lecture 20 – Searching

---

Adam Poliak

03/30/2023

# Announcements

- HW07
  - Due Monday 04/03
- Midsemester feedback
- Office hours today (3-4:45pm)
- Schedule:
  - Thursday 04/06 – NO CLASS
  - Tuesday 04/11 – remote class
  - Thursday 04/13 – Midterm

# Outline

- Reviewing relationships!
- Searching
- Sorting - possible

# Subtyping vs Subclassing

## Interfaces (subtyping)

- **implements**
- Guarantees same types have same functions
  - Though the same functions are implemented differently
- A class can implement multiple interfaces
- An interface can extend another interface

## Inheritance (subclassing)

- **extends**
- Reuses implementations
- Consequences:
  - Dependent on base class
  - Changes in superclass affects all subclasses
  - Can re-use code inside classes
- A class can extend just one parent class

# Outline

- Reviewing relationships!
- **Searching**
- Sorting - possible

# Searching

Finding whether an item is in a collection

Applications:

- Specific email in an inbox
- Word in a document
- Course in a list of course offerings
- Professor is on RateMyProfessor
- ...

# Common search problems

- Is an item in an array?
  - Returns: True or False
- Where in an array is the item?
  - Returns: the index (an integer)
    - Standard: -1 if the item is not found
- How many times does the item appear in an array?
  - Returns: a count (an integer)
- What is the min, max, or average value in an array?
  - Returns: the value (double)

# Linear Search

These previous approaches are examples of linear search

Check each item in a collection one by one

Why is this call linear search?

Time it takes to search increases *linearly* with the size of the list



# Linear Search

What happens (in terms of speed) when the list is very large?

The search becomes slower

In what cases do we do the most work (i.e. perform the most comparisons)?

When the item is not in the list

In what cases do we do the least amount of work?

When the item is the first element in the list

# Binary Search

If we could change the list, is there a way to search more efficiently?

Yes, if the list is sorted

# Guessing game – in class exercise

Pair up:

- Person A chooses a number between 1 and 100
- Person B guesses the number
- Until the guess is correct:
  - Person A tells whether the guess is too high or too low
  - Person B guesses again

# Binary Search

Assuming list is sorted in ascending order

High-level Algorithm:

- Step 1: Find the midpoint of the list:
  - if the search value is at the midpoint – we are done!
  - if the value we are searching for is above the midpoint,
    - Search right: cut our list in half and repeat step 1 with the right half of the list
  - If the value we are searching for is below the midpoint
    - Search left: cut out list in half and repeat step 1 with the left half of the list

# Binary Search – Initial Values

lowIndex, highIndex, midIndex

lowIndex = 0

highIndex = length of the array - 1

midIndex =  $\frac{\text{lowIndex} + \text{highIndex}}{2}$

# Binary Search – Initial Values

lowIndex, highIndex, midIndex

If value at midIndex == searchValue:

Success!

If value at midIndex < searchValue:

lowIndex = midIndex + 1

update midIndex

If value at midIndex > searchValue:

highIndex = midIndex - 1

update midIndex















# Binary search

String[] ls = {<sup>0</sup>-20, <sup>1</sup>-4, <sup>2</sup>44, <sup>3</sup>58, <sup>4</sup>99, <sup>5</sup>145}

Search for 30

low	mid	high	ls[mid]
0	2	5	44
0	0	1	-20
1	1	1	-4

# Binary search

String[] ls = {<sup>0</sup>-20, <sup>1</sup>-4, <sup>2</sup>44, <sup>3</sup>58, <sup>4</sup>99, <sup>5</sup>145}

Search for 30

low	mid	high	ls[mid]
0	2	5	44
0	0	1	-20
1	1	1	-4
2		1	Not found!









# Binary search w/ Strings

0 1 2 3 4 5 6 7

```
String[] ls = {"bear", "bird", "bug", "cat", "cow", "dog", "fish", "lion"};
```

Search for "cow"

low	mid	high	ls[mid]
0	3	7	"cat"
4	5	7	"dog"
4	4	4	"cow"!







# Binary search

0 1 2 3 4 5 6 7  
String[] ls = {"bear", "bird", "bug", "cat", "cow", "dog", "fish", "lion"};

Search for "elephant"

low	mid	high	ls[mid]
0	3	7	"cat"
4	5	7	"dog"
6	6	7	"fish"

# Binary search

0 1 2 3 4 5 6 7  
String[] ls = {"bear", "bird", "bug", "cat", "cow", "dog", "fish", "lion"};

Search for "elephant"

low	mid	high	ls[mid]
0	3	7	"cat"
4	5	7	"dog"
6	6	7	"fish"
6		6	

# Sorting