6.4 Introduction to Binary Search Trees

THE MULTISET ADT (Behaviours)
AKA COLLECTION ADT

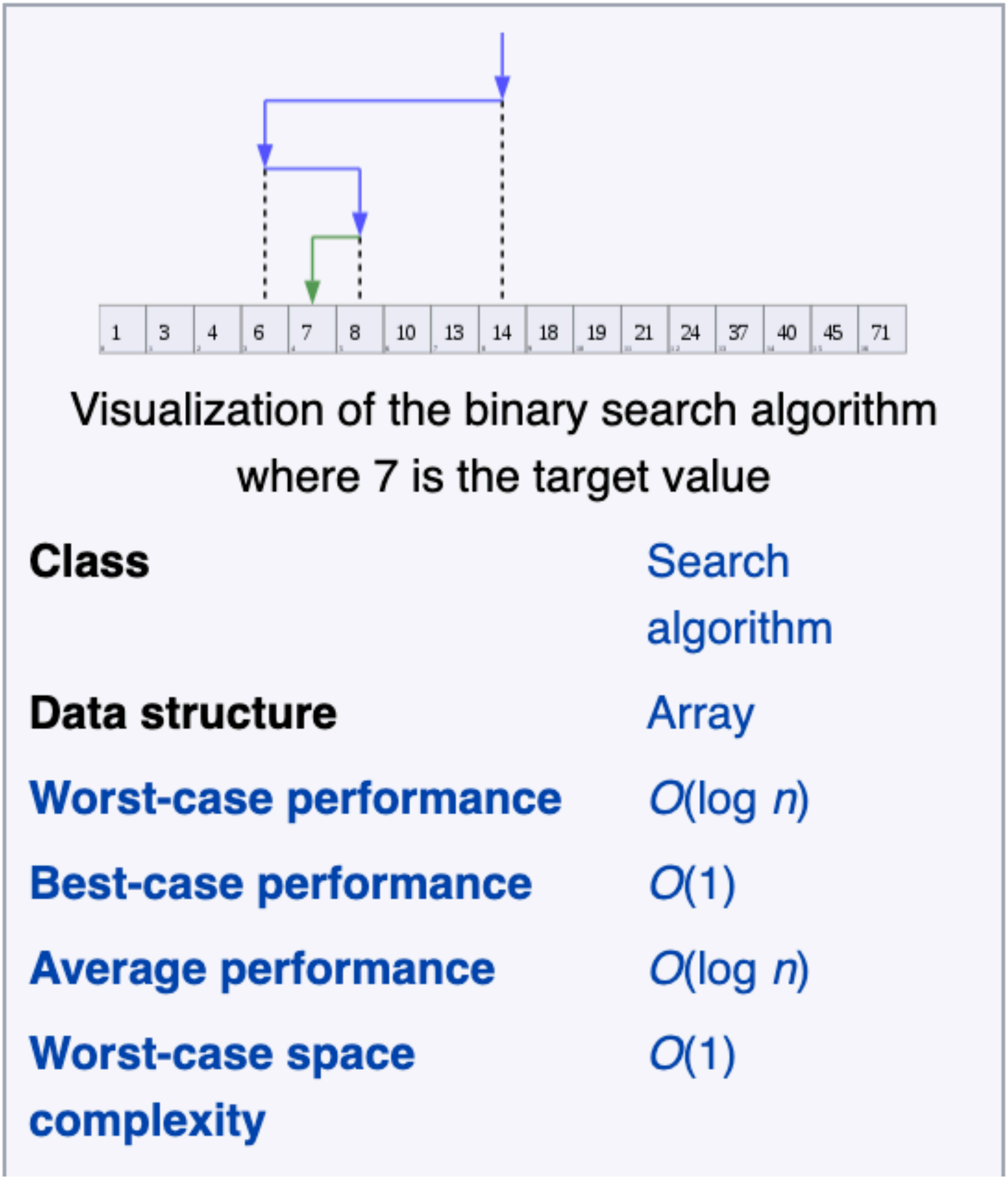
- Check whether the collection is empty
- Check whether a given item is in the collection
- Add a given item to the collection
- Remove a given item from the collection
- Allows user to choose which item to remove unlike container-based ADTs (Stacks & Queues)

SEARCHING IN LISTS:

- Behaviour that we have learned… iterate through every item and check
- Additional structure to data == new, more efficient algorithms WOOOT WOOT 😎 🔥 💪🏿
- If list is sorted… **binary search** is way more efficient!

BINARY SEARCH:

- Compares target value to the middle element of the list.
- If they aren't equal, the half where the target can't lie is eliminated!
- Search continues on remaining half and process is repeated until target value is found.
- If search ends with remaining half empty … target isn't in the list.



Visualization of the binary search algorithm where 7 is the target value

Class	Search algorithm
Data structure	Array
Worst-case performance	$O(\log n)$
Best-case performance	$O(1)$
Average performance	$O(\log n)$
Worst-case space complexity	$O(1)$

BINARY SEARCH TREES:

- Binary structure of trees + binary tree -> "sorted tree"
- A tree in which *every item has at most two subtrees*
- An item in the binary tree satisfies binary search property:
  - Its value >= all items in its left subtree
  - Its value <= all items in its right subtree
- EVERY item in the tree satisfies the binary search property!
- Naturally represent sorted data