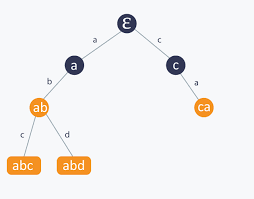
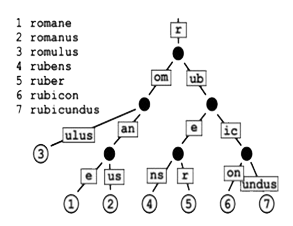
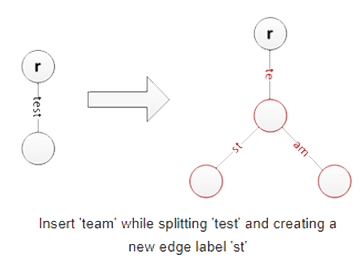
**RADIX TREE**

** A radix tree is a data structure that represents a space-optimization Trie. By traversing from the root to any leaf, concatenating all the labels of edges along the way, you can find any string.**

**INSERTION**

1. **Input the root node (the one without any character value assigned to it) and the whole word.**
2. **Then iterates through the word, one character at a time, starting with the first character.**
3. **Checks whether the current “node” (At the beginning of the process which points to root node)**

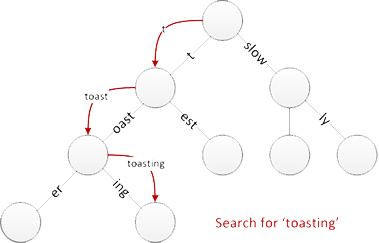
**has a child node with that character.**

1. **If found, it just increments the counter of that node to indicate that it has got a repeated**

**occurrence of that character.**

1. **If not found then it simply adds a new node as a child of the current node.**
2. **In both cases (4 & 5), it assigns the child node as the “current node” (which means in the next**

**iteration it will start from here) before it starts with the next character of the word.**

**Searching**

1. **Starts with the root node and the prefix to search for.**
2. **Takes one character at a time from the prefix and searches through the children of the**

**“current node” (at the beginning which points to the root node) to find a node containing**

**that character.**

1. **If found, it reassigns that child node as the “current node” (which means in the next iteration**

**step it will start from here)**

1. **If not found then it returns False to indicate that the prefix does not exist.**

**Deletion**

**First, we delete the corresponding leaf. Then, if its parent only has one child remaining,**

**we delete the parent and merge the two incident edges.**

**Applications**

**Trie are used in many string search applications such as auto-complete, text search,**

**and prefix matching where else Radix Tree, a kind of trie that are often used in IP routing.**