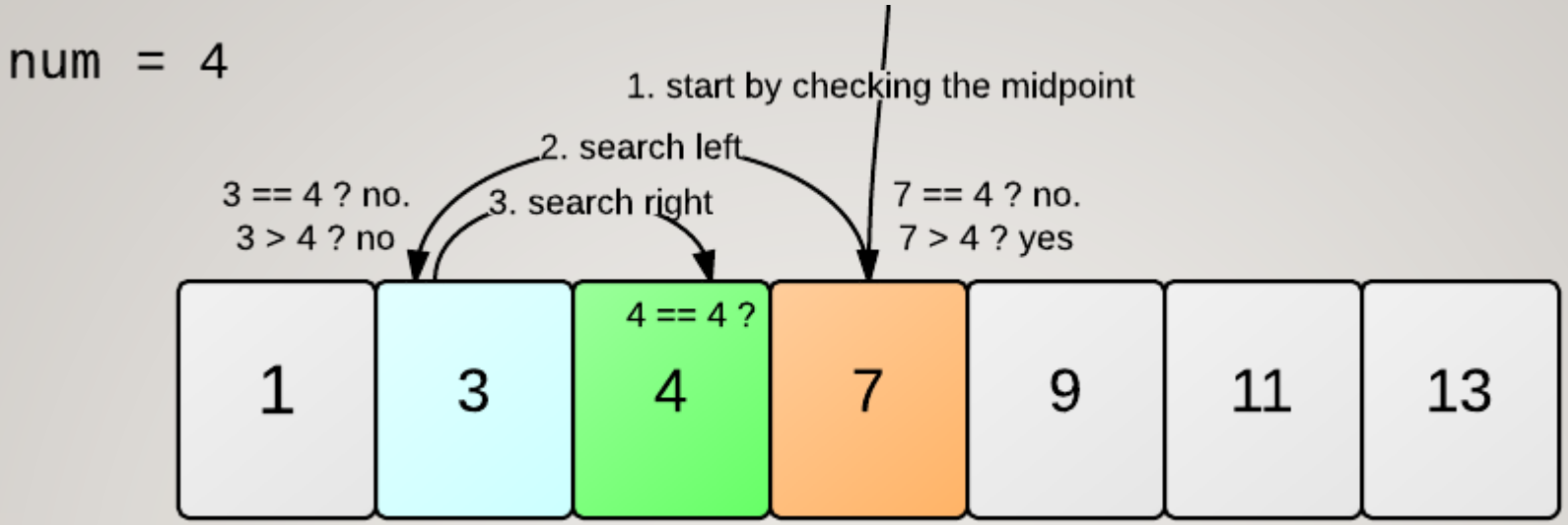


Searching and *Sorting* Algorithms

on linear arrays

ALGORITHMS

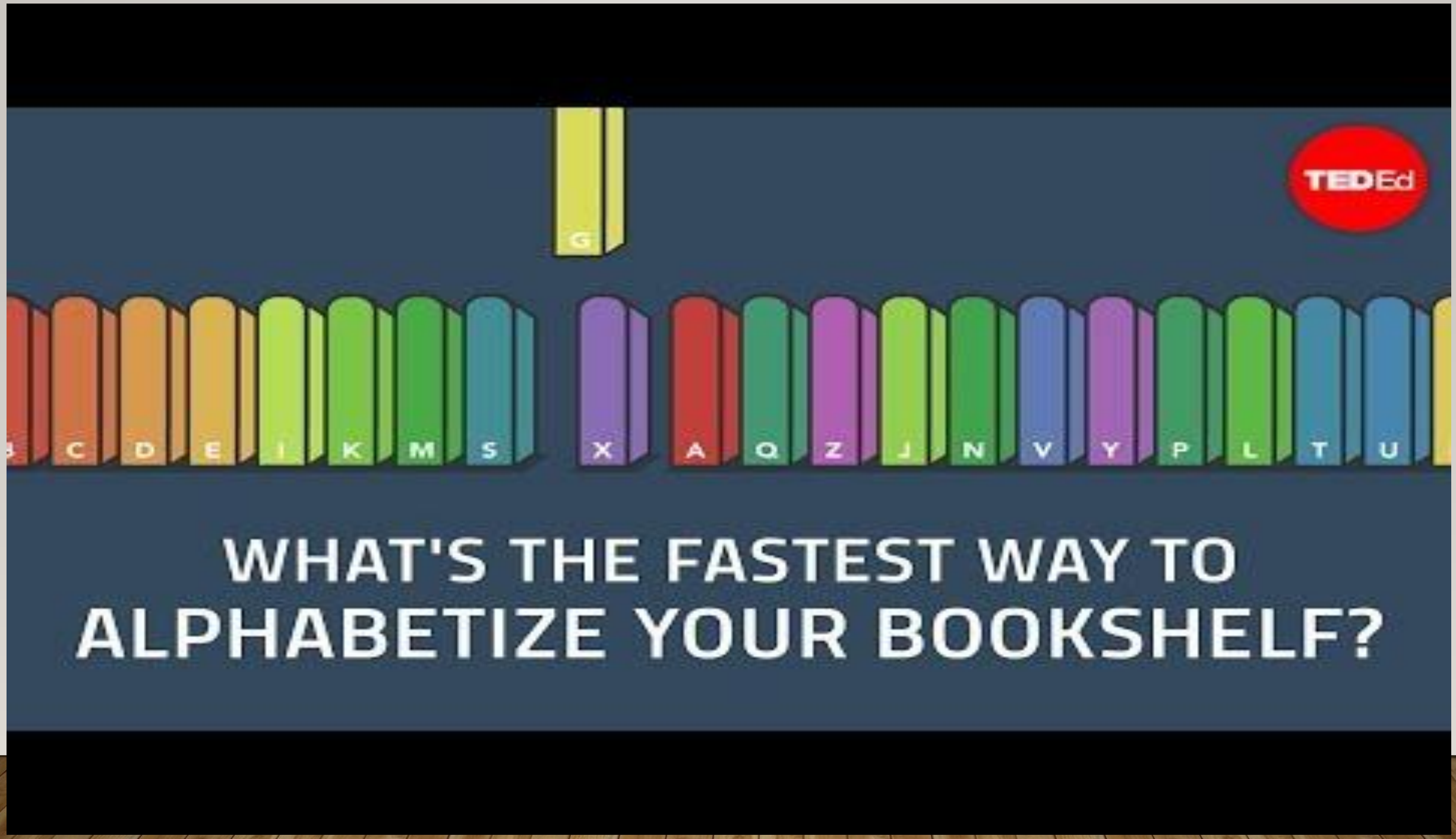
Learning Aims

“To develop an understanding of searching and sorting algorithms on arrays”

4.2.1

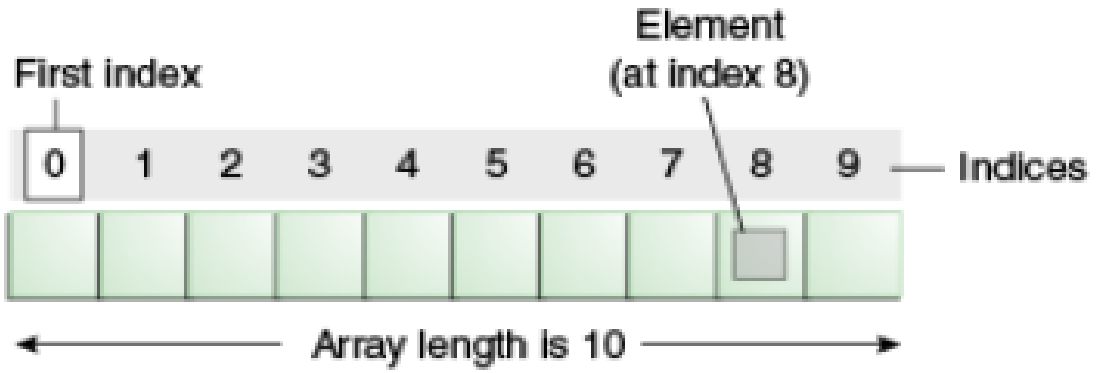
Learning Objectives

To Describe the characteristics of standard algorithms on linear arrays


G
TED-Ed
C D E I K M S X A Q Z J N V Y P L T U
WHAT'S THE FASTEST WAY TO
ALPHABETIZE YOUR BOOKSHELF?

The four key standard algorithms:

- Sequential **search**
- Binary **search**
- Bubble **sort**
- Selection **sort**



TODAY'S TASK

Use canva.com for your poster

Each of you will take one search or sort algorithm and perform some research:

- What is it? (diagrams please)
- How does it work? (include adv. and disadv.)
- What is the standard pseudocode? (IB notation)
- Code your chosen algorithm in Java and include in the poster?
- Research the algorithms "time complexity"

All in a big, bold poster please…

You will present to each other once finished…

Sequential **search**

Binary **search**

Bubble **sort**

Selection **sort**

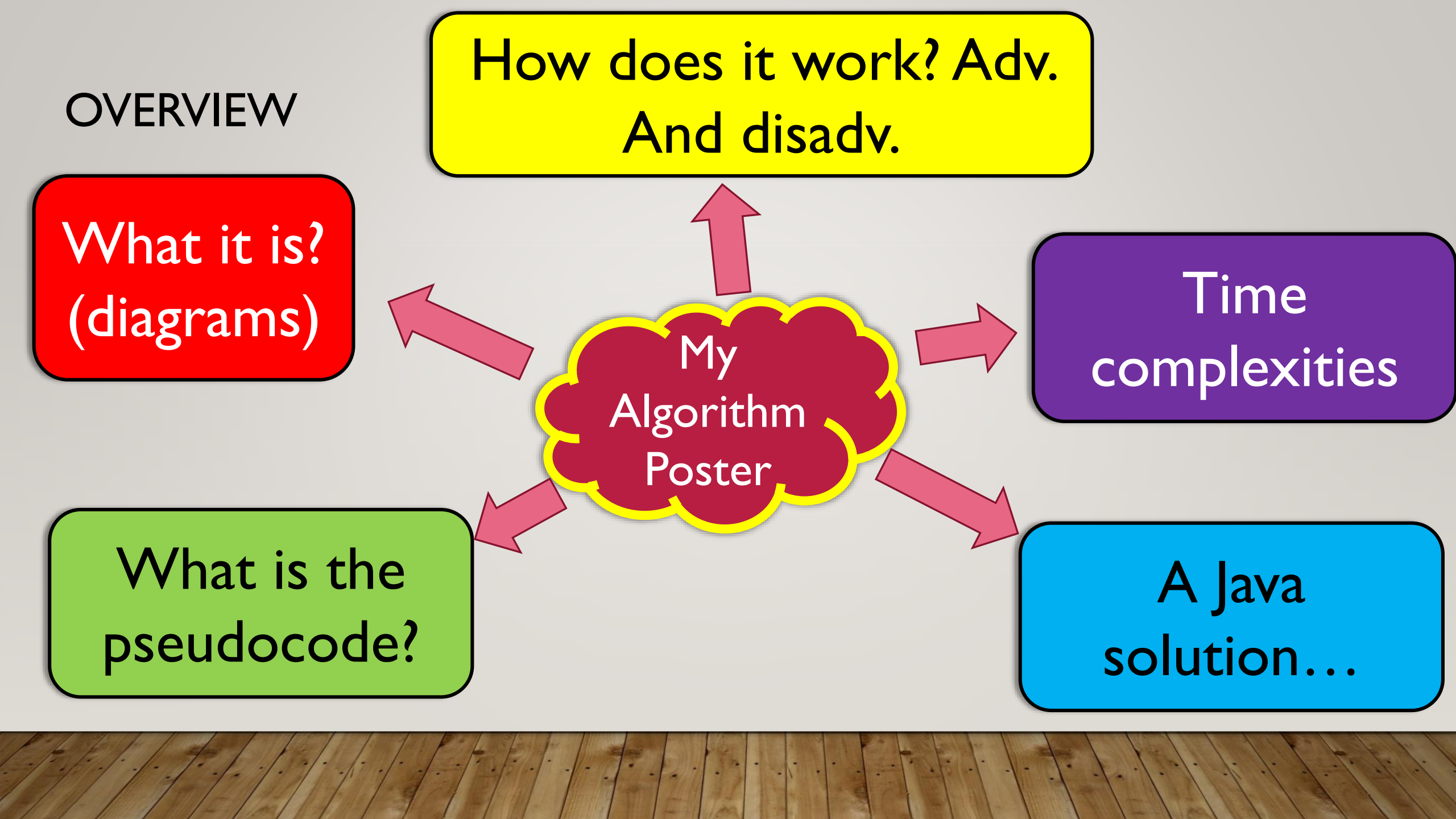
ASSESSMENT

- This will be marked
- You will be assessed in several ways including the holistic approach to your poster (effort), a chosen Java solution as well as an MCQ and online test.
- You will present your ideas (poster) to classmates to inform them of the assigned algorithm, so it must be of high quality to allow the others a chance to understand the other algorithms they have not researched!
- The more you research the more information you can provide to your peers.
- You will be completing some short online tests via the TestandTrack system after – good practice!

BREAKDOWN OF SCORES (WEIGHTINGS)

- Poster - (15%)
- Java Solution - (35%) – 4 algorithms (extra points for innovation)
- MCQ - (30%)
- Online Assignment via TestandTrack – (20%)

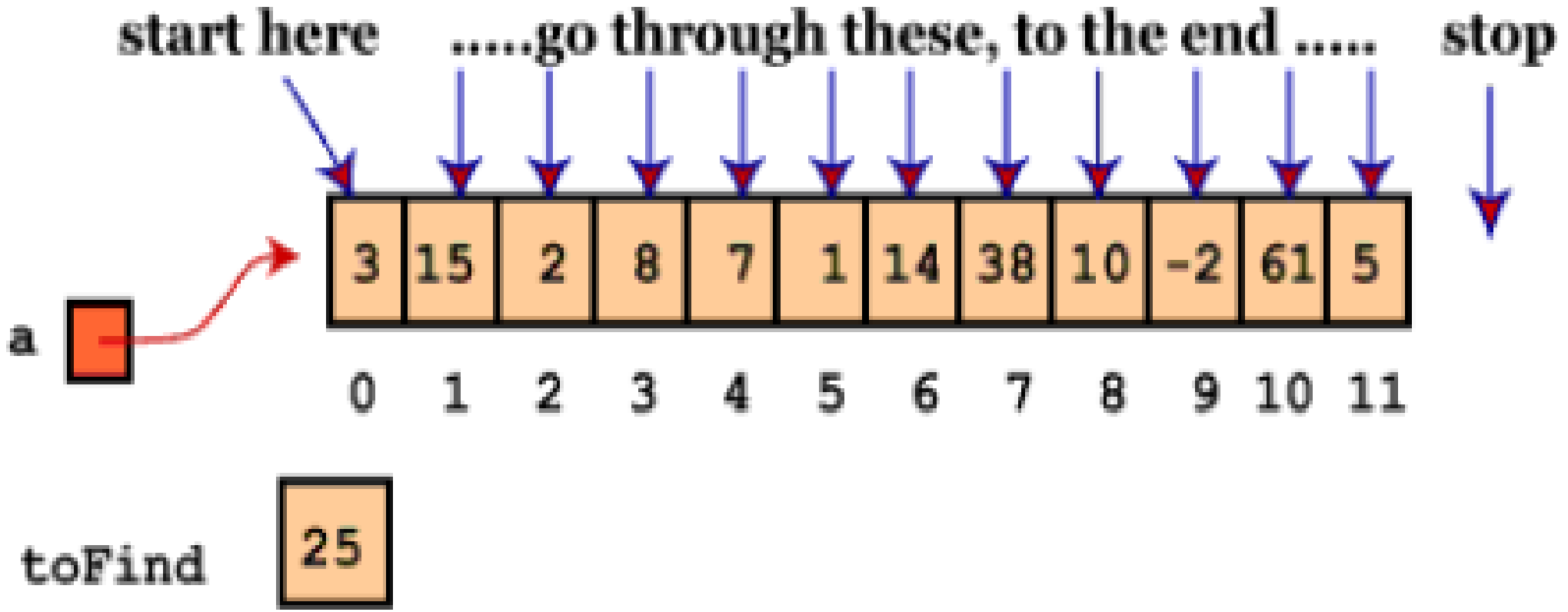
OVERVIEW


How does it work? Adv. And disadv.
What it is? (diagrams)
My Algorithm Poster
Time complexities
What is the pseudocode?
A Java solution…

SEQUENTIAL SEARCH

- Linear search or sequential search is an algorithm to find an item in a list.
- It starts at the first element and compares each element to the one it's looking for until it finds it.
- Commonly used with collections (which are unsorted lists of items) and text/csv file reading.

EXAMPLE


start here
.....go through these, to the end .....
stop
3
15
2
8
7
1
14
38
10
-2
61
5
a
0
1
2
3
4
5
6
7
8
9
10
11
toFind
25

```
NAMES = "Bob","Betty","Kim","Lucy","Dave"

output "These names start with D"

loop while NAMES.hasNext()
    NAME = NAMES.getNext()
    if firstLetter(NAME) = "D" then
      output NAME
    end if
end loop
```

Sequential Search (Pseudocode)

http://www.youtube.com/watch?v=CX2CYIJLwfg

```
linearSearch(key, array[]):

    for (i = 0; i < length(array); i++):
        if (array[i] == key):
            return i
```

BINARY SEARCH

- **Binary search**, also known as **half-interval search**, is a search algorithm that finds the position of a target value within a sorted array.
- It works by comparing the target value to the **middle element** of the array;
- If they are unequal, the lower or upper half of the array is eliminated depending on the result and the search is repeated in the remaining sub-array until it is successful.
- It only applies to **SORTED arrays** (where there are usually no duplicate values, or duplicates do not matter)

http://www.youtube.com/watch?v=D5SrAga1pno

Binary Search Tree Properties

- The left subtree of a node only contains values that are less than or equal to the node's value.
- The right subtree of a node only contains values that are greater than or equal to the node's value.
- Both left and right subtrees of a node are are also binary search trees.

TASK

1. Turn to page 211 and convert the IB Pseudocode to Java or Python
2. Complete a tracetable of the algorithm using 112 as the search target.

Binary search (IB Pseudocode)

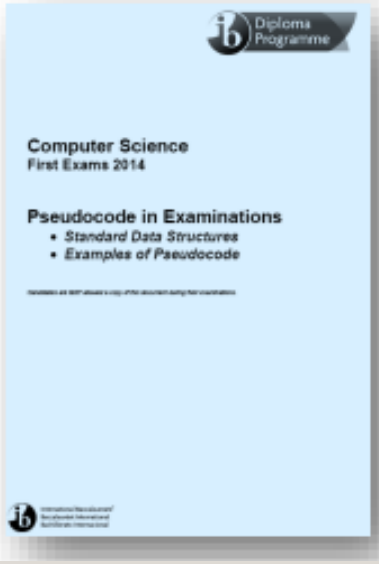
```
ID = [1001,1002,1050,1100,1120,1180,1200,1400]
NAME = ["Apple","Cherry","Peach","Banana","Fig","Grape","Olive","Mango"]

output "Type the ID number that you wish to find"
input TARGET

LOW = 0
HIGH = 7
FOUND = -1

loop while FOUND = -1 AND LOW <= HIGH
  MID = LOW + HIGH  div 2
  if ID[MID] = TARGET then
      FOUND = MID
  else if TARGET < ID[MID] then
      HIGH = MID - 1
  else
      LOW = MID + 1
  end if
end while

if FOUND >= 0 then
    output TARGET , ":" , NAME[FOUND]
else
    output TARGET , " was not found"
end if
```

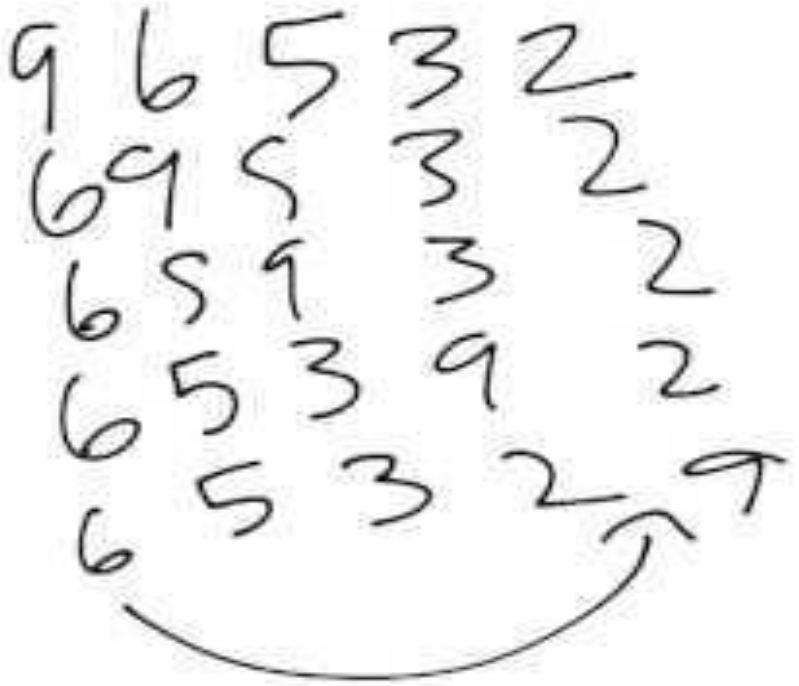


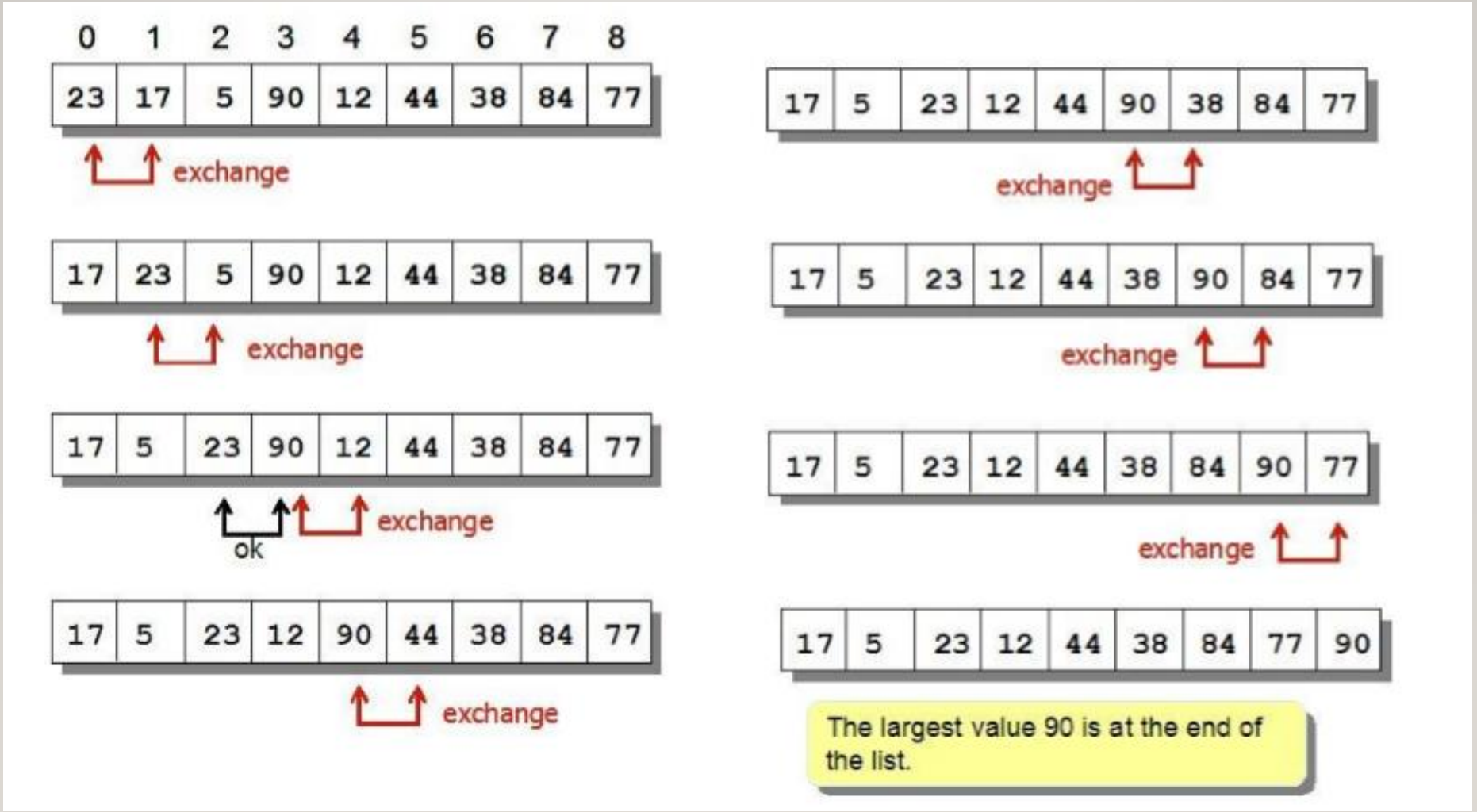
```java
/**
 * an implementation of a binary search algorithm. It will find
 * the item in the list provide the list is already sorted
 * and return the index where it found it or a -1 if not found
 * @param array the array of integer to search through
 * @param itemToFind the integer to search for in the array
 * @return the index found at, or -1 if not found
 */
public static int binarySearch(int[] array, int itemToFind) {
    int high = array.length - 1;
    int low = 0;
    while(low <= high) {
        int mid = (high + low) / 2;
        if      (array[mid] == itemToFind) return mid;
        else if (array[mid] >  itemToFind) high = mid - 1;
        else if (array[mid] <  itemToFind) low  = mid + 1;
    }
    return -1;
}
```

BUBBLE SORT

- Bubble sort is a simple sorting algorithm that repeatedly steps through the list to be sorted, **compares each pair** of adjacent items and **swaps them if they are in the wrong** order.
- The pass through the list is **repeated until no swaps are needed**, which indicates that the list is sorted.
- The algorithm, which is a comparison sort, is named for the way smaller elements "bubble" to the top of the list.
- Although the algorithm is simple, it is **too slow and impractical for most problems**

http://www.youtube.com/watch?v=8Kp-8OGwphY




0 1 2 3 4 5 6 7 8
23 17 5 90 12 44 38 84 77
exchange
17 23 5 90 12 44 38 84 77
exchange
17 5 23 90 12 44 38 84 77
ok
exchange
17 5 23 12 90 44 38 84 77
exchange
17 5 23 12 44 90 38 84 77
exchange
17 5 23 12 44 38 90 84 77
exchange
17 5 23 12 44 38 84 90 77
exchange
17 5 23 12 44 38 84 77 90
The largest value 90 is at the end of the list.

```
NUMS = [15,30,85,25,40,90,50,65,20,60]

output "Before sorting"
loop C from 0 to 9
    output NUMS[C]
end loop

loop PASS from 0 to 8
    loop CURRENT from 0 to 8
        if NUMS[CURRENT] < NUMS[CURRENT + 1] then
            TEMP = NUMS[CURRENT]
            NUMS[CURRENT] = NUMS[CURRENT+1]
            NUMS[CURRENT+1] = TEMP
        end if
    end loop
end loop
```

IB PSEUDOCODE (BUBBLE SORT)

SELECTION SORT

Selection sort is a sorting algorithm and it is **inefficient** on **large lists**

Selection sort is noted for its **simplicity**, and it has performance advantages over more complicated algorithms in certain situations, **particularly where memory is limited**.

The algorithm **divides** the input list into two parts: the sublist of items **already sorted**, which is built up from left to right at the front (left) of the list, and the sublist of **items remaining to be sorted** that occupy the rest of the list.

Initially, the sorted sublist is **empty** and the unsorted sublist is **the entire input list**.

The algorithm proceeds by finding the **smallest** (or **largest**, depending on sorting order) element in the unsorted sublist, exchanging (swapping) it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right.

http://www.youtube.com/watch?v=f8hXR_Hvybo

```
A - an array containing the list of numbers
numItems - the number of numbers in the list

for i = 0 to numItems - 1
    for  j = i+1 to numItems
        if A[i] > A[j]
            // Swap the entries
            Temp = A[i]
            A[i] = A[j]
            A[j] = Temp
        end if
    end loop
end loop
```

SELECTION SORT
(IB PSEUDOCODE)

RECAP

1. What did we do last lesson?

BREAKDOWN OF TASKS

- Poster - (15%) 
- Java Solution - (35%) – 4 algorithms 
- MCQ - (30%) 
- Online Assignment via TestandTrack – (20%) 

TODAYS TASK

*(**Bubble** Sort, **Binary** Search, **Linear** Search, **Selection** Sort)*

- Continue to develop your Java solution for each of the algorithms we have studied

```
LOW = 0
HIGH = 7
FOUND = -1

loop while FOUND = -1 AND LOW <= HIGH
  MID = LOW + HIGH  div 2
  if ID[MID] = TARGET then
      FOUND = MID
  else if TARGET < ID[MID] then
      HIGH = MID - 1
  else
      LOW = MID + 1
  end if
end while

if FOUND >= 0 then
    output TARGET , ":" , NAME[FOUND]
else
    output TARGET , " was not found"
end if
```

BinarySearchArray.java

```
import java.util.*;
    public class BinarySearchArray {
    public static void main(String[]args) {
            String [] A = {"Anna","Bill","David","Faisal","Jasmine","Jumal","Ken","Michela","Pavel"
            System.out.println("Please enter name to search for: ");
            Scanner kb = new Scanner (System.in);
            String itemsought = kb.nextLine();
            int itemFound = 0;
            int searchFailed = 0;
            int top = A.length-1;
            int bottom = 0;

            while (! itemFound) & (! searchFailed){

                midpoint = int((top + bottom)/2)
                print("top, bottom, midpoint", top, bottom, midpoint)
                if A[midpoint]==itemSought:
                    itemFound = 1
                else:
                    if bottom > top:
                        searchFailed = 1
                    else:
                        if A[midpoint]<itemSought:
                            bottom = midpoint + 1
                        else:
                            top = midpoint - 1

            #endwhile
            if itemFound:
                print("item is at position ",midpoint)
            else:
                print ("item is not in the array")
    }
}
```

TRACING

1. You *may* be asked to trace an algorithm in the exam (actually quite likely).
2. Understanding algorithms by simply looking at them is difficult, therefore we can use a **trace table** to make this easier.
3. **Tracing** is a technique used to test an algorithm and predict step by step how the computer will run the algorithm. It can be used to understand or predict what an algorithm is doing and to identify potential logic errors (when the program compiles but does not produce the expected output).

```
int[] array = {3, 8, 2, 5};
int total = 0;
for(int i=1; i < array.length; i++)
{
    array[i] = array[i - 1];
    total = total + array[i];
}
```

Trace Table

	array				
i	0	1	2	3	total

PROBLEM 1

```
1  number = 5
2  factorial = number
3
4  WHILE number>2
5      number = number - 1
6      factorial = factorial * number
7  END WHILE
8
9  OUTPUT(factorial)
```

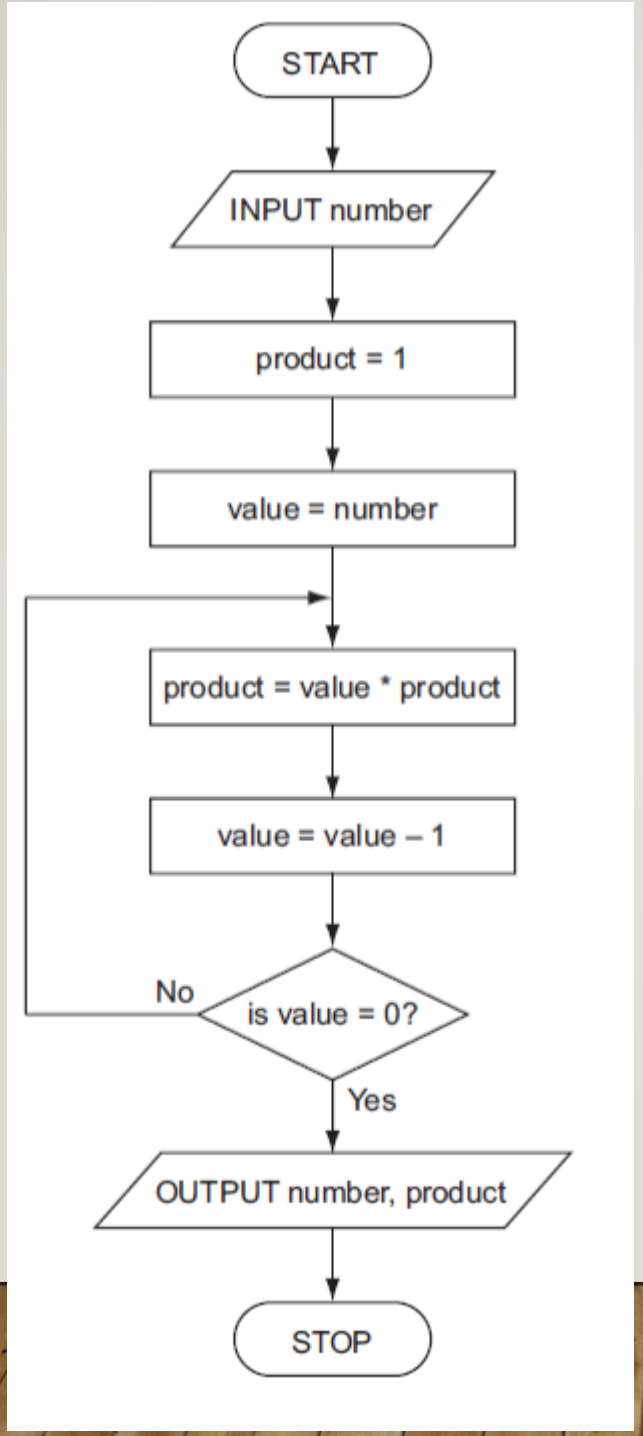
number	factoral	Output

PROBLEM 2

```
1   FOR i FROM 1 TO 20
2      IF i MOD 3 == 0 AND i MOD 5 == 0 THEN
3         OUTPUT "Fizz-Buzz"
4      ELSE IF i MOD 3 == 0 THEN
5         OUTPUT "Fizz"
6      ELSE IF i MOD 5 == 0 THEN
7         OUTPUT "Buzz"
8      ELSE
9         OUTPUT i
10     END IF
11  NEXT i
12
13  OUTPUT("The End")
```

Line Number	i	i MOD 3 == 0	i MOD 5 == 0	OUTPUT

PROBLEM 3



number	product	value	OUTPUT

SOLUTION PROBLEM 3

number	product	value	OUTPUT
5	1	5	
(5)	5	4	
(5)	20	3	
(5)	60	2	
(5)	120	1	
(5)	(120)	0	
			5, 120

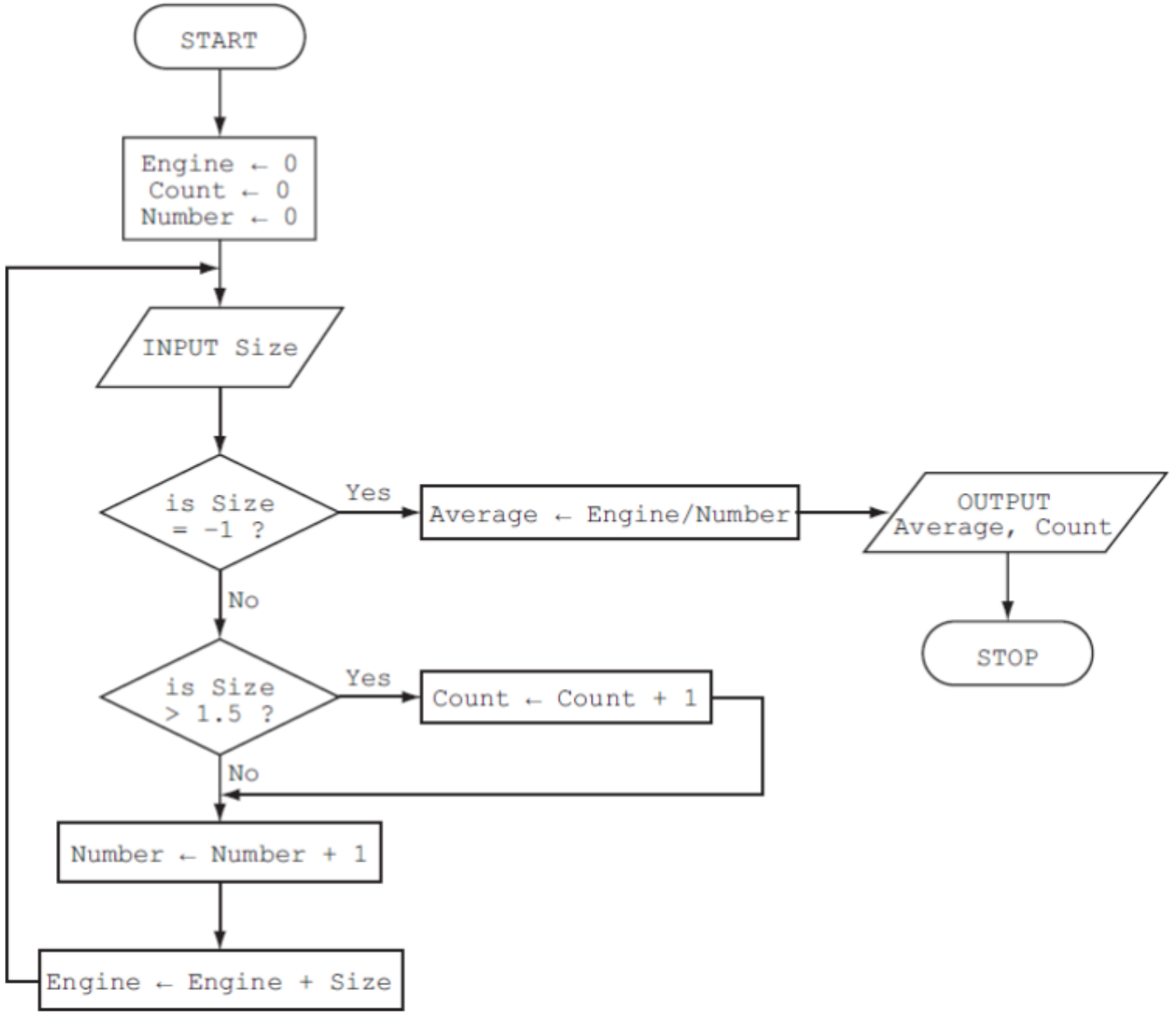
PROBLEM 4

1.8, 2.0, 1.0, 1.3, 1.0, 2.5, 2.0, 1.3, 1.8, 1.3, –1

Engine	Count	Number	Size	Average	OUTPUT

The flowchart inputs the size of a number of car engines; a value of –1 stops the input.

This information is output: *average engine size* and *number of engines with size > 1.5*



SOLUTION PROBLEM 4

Engine	Count	Number	Size	Average	OUTPUT
0	0	0	1.8		
1.8	1	1	2.0		
3.8	2	2	1.0		
4.8		3	1.3		
6.1		4	1.0		
7.1		5	2.5		
9.6	3	6	2.0		
11.6	4	7	1.3		
12.9		8	1.8		
14.7	5	9	1.3		
16.0		10	–1		
				1.6	
					1.6, 5

PROBLEM 5

Construct a trace table for the following algorithm

```
A = 3
B = 7
loop while B >= A
   A = A + 1
   output(B - A)
   B = B - 1
end loop
```

[4]

SOLUTION

*Award **[4 max]**.*

*Award **[1]** for a trace table with at least three columns.*
*Award **[1]** for each correct column (out of the four columns – A, B, B >= A, output).*

A	B	B >= A	output
3	7	true	
4	6		3
		true	
5	5		1
		true	
6	4		–1
		false	

[4]

PROBLEM 6

8. Construct a trace table for the following algorithm.

```
K = 1
N = 1
M = 2
loop while K < 5
    output(N,M)
    K = K + 1
    N = N + 2
    M = M * 2
end loop
```

[5]

SOLUTION

8. *Award **[5 max]***

*Award **[1]** for a trace table with at least three columns (headings K, N, M, K<5 and output);*

*Award **[1]** for each correct output up to **[4 max]***

K	N	M	K<5	OUTPUT
1	1	2	TRUE	1 2
2	3	4	TRUE	3 4
3	5	8	TRUE	5 8
4	7	16	TRUE	7 16
5	9	32	FALSE	

[5 max]