Introduction to Computer Science:

Algorithm

April 2020

Honguk Woo

How do you solve problems?

Ask questions

- What do I know about the problem?
- What is the information that I have to process in order the find the solution?
- What does the solution look like?
- What sort of special cases exist?
- How will I recognize that I have found the solution?

- Similar problems come up again and again in different guises
- A good programmer recognizes a task or subtask that has been solved before and plugs in the solution

How do you approach this ?

Given a sorted array of integers containing duplicates, count occurrences of a number provided. If the element is not found in the array, report that as well.

For example,

Input: A[] = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]
target = 5

Output: Element 5 occurs 3 times

Input: A[] = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]
target = 6

Output: Element 6 occurs 2 times

Count Occurrence in Python

- range(a) returns a sequence of numbers starting from 0 to a-1.

```
def countOccurrences(arr, x):
    res = 0
    for i in range(len(arr)):
        if x == arr[i]:
            res += 1
    return res

arr = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]
target = 6
print (countOccurrences(arr, target))
```

Strategies – algorithm pattern or paradigm

- Algorithmic pattern, or algorithmic paradigm, is a method, strategy, or technique of solving a problem
  - How to think like a computer scientist

- **e.g., Divide and Conquer**
  - Break up a large problem into smaller units and solve each smaller problem
  - Applies the concept of abstraction
  - The divide-and-conquer approach can be applied over and over again until each subtask is manageable
  - e.g., Binary search

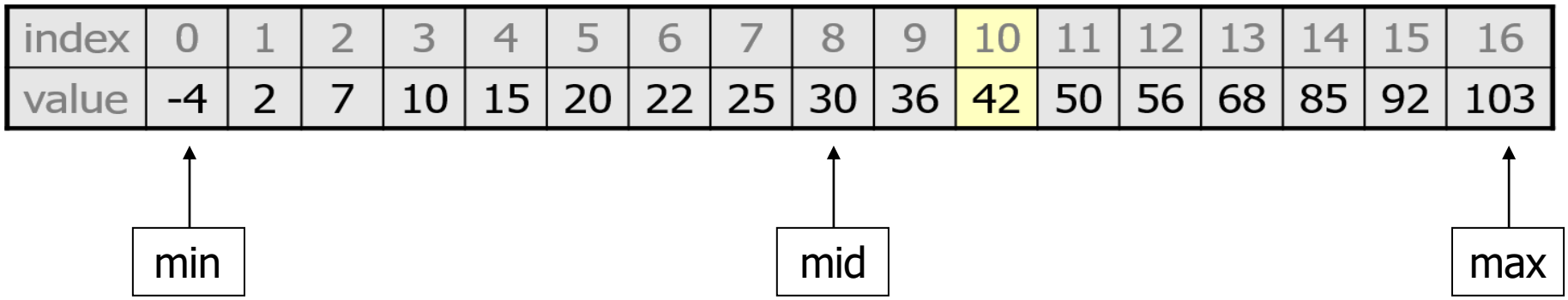
Why do we need a good strategy ?

- Sequential Search
  - How many elements will it need to examine?
  - e.g., Searching the array below for the value **42**:

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	68	85	92	103

Why do we need a good strategy ?

- Binary Search : search by successively eliminating half of the elements
  - How many elements will it need to examine?
  - Example: Searching the array below for the value **42**:



Why do we need a good strategy ? Complexity

- For an array of size N, it eliminates ½ until 1 element remains

  N, N/2, N/4, N/8, ..., 4, 2, 1

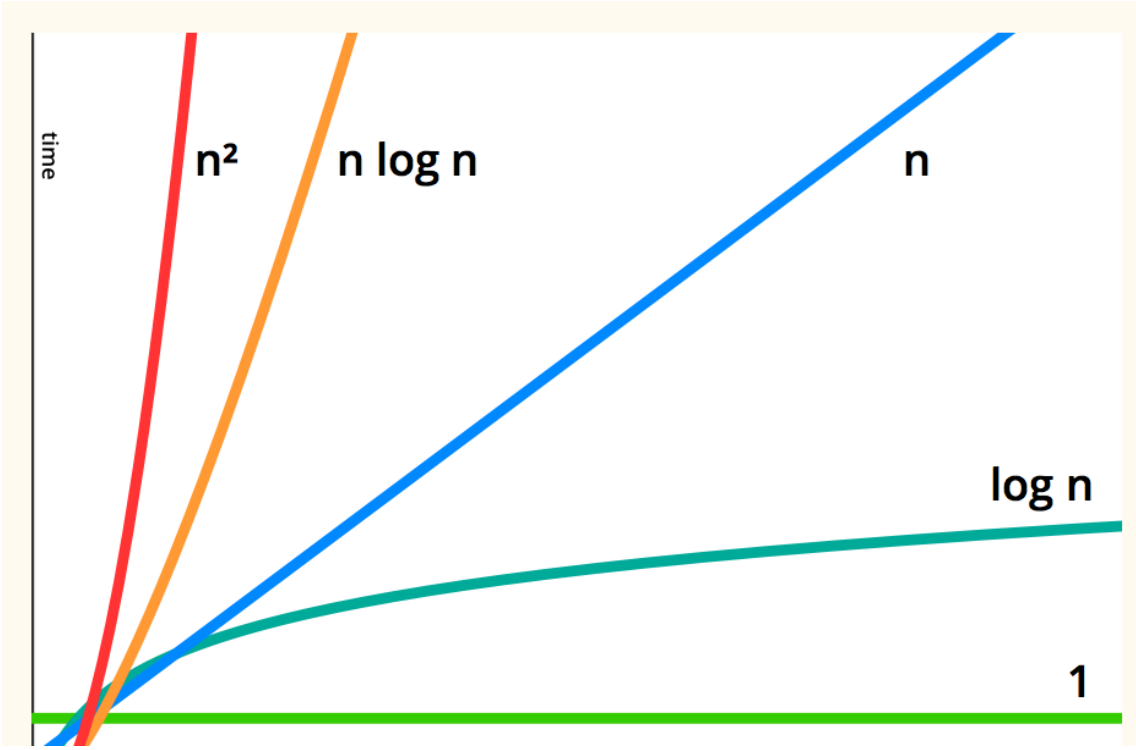
  How many divisions ?

  - How many times do I have to multiply by 2 to reach N?
    1, 2, 4, 8, ..., N/4, N/2, N
    $2^x = N$
    $x = \log_2 N$

- Binary search : **logarithmic** complexity

Complexity graph : how various complexity grows

- If having millions of elements in sorted array
  - Sequential search : N
  - Binary search : log N



Computer Problem-Solving

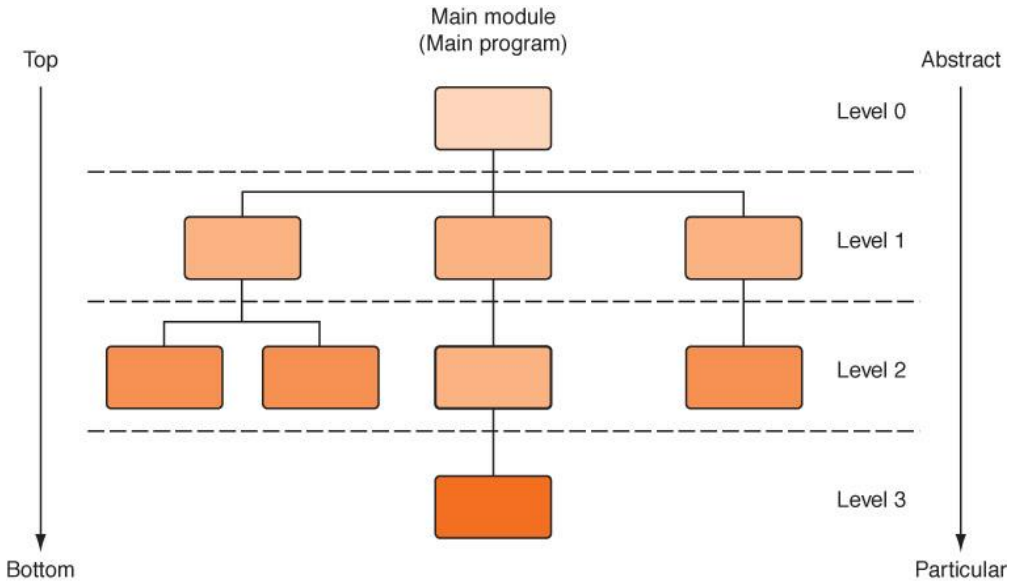
- Analysis and Specification Phase
  - Analyze, Specification
- **Algorithm Development Phase**
  - Develop algorithm, Test algorithm
- Implementation Phase
  - Code algorithm, Test algorithm
- Maintenance Phase
  - Use, Maintain

Algorithms

- **Algorithm**
  - A set of unambiguous instructions for solving a problem or subproblem in a finite amount of time using a finite amount of *data*
  - **Abstract Step**
    - An algorithmic step containing unspecified details
  - **Concrete Step**
    - An algorithm step in which all details are specified

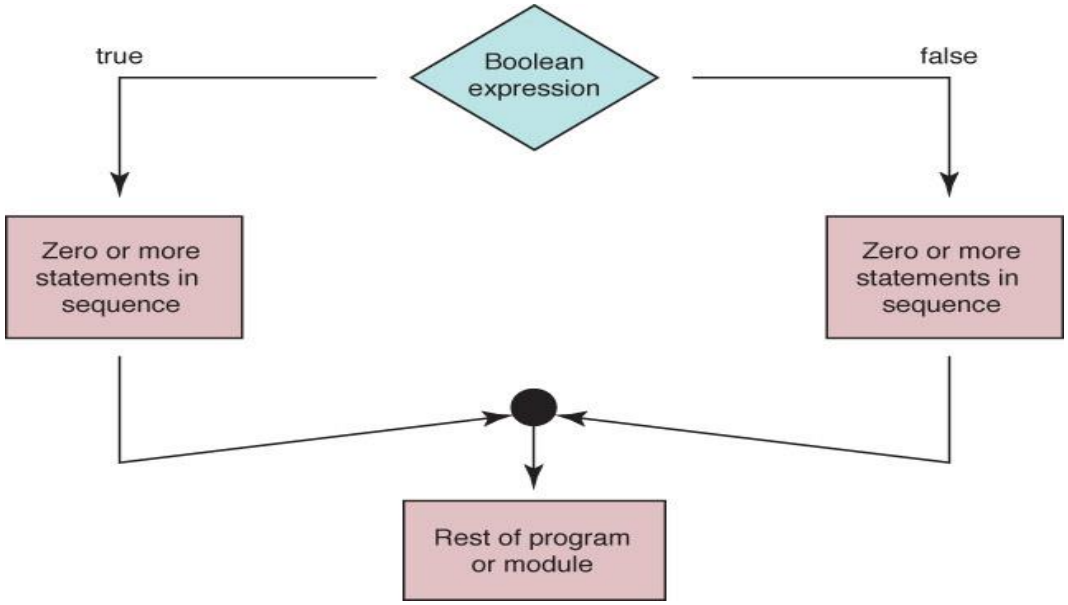
Top-Down Design

- Top-down design focuses on the **tasks** to be done to develop computer solutions to a problem
  - Process continues for as many levels as it takes to make every step concrete
  - Name of (sub)problem at one level becomes a module at next lower level



Control Structures

- **Control structure**
  - An instruction that determines the order in which other instructions in a program are executed
  - *Selection : **if statement***



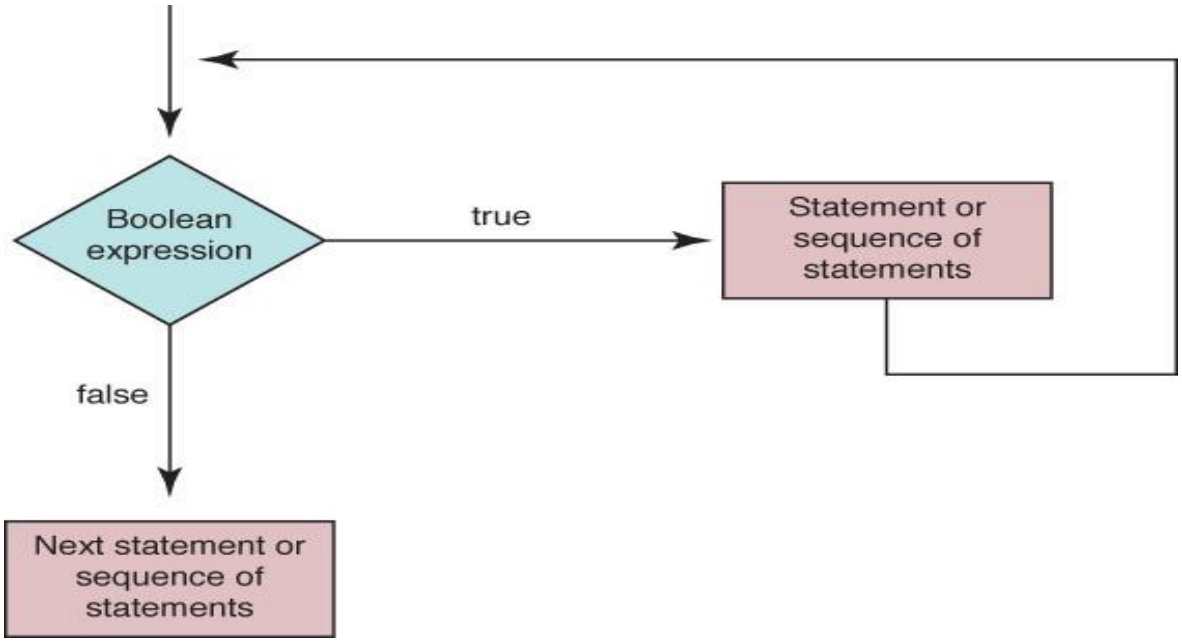
Selection : if-statement

- Problem: Write the appropriate dress for a given temperature
  - Write "Enter temperature“
  - Read temperature
  - Determine Dress

```
IF (temperature > 90)
   Write “Texas weather: wear shorts”
ELSE IF (temperature > 70)
   Write “Ideal weather: short sleeves are fine”
ELSE IF (temperature > 50)
   Write “A little chilly: wear a light jacket”
ELSE IF (temperature > 32)
   Write “Philadelphia weather: wear a heavy coat”
ELSE
   Write “Stay inside”
```

Loop Statement

- **For and While**



Loop Statement

- Problem: Calculate Square Root
  - Read in square
  - Calculate the square root
  - Write out square and the square root

```
Read in square
Set guess to square/4
Set epsilon to 1
WHILE (epsilon > 0.001)
   Calculate new guess (guess + (square/guess)) / 2.0
   Set epsilon to abs(square - guess * guess)
Write out square and the guess
```

Composite Data Types

- **Records**
  - A named **heterogeneous** collection of items in which individual items are accessed by name. For example, we could bundle name, age and hourly wage items into a record named *Employee*
- **Arrays**
  - A named **homogeneous** collection of items in which an individual item is accessed by its position (**index**) within the collection

What about C struct, array ?

What about Python list, tuple, dic ?

Composite Data Types

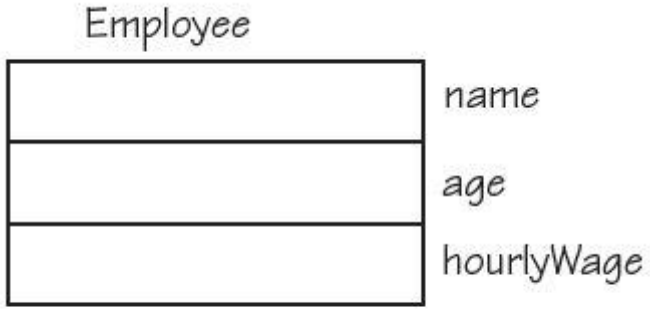


FIGURE 7.6 Record Employee

Following algorithm, stores values into the fields of record:

Employee employee // Declare and Employee variable
Set employee.name to "Frank Jones"
Set employee.age to 32
Set employee.hourlyWage to 27.50

```
class Employee:
    def __init__(self,name,age, wage):
        self.name = name
        self.age = age
        self.wage = wage

    def __str__ (self):
        return self.name + " is " + str(self.age) + " years old"

a = Employee("Tom John", 32, 27.50)
print(a)
```

Composite Data Types

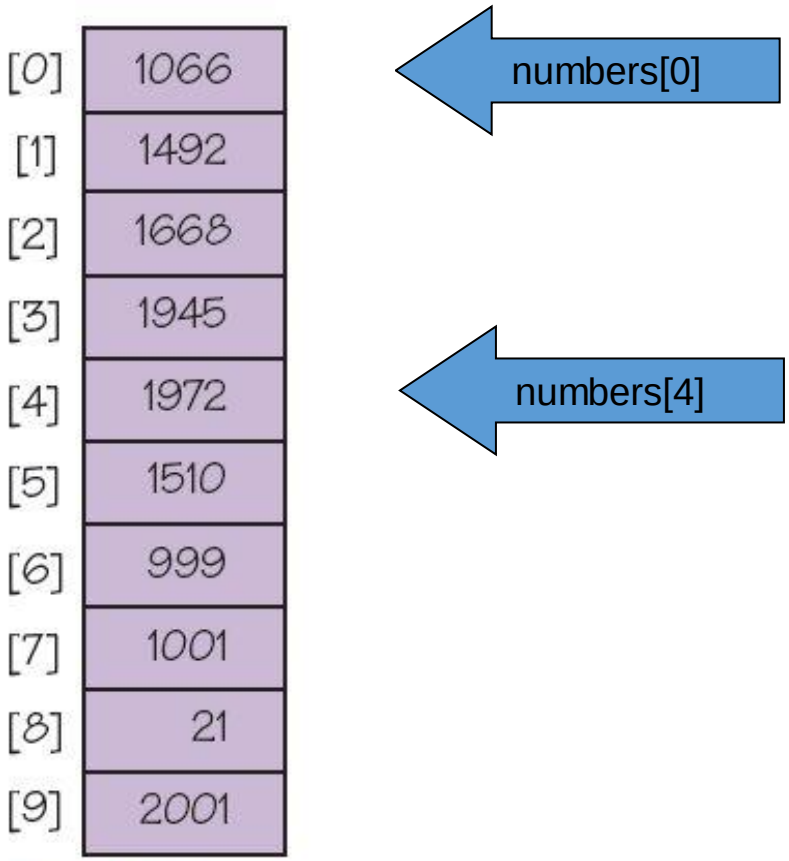


FIGURE 7.5 An array of ten numbers

Sequential Search of an **Unsorted Array**

- A sequential search examines each item in turn and compares it to the one we are searching.
- If it matches, we have found the item. If not, we look at the next item in the array.
- We stop either when we have found the item or when we have looked at all the items and not found a match
- Thus, a loop with two ending conditions

```
Set Position to 0
Set found to FALSE
WHILE (position < length AND NOT found )
        IF (numbers [position] equals searchitem)
                Set Found to TRUE
        ELSE
                Set position to position + 1
```

Sequential Search in a **Sorted Array**

- If items in an array are sorted, we can stop looking when we pass the place where the item would be it were present in the array

```
Set found to TRUE if searchItem is there
Set index to 0
Set found to FALSE
WHILE (index < length AND NOT found)
  IF (data[index] equals searchItem)
        Set found to TRUE
  ELSE IF (data[index] > searchItem)
        Set index to length
  ELSE
        Set index to index + 1
```

again, **Binary Search**

- Sequential search
  - Search begins at the beginning of the list and continues until the item is found or the entire list has been searched
- Binary search (list must be sorted)
  - Search begins at the middle and finds the item or eliminates half of the unexamined items; process is repeated on the half where the item might be

Binary Search

```
Set first to 0
Set last to length-1
Set found to FALSE
WHILE (first <= last AND NOT found)
   Set middle to (first + last)/ 2
          IF (item equals data[middle]))
                    Set found to TRUE
          ELSE
                    IF (item < data[middle])
                              Set last to middle – 1
                    ELSE
                              Set first to middle + 1
RETURN found
```

Binary Search

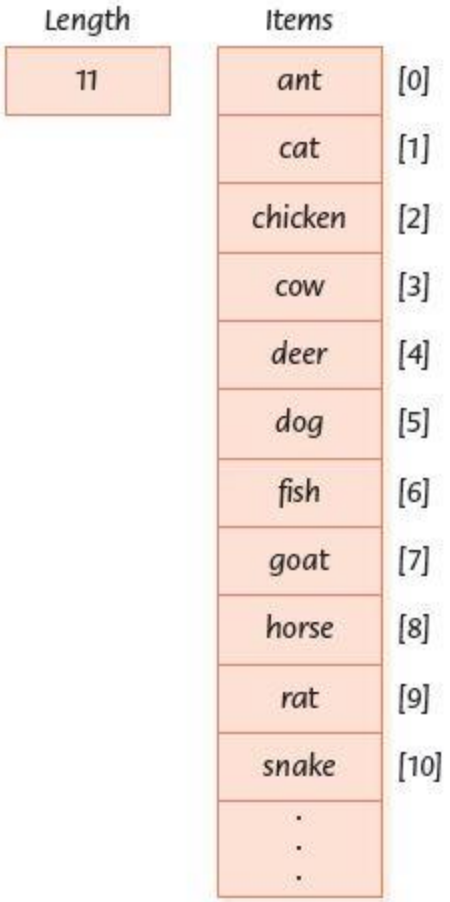


FIGURE 7.9 Binary search example

Searching for *cat*

First	Last	Middle	Comparison	
0	10	5	*cat* < *dog*	
0	4	2	*cat* < *chicken*	
0	1	0	*cat* > *ant*	
1	1	1	*cat* = *cat*	**Return: true**

Searching for *fish*

First	Last	Middle	Comparison	
0	10	5	*fish* > *dog*	
6	10	8	*fish* < *horse*	
6	7	6	*fish* = *fish*	**Return: true**

Searching for *zebra*

First	Last	Middle	Comparison	
0	10	5	*zebra* > *dog*	
6	10	8	*zebra* > *horse*	
9	10	9	*zebra* > *rat*	
10	10	10	*zebra* > *snake*	
11	10		*first* > *last*	**Return: false**

FIGURE 7.10 Trace of the binary search

again, How do you approach this ?

Given a sorted array of integers containing duplicates, count occurrences of a number provided. If the element is not found in the array, report that as well.

For example,

```
Input: A[] = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]
target = 5

Output: Element 5 occurs 3 times
```

```
Input: A[] = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]
target = 6

Output: Element 6 occurs 2 times
```

(recursive) Binary Search in Python

```
def binarySearch (arr, l, r, x):
    if r >= l:
        mid = l + (r - l)//2
        if arr[mid] == x:
            return mid
        elif arr[mid] > x:
            return binarySearch(arr, l, mid-1, x)
        else:
            return binarySearch(arr, mid+1, r, x)
    else:
        return -1


binarySearch([-1, 5, 9, 11, 16, 19, 21], 0, 6, -2)
```

use Binary Search for count occurrence

```
def countOccurrences(arr, x):
    res = 0
    i = binarySearch(arr, 0, len(arr)-1, x);
    j = i
    while (j >= 0 and arr[j] == x):
        res += 1
        j -= 1

    l = len(arr)
    j = i+1
    while (j < l and arr[j] == x):
        res += 1
        j += 1

    return res


arr = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]
target = 6
print (countOccurrences(arr, target))
```

Quiz : better one ?

- Can we find the first occurrence of x by Binary Search ?
- Can we find the last occurrence of x by Binary Search ?

```
#binary search first occurrence
def binarySearch_f_ccurrence (arr, l, r, x):
    if r >= l:
        mid = l + (r - l)//2
        if arr[mid] == x and ( mid == 0 or arr[mid-1] != x):
            return mid
        elif arr[mid] >= x:
            return binarySearch_f_ccurrence (arr, l, mid-1, x)
        else:
            return binarySearch_f_ccurrence (arr, mid+1, r, x)
    else:
        return -1


a = [-1, 5, 9, 11, 11, 16, 19, 20, 21, 21]
binarySearch_f_ccurrence(a, 0, len(a)-1, 11)
```

Another binary search (without recursion)

```
def binary_search(a_list, n, x):
    first = 0
    last = n - 1
    found = False
    while first <= last and not found:
        midpoint = (first + last) // 2
        if a_list[midpoint] == x:
            found = True
        else:
            if x < a_list[midpoint]:
                last = midpoint - 1
            else:
                first = midpoint + 1

    if (found):
        return midpoint
    else:
        return -1

test_list = [0, 1, 2, 8, 13, 17, 19, 32, 42,]
print(binary_search(test_list, len(test_list), 19))
print(binary_search(test_list, len(test_list), 27))
```

About assignment 2

- will be about Game programming with Pygame
- Snake game (wormy)
  - Most parts of code were provided in the zip file
  - Fill your code in *worm.py* as explained in the game description

- Install pygame on your environment
  - **Getting started**
    - https://www.pygame.org/wiki/GettingStarted
  - Tutorials
    - https://realpython.com/pygame-a-primer/

- Play with some sample codes
  - http://programarcadegames.com/index.php?chapter=example_code
  - https://www.pygame.org/tags/all

Anatomy of pygame : first barebone code

```
import pygame  # pygame frame
pygame.init()     #  init all the modules in pygame
screen = pygame.display.set_mode((400, 300))  # launch the window with specified size and get surface
done = False
is_blue = True
x = 30
y = 30

clock = pygame.time.Clock()

while not done:
    for event in pygame.event.get():    # get window events
        if event.type == pygame.QUIT:
            done = True
        if event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:    # input event type and key
            is_blue = not is_blue    # change color option
    pressed = pygame.key.get_pressed()    # another way to access key events
    if pressed[pygame.K_UP]: y -= 3
    if pressed[pygame.K_DOWN]: y += 3
    if pressed[pygame.K_LEFT]: x -= 3
    if pressed[pygame.K_RIGHT]: x += 3
    screen.fill((0, 0, 0))      # reset the surface
    if is_blue:
        color = (0, 128, 255)
    else:
        color = (255, 100, 0)
    pygame.draw.rect(screen, color, pygame.Rect(x, y, 60, 60))    # parameters : surface, color, rectangle

    pygame.display.flip()  # update the screen
    clock.tick(60)    # set framerate , 60 frames per second
```

https://nerdparadise.com/programming/pygame/part1