LECTURE 5

STRING ALGORITHM

Benjamin Li

10/8, 2021

Trie Data Structure

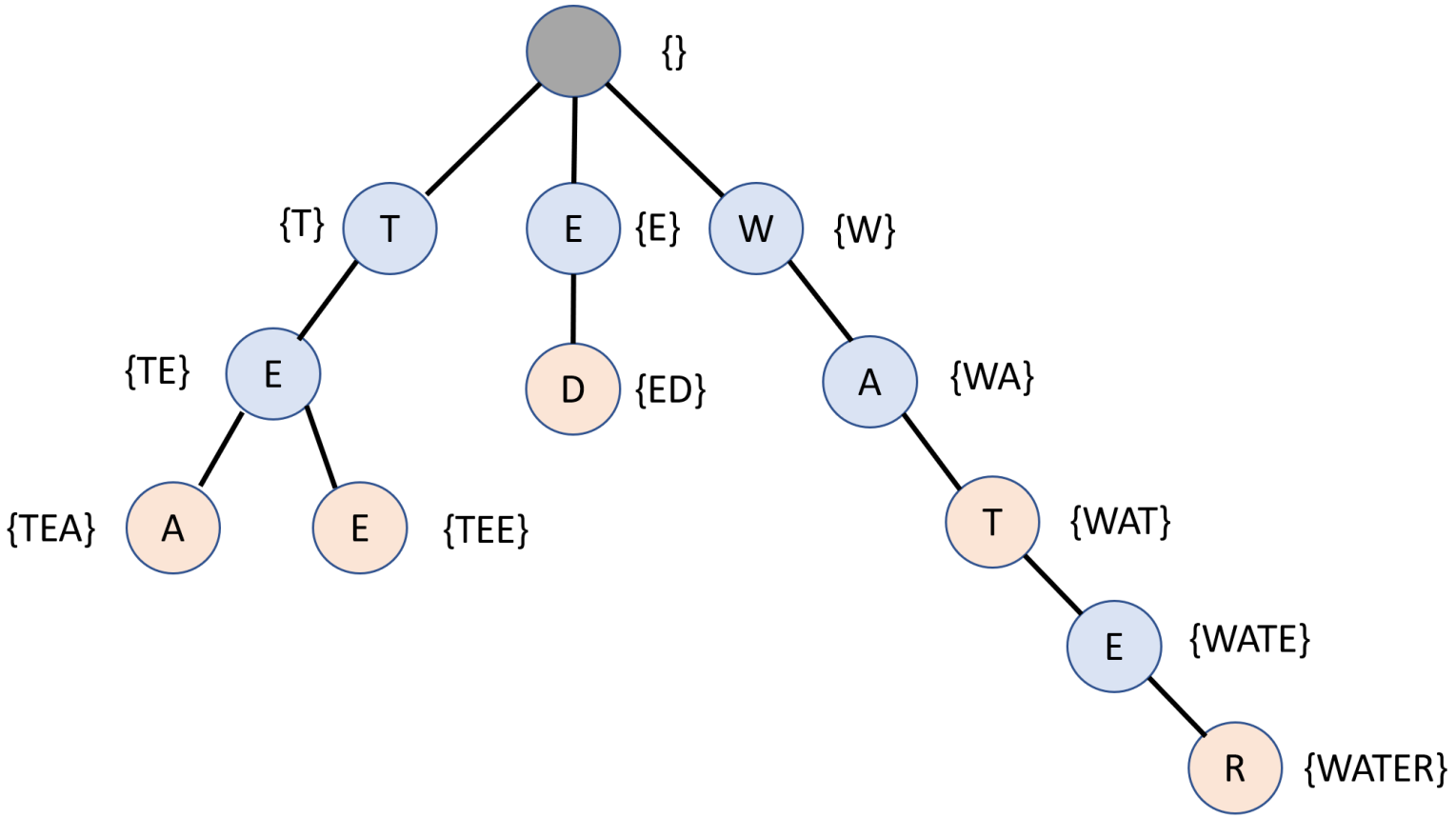
A trie is a type of tree, often used to store characters

- Each node has a character value
- Each branch contains a word
- Use those words to create a dictionary

Tries are very efficient

- Construction - `O(# words * L)` where `L` is average length of a word
- Insertion - `O(L)`
- Lookup - `O(L)`

Example Trie



Aho Corasick - Linear Search Algorithm

Algorithm to find all strings from a given set in a text

- `O(n + m + z)`
  - `n` - length of the text
  - `m` - total number of characters in all words in set
  - `z` - total occurrences of words in text

We keep state within the trie, so that we don't need to restart when a word doesn't match

- Do this by building failure links and output links
- Failure links point to start of next potential keyword (longest proper prefix)
- Output link joins keywords reachable from each node via a suffix link

Obscene Word Filter

Given a list of obscene words, and a string of text, determine if each obscene word appears in that text.

Example

- Obscene Words: homework, work, segfault, interviews
- Text: "I hate coding interviews because I always segfault"

Output:

homework does not exist in the string

work does not exist in the string

segfault exists in the string

interviews exists in the string

Longest Palindromic Sub-string

Problem - Given a string, find the longest palindrome in the string

Easily solved in $O(n^2)$ using two-pointer or expand-from-center.

What if our string is 100,000 characters long?

Longest Palindromic Sub-string

We want to be able to solve this problem in `O(n)` time.

Exploit some properties of palindromes

- Palindromes expand around their centers, but doing every center is wasteful
- Palindromes are symmetric around their centers

Manacher's Algorithm

By using a hybrid two pointer approach and expansion, we reduce the problem to `O(n)` with some smart bounds checking.