Data Structures and Algorithms

Week 9 - Tries

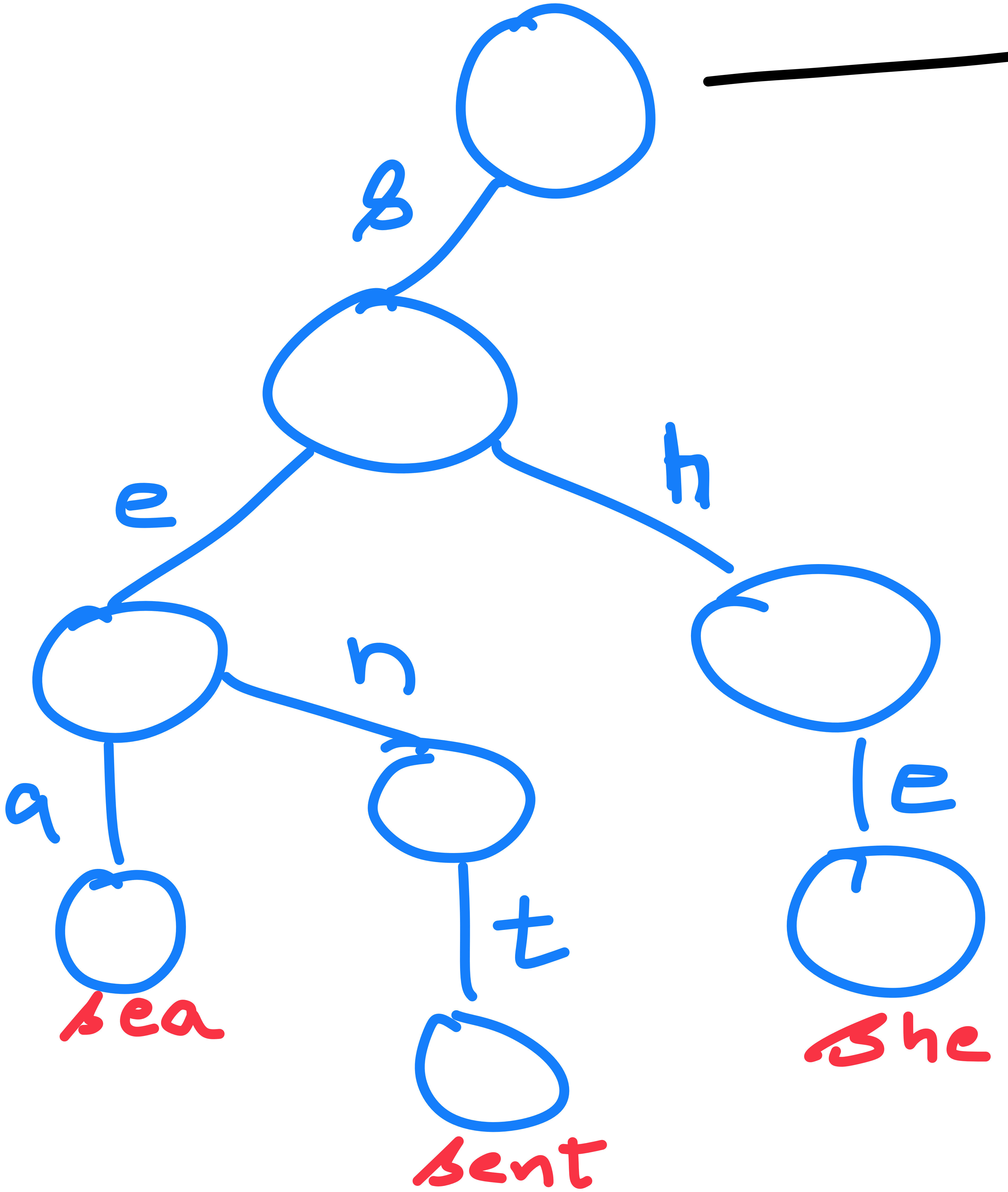
Subodh Sharma and Rahul Garg

{svs,rahulgarg}@iitd.ac.in.

Trie

- Pronounced as “try”
- They are special kind of trees
  - **K-ary search trees** used for finding keys within a set.
  - Keys are often **strings** but could be used for ordered lists of any type
    - The links between the nodes encode **single characters** of the key
    - Accessing a key requires a Depth-first traversal of the object
- Invented by - Edward Fredkin (1960), Axel Thue (1912), and René de la Briandais (1959)
- Unlike BST — there are not necessarily binary trees and not every node has a valueß
- Unlike AVL — there are not necessarily balanced

Trie - An Example



Root represents Empty String

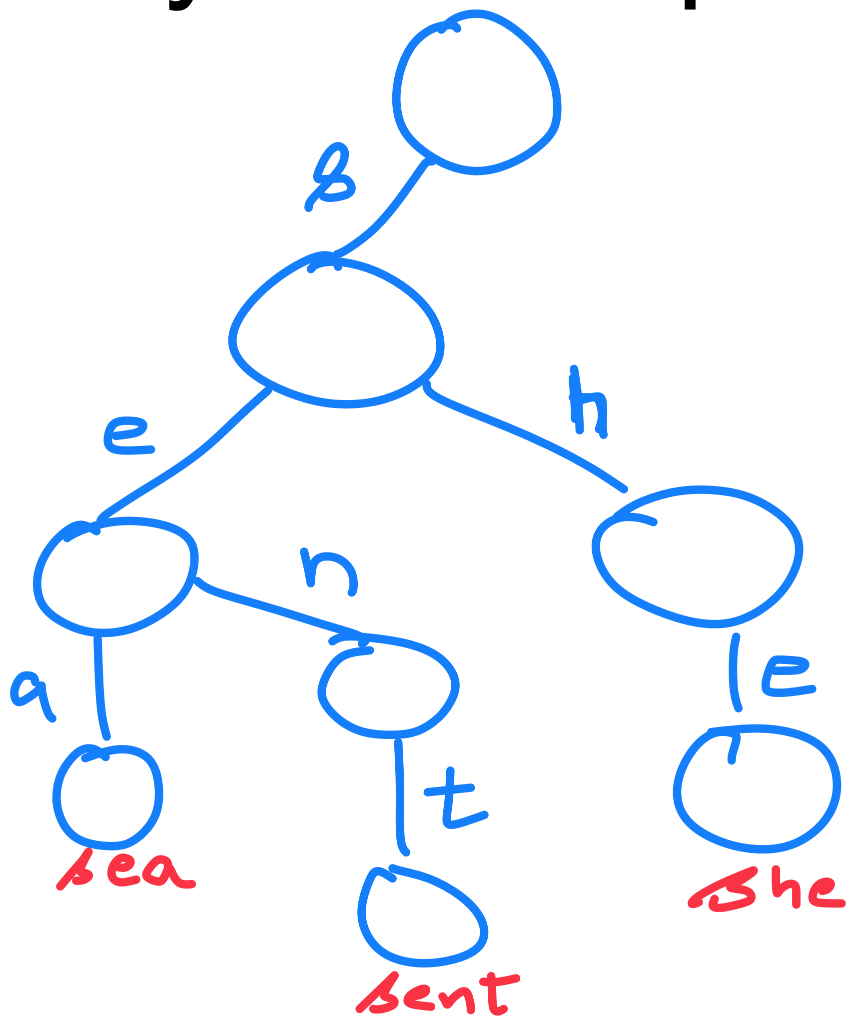
* All children of a node have a common prefix
* Note that the value of a key is distributed across the data structure!

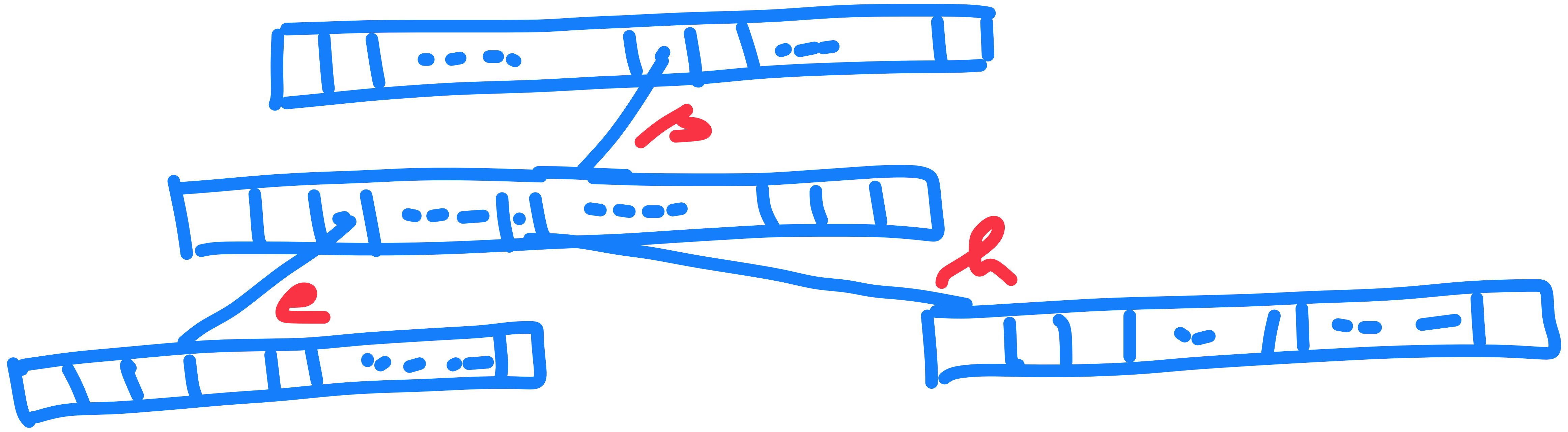
Trie — In other Words

- String-indexed structure to store dictionary list of words
  - Words can be searched in a manner that leads to efficient generation of **completion lists**
- Trie is an ordered tree data structure used in representation of a set of strings over a finite alphabet set.
- **What are they good for?**
  - Used in string searching algorithms — approximate string matching, spell checking, predictive text (autocomplete)
- **What operations Trie supports?**
  - Insertion, Deletion, LookUp,

Operationalisation

- Each node (other than the root) is pointed to by just its parent
- Each node contains n links where n is the cardinality of the Alphabet
  - Many of these links are **nil** (indicating string termination)
- An efficient way is to implement these links as a bitvector.





Insertion

- Algorithm:
    - For every char $\in$ word:
        - check if the char exists in the current node's children
        - If not, then add a TrieNode
    - Once the word end is reached, set *EndofWord* to true

Deletion

- Algorithm:
  - Recursively travel until the *EndOfWord* is reached
  - If the word-end is reached and *EndOfWord* is true, then set it to false and return whether the current node has no children
  - During backtracking, check if the node has no children then remove it

LookUp or Search

- Algorithm:
  - Traverse the Trie starting from the root node character by character
  - If any char $\in$ word not found in the Trie — Halt and return false
  - If the *EndOfWord* is reached, return true

QUIZ 2

- Derive, in the worst case, the number of swaps needed to insert a new element in a priority queue.
- Suppose you are given two max-priority queues Q1 and Q2, implemented as binary heaps, with sizes **n** and **m**, respectively. Derive the time complexity to merge Q1 and Q2 by performing repeated inserts.