Intro to Binary Search Trees

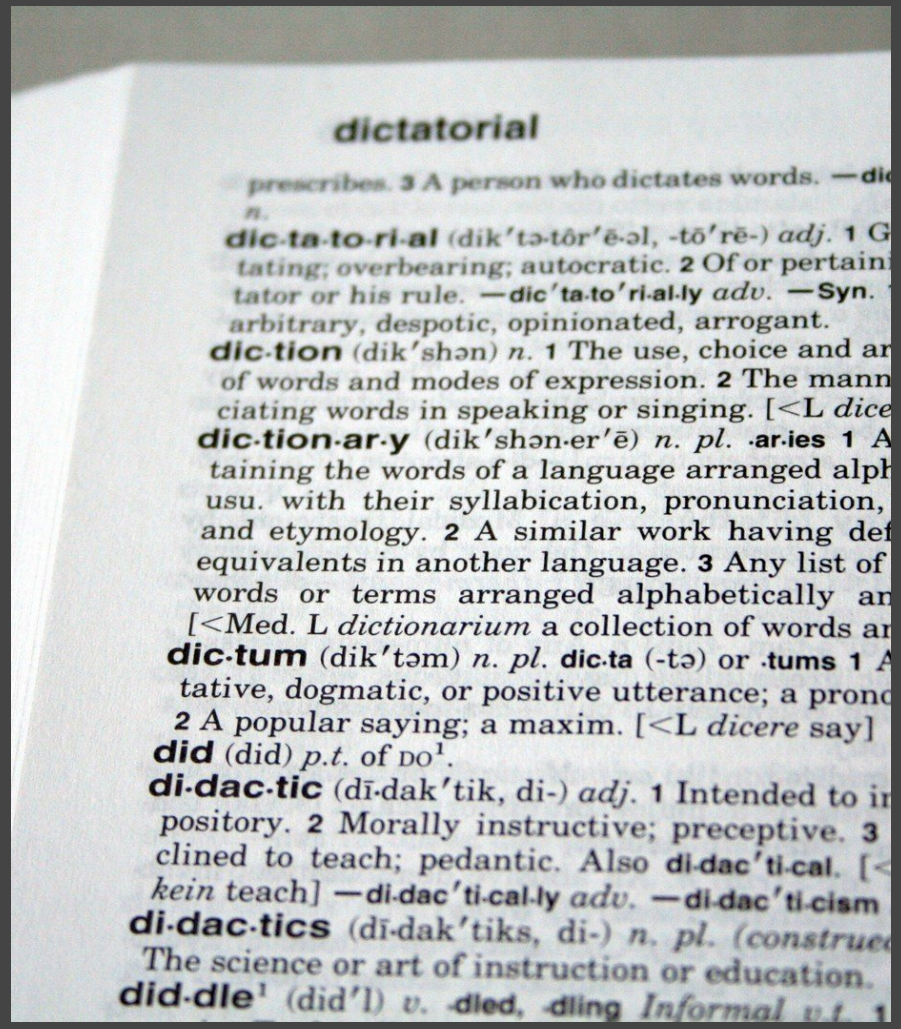
Learn to Code with Rust

Binary Search

- "Binary" means consisting of two parts.
- **Binary search** is a search algorithm for finding a target value in a sorted collection.
- **Binary search** splits the search area into two parts with each turn.
- Each step in a binary search *halves* the search area.

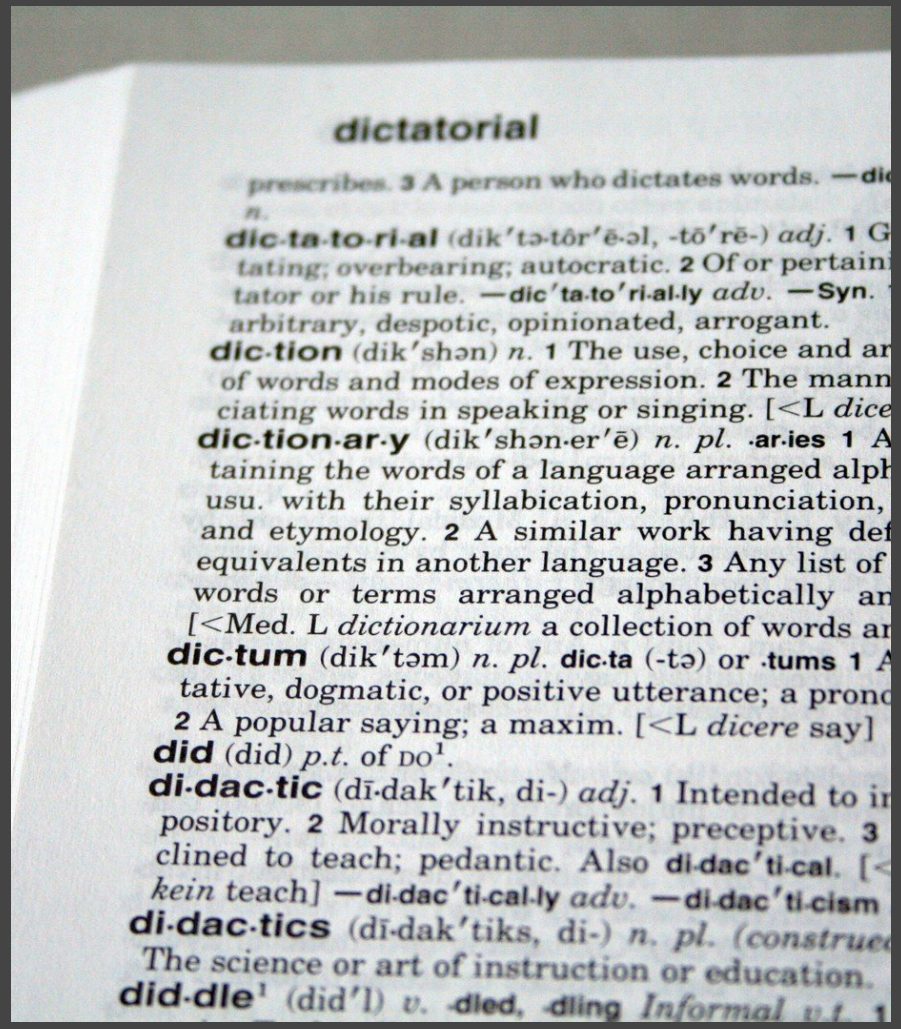
Real Life Example: Dictionary

- A **dictionary** is an ordered collection of words.
- Searching for a target word one-by-one could involve thousands of steps / comparisons.

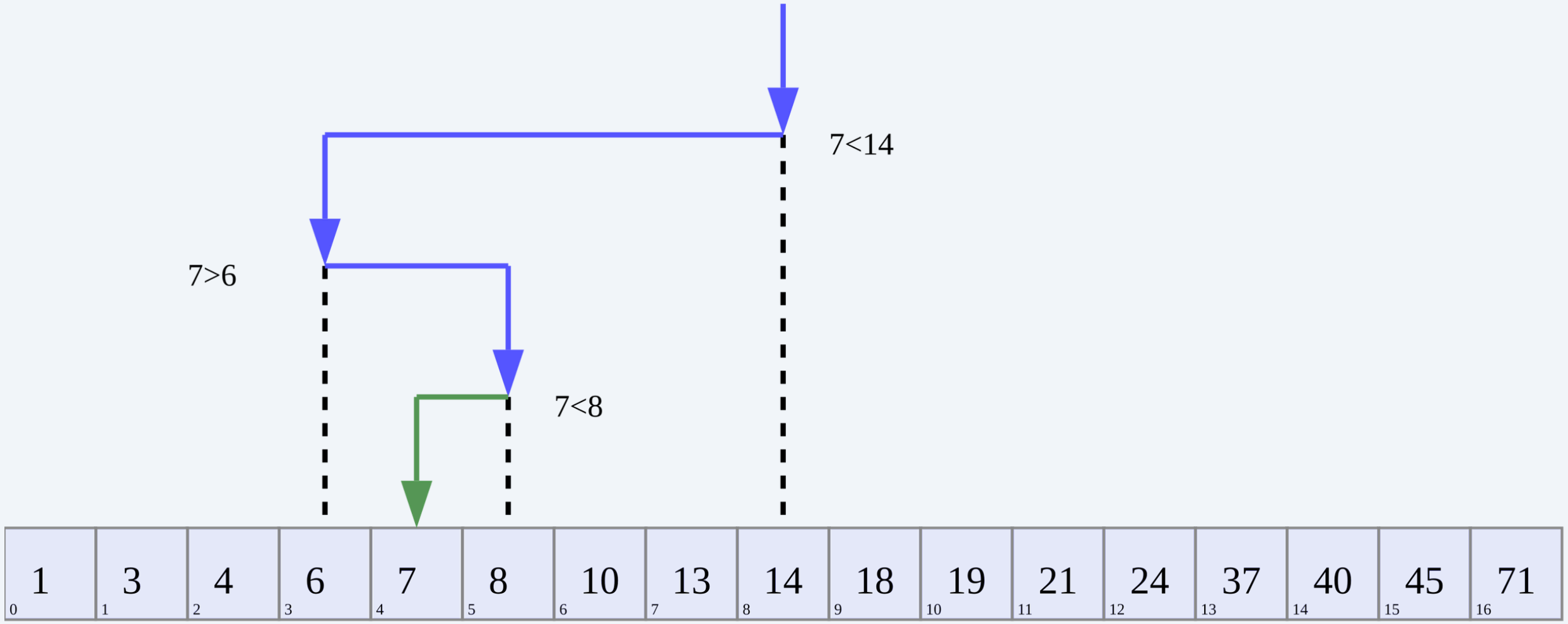


Real Life Example: Dictionary

- Navigate to the middle word in the dictionary and check if it is equal to the target word. If so, the search is over.
- If the target word appears earlier in the alphabet, repeat the process in the first half of the dictionary.
- If the target word appears later in the alphabet, repeat the process in the second half of the dictionary.
- If the dictionary has 1,000,000 words, binary search will locate the target in at most 20 checks (worst case scenario).

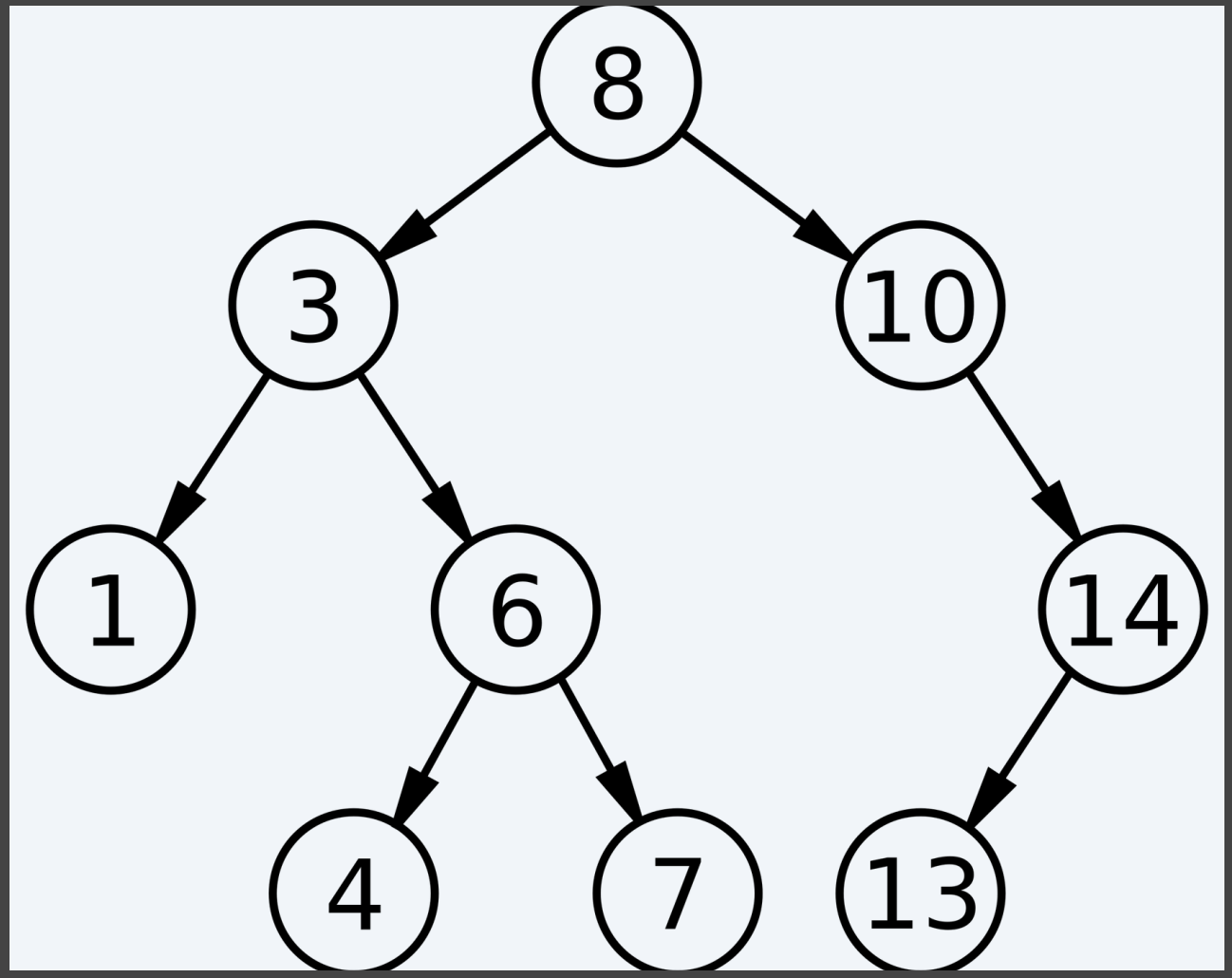


Rust Example: Array/Vector



Binary Search Trees

- A **binary search tree** is a tree data structure that consists of nodes with values.
- Each node has between 0 and 2 children.
- A node's value is greater than the values of all children in its left subtree.
- A node's value is smaller than the values of all children its right subtree.



Balanced and Unbalanced Trees