CMSC201
Computer Science I for Majors

Lecture 21 – Searching and Sorting

Last Class We Covered

- Hexadecimal numbers
  - Will be on the final exam!
- Printing in color
  - Will not be on the final exam!

Any Questions from Last Time?

Today's Objectives

- To learn about some sorting algorithms
  - Bubble Sort
  - Selection Sort
  - Quicksort
- To learn about searching algorithms
  - Linear search
  - Binary search

Sorting

Sorting Algorithms

- Sorting algorithms put the elements of a list in a specific order

- A sorted list is necessary to be able to use certain other algorithms
- Like search algorithms!
  - There must be an order to be able to search – sorting once means we can search quickly forever

Sorting Algorithms

- There are many different ways to sort a list
- What method would you use?

- Now imagine you can only look at ***at most*** two elements at a time
  - What method would you use now?

- Computer science has a number of commonly used sorting algorithms

Bubble Sort

Bubble Sort Algorithm

- Let's take a look at a common sorting method!

1. We look at the first pair of items in the list, and if the first one is bigger than the second one, we swap them
2. Then we look at the second and third one and put them in order, and so on
3. Once we hit the end of the list, we start over at the beginning
4. Repeat until the list is sorted!

Bubble Sort Example

```
[ 4, 8, 1, 10, 13, 14, 6]
```

First pass:

4 and 8 are in order

8 and 1 should be swapped:

```
[ 4, 1, 8, 10, 13, 14, 6]
```

8 and 10 are in order

10 and 13 are in order

13 and 14 are in order

6 and 14 should be swapped:

```
[ 4, 1, 8, 10, 13, 6, 14]
```

Bubble Sort Example (Cont)

```
[ 4, 1, 8, 10, 13, 6, 14]
```

Second pass:

4 and 1 should be swapped:

```
[ 1, 4, 8, 10, 13, 6, 14]
```

4 and 8 are in order

8 and 10 are in order

10 and 13 are in order

13 and 6 should be swapped:

```
[ 1, 4, 8, 10, 6, 13, 14]
```

13 and 14 are in order

Bubble Sort Example (Cont)

```
[ 1, 4, 8, 10, 6, 13, 14]
```

Third pass:

10 and 6 should be swapped:

```
[ 1, 4, 8, 6, 10, 13, 14]
```

Fourth pass:

8 and 6 should be swapped:

```
[ 1, 4, 6, 8, 10, 13, 14]
```

Bubble Sort Video

Video from https://www.youtube.com/watch?v=lyZQPjUT5B4

Selection Sort

Selection Sort Algorithm

- Here is a very simple way of sorting a list:

1. Find the smallest number in a list
2. Move that to the end of a new list
3. Repeat until the original list is empty

- Unfortunately, it's also pretty slow!

Selection Sort Video

Video from https://www.youtube.com/watch?v=Ns4TPTC8whw

Quicksort

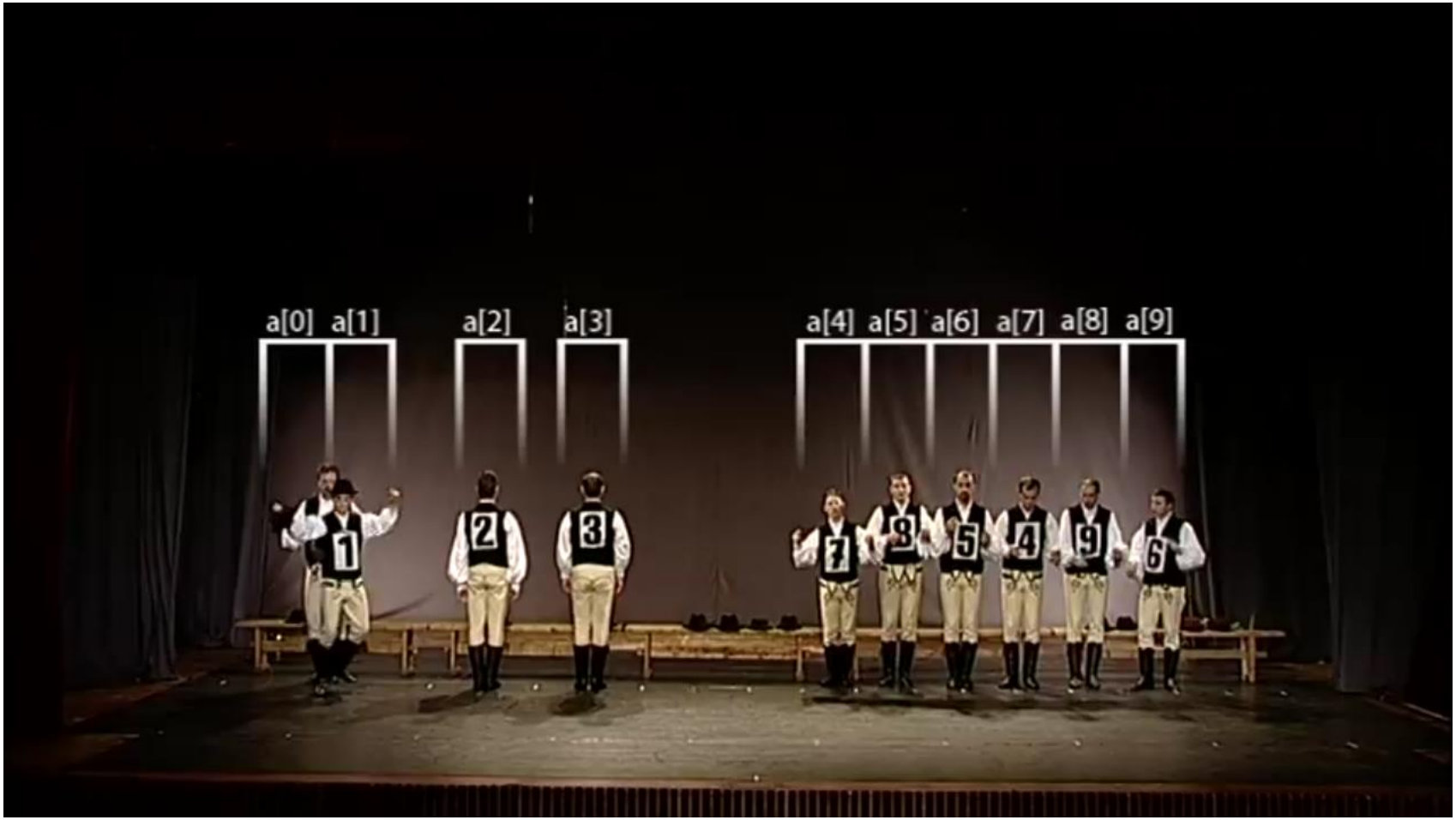
Quicksort Algorithm

- Here's one more method:

1. Start with any number (the first one works)
2. Put everything less than that number on the left of it and everything greater than it on the right of it
3. Quicksort the left side and the right side

- Does this method remind you of anything?

Quicksort Video



Video from https://www.youtube.com/watch?v=ywWBy6J5gz8

Search

Motivations for Searching

- Want to know if something exists
  - Python can do this for us!

- Want to know where something exists
  - Python can actually do this for us too!
  - `raceWinners.index("#718")`

- But **how** does Python does this?

Exercise: `find()`

- Write a function that takes a list and a variable and returns the index of the variable in the list
  - If it's not found, return -1
  - You can't use `.index()`!

```
def find(searchList, var)
```

Exercise: `find()` Solution

```
def find(searchList, var):
    for i in range(len(searchList)):
        if searchList[i] == var:
            return i

    # outside the loop, means that
    # we didn't find the variable
    return -1
```

Linear Search

- You just programmed up a search function!

- This algorithm is called ***linear search***
- It's a common, fundamental algorithm in CS

- It's especially useful when our information isn't in a sorted order
  - But it isn't very fast

Searching Sorted Information

- Now, imagine we're looking for information in something sorted, like a phone book
- We know someone's name (it's our "variable"), and want to find their number in the book

- What is a good method for locating their phone number?
  - Think about how a person would do this

Algorithm in English

- Open the book midway through.
  - If the person's name is **on** the page you opened to
    - You're done!
  - If the person's name is **after** the page you opened to
    - Tear the book in half, throw the first half away and repeat this process on the second half
  - If the person's name is **before** the page you opened to
    - Tear the book in half, throw the second half away and repeat this process on the first half
- This is rough on the phone book, but you'll find the name!

Binary Search

Binary Search

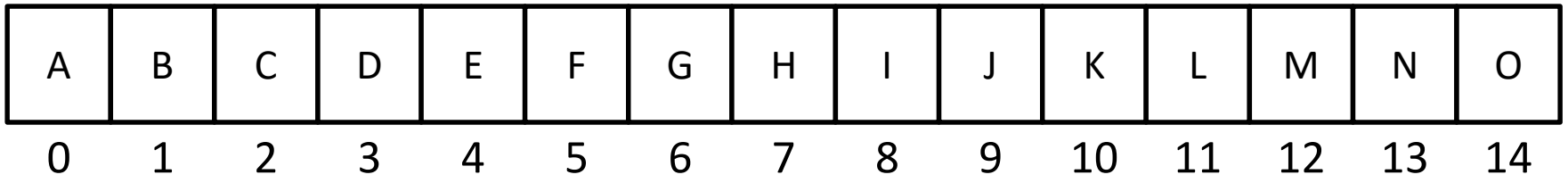
- The algorithm we just demonstrated is better known as ***binary search***

- Binary search is a ***divide and conquer*** algorithm
  - We've talked about these before, remember?

- Binary search is only usable on sorted lists
  - Why?

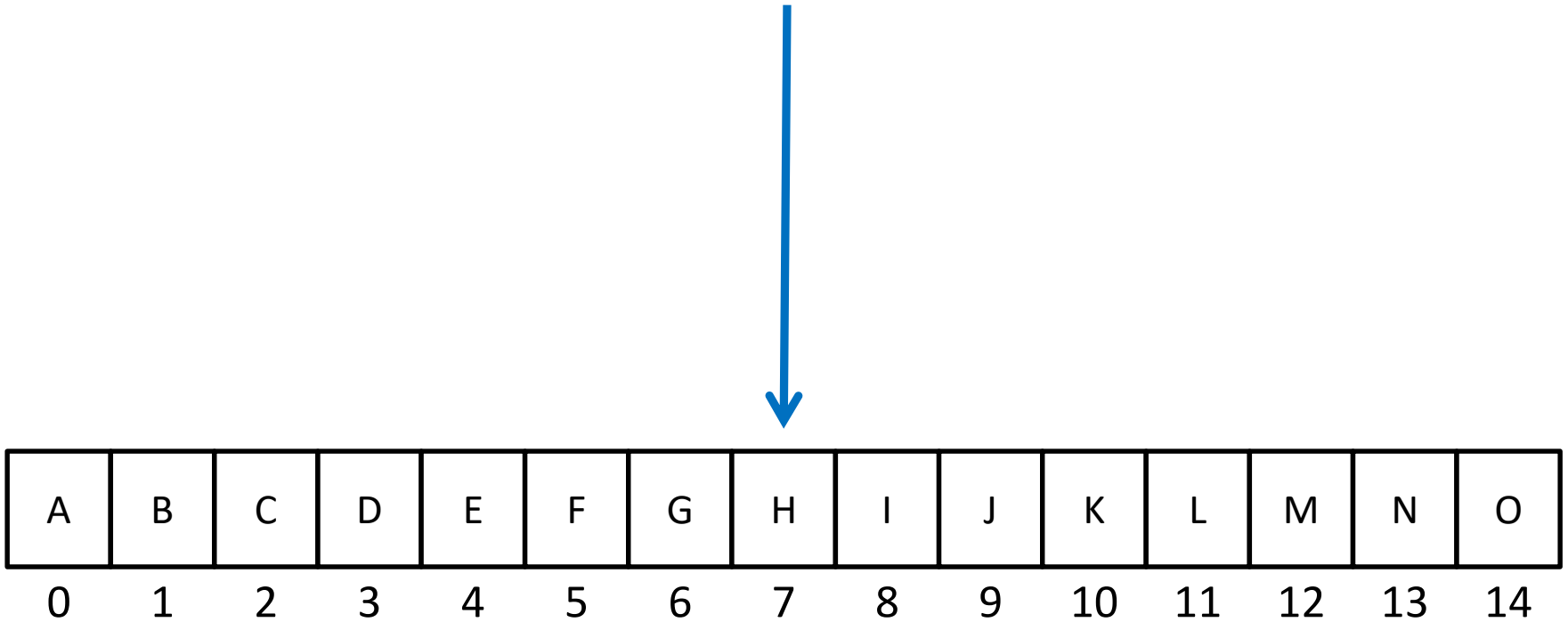
Binary Search Example

- Find the letter “J” using binary search



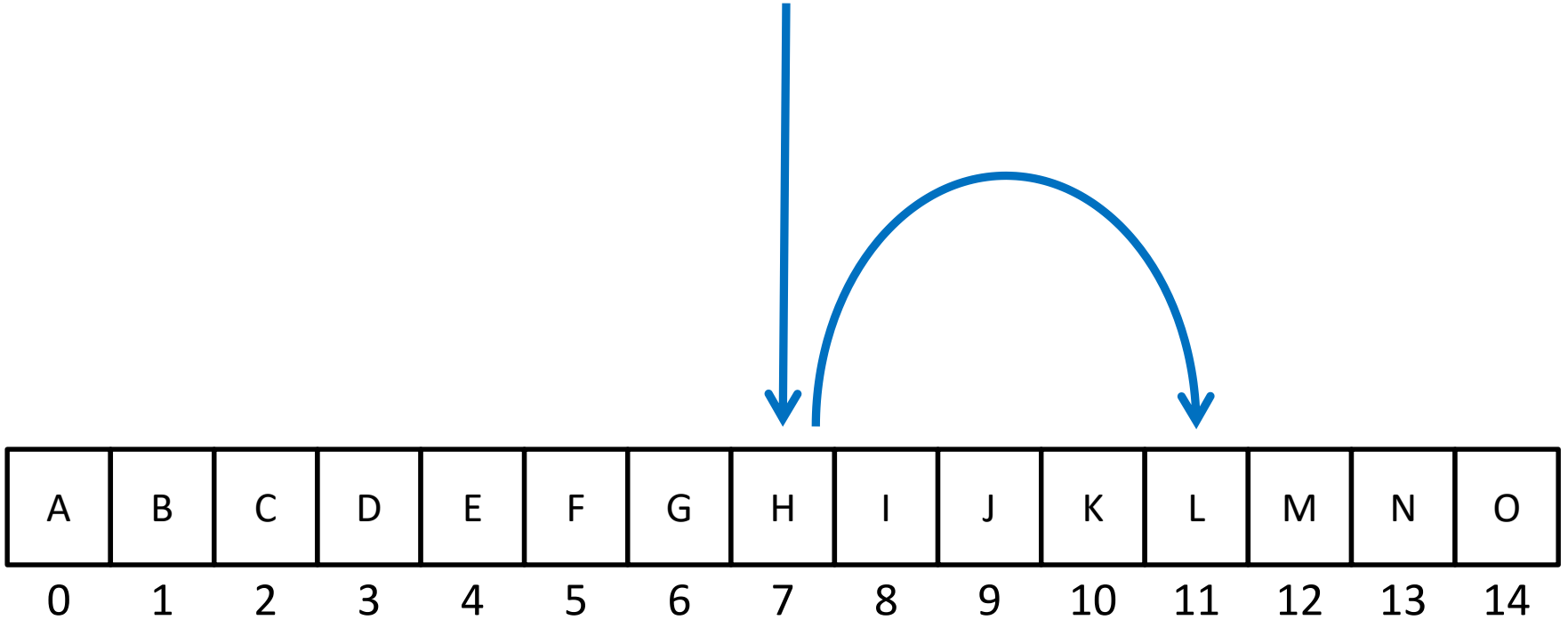
Binary Search Example

- Find the letter “J” using binary search



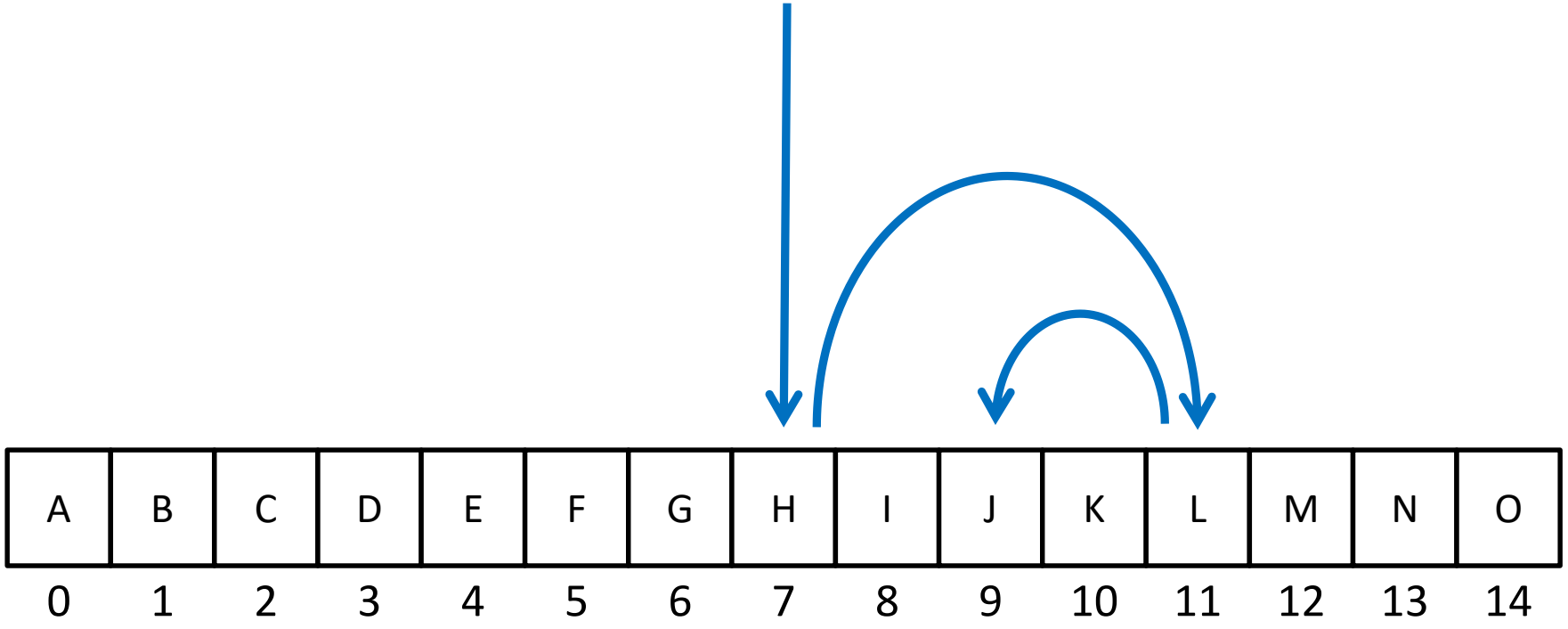
Binary Search Example

- Find the letter “J” using binary search



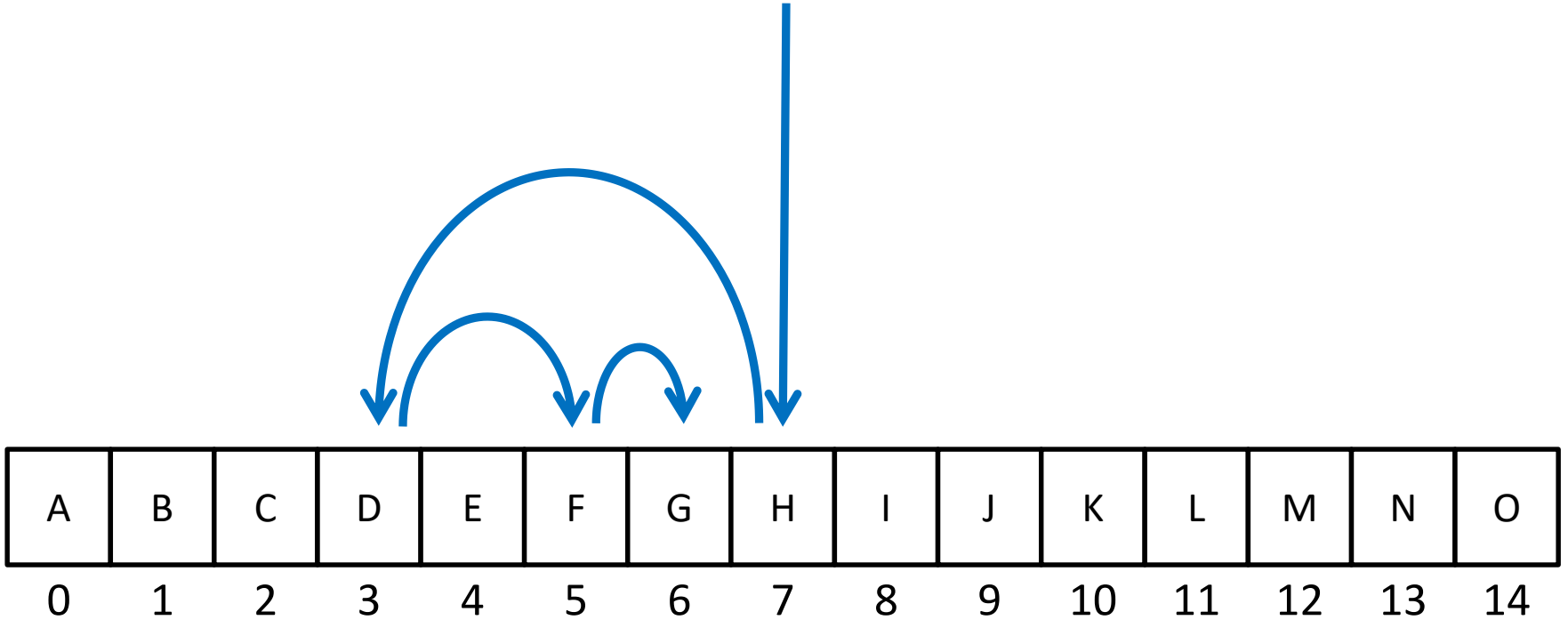
Binary Search Example

- Find the letter “J” using binary search



Binary Search Example

- Find the letter “G” using binary search



Solving Binary Search

- Binary search is a problem that can be broken down into
  - Something simple (breaking a list in half)
  - A smaller version of the original problem (searching that half of the list)

- That means we can use ... recursion!

Time for…

LIVECODING!!!

Exercise: Recursive Binary Search

- Write a recursive binary search!

- To make the problem slightly easier, make it "checking to see if something is in a sorted list"
  - Simply return True or False to answer this

- If there's no "middle" of the list, we'll just look at the lower of the two "middle" indexes

Exercise: Recursive Binary Search

- Write a recursive binary search!
- Remember to ask yourself:
  - What is our base case(s)?
  - What is the recursive step?

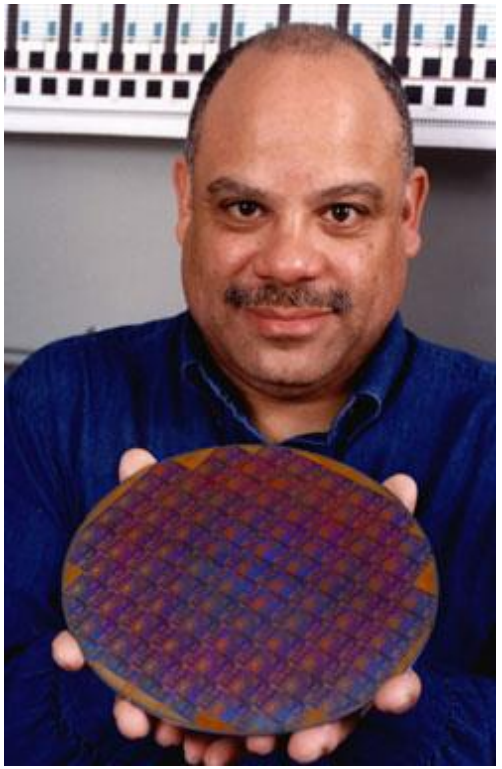
```
def binarySearch(theList, item):
```

- A hint: in order to get the number at the middle of the list, use this bit of code:

```
myList[len(theList) // 2]
```

Daily CS History

- Mark Dean
  - Holds 3 of 9 patents for the IBM PC
  - Part of team that developed the ISA bus (used to connect I/O devices)
  - Led design of the 1-gigahertz chip
  - Computing visionary
    - Predicted the tablet computer in 1999
    - https://web.archive.org/web/20121020094411/http://www.usnews.com/usnews/culture/articles/000103/archive_034033.htm



Announcements

- Project 3 design is due on Friday, May 3rd
  - Project itself is due on Friday, May 10th
- Survey #3 out on Monday, May 6th

- Course evaluations are (not out yet)
- Final exam is when?
  - Friday, May 17th from 6 to 8 PM

Image Sources

- Sorting video screenshots:
  - Bubble sort:
    - https://www.youtube.com/watch?v=lyZQPjUT5B4
  - Selection sort:
    - https://www.youtube.com/watch?v=Ns4TPTC8whw
  - Quicksort:
    - https://www.youtube.com/watch?v=ywWBy6J5gz8

- Mark Dean:
  - http://www.blackpast.org/aah/dean-mark-1957
    - http://www.blackpast.org/files/blackpast_images/Mark_Dean__Stanford_University_News_Archive_.jpg