# Computer Science

# Binary search

# Lesson Objectives

Students will learn:

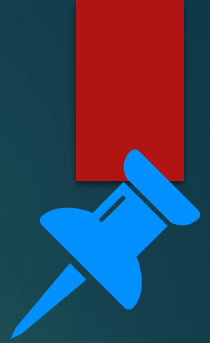
- Binary search algorithm
- How is an element searched in a list using a binary search algorithm?
- Pseudocode for a binary search algorithm

# KNOWING WHAT YOU KNOW

- Go to: https://joinmyquiz.com
- Write your name and grade level
- Join code: _____

1.

# Content



# Searching and sorting algorithms

- Sorting algorithms arrange the data in particular order.
- Searching algorithms are used to search for data in a list.

# Binary search algorithm

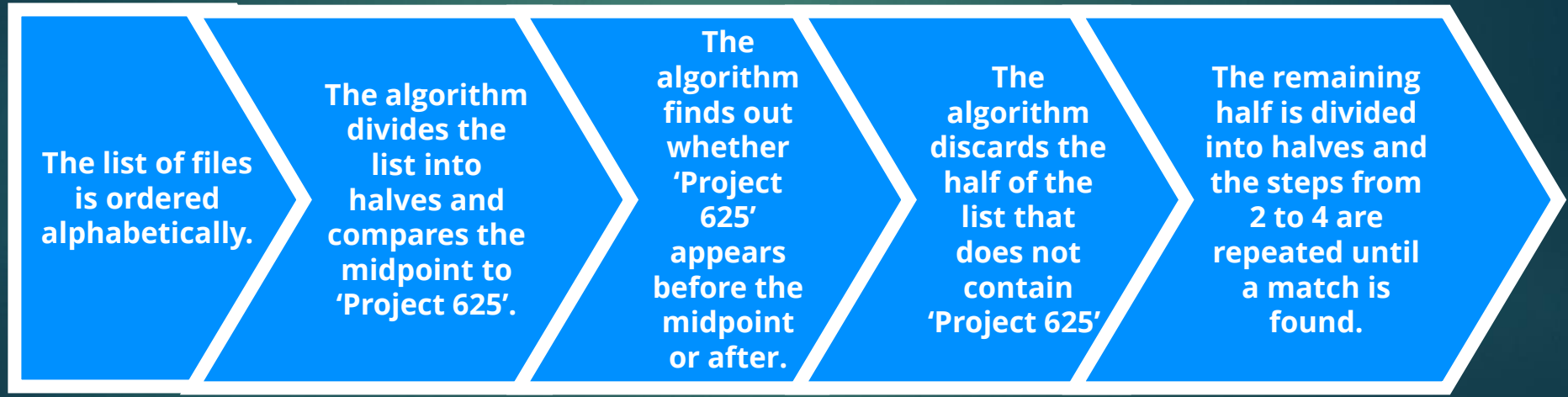
- When a list is input to a binary search algorithm, the algorithm keeps dividing into half until the item is matched with the one in the list.

# Ordered list

- An ordered list is a list of items that are arranged sequentially in a particular order.
- For example: products on an online shopping website are arranged according to their price and files in Windows Explorer are arranged in various orders such as name, size, date and type.
- Binary search algorithms work faster with an ordered list.

# Binary search algorithm

Let's us search a file with file name 'Project 625' in a folder.



# Binary search algorithm: Pseudocode

Let's use an example of searching a file with the name 'Project 625' in a folder using a binary search algorithm.

Setting the variables.

*INPUT user inputs 'Project 625' in File explorer*

*file_name='Project 625'*

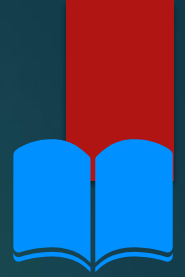
*file_found=FALSE*

*(file_found turns TRUE only when the match is found)*

# Binary search algorithm: Pseudocode

Using while loop and if condition to compare the file names.

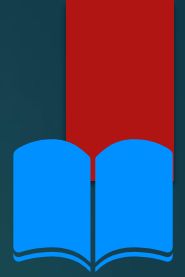
```
WHILE file_found = FALSE:
    Find the midpoint of the list
    IF  file_name=record at midpoint of the list THEN
        file_found=TRUE
    ELSE IF file_name is in the first half of the list THEN
        discard the second half of the list
    ELSE
        discard the first half of the list
    ENDIF
 END WHILE
OUTPUT file_name, file_size, file_data and file_type
```



# Pseudocode

- Let us consider an ordered list with a certain length_of_list.
- lower_bound is the position of first element and upper_bound is the position of last element.
- In the table below, a list of 10 elements is shown. Here the **lower_bound is 0**, **upper_bound is 9** and length_of_list is 10.

| Position | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Element | A | B | C | D | E | F | G | H | I | J |



# Pseudocode

Setting the variables

```
item_to_be_found=(“enter the item to be found”)
lower_bound=0
upper_bound=length_of_list-1
item_found=false
```

- Using while loop to check whether a match has been found.
- Using if condition to compare items

```
while (item_found==false and lower_bound <= upper_bound)
        midpoint= round ((lower_bound+upper_bound)/2)
        if list[midpoint]=item then
                item_found=true
        elseif list[midpoint] < item then
                lower_bound=midpoint+1
        else
                upper_bound=midpoint-1
        endif
endwhile
if (item_found==true) then
        print (“item found at ”, midpoint )
else
        print (“item not present”)
endif
```

# Analysing Pseudocode

- Let us analyse this pseudocode by using some values.
- Let item_to_be_found= G in the ordered list.

| Position | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Element | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** |

- midpoint = round ((0+9) /2) = round (4.5) =5
- list [midpoint]= list [5]= F
- Because list[midpoint] < item, the statement lower_bound=midpoint +1 is executed.

| P | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| E | A | B | C | D | E | F | G | H | I | J |

```
while (item_found==false and lower_bound <= upper_bound)
        midpoint= round ((lower_bound+upper_bound)/2)
        if list[midpoint]=item then
                item_found=true
        elseif list[midpoint] < item then
                lower_bound=midpoint+1
        else
                upper_bound=midpoint-1
        endif
endwhile
```

- Now the lower_bound becomes 6. The lower half is now discarded. Therefore,

| | | | | |
|---|---|---|---|---|
| P | 6 | 7 | 8 | 9 |
| E | G | H | I | J |

- midpoint = round ((6+9) /2) = round (7.5) =8
- list [midpoint]= list [8]= I
- Because list[midpoint] > item, the statement upper_bound=midpoint -1 is executed.

```
while (item_found==false and lower_bound <= upper_bound)
        midpoint= round ((lower_bound+upper_bound)/2)
        if list[midpoint]=item then
                item_found=true
        elseif list[midpoint] < item then
                lower_bound=midpoint+1
        else
                upper_bound=midpoint-1
        endif
endwhile
```

- Now the upper_bound becomes 7. The upper half is now discarded. Therefore,

| | | |
|---|---|---|
| P | 6 | 7 |
| E | G | H |

- midpoint = round ((6+7) /2) = round (6.5) =7
- list [midpoint]= list [7]= H
- Because list[midpoint] > item, the statement upper_bound=midpoint -1 is executed.

```
while (item_found==false and lower_bound <= upper_bound)
        midpoint= round ((lower_bound+upper_bound)/2)
        if list[midpoint]=item then
                item_found=true
        elseif list[midpoint] < item then
                lower_bound=midpoint+1
        else
                upper_bound=midpoint-1
        endif
endwhile
```

- Now the upper_bound becomes 6. The upper half is now discarded. Therefore,

| | |
|---|---|
| P | 6 |
| E | G |

- midpoint = round ((6+6) /2) = round (6) =6
- list[midpoint]= list [6] =G, the statement item_found =true is executed.

```
while (item_found==false and lower_bound <= upper_bound)
        midpoint= round ((lower_bound+upper_bound)/2)
        if list[midpoint]=item then
                item_found=true
        elseif list[midpoint] < item then
                lower_bound=midpoint+1
        else
                upper_bound=midpoint-1
        endif
endwhile
```

# Analysing Pseudocode

- The while  loop ends, and the output is: item found at 6

```
if (item_found==true) then
        print (“item found at ”, midpoint )
else
        print (“item not present”)
endif
```

# KNOWING WHAT YOU LEARNED

- Go to: https://joinmyquiz.com
- Write your name and grade level
- Join code: _____

2.

# Activity

# Activity-1

Duration: 15 minutes

1. Here is a list of 9 elements.

| Position | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Element | **P** | **Q** | **R** | **S** | **T** | **U** | **V** | **W** | **X** |

User wants to search the element P from the above list. What steps are followed in a binary search algorithm to find out P from the above list?

# Activity-1

Duration: 15 minutes



1. Here is a list of 9 elements.

User wants to search the element P from the above list. What steps are followed in a binary search algorithm to find out P from the above list?

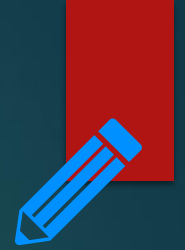
3.

End of topic questions

# End of topic questions

1. Using the pseudocode of a binary search algorithm, create a flowchart for this algorithm.

```
item_to_be_found=("enter the item to be found")
lower_bound=0
upper_bound=length_of_list-1
item_found=false
while (item_found==false and lower_bound <= upper_bound)
        midpoint= round ((lower_bound+upper_bound)/2)
        if list[midpoint]=item then
                item_found=true
        elseif list[midpoint] < item then
                lower_bound=midpoint+1
        else
                upper_bound=midpoint-1
        endif
endwhile
```



# End of topic questions

2. How is a binary search algorithm different from a linear search algorithm?