EASTERN WASHINGTON UNIVERSITY
start something big

Prefix Tree (Trie)

Computer Science Department
Eastern Washington University
Yun Tian (Tony) Ph.D.

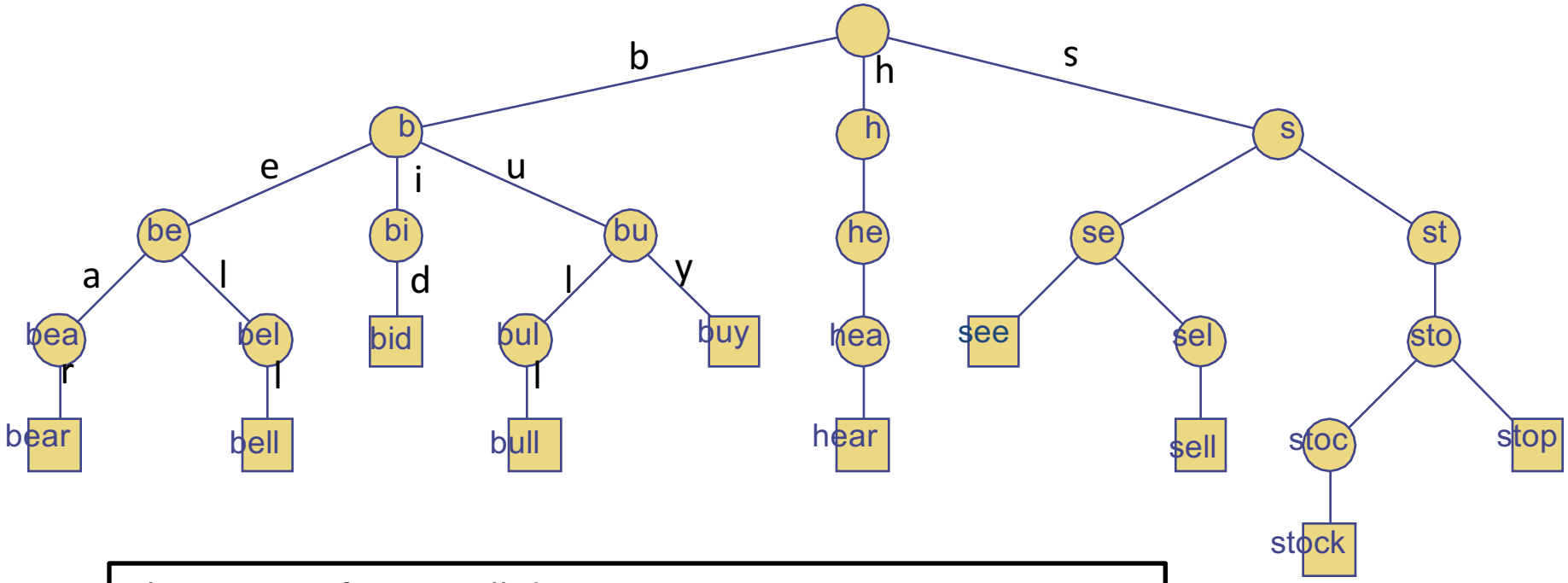
Prefix Tree(also called Trie)

- Concept
- Application
- Demonstration

Motivation

- A prefix tree or a Trie (pronounced as [tri:] or try) is a compact data structure for representing a set of strings, such as all the words in a text. (or a dictionary)
- A Tries supports pattern matching queries in time proportional to the pattern size.(length of searched string) O(d), d is the length of pattern string.

Trie

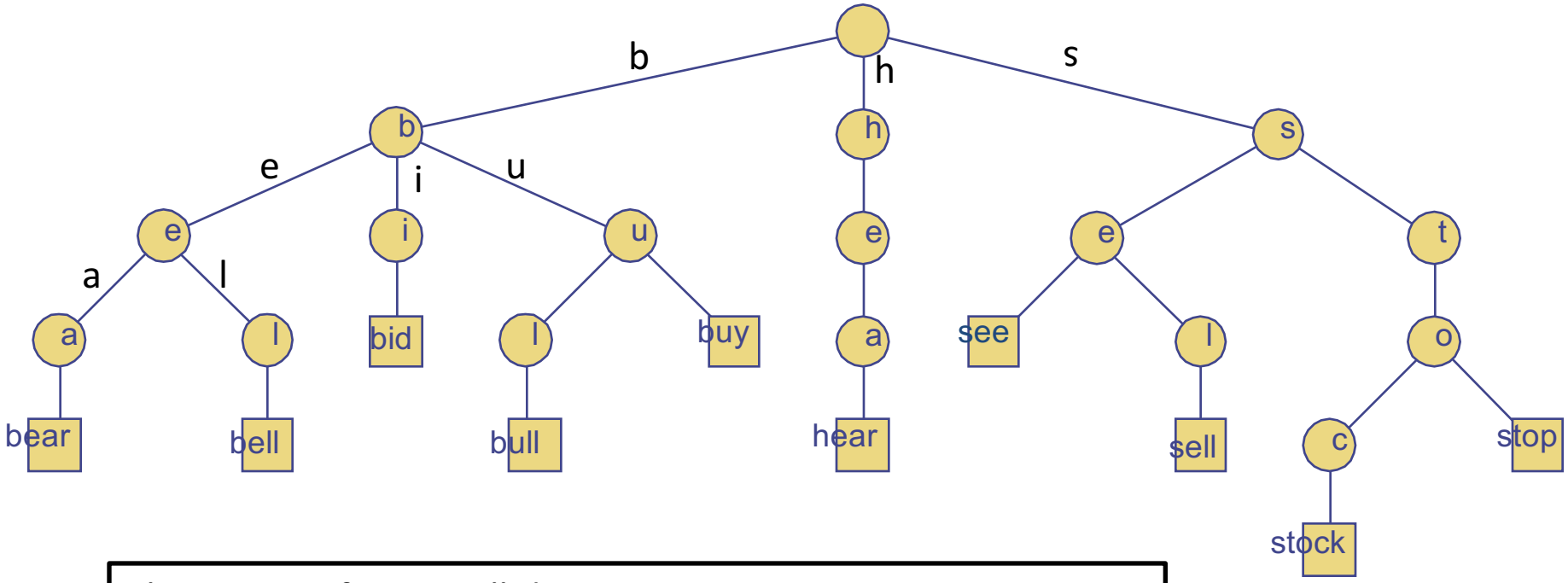


This is a Trie for a small dictionary:
{ be, bear, bell, bid, bull, buy, he, hear, see, sell, stock, stop},
Here, we only allow lower-case English words.

Concept of Prefix Tree(Trie)

- Each node in Trie has a unique string **implicitly** associated with it.
  - The string associated with a Trie node is a **prefix** of some English word.
  - You do not need store the prefix into the Trie node, instead its position in the tree defines the key with which it is associated.
  - All the descendants of a node have a common prefix of the string associated with that node, and the root is associated with the empty string.

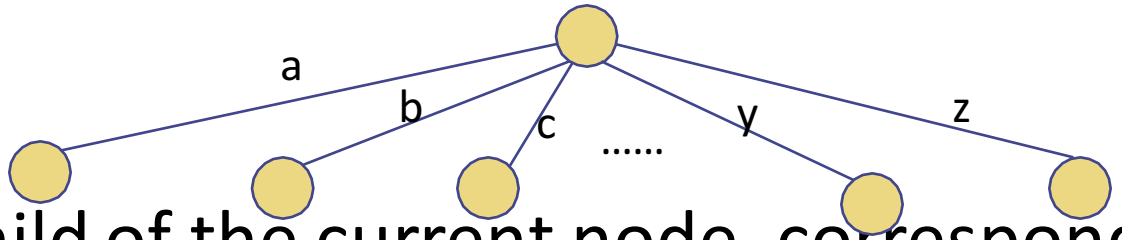
Trie



This is a Trie for a small dictionary:
{ be, bear, bell, bid, bull, buy, he, hear, see, sell, stock, stop},
Here, we only allow lower-case English words.

Concept of Prefix Tree(Trie)

- Each Trie node could have at most 26 children, if we only consider lowercase words.



- Each child of the current node, corresponds to a single letter,
  - Meaning we append the corresponding letter to the current prefix we have in current node, resulting in a longer prefix in its child.
  - Usually, the branches associated with each letter are sorted.

Concept of Trie

- Though Tries are most commonly used for searching character strings, they don't need to be.
  - The same algorithms can easily be adapted to serve similar functions of ordered lists of any construct, e.g., permutations on a list of digits or shapes.
  - A **Bitwise Trie** consists of individual bit in each node, to represent integers, address in computer.

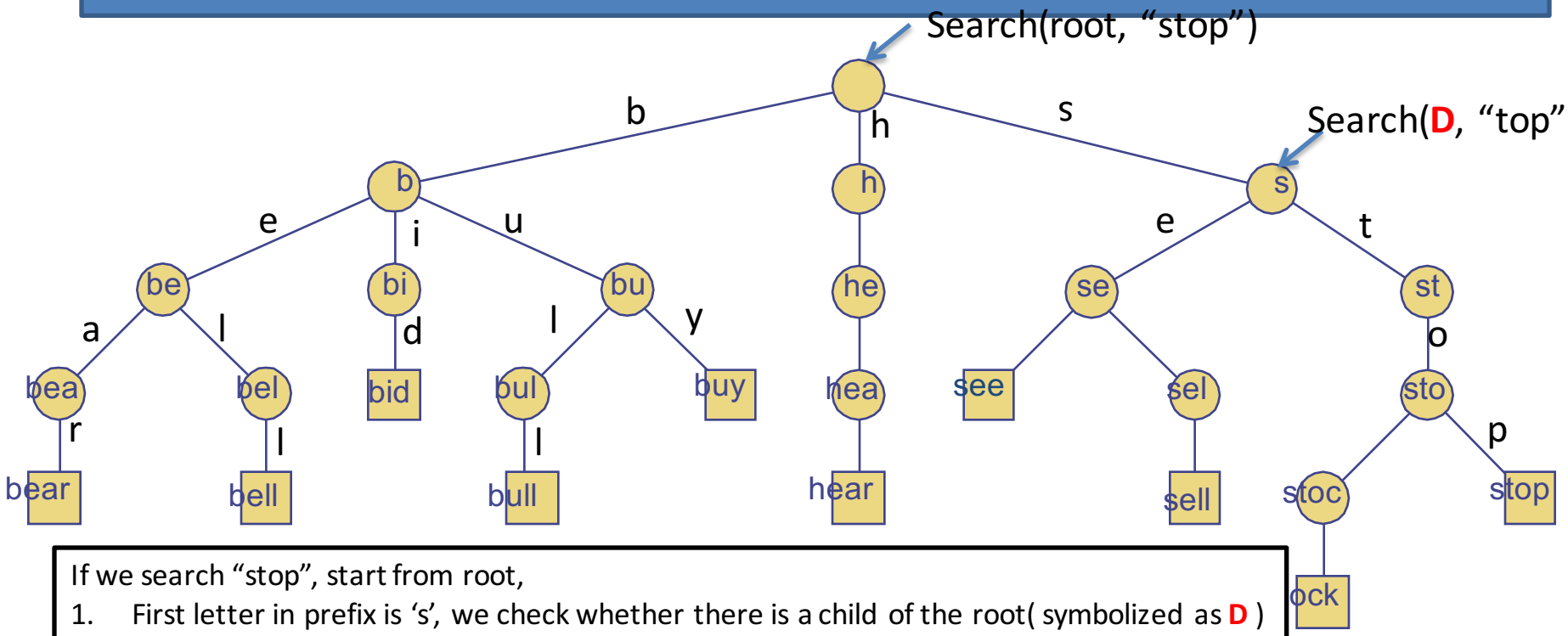
Operations on Trie

- We mainly focus on the insertion and search.
  - We will not talk about deletion this time.
- Once we created the Trie, we can search and find whether a word in the Trie or not?
  - Most frequently used scenario
- Or we can search the Trie, to tell whether a string(partial word) we sent in could possibly be a prefix of any valid English word in the Trie.
  - Or it returns all words that has the prefix passed in.
- A Trie could contain all Valid English words,
  - served as a dictionary.

Operations on Trie

- E.g if we search “zddd”, the trie returns that No English words start with “zddd”.
- E.g. if we search “bu” in the Trie, it returns the words that possibly have that prefix are: bull, buy, …etc by using the small dictionary on slide 6.
- We will use this feature in next project.

Search a String in Trie



If we search “stop”, start from root,

1. First letter in prefix is ‘s’, we check whether there is a child of the root(symbolized as **D**) that is associated with ‘s’,
2. If there is not such child, return not found.
3. If this is a base case, return true.(prefix to be searched contains one letter, and current tree node has a child associated with that letter.)
4. Otherwise, search the rest of the prefix →’top’ in the subtree rooted at **D**.

Next Class

Other trees and project ideas