**Trie : Algorithm**

**1)Algorithm 1:** To create a newnode

Input: Data ‘data’ to inserted into tree/Node

Output: Newnode created

1. Declare a newnode
2. i=0
3. Repeat until i<N
   1. newnode->children[i]=0
   2. Increment i

4.newnode->data=data

5.newnode->is\_end=0

6. return newnode

**2)Algorithm 2: Used to insert a word into trie**

**Input:** Word ‘word’ to be inserted into Trie

**Output:** Node inserted

1.If root is NULL

1.root=getnode(‘\0’)

2.Create a node temp=root

3.i=0

4. Repeat until word[i]!=’\0’

4.1 idx=(int)word[i]-‘a’:to get the relative position of the character in the alphabets

4.2 if temp->children[idx] is NULL

4.2.1 temp->children[idx]=getnode(word[i])

4.3 temp=temp->children[idx]

5. temp->is\_end=1

6. return temp

3) **Algorithm 3 : To search for a word in trie**

**Input** : word to search in the trie

**Output** : True if word found, else false

1. if root equals NULL

1.1 return false

2. Assign pointer temp equal to root

3. temp=temp->children[int(word[0])-‘a’]

4. if temp equals NULL

4.1 return false

5. for i equals 0 till word[i]!=’\0’, increment i by 1

5.1 if temp equals NULL

5.1.1 return flase

5.2 if temp->data equals word[i]

5.2.1 if word[i+1] equals ‘\0’

5.2.1.1 break

5.2.2 else

5.2.2.1 temp=temp->children[(int)(word[i+1]-‘a’]

5.3 else

5.3.1 return false

6. if temp->is\_end not equals 1

6.1 return false

7. return true

**4)Algorithm 4: To delete a word from Trie**

**Input:** Word ‘word’ to be deleted from Trie, node temp and level

**Output:** Node deleted

1.If level is word\_len(word)

1. if temp-> is\_end is True
   1. temp->is\_end = 0
2. if isempty(temp) is True
   1. free(temp)
   2. temp= NULL
3. return temp

2. ind =word[level] – ‘a’

3.temp ->children[ind] = delete\_rec(temp ->children[ind],word,level+1)

4. if isempty(temp) is true and temp->is\_end is 0

1. free(temp)
2. temp = NULL

5.return temp