And now for something completely different . . .

Python Searching

Searching

Searching

Introduction to Searching

The topics of searching and sorting occupy prominent positions in computer science.

It has been estimated that anywhere from 25% to 50% of all computing time in the commercial world is spent on searching and sorting data.

Searching

Sequential Search

One of the most straightforward and elementary searches is the sequential search, also known as the linear search.

Sequential Search

You start at the beginning of a sequence and go through each item one by one, in the order they exist in the list, until you find the item you are looking for. . .

A dictionary is a book that lists (in alphabetical order) the words of a language and against each word a description of its meaning. . .

bliss *n.* perfect happiness

You're guaranteed to find any word in a dictionary, if you start at the beginning and check every word one after the other. . .

Linear search requires you to look at, on average, half of the elements in the list.

In the worst case scenario, you may have to search the entire list to find the item of interest.

Advantages of Sequential Search?

Easy to implement

Data does not need to be sorted

Sequential (Linear) Search Animation

Click here to see the animation:

`http://cs.armstrong.edu/liang/animation/web/LinearSearchNew.html`

A Sequential Search program

```
list1 = [11,27,36,44,51,22,65,1,78]
numbertofind = int(input("Enter a number\n"))
found = 0
for i in list1:
  if numbertofind == i:
     print (numbertofind, " at index: ",list1.index(numbertofind))
     found = 1
if found == 0:
  print ("Number not found")
```

10-01.py

Another Sequential Search program

```
mylist = [10,11,3,4,55,12,23,14,16]
n = len(mylist)
print (n)
for i in range(n):
    print (mylist[i], end=" ")
search = int(input("\nPlease enter a number to search for: "))
print (search)
found = False
for i in range(n):
    if mylist[i] == search:
        found = True
        index = i
print()
if found == True:
    print (str(search) + " found at index " + str(index))
else:
    print (str(search) + " not found")
```

10-02.py

Sequential (Linear) Search Video

Click here to see the video:

https://youtu.be/eQN-EB8Wsls

Searching

10.1 Introduction to Searching

10.2 The Sequential Search

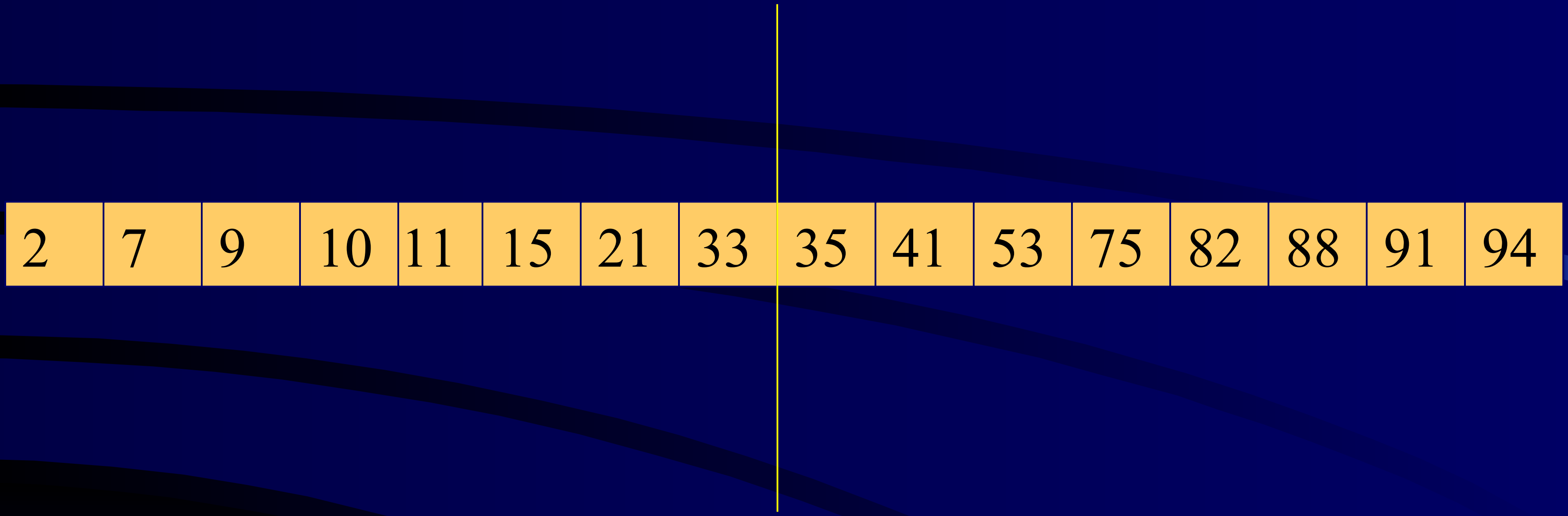
10.3 The Binary Search

Binary Search

Binary search is a more specialized algorithm than sequential search as it takes advantage of the fact that the data has been sorted. . .

Binary Search

The underlying idea of binary search is to divide the sorted data into two halves and to examine the data at the point of the split.


2
7
9
10
11
15
21
33
35
41
53
75
82
88
91
94

Binary Search

Since the data is sorted, we can easily ignore one half or the other depending on where the value we're looking for lies in comparison to the value at the split. This makes for a much more efficient search than linear search.

Binary Search Algorithms

A binary search algorithm involves breaking down the problem into a smaller version of itself, and hence is an ideal candidate for a technique known as *recursion*. . .

Recursion

Recursion is a method for solving a problem in which the problem is broken down into a smaller version of itself.

Recursion

A recursive solution to a programming problem uses a function which calls itself.   We study recursion in detail in a separate unit.

Binary Search

Binary search involves binary decisions, decisions with two choices. At each step in the process you can eliminate half of the data you're searching. . .

Binary Search

If you have an list of 16 numbers (N = 16), you can find any one number in at most, 4 steps:

```
16 -> 8 -> 4 -> 2 -> 1
```

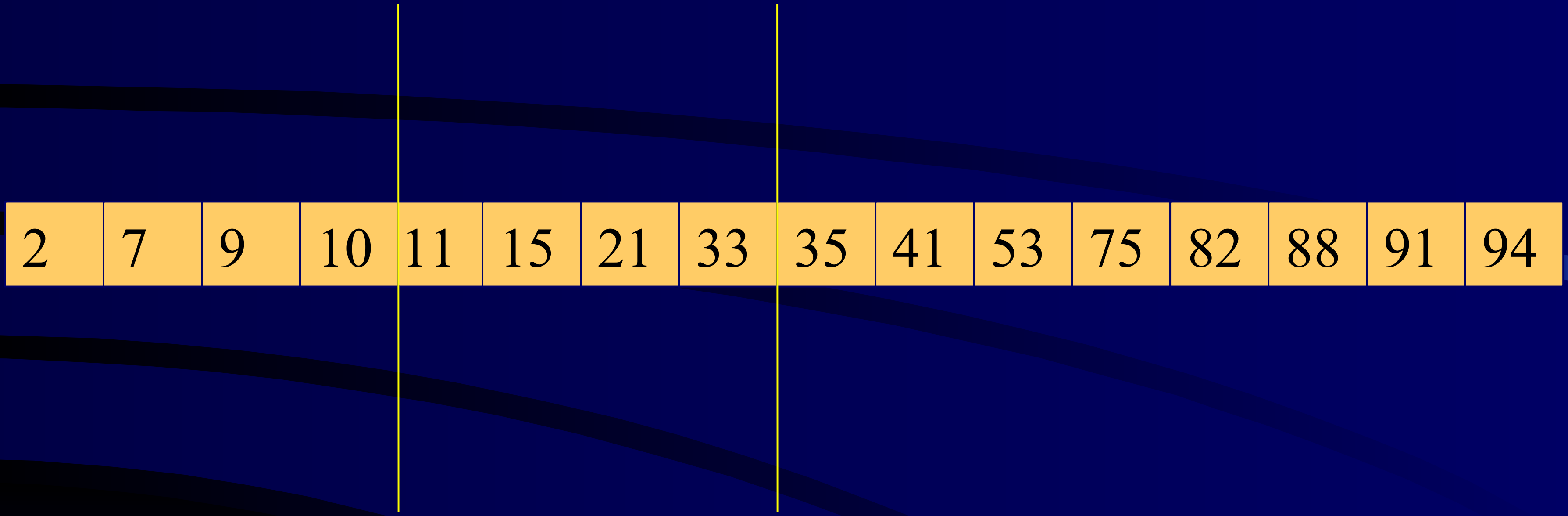
$\text{Log}_2 16 = 4$

Searching for number 7

2	7	9	10	11	15	21	33	35	41	53	75	82	88	91	94

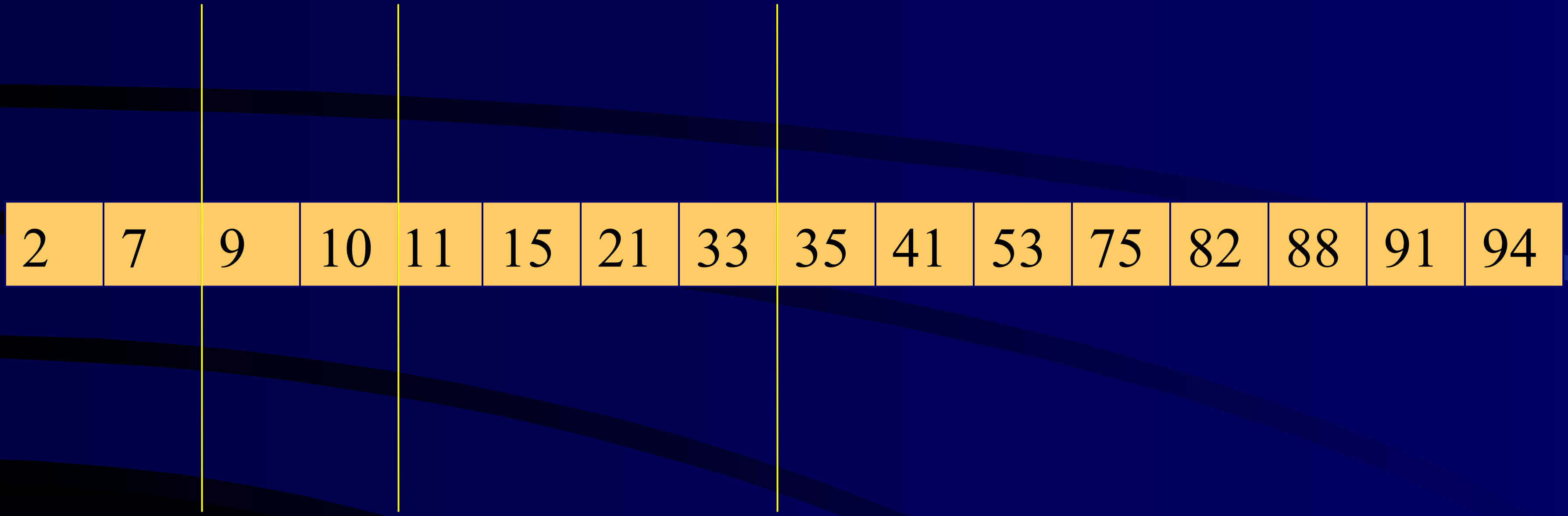
is 7 <= 33?

Searching for number 7



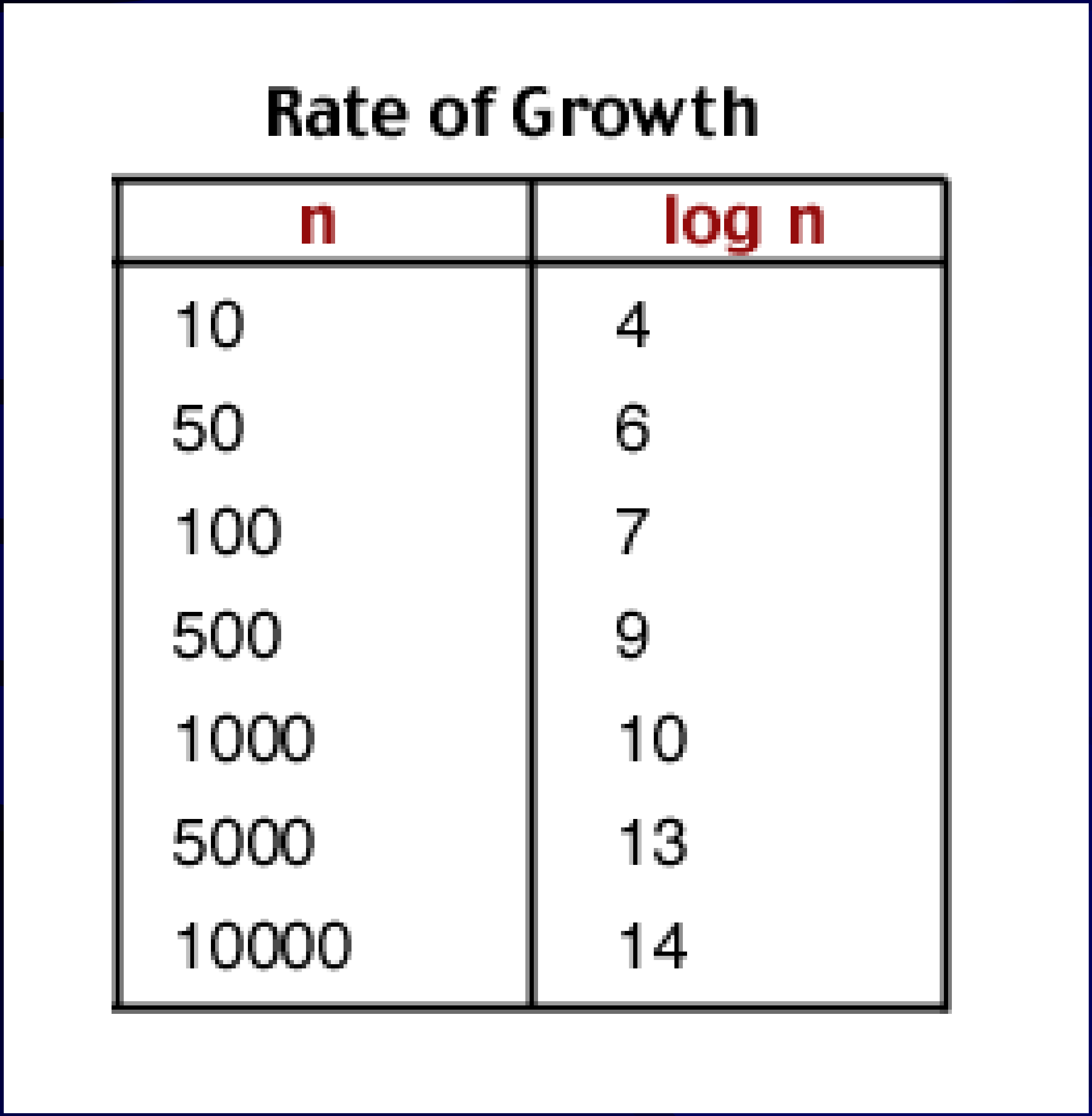
is 7 <= 10?

Searching for number 7



is 7 <= 7?

7 found!



Rate of Growth

n	log n
10	4
50	6
100	7
500	9
1000	10
5000	13
10000	14

Binary Search

In general, binary search operates on one of two data structures: lists and trees.In this course, we focus on lists.

Binary Search Animation

Click here to see the animation:

http://cs.armstrong.edu/liang/animation/web/BinarySearchNew.html

Binary Search Video

Click here to see the video:

https://youtu.be/_G7ivNb6BPk

Binary Search Python Code

Click here to see the binary search programs `10-04.p` and `10-05.py:`

`http://www.annedawson.net/python3programs.html`

This presentation uses the following program files:

`10-01.py  to 10-05.py`

Click here to see Python code:

`http://www.annedawson.net/python3programs.html`

Last updated: Friday 5th January 2018, 7:00 PT, AD