Program Structure and Algorithms

Sid Nath

Lecture 8

Agenda

- Administrative
  - Midterm today!
- Lecture
  - Tries (~1hr)
- Midterm (remaining time, must submit after 2h 30min)

Tries (Prefix Trees) [Optional]

- Tree-based data structure to store collection of strings for efficient reTRIEval
- Useful to implement dictionaries, autocompletions, sort collections of strings
- Idea: if two strings have a common prefix, they have the same ancestor(s) in the trie
- More efficient than a hash table for prefix-based searching

Trie vs. Hash Table

- Problem: Implement a dictionary of words with following operations: (i) add a word, (ii) search a word, (iii) remove a word
- Let the length of word be N
- Using hash table, all operations are O(N)
  - Computing the hash code will traverse the entire word
  - Tries will also do these operations in O(N)
- Problem: Given a prefix string, find all words in the dictionary
  - Need hashcode for every word, check prefix is the same: O(#words * maxLength of word)
  - This is much more efficient if we use a Trie!

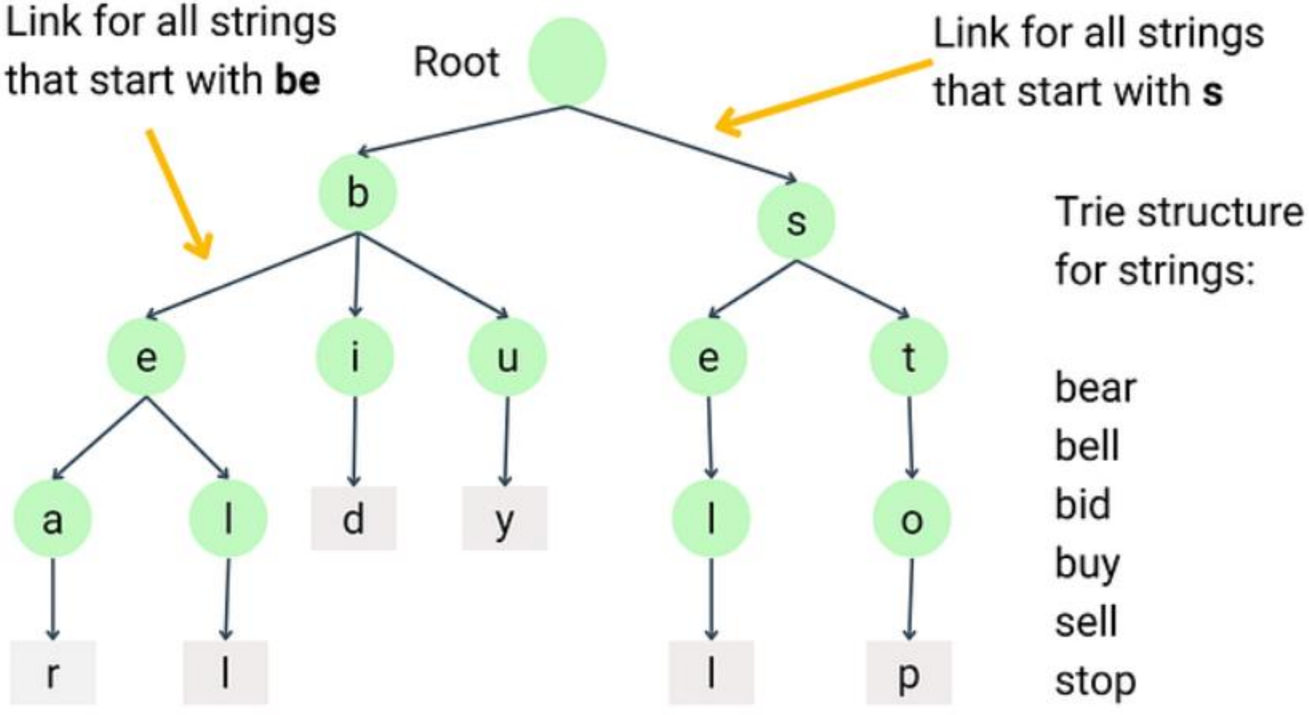
Trie Properties

- Every trie has an empty root node, with links to other nodes
- Each node represents a string and each edge represents a character
- Every node consists of an array of pointers
  - each index represents a character, and
  - a bit to indicate if any string ends at the current node
  - pointer to child nodes
- Can represent alphabets, numbers, special characters
- Each path from the root to any node represents a prefix or the end of a word

Trie Properties

- For lowercase English alphabets (a to z), array of pointers is of size k = 26
  - Any word can start with a – z
  - Next letter can be a – z, and so on…

Trie Properties



Link for all strings that start with **be**

Link for all strings that start with **s**

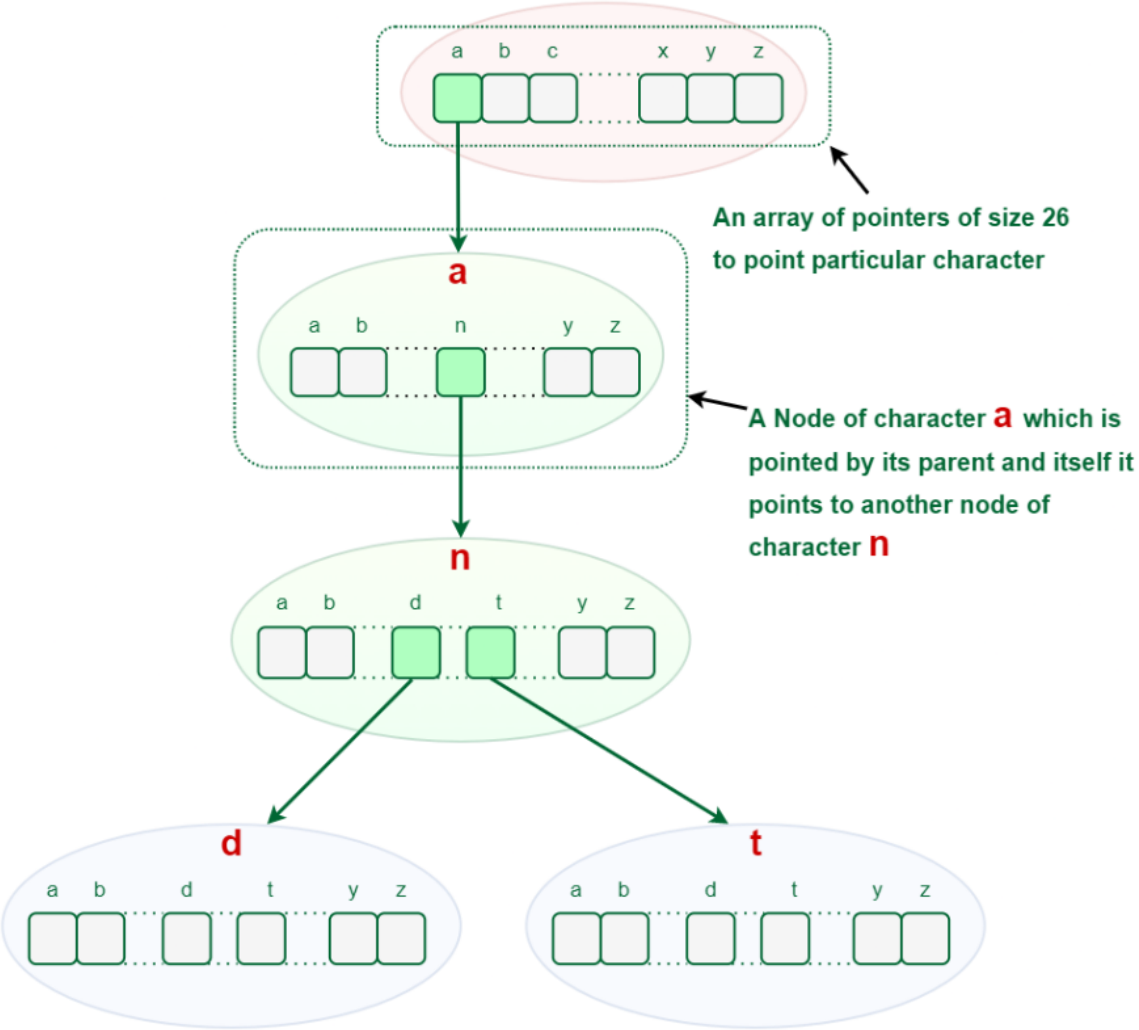
Trie structure for strings:

bear
bell
bid
buy
sell
stop

Trie: Store “and”, “ant”

- Init array of size 26 with all characters with NULL pointers
- Iterate over all characters of “and”
  - Mark the position character as filled, if not filled
  - The last character’s bit (isEnd) is True
- Iterate over all characters of “ant”
  - Mark the position character as filled, if not filled (e.g., “a” and “n”)
  - “t” is a new branch from “n”

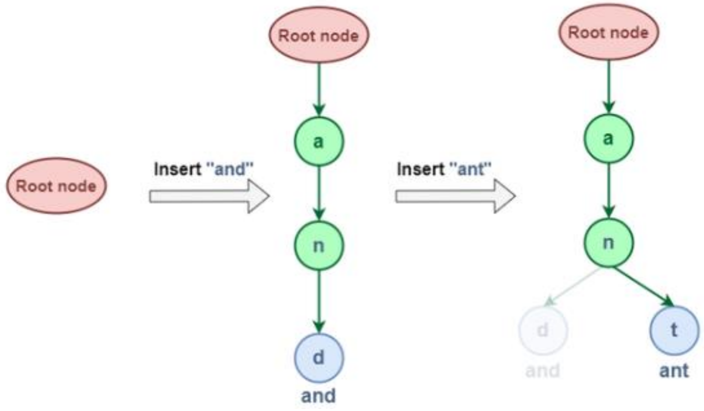
Trie: Store “and”, “ant”

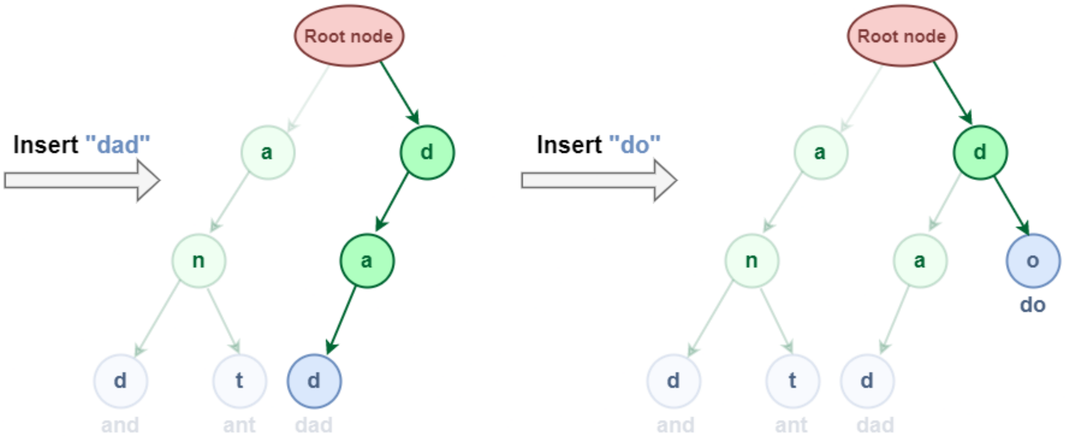


An array of pointers of size 26 to point particular character

A Node of character a which is pointed by its parent and itself it points to another node of character n

Trie: Insertion



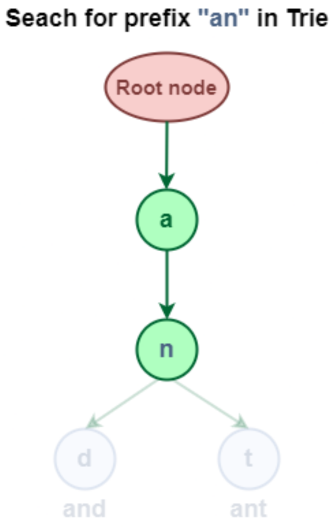


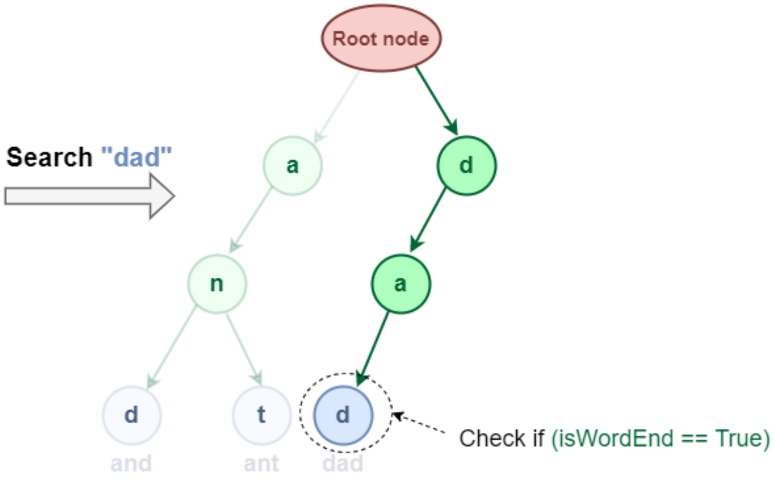
Trie: Insertion

- Start from the root node
- Iterate over each character in the string (left to right)
  - If current node has a child corresponding to that letter, go to child node
  - Else, create a new node as a child of the current node, and point the current character to this new node, go to child node
  - Mark end of word bit as True for the last node
  - In Python, make use of ord() to get the integer representing the character

Trie: Search

- Find whether a word exists in the Trie
- Find whether any word that starts with the given prefix exists in the Trie
- Idea #1: When array of pointers in the current node does not point to the current character of the word, return False
- Idea #2: Last character of word, must have word end bit set to True in Trie



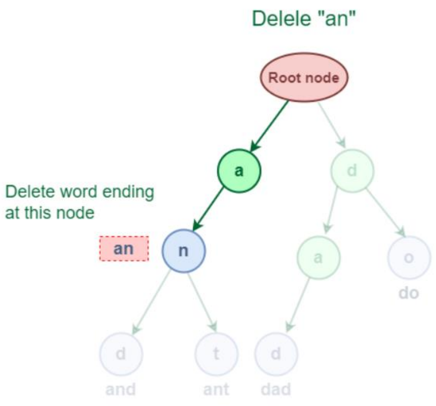


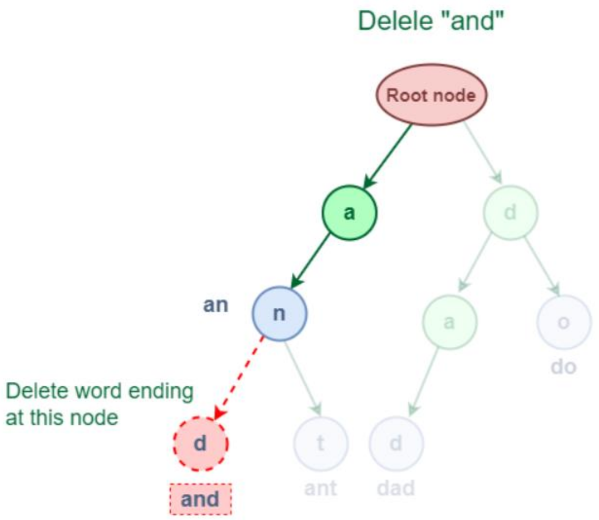
Trie: Search

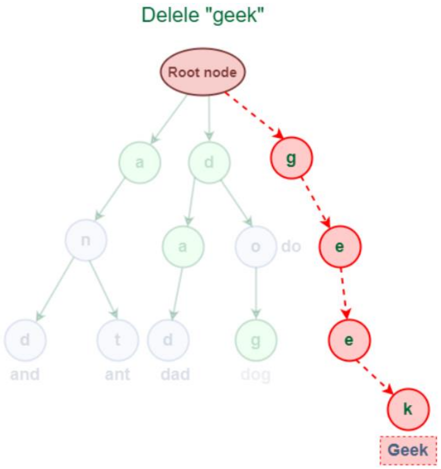
- Start from the root node
- Iterate over each character in the string (left to right)
  - If current node has a child corresponding to that letter, go to child node
  - Else, return False
  - Return True, if last node has end of word bit set to True

Trie: Deletion

- Case 1: Deleted word is a prefix of other words
- Case 2: Deleted word shares a common prefix with other words
- Case 3: Deleted word does not share any common prefix with other words







Trie: Deletion

- Case 1: Deleted word is a prefix of other words

- Case 2: Deleted word shares a common prefix with other words
  - Delete all nodes starting from the node corresponding to the end of the prefix to the last character of the deleted word

- Case 3: Deleted word does not share any common prefix with other words
  - Delete all the nodes

Trie Applications

- Autocompletion (e.g., in search engine)
- Longest prefix matching (e.g., for IP routing)
- Spell checker

Lecture 8 summary

- Tries