Binary Search Trees

Performance

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License by Christine Alvarado, Mia Minnes, and Leo Porter, 2015.

By the end of this video you will be able to…

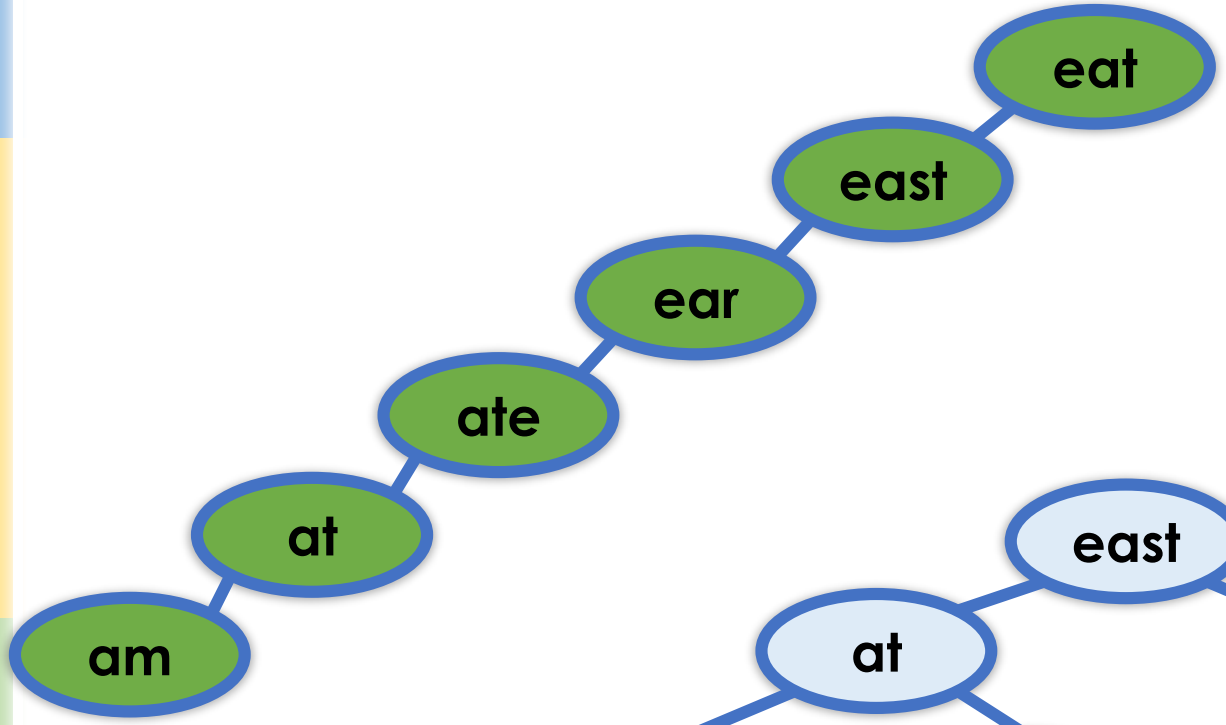
- Explain the running time performance of **`isWord()`** in a BST
- Compare the performance of linked lists and BSTs

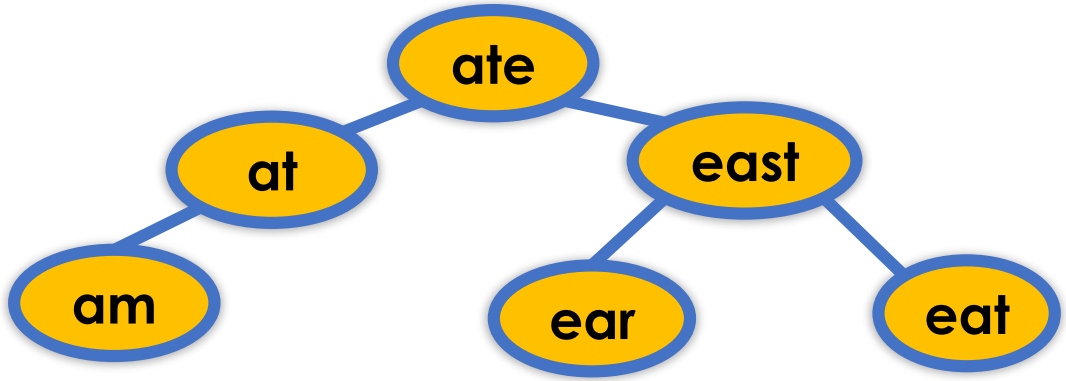
Storing a dictionary as a BST

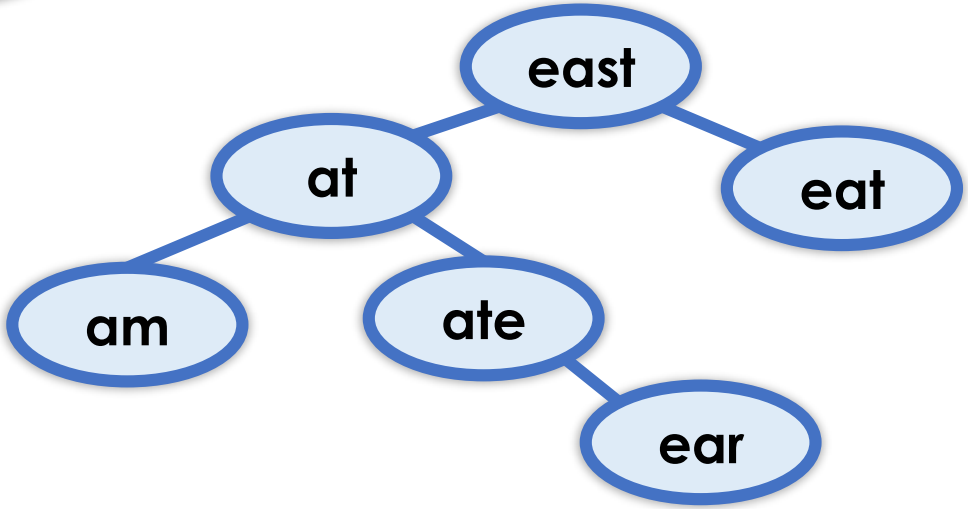
{ am, at, ate, ear, eat, east }

BST?

{ am, at, ate, ear, eat, east }



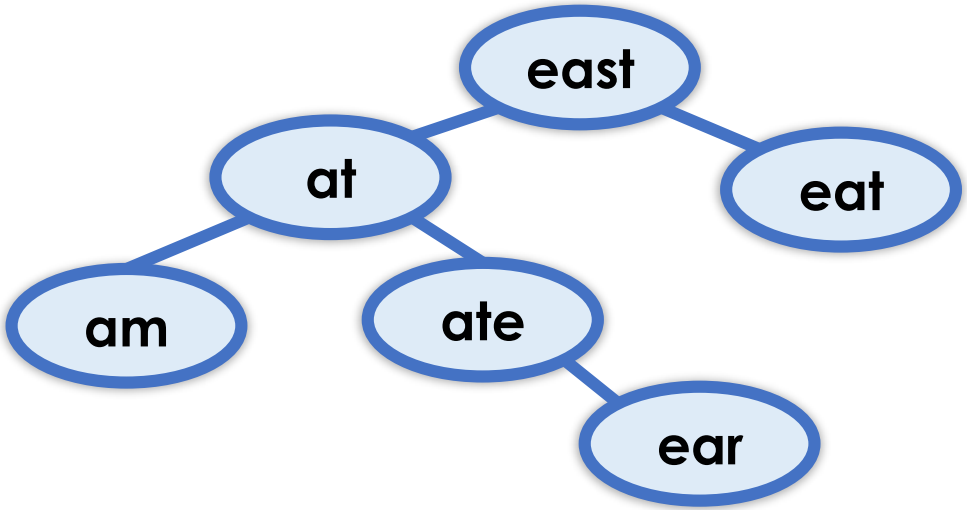




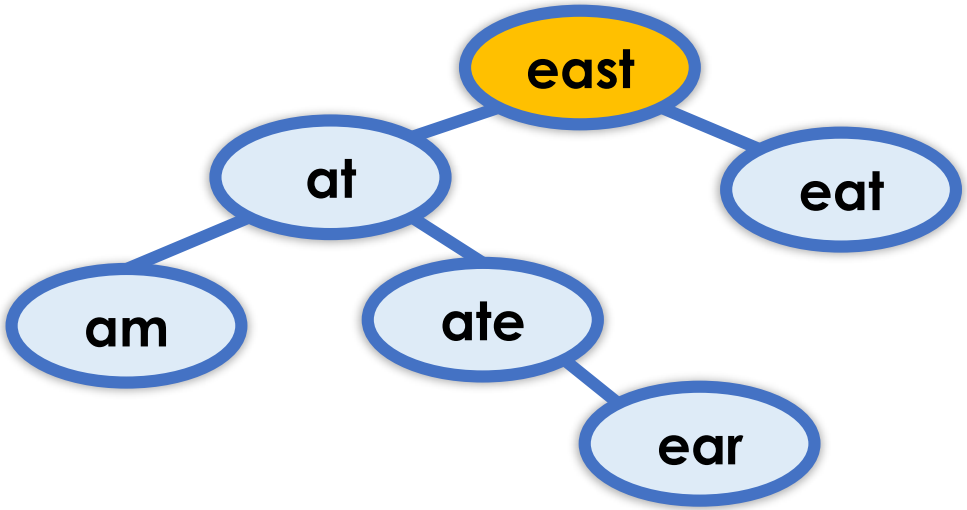
`isWord(String wordToFind)`

- Start at root
- Compare word to current node
  - If current node is null, return false
  - If wordToFind is less than word at current node, continue searching in left subtree
  - If wordToFind is greater than word at current node, continue searching in right subtree
  - If wordToFind is equal to word at current node, return true

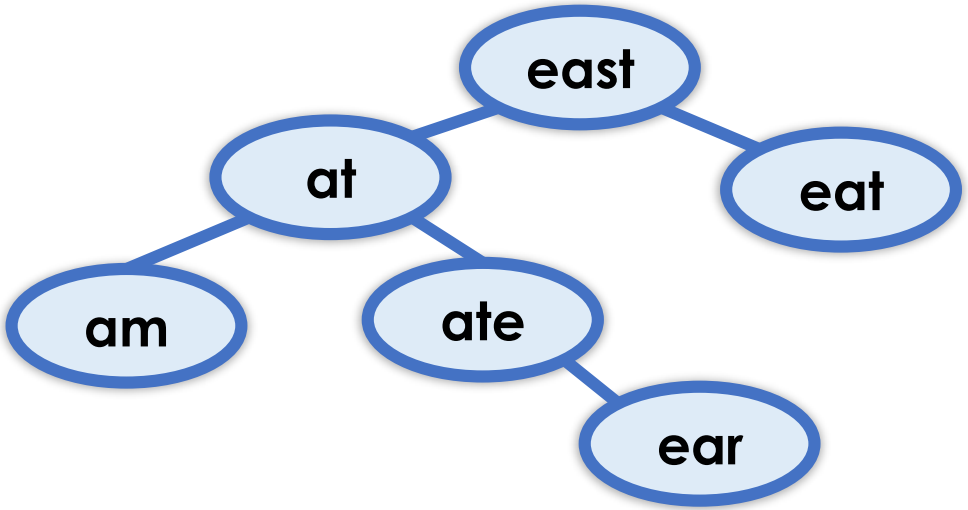
isWord(east)



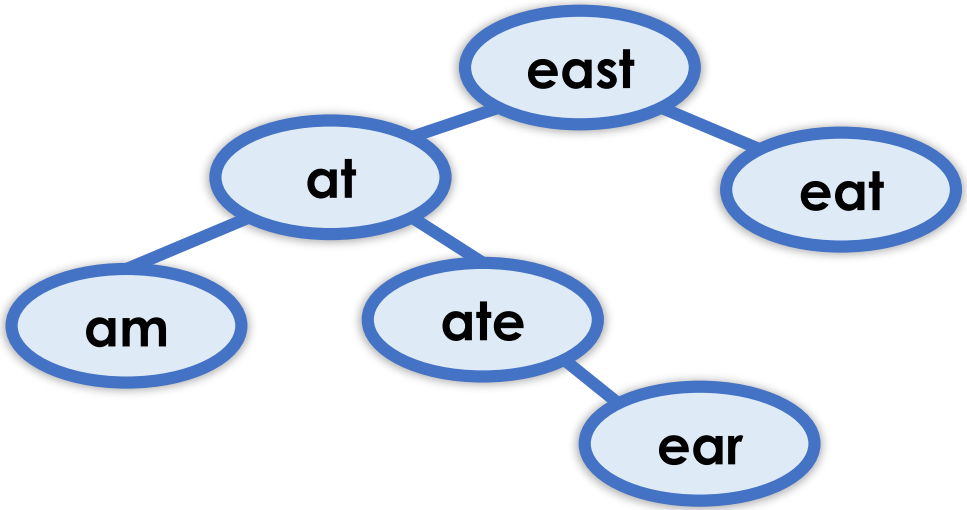
isWord(east)



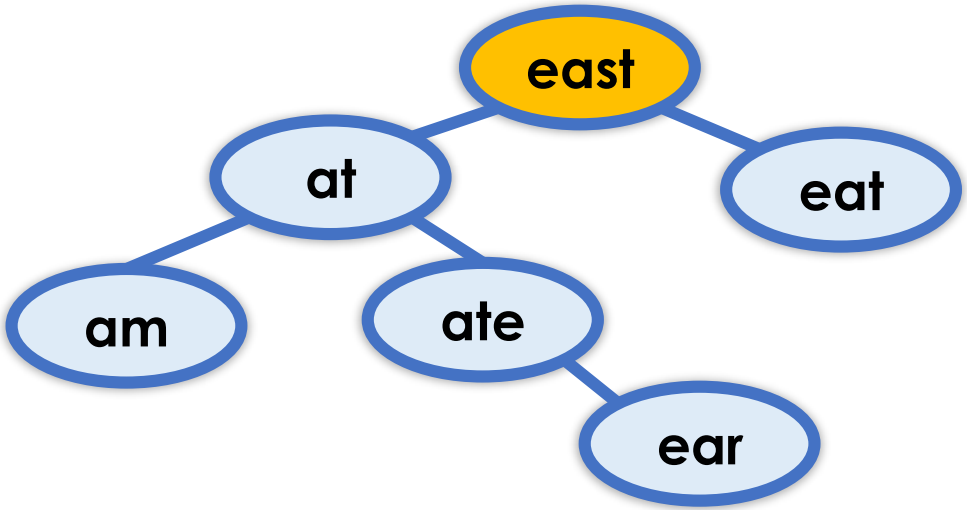
`isWord(east)` Best case: O(1)



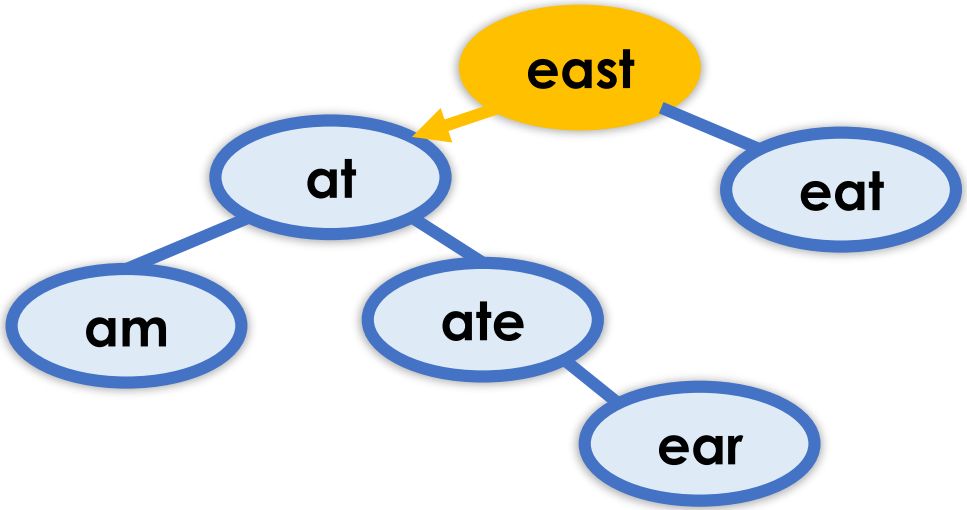
isWord(a)



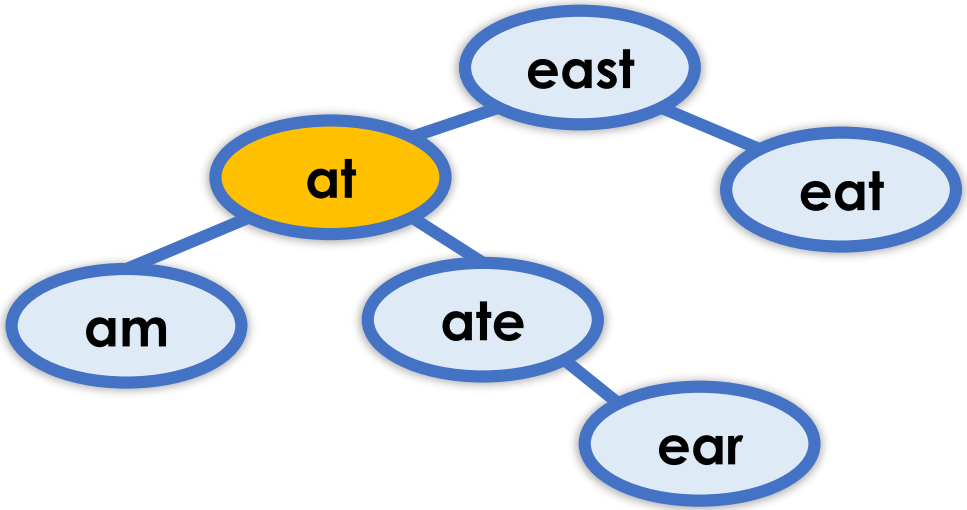
isWord(a)



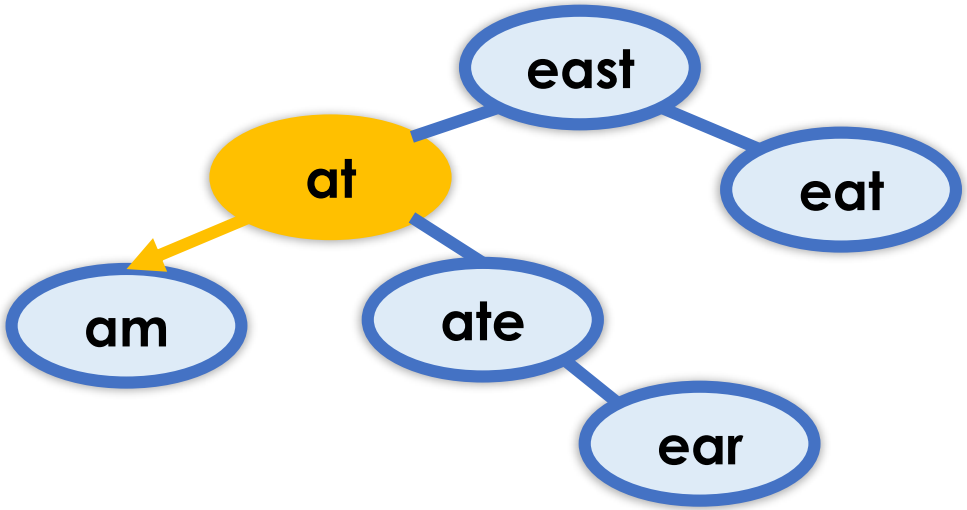
isWord(a)



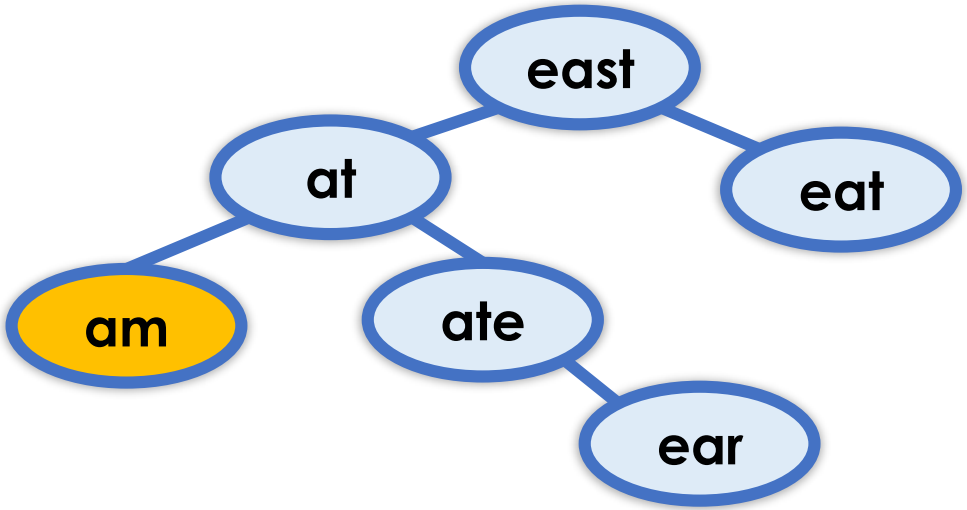
isWord(a)



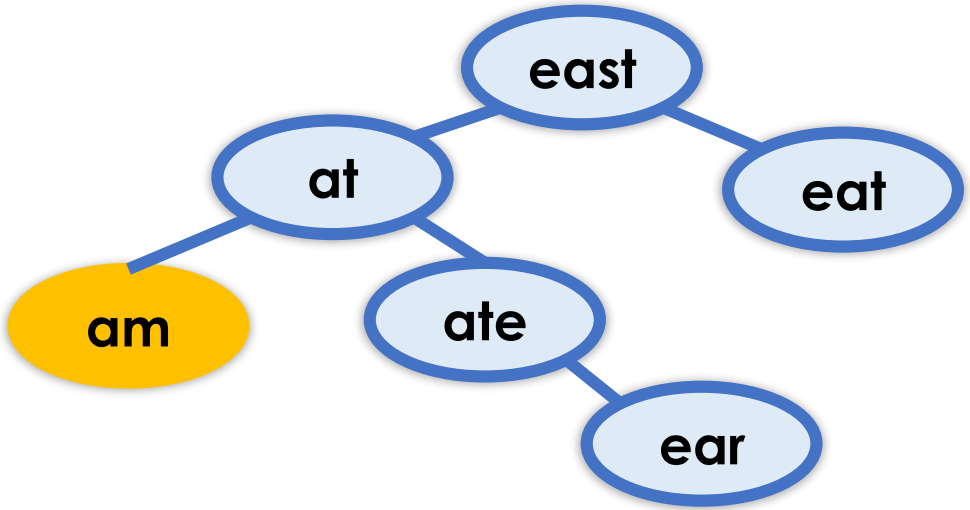
isWord(a)



isWord(a)

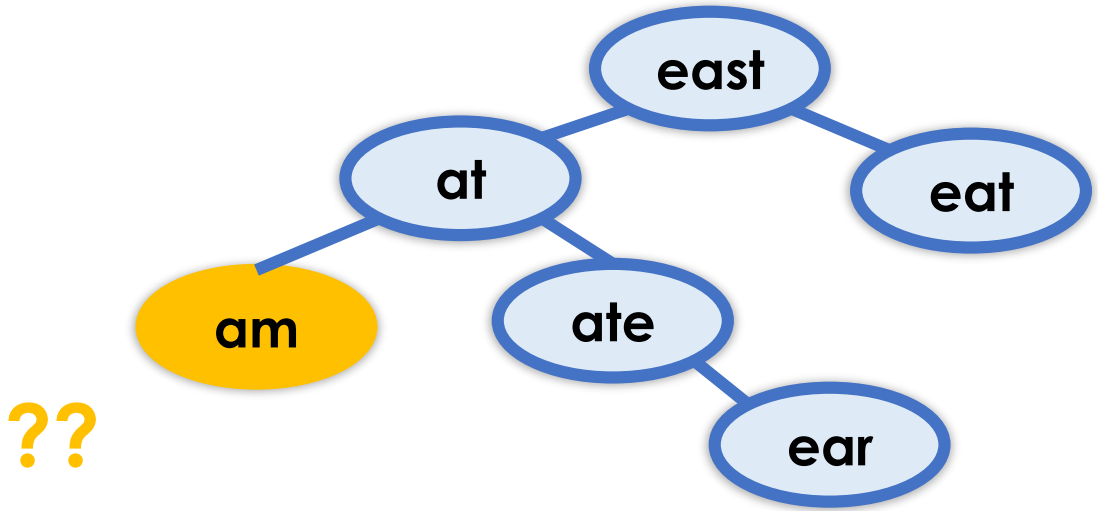


isWord(a)



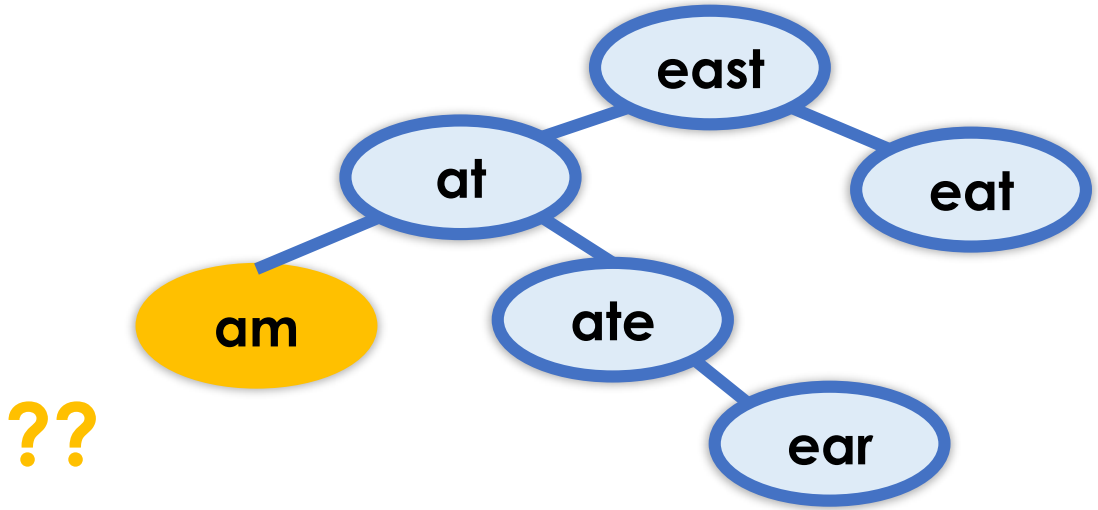
??

`isWord(a)`



Compared with 3 out of 7 words

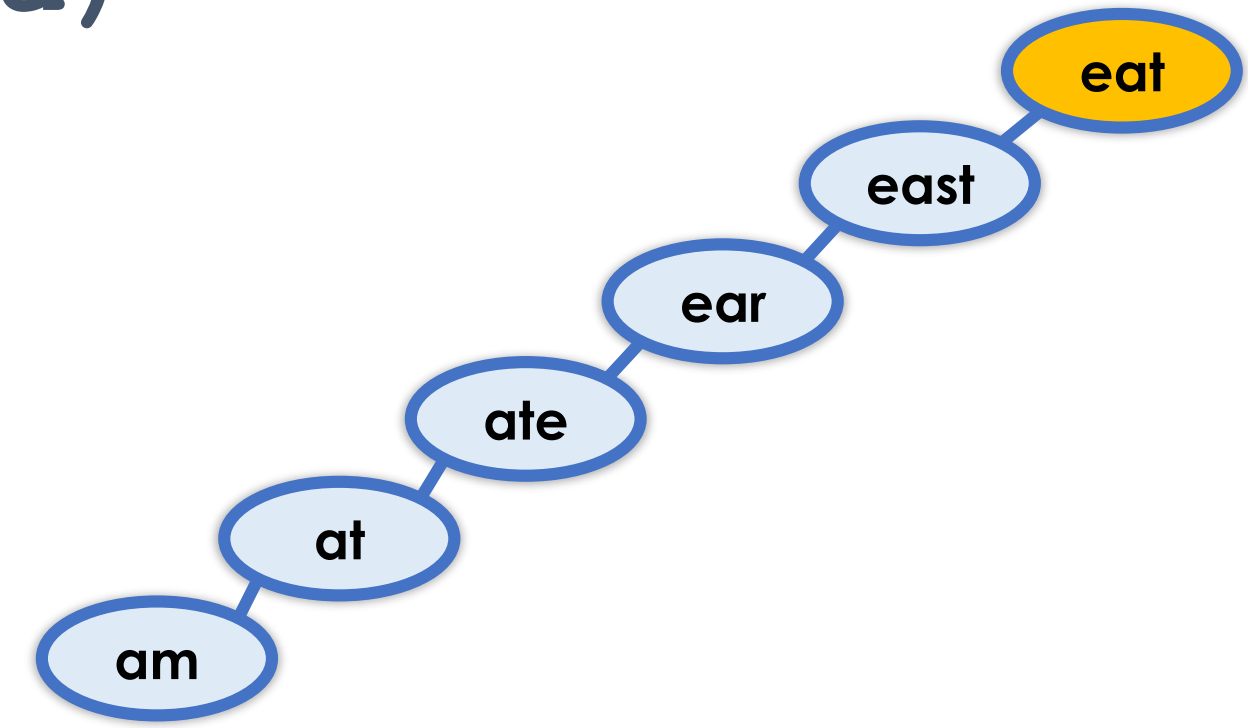
`isWord(a)`



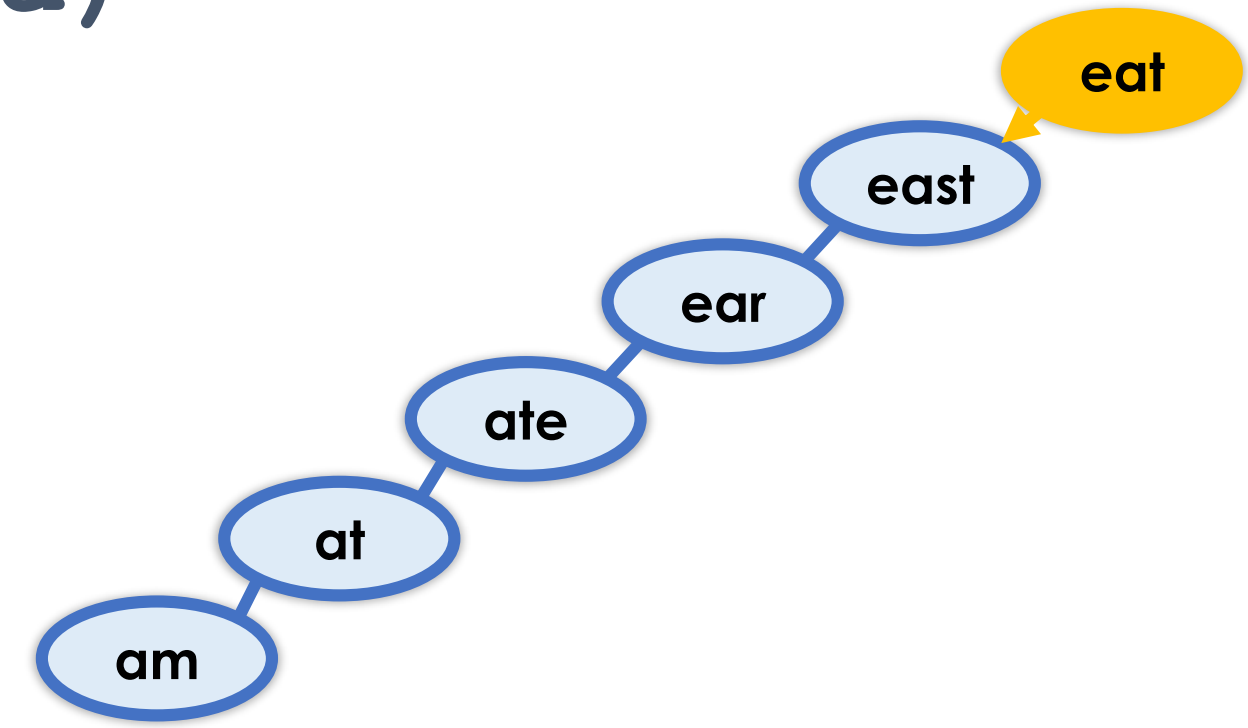
??

Worst case is O(??)

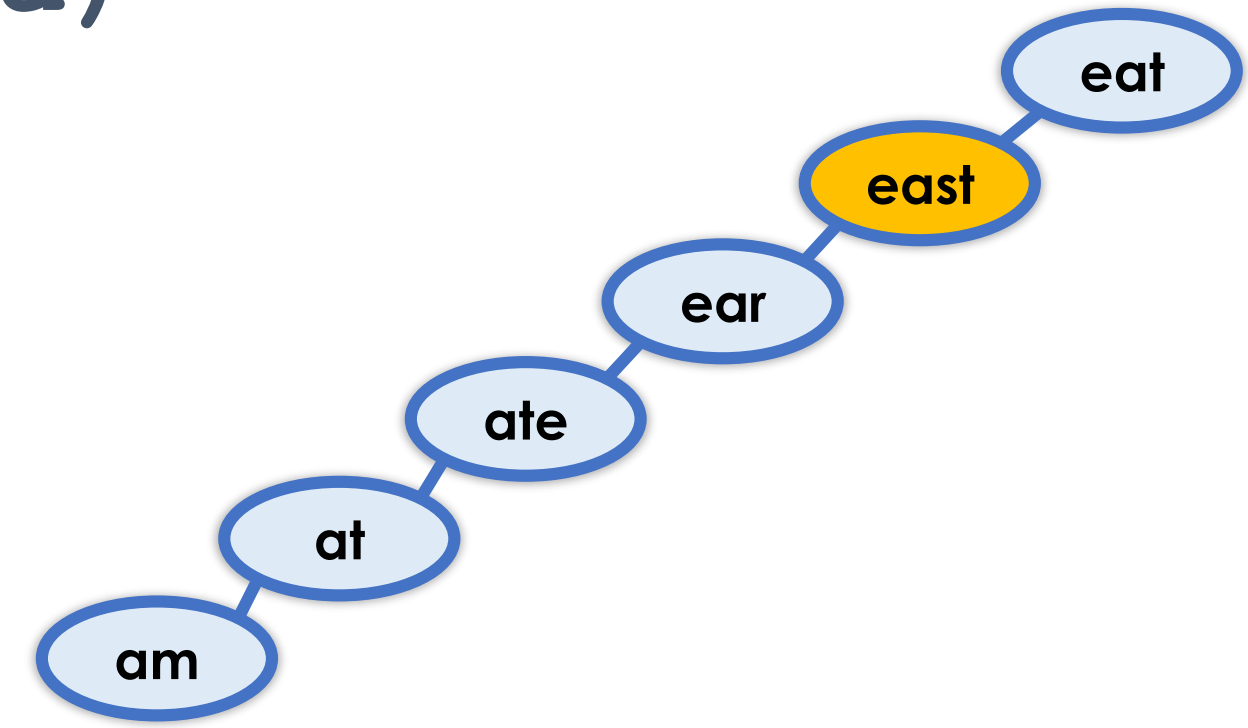
isWord(a)



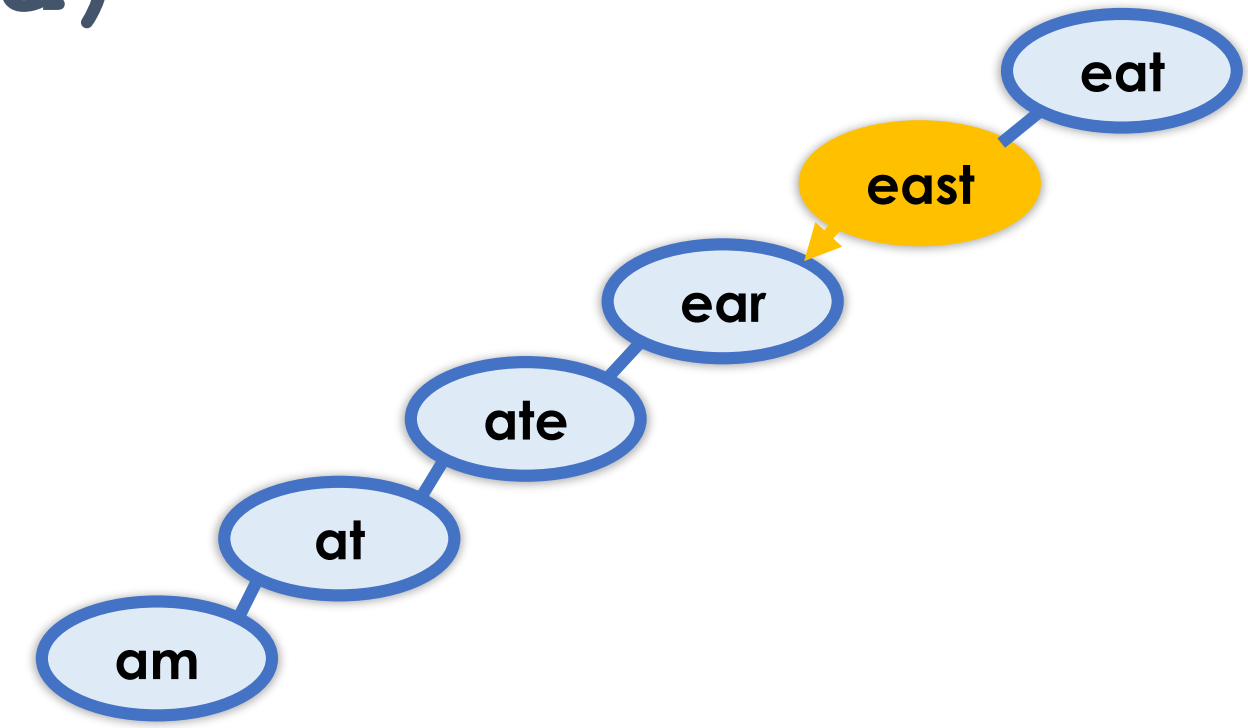
isWord(a)



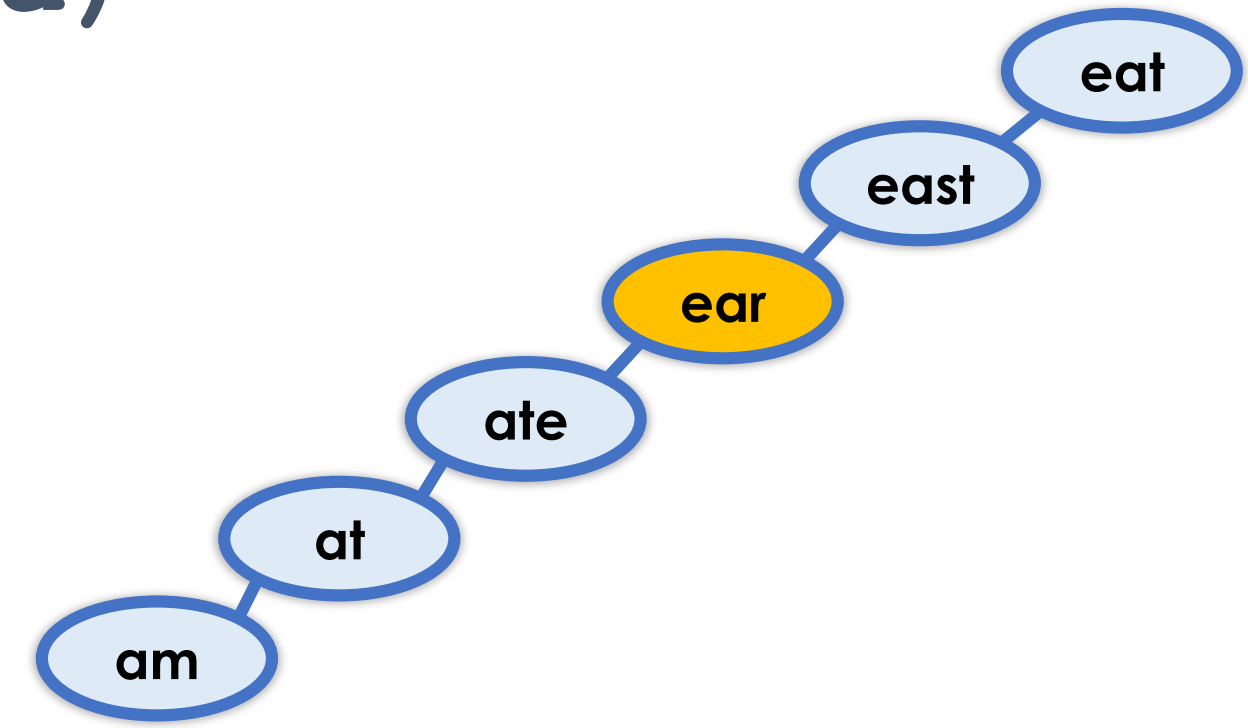
isWord(a)



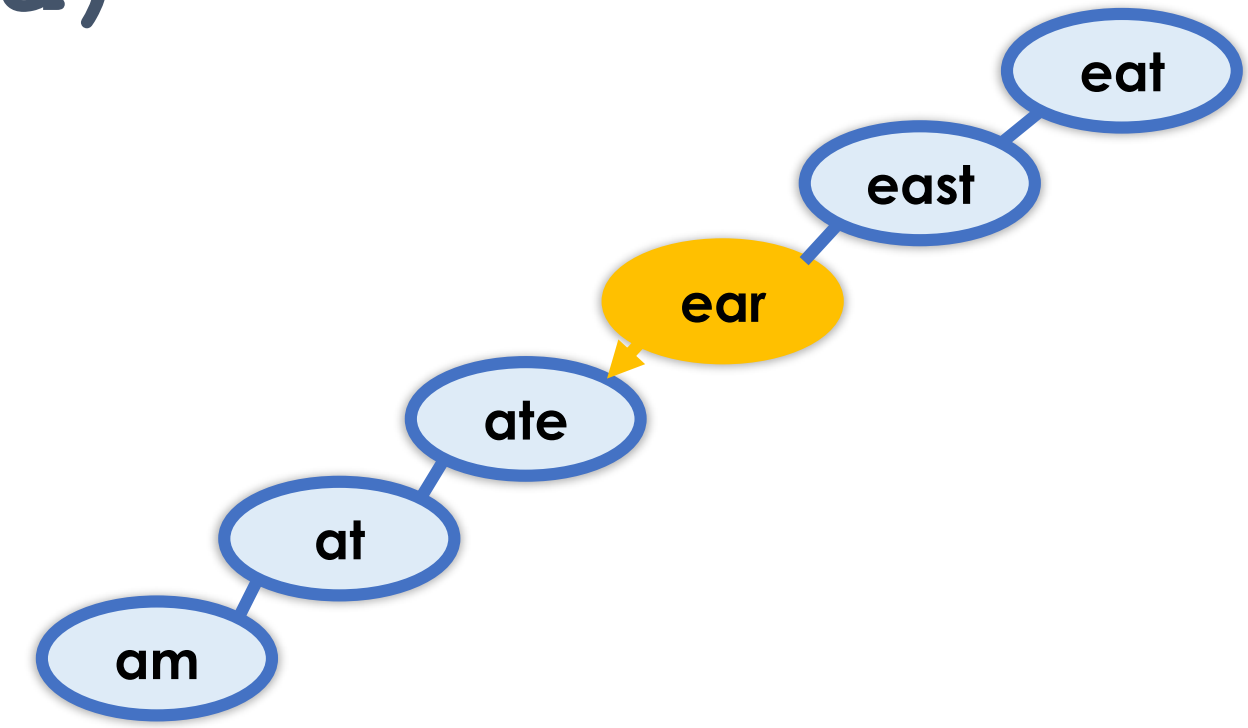
isWord(a)



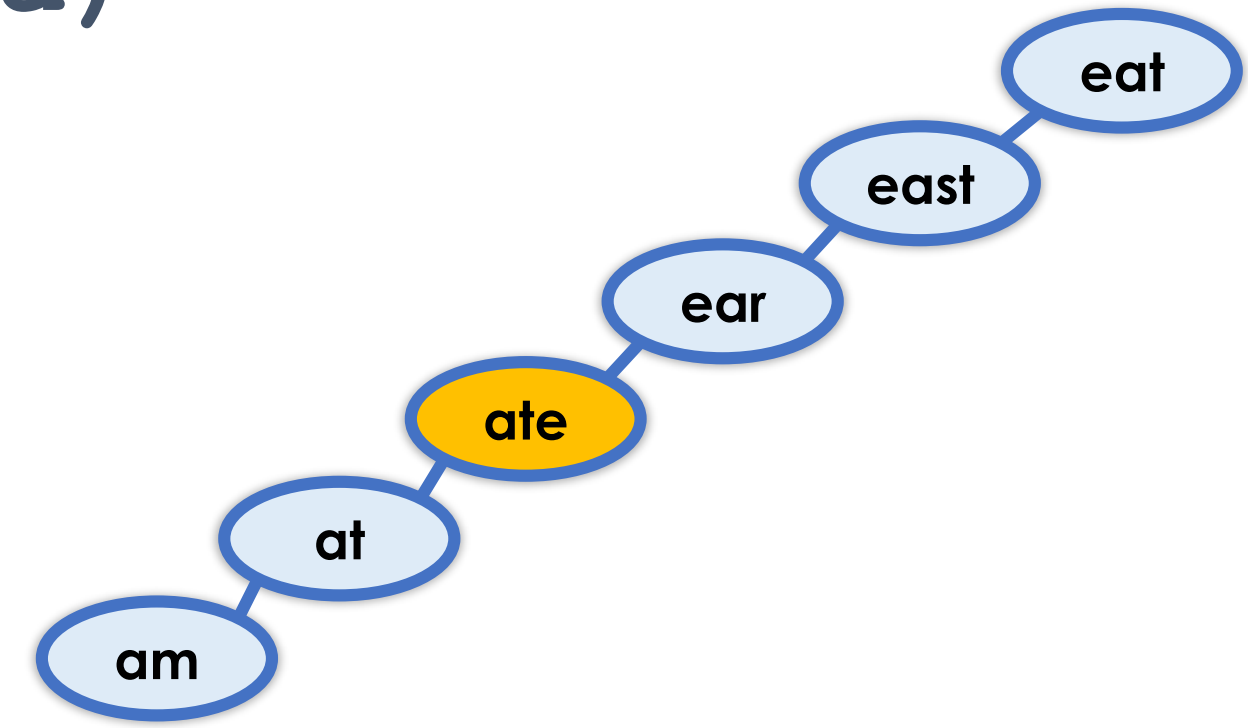
isWord(a)



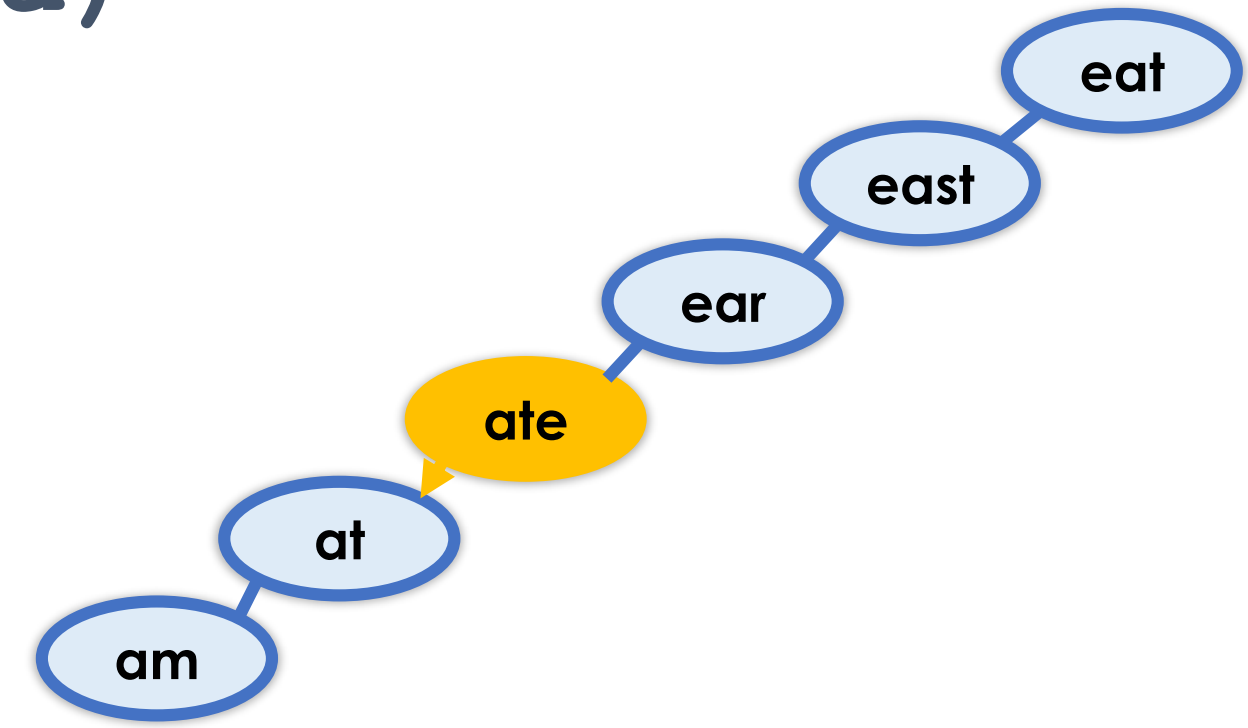
isWord(a)



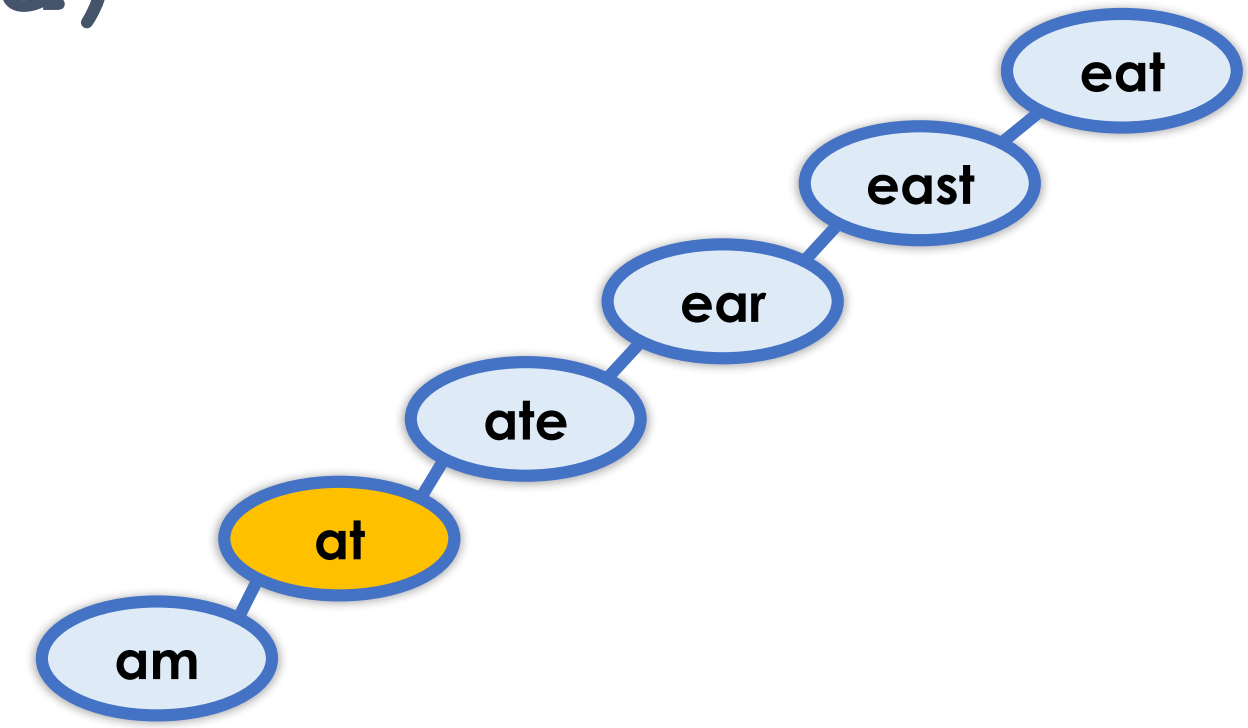
isWord(a)



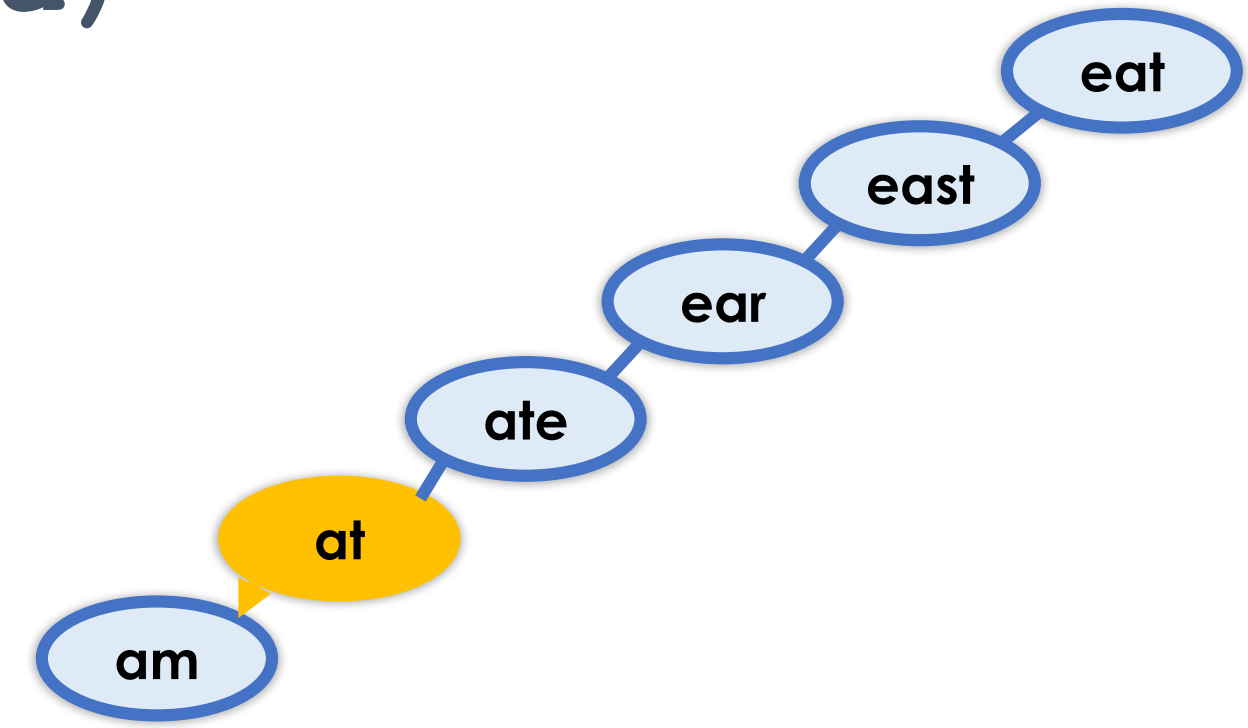
isWord(a)



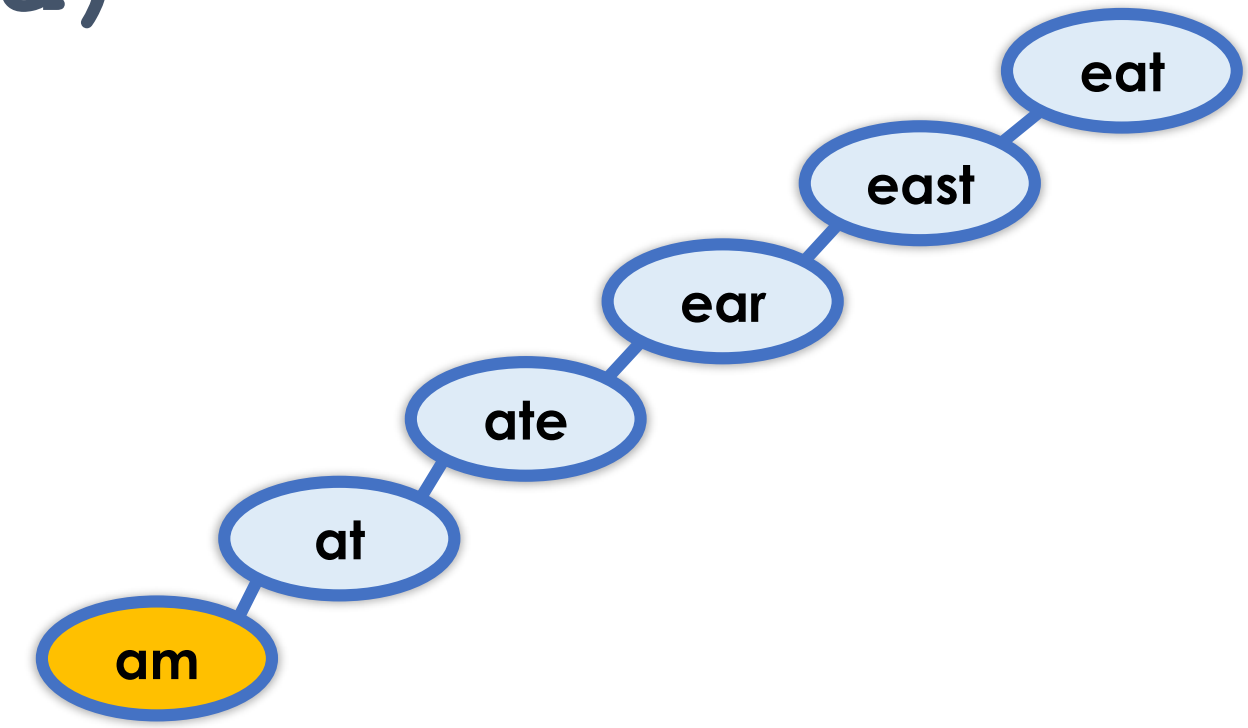
isWord(a)



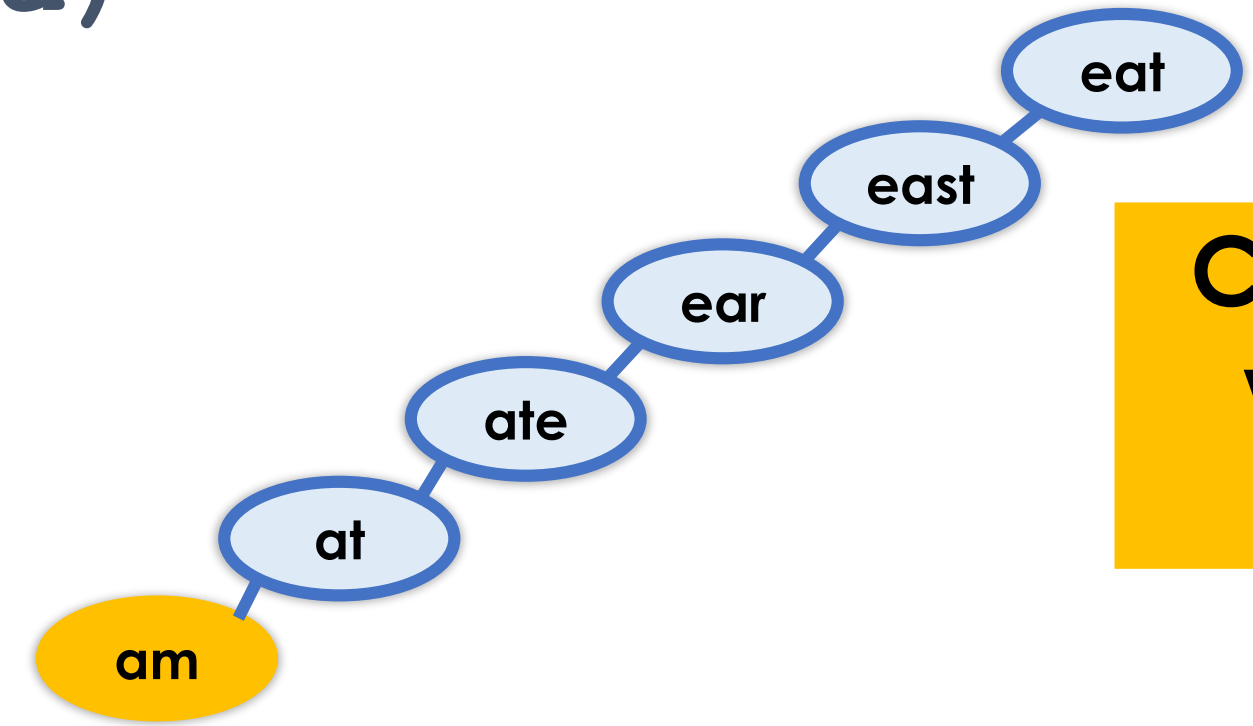
isWord(a)



isWord(a)



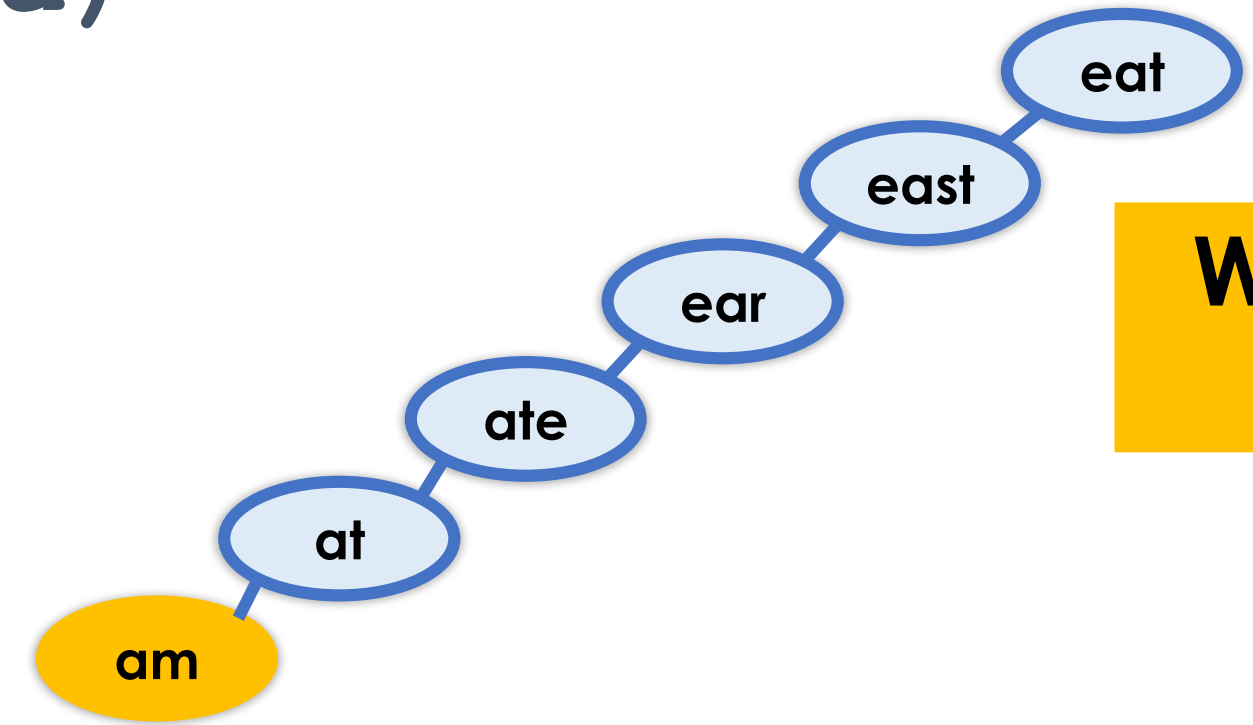
isWord(a)



??

Compared with all 7 words

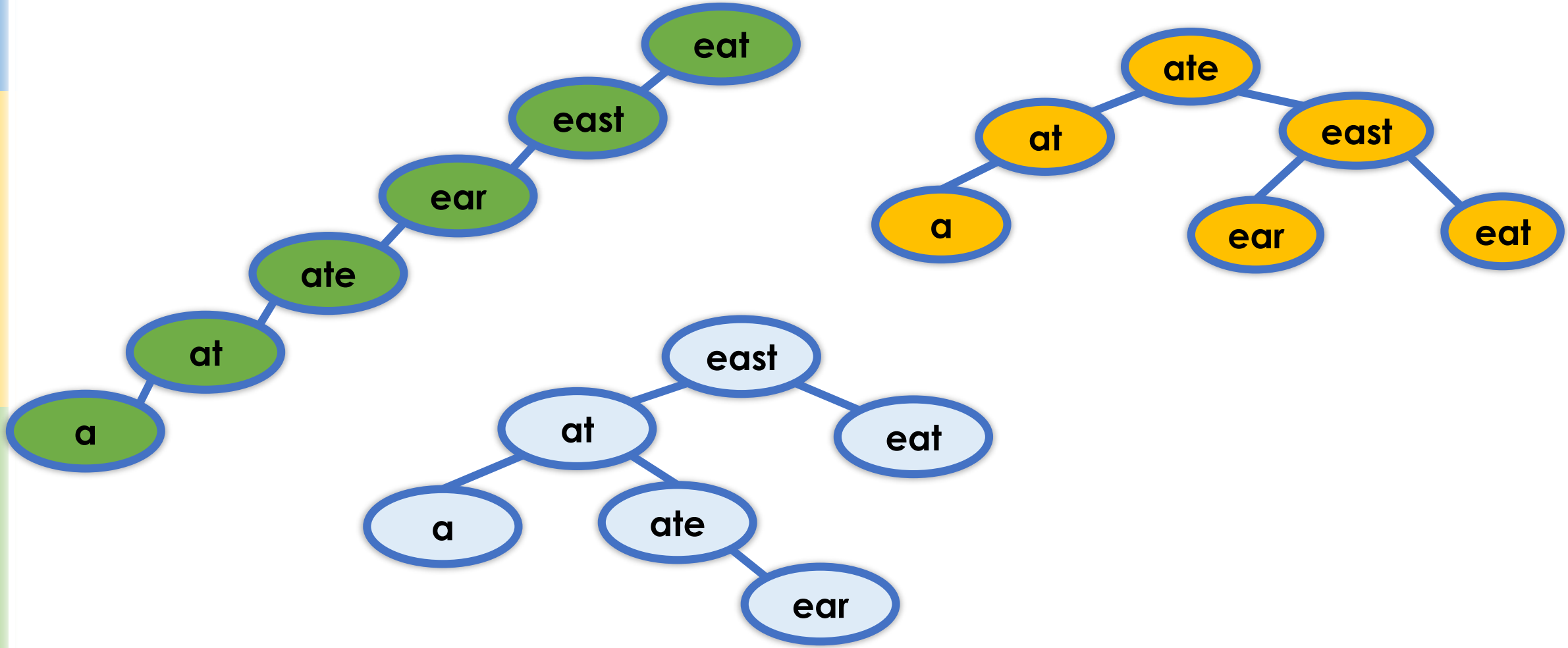
isWord(a)



??

Worst case O(n)

`isWord(String wordToFind)`

- Start at root
- Compare word to current node
  - If current node is null, return false
  - If wordToFind is less than word at current node, continue searching in left subtree
  - If wordToFind is greater than word at current node, continue searching in right subtree
  - If wordToFind is equal to word at current node, return true

Max distance until leaf?