IT1312

Data Structures & Algorithms

Topic 03b : Search – Binary Search

Learning Outcomes

- Describe Binary Search Algorithms
- Know the differences between Sequential Search (Linear) and Binary Search
- Define the Complexity (Big-O) of Binary Search

Search Algorithms

- Searching is the process of finding an item within a sequence using a **search key** to identify the specific item.

- **Sequential Search** (or Linear Search)
  - Unsorted List
  - Sorted List
- **Binary Search**

Binary Search

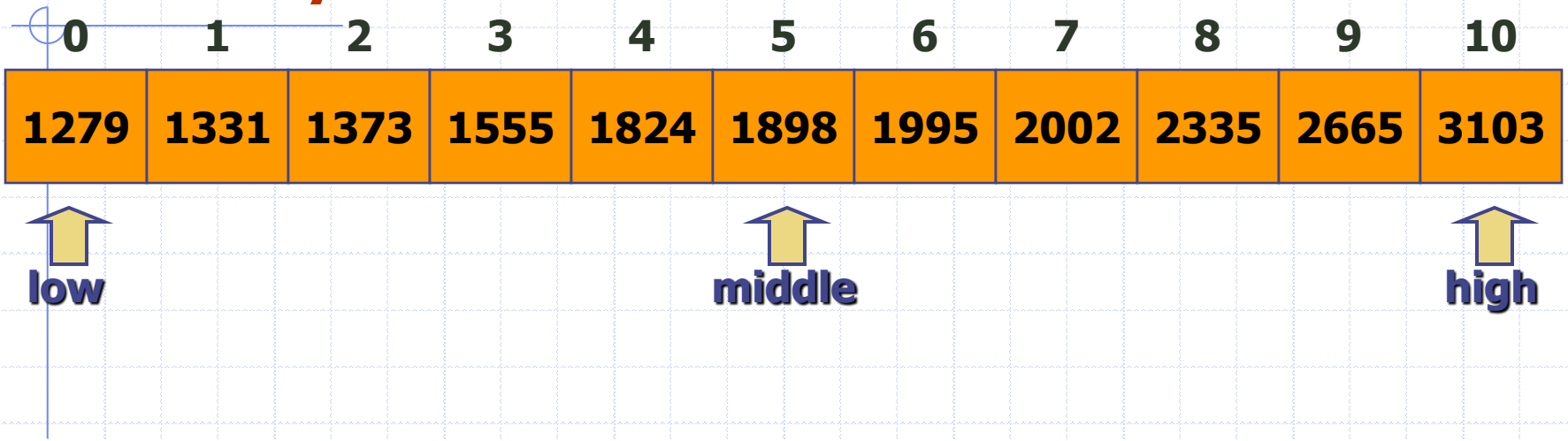
- It is used when list is **sorted**
- It is **more efficient** than sequential / linear search
- Adopts the **divide and conquer** strategy:
  - Recursively breaking down a problem into two or more sub-problems, until these become simple enough to be solved directly.
  - The solutions to the sub-problems are then combined to give a solution to the original problem.

Binary Search

- **Algorithm:**
  - Start with the middle item in the list and compare it against the target value
  - If a match is found, return position in the list containing the target value
  - Else determine if item is in the left-half or right-half of the list
  - Repeat search using the **middle** item of that sub-list

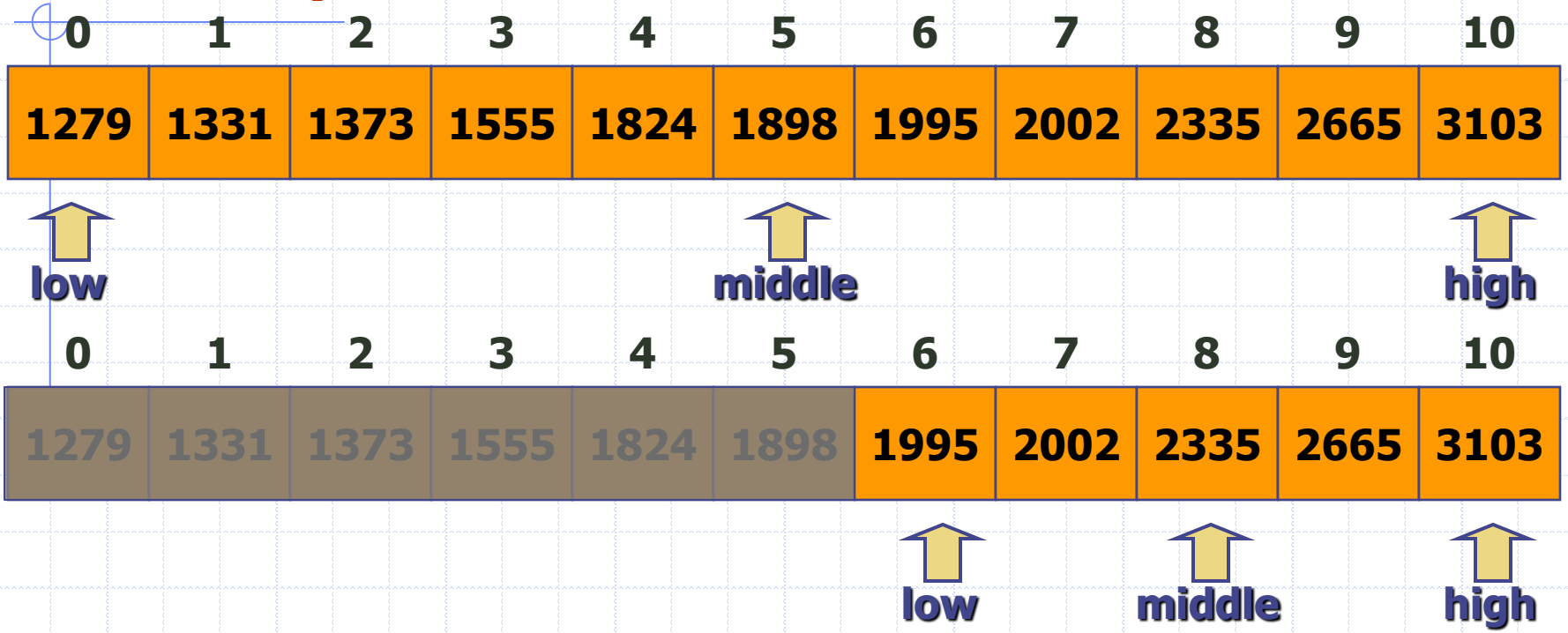
Binary Search

Searching for → **1995**

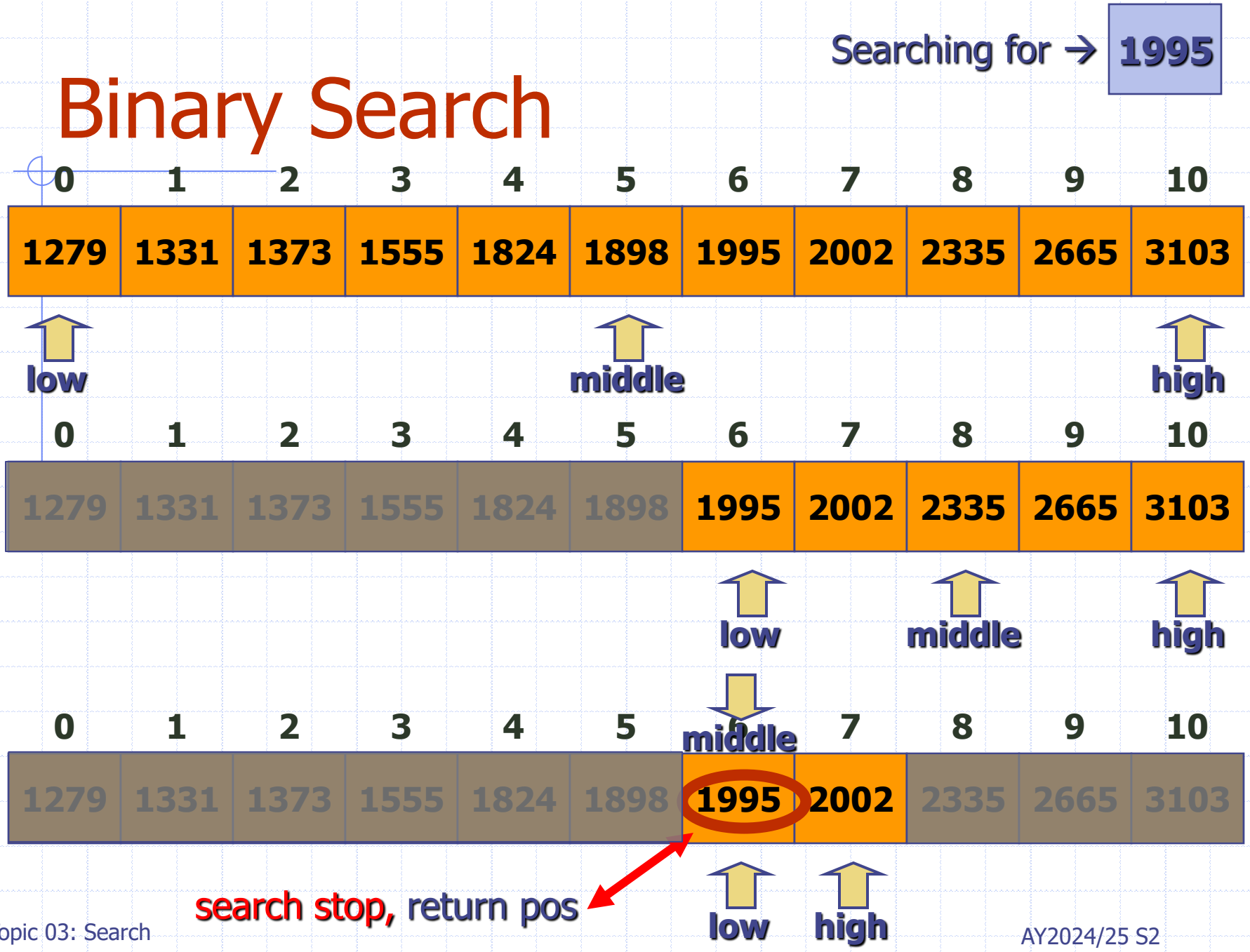


Binary Search

Searching for → 1995

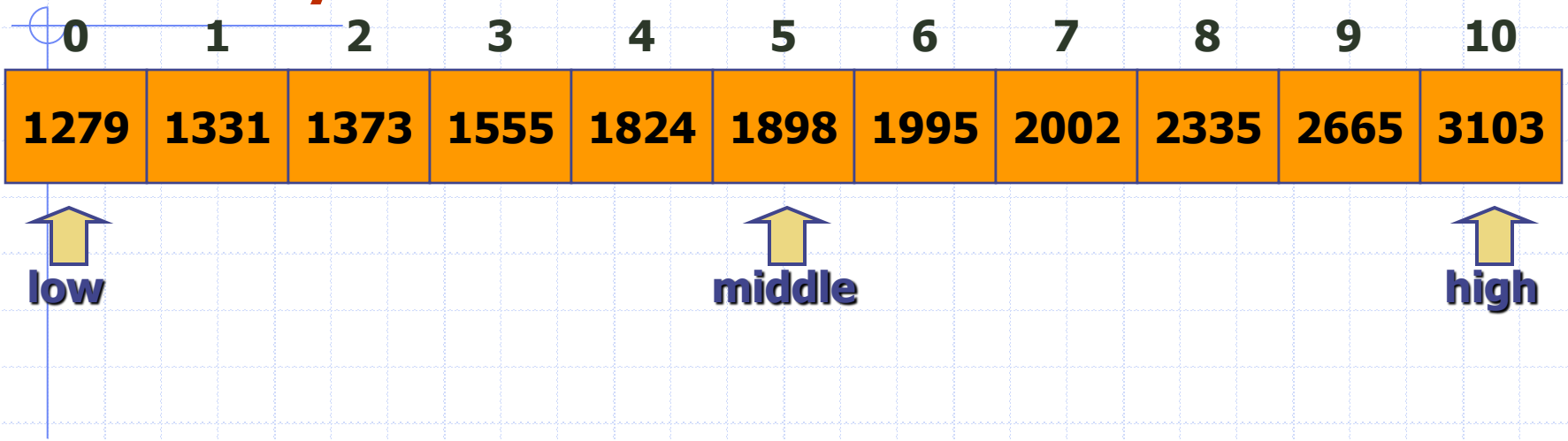


Binary Search



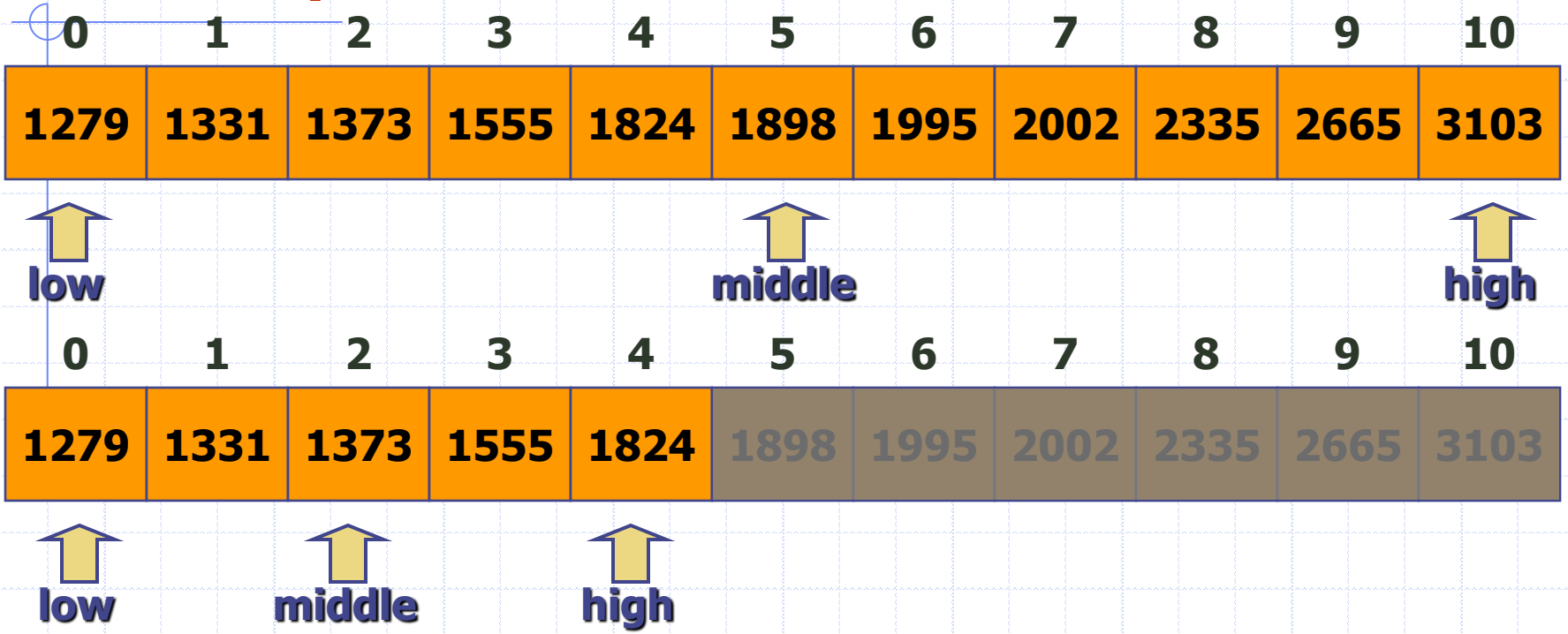
Binary Search

Searching for → **1400**

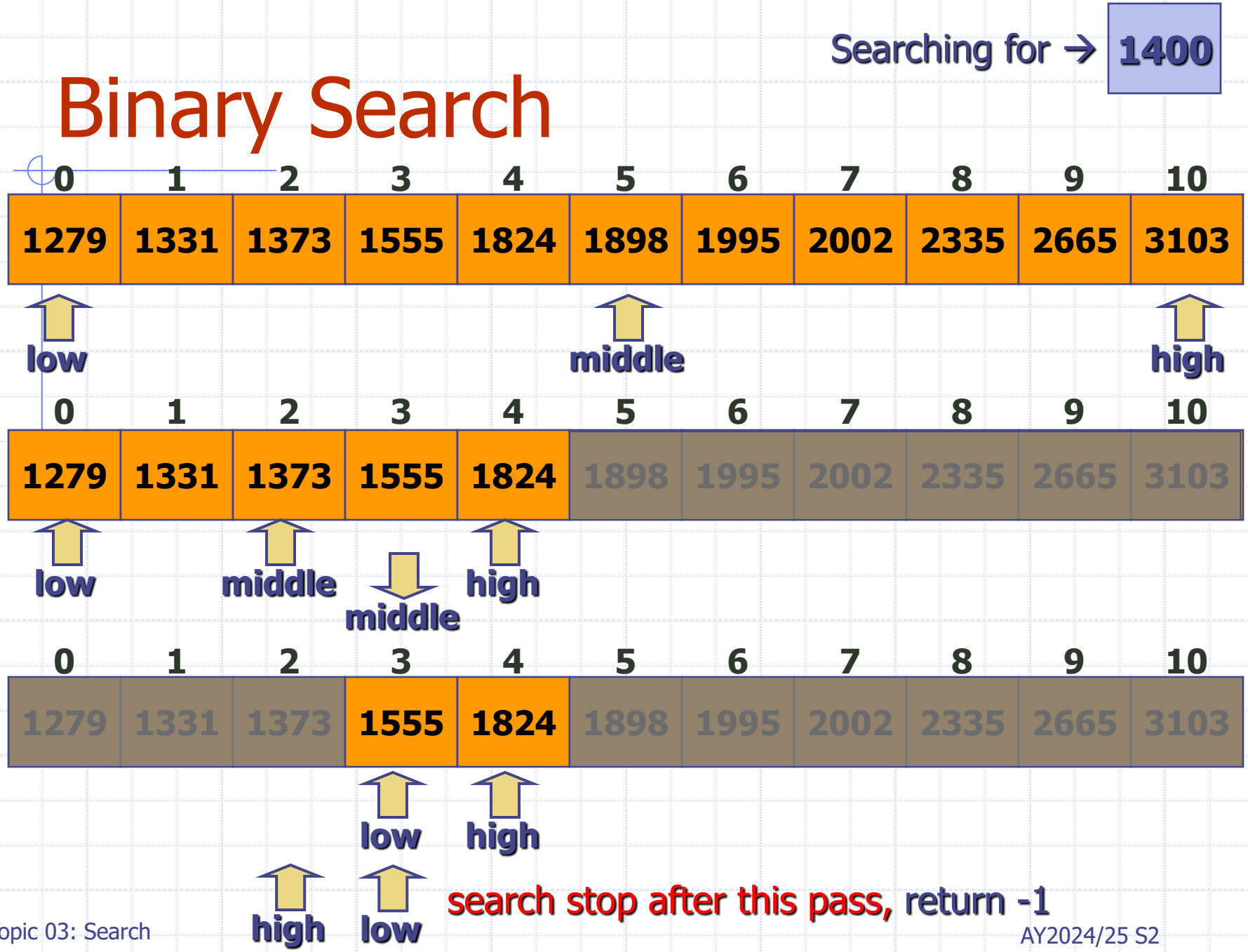


Binary Search

Searching for → **1400**



Binary Search



Binary Search

- **Two situations when a search will stop:**
  - **When search item is found**
    - Return the position in array
  - **When search item is not found**
    - Return -1
    - How do we know that search item is not found?
      - **When high < low**

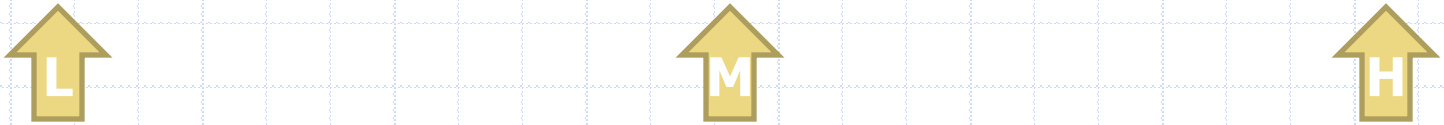
high & low are referring to the index of the list

Binary Search – Activity

42 < 64

Find the index position of the value 64

Index Position	0	1	2	3	4	5	6	7	8	9	10
Array Elements	10	23	25	34	36	42	64	74	87	92	99



Pass	Low Pointer Index Position	Middle Pointer Index Position	High Pointer Index Position	Found (Yes/No)
1	0	5	10	No
2				
3				

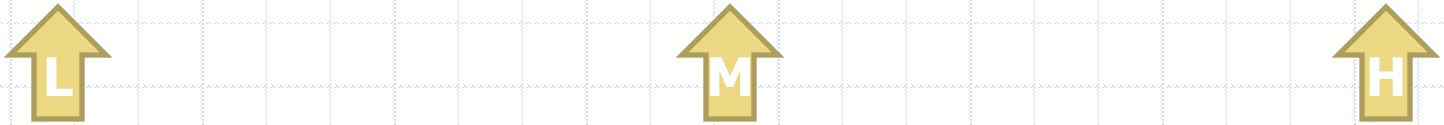
Binary Search – Activity

42 < 64

Move to M + 1

- Find the index position of the value 64

Index Position	0	1	2	3	4	5	6	7	8	9	10
Array Elements	10	23	25	34	36	42	64	74	87	92	99



Pass	Low Pointer Index Position	Middle Pointer Index Position	High Pointer Index Position	Found (Yes/No)
1	0	5	10	No
2	6	8	10	
3				

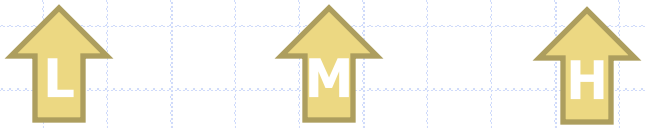
Binary Search – Activity

87 > 64

H move to M – 1

- Find the index position of the value 64

Index Position	0	1	2	3	4	5	6	7	8	9	10
Array Elements	10	23	25	34	36	42	64	74	87	92	99



Pass	Low Pointer Index Position	Middle Pointer Index Position	High Pointer Index Position	Found (Yes/No)
1	0	5	10	No
2	6	8	10	No
3			7	

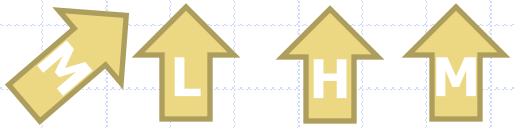
Binary Search – Activity

64 == 64

- Find the index position of the value 64

Move to new M

Index Position	0	1	2	3	4	5	6	7	8	9	10
Array Elements	10	23	25	34	36	42	64	74	87	92	99



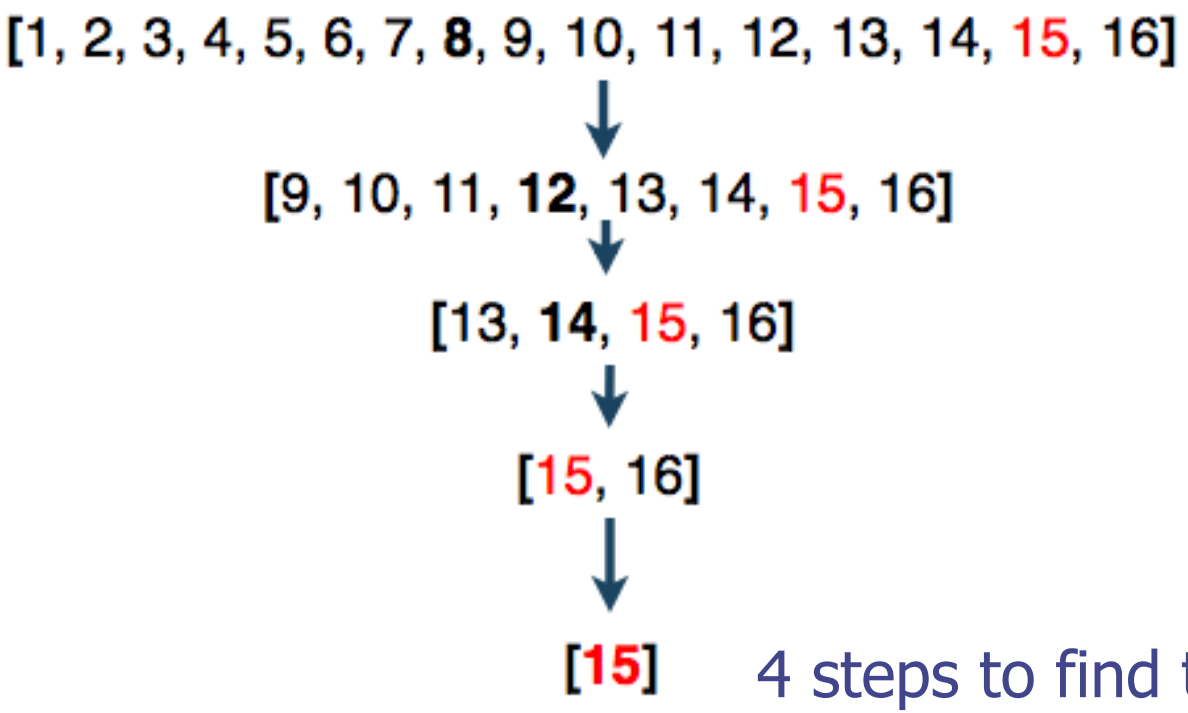
Pass	Low Pointer Index Position	Middle Pointer Index Position	High Pointer Index Position	Found (Yes/No)
1	0	5	10	No
2	6	8	10	No
3	6	6	7	Yes!

Binary Search Complexity

- In Binary search, the best case is O(1), meaning the item of your search is located at the midpoint.
- The worst & average case is log base 2 of n or: $\log_2 n = \text{exponent}$
- Complexity of **O ($\log_2$ n)**
- if there were 16 elements (n = 16), then it would take, at worse case, 4 steps to find the number (exponent = 4).

For example:

- Here is a list of 16 elements.
- It takes 4 operations $\log_2 16 = 4$



4 steps to find the number 15

Video Example

- Let's see a video to sum up Binary Search
  - *https://youtu.be/P3YID7liBug*

More examples

Reference:

https://www.mathwarehouse.com/programming/images/binary-vs-linear-search/binary-and-linear-search-animations.gif

https://www.freecodecamp.org/news/the-complexity-of-simple-algorithms-and-data-structures-in-javascript-11e25b29de1e/

Summary

- Binary Search, list must be Sorted
- More efficient than Linear Search
- Adopt divide and conquer method
- Complexity of **O ($\log_2$ n)**
- Next, we will explore other algorithms such as sorting and many more