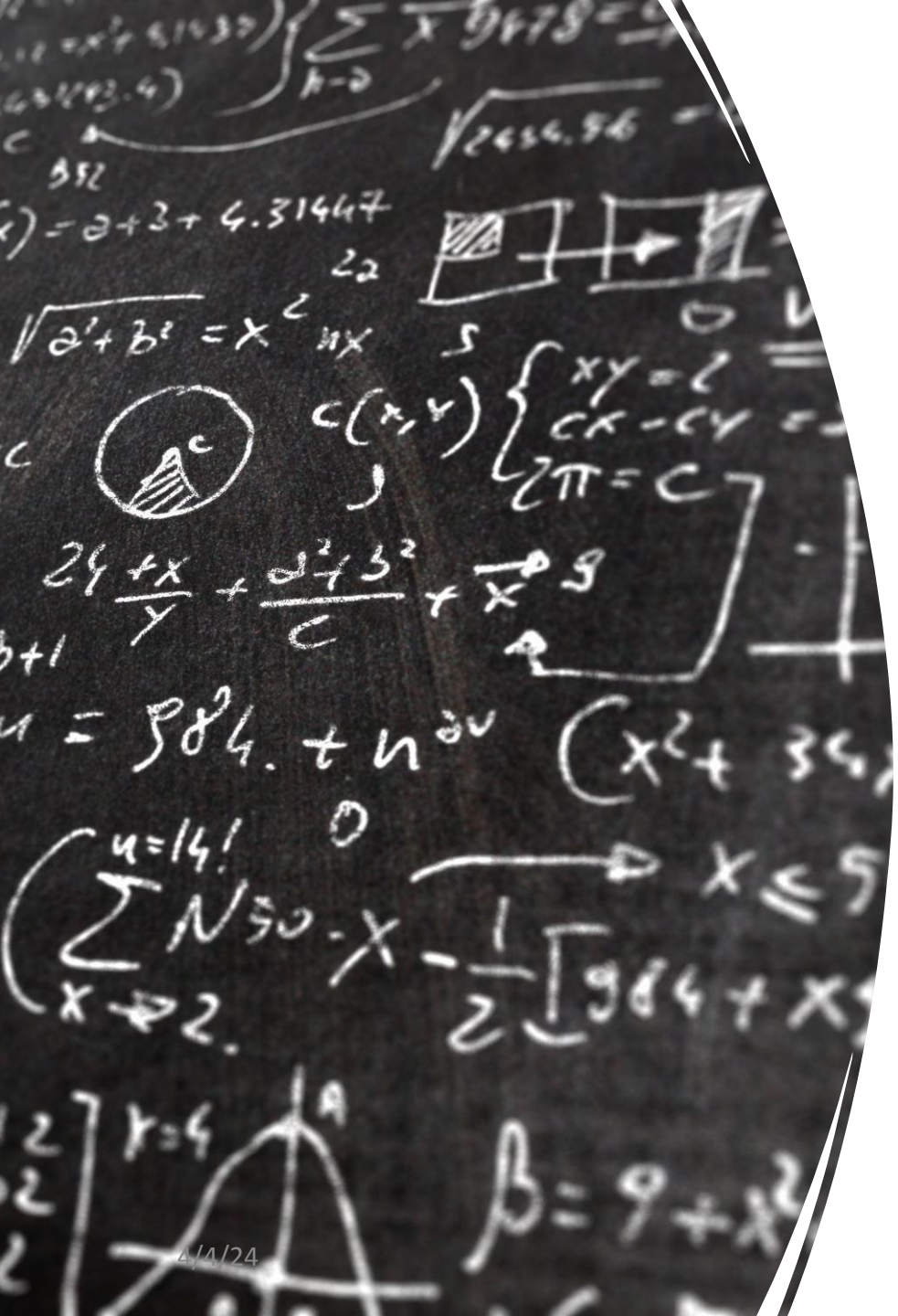


CS 113 – Computer Science I

Lecture 19 – Searching!

Thursday 4/4/2024

Announcements

HW06 and Lab06 due tonight

HW07 - class design will be released tonight.

Due 4/11

Outline

- Reading in a CSV
- Interfaces review
- Searching

Processing CSVs

CSV files

- Comma separated values
- Each line corresponds to a row, and within each line, the values for each column are separated by commas
- First line typically contains “headers” describing each column
- courses.csv

Reading CSVs

- Method you'll need: string `split`()
  - split a string into an array of substrings based on a specified delimiter
  - Split.java

- Now let's read in a CSV
  - ReadCsv.java

Interfaces review

Interfaces

- An interface is a contract - A set of shared methods that users **must** implement
- create a program to calculate the area of different shapes, such as circles, rectangles, triangles etc.
- For each shape, you should be able to print the shape name and area
- Every time someone adds a new shape, they **must** include the methods for getName() and getArea()

Interfaces

An interface is not a class!

A class is what an object **is**

An interface is what an object **does**

- can not be instantiated
- no constructors
- incomplete methods

Interfaces Example

Vehicle interface

Review: Comparing objects

Recall: variables for objects are references (pointers) to objects

==

compares whether the two references are the same

Object.equals(Object obj)

compares two objects

Every (base) class should implement this

Searching

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)

Searching in Bank

Do we have an account for a specific person/name

Searching in our array of Animals

How many animals in our collection are less than 5lbs?

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

Guessing game

- I chooses a number between 1 and 100
- you guess the number
- Until the guess is correct:
    - I tell you whether the guess is too high or too low
    - You guess again

How many guesses did we have to make?

Binary Search

Let’s use this idea to implement a faster search.

Assume we have a list of ***sorted*** integers.

Is there a way to search more efficiently?

Binary Search – Initial Values

Where should we start our search?

Think of our game!

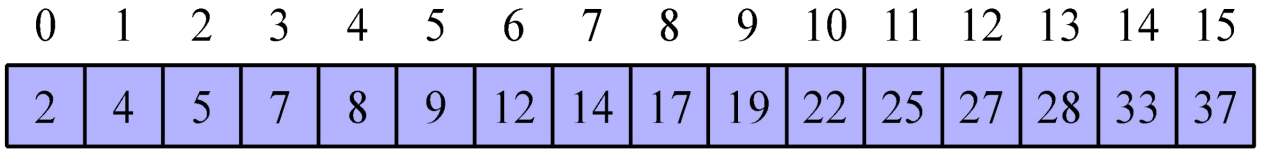
Binary Search

1. Calculate midpoint

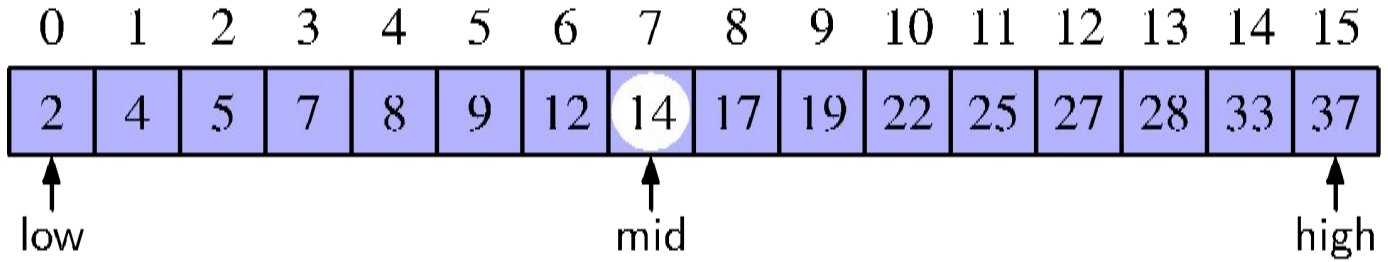
2. Compare the value at the midpoint with the target value
   a. if equal:
      i. return index
   b. if target value < midpoint value:
      i. **search** the left portion of the list
   c. if target value > midpoint value:
      i. **search** the right portion of the list

Binary Search

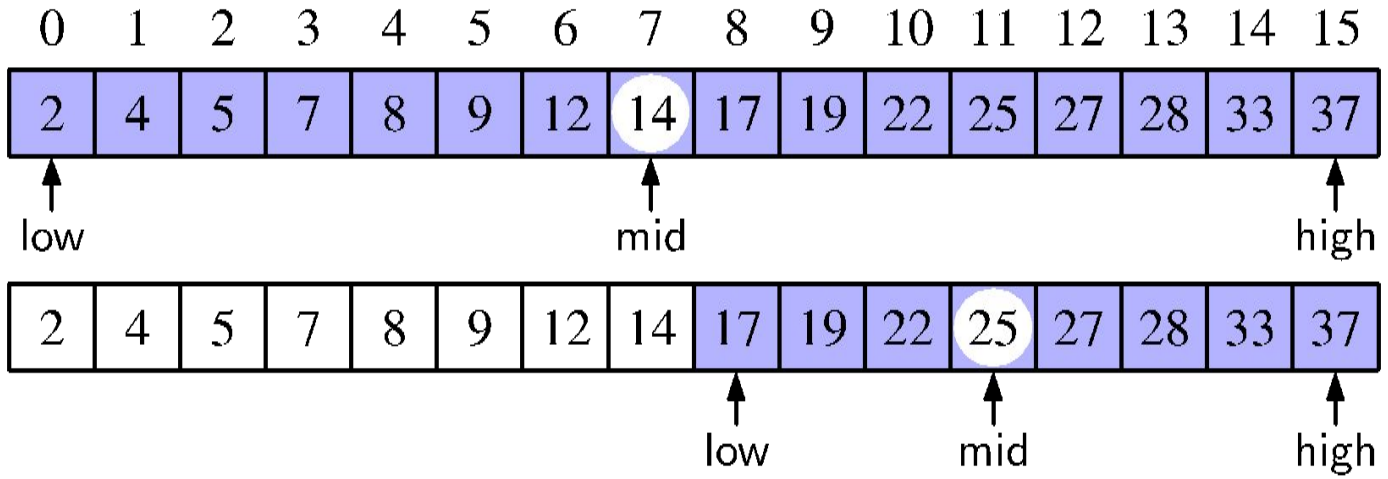
Search for an integer (22) in an ordered list



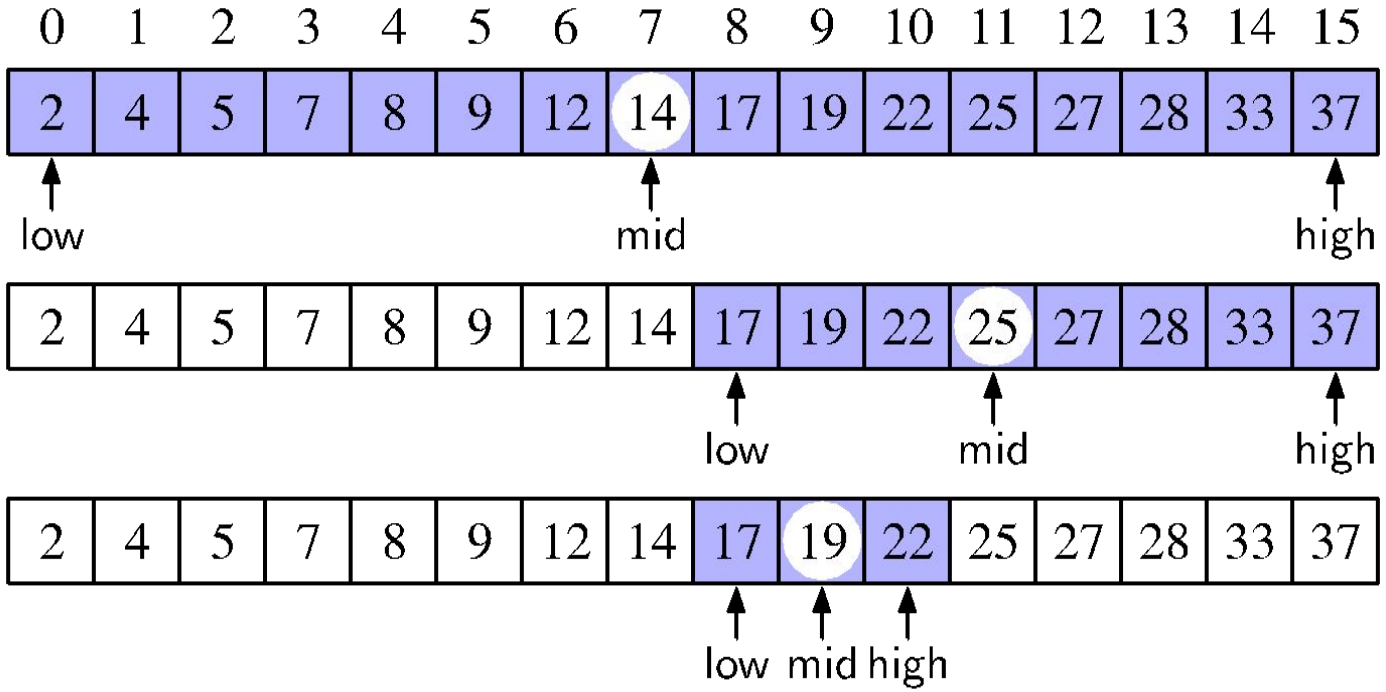
target = 22


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
2 4 5 7 8 9 12 14 17 19 22 25 27 28 33 37
low
mid
high

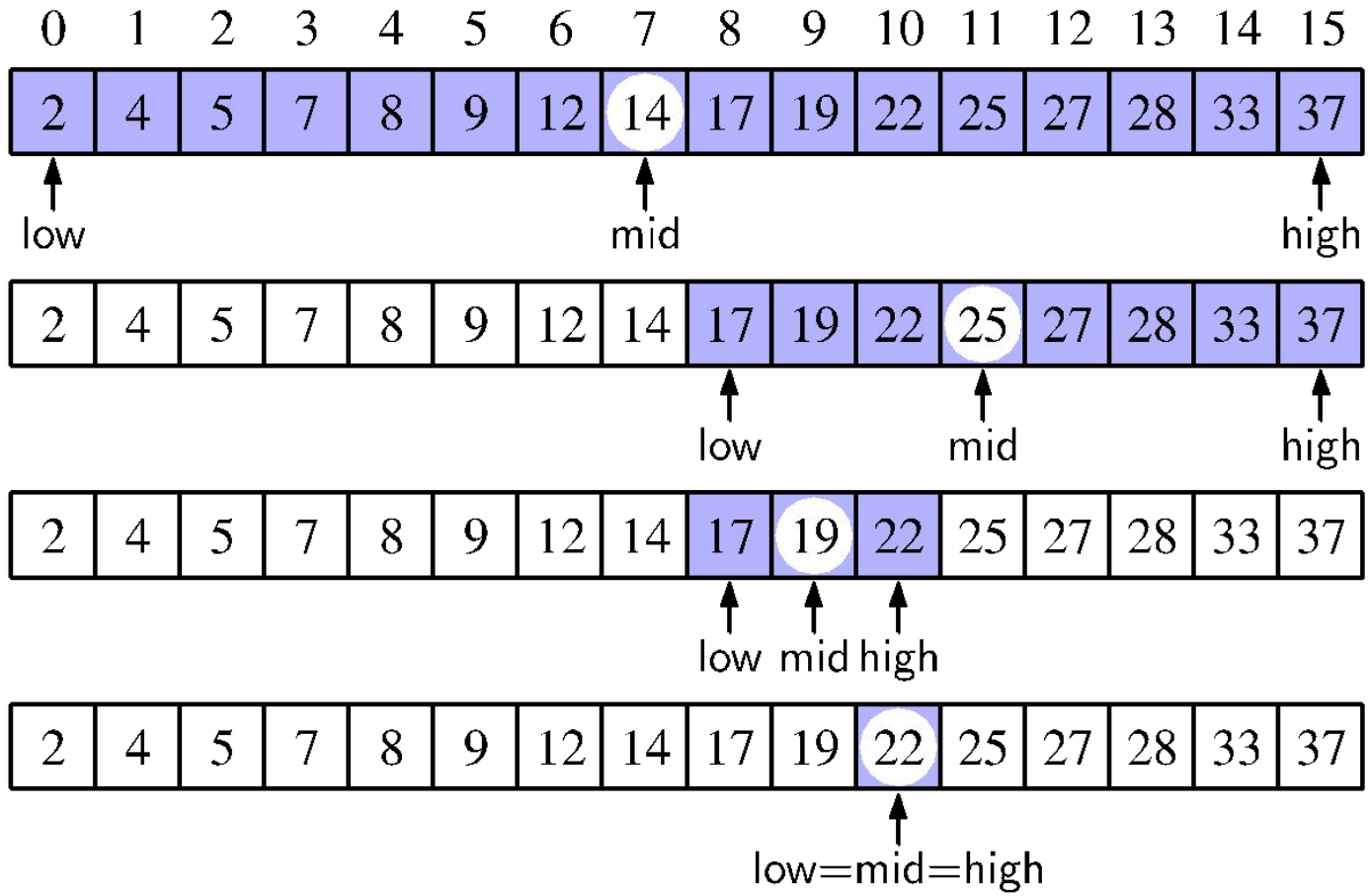
target = 22


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
2 4 5 7 8 9 12 14 17 19 22 25 27 28 33 37
low
mid
high
2 4 5 7 8 9 12 14 17 19 22 25 27 28 33 37
low
mid
high

target = 22


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
2 4 5 7 8 9 12 14 17 19 22 25 27 28 33 37
low
mid
high
2 4 5 7 8 9 12 14 17 19 22 25 27 28 33 37
low
mid
high
2 4 5 7 8 9 12 14 17 19 22 25 27 28 33 37
low mid high

target = 22


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
2 4 5 7 8 9 12 14 17 19 22 25 27 28 33 37
low
mid
high
2 4 5 7 8 9 12 14 17 19 22 25 27 28 33 37
low
mid
high
2 4 5 7 8 9 12 14 17 19 22 25 27 28 33 37
low mid high
2 4 5 7 8 9 12 14 17 19 22 25 27 28 33 37
low=mid=high

Binary search

String[] ls = {-20, -4, 44, 58, 99, 145}

Search for 99

low	mid	high	ls[mid]

Binary search w/ Strings

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

Search for “cow”

low	mid	high	ls[mid]

Binary Search

- efficient search in a sorted list
- can be implemented **recursively**

Search steps:

1. Calculate midpoint
2. Compare the value at the midpoint with the target value
   a. if equal:
      i. return index
   b. if target value < midpoint value:
      i. **search** the left portion of the list
   c. if target value > midpoint value:
      i. **search** the right portion of the list

Binary Search Implementation