

King Fahd University of Petroleum & Minerals

**By: Fares Alnzlawe – 201867080 – Section 57**

**Major: Software Engineering**

**For: Mustafa Alturki, ICS 202**  
  
  
  
  
  
**“Trie data structure project”**

Tuesday, 14 December 2021

import java.io.File;

import java.io.FileNotFoundException;

import java.util.ArrayList;

import java.util.LinkedList;

import java.util.Queue;

import java.util.Scanner;

import java.util.Stack;

public class Trie {

    protected TrieNode root;

    public Trie() {

        root = new TrieNode('\0');

    }

    boolean contains(String s) {

        TrieNode tmp = root;

        char tmpC;

        for (int i = 0; i < s.length(); i++) {

            tmpC = s.charAt(i);

            if (tmp.child[tmpC - 'A'] == null) {

                return false;

            } else {

                tmp = tmp.child[tmpC - 'A'];

            }

        }

        return tmp.isWord;

    }

    boolean isPrefix(String p) {

        TrieNode tmp = root;

        char tmpC;

        for (int i = 0; i < p.length(); i++) {

            tmpC = p.charAt(i);

            if (tmp.child[tmpC - 'A'] == null) {

                return false;

            } else {

                tmp = tmp.child[tmpC - 'A'];

            }

        }

        return true;

    }

    void insert(String s) {

        s = s.toUpperCase();

        TrieNode tmp = root;

        for (int i = 0; i < s.length(); i++) {

            char tmpC = s.charAt(i);

            if (tmp.child[tmpC - 'A'] == null)

                tmp.child[tmpC - 'A'] = new TrieNode(tmpC, tmp);

            tmp = tmp.child[tmpC - 'A'];

        }

        tmp.isWord = true;

    }

    void delete(String s) {

        if (!contains(s)) {

            System.out.println("the word does not exist");

            return;

        }

        Stack<TrieNode> stack = new Stack<>();

        TrieNode tmp = root;

        char tmpC;

        for (int i = 0; i < s.length(); i++) {

            tmpC = s.charAt(i);

            tmp = tmp.child[tmpC - 'A'];

            stack.push(tmp);

        }

        boolean deleteAble;

        int letterNum = 0;

        while (!stack.isEmpty()) {

            tmp = stack.pop();

            letterNum++;

            deleteAble = true;

            for (int i = 0; i < 26; i++) {

                if (tmp.child[i] != null) {

                    deleteAble = false;

                }

            }

            if (!deleteAble && letterNum == 1) {

                tmp.isWord = false;

            }

            if (letterNum > 1) {

                if (tmp.isWord)

                    deleteAble = false;

            }

            if (deleteAble) {

                tmp = null;

            } else {

                break;

            }

        }

        System.out.println("Done!");

    }

    boolean isEmpty() {

        TrieNode tmp = root;

        for (int i = 0; i < 26; i++) {

            if (tmp.child[i] != null)

                return false;

        }

        return true;

    }

    void clear() {

        root = new TrieNode('\0');

    }

    String[] allWordsPrefix(String p) {

        p = p.toUpperCase();

        if (!isPrefix(p)) {

            System.out.println("No word contains this prefix");

            return null;

        }

        ArrayList<String> words = new ArrayList<>();

        TrieNode tmp = root;

        char tmpC;

        for (int i = 0; i < p.length(); i++) {

            tmpC = p.charAt(i);

            tmp = tmp.child[tmpC - 'A'];

        }

        TrieNode startP = tmp;

        ArrayList<TrieNode> allw = awd(startP);

        for (int i = 0; i < allw.size(); i++) {

            words.add(toWord(allw.get(i)));

        }

        return words.toArray(new String[words.size()]);

    }

    String toWord(TrieNode node) {

        String word = "";

        Stack<Character> st = new Stack<>();

        while (node.prev != null) {

            st.push(node.info);

            node = node.prev;

        }

        while (!st.isEmpty()) {

            word += st.pop();

        }

        return word;

    }

    ArrayList<TrieNode> awd(TrieNode node) {

        ArrayList<TrieNode> list = new ArrayList<>();

        for (int i = 0; i < 26; i++) {

            if (node.child[i] != null) {

                list.addAll(awd(node.child[i]));

            }

        }

        if (node.isWord) {

            list.add(node);

        }

        return list;

    }

    int size() {

        return size(root);

    }

    int size(TrieNode node) {

        int size = 1;

        for (int i = 0; i < 26; i++) {

            if (node.child[i] != null) {

                size = size + size(node.child[i]);

            }

        }

        return size;

    }

}

import java.util.Scanner;

public class driver {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int a;

        Trie te=new Trie();

        while (true) {

            System.out.println("Trie Project: Enter your choice");

            System.out.println(

                    "1) Create an empty trie\n2) Create a trie with initial letters\n3) Insert a word\n4) Delete a word\n5) List all words that begin with a prefix\n6) Size of the trie\n7) End");

            a = sc.nextInt();

            if (a == 1) {

                te = new Trie();

                System.out.println("Empty trie created");

            } else if (a == 2) {

                System.err.println("Enter your set");

                String wwww = sc.next();

                System.out.println();

            } else if (a == 3) {

                System.out.println("Enter a word to insert");

                te.insert(sc.next());

            } else if (a == 4) {

                System.out.println("Enter a word delete");

                te.delete(sc.next());

            } else if (a == 5) {

                System.out.println("Enter a prefix");

                String[] words = te.allWordsPrefix(sc.next());

                for (int i = 0; i < words.length; i++) {

                    System.out.println(words[i]);

                }

            } else if (a == 6) {

                System.out.println(te.size());

            } else if (a == 7) {

                System.exit(0);

            } else {

                System.out.println("Wrong input. Try again");

            }

            sc.nextLine();

        }

    }

}

public class TrieNode {

    char info;

    TrieNode[] child;

    TrieNode prev;

    boolean isWord = false;

    public TrieNode() {

        child = new TrieNode[26];

    }

    public TrieNode(char el) {

        info = el;

        child = new TrieNode[26];

    }

    public TrieNode(char el, TrieNode prev){

        info = el;

        child = new TrieNode[26];

        this.prev = prev;

    }

}