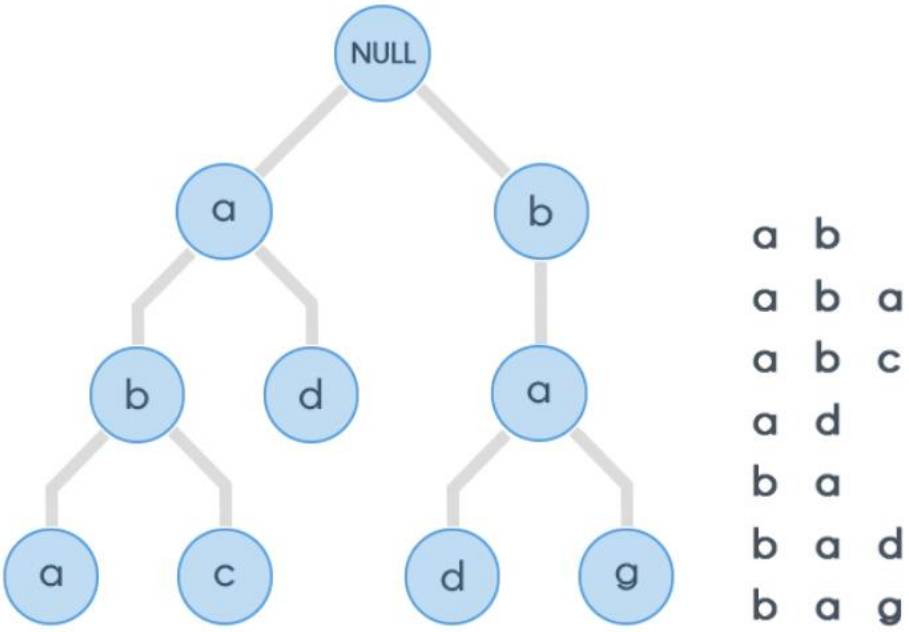
A Trie is a special data structure used to store strings that can be visualized like a graph. It consists of nodes and edges. Each node consists of at max 26 children and edges connect each parent node to its children. These 26 pointers are nothing but pointers for each of the 26 letters of the English alphabet A separate edge is maintained for every edge. Strings are stored in a top to bottom manner on the basis of their prefix in a trie. All prefixes of length 1 are stored at until level 1, all prefixes of length 2 are sorted at until level 2 and so on. For example, consider the following diagram



Q1: Implement the basic trie structure and operations like insert, delete, and find/query. Using this implementation, solve the following question.

Q2: You are given a Trie with a set of strings stored in it. Now the user types in a prefix of his search query, we need to give him all recommendations to auto-complete his query based on the strings stored in the Trie. We assume that the Trie stores past searches by the users.

For example if the Trie store {“abc”, “abcd”, “aa”, “abbbaba”} and the User types in “ab” then he must be shown {“abc”, “abcd”, “abbbaba”}.