Longest prefix matching – A Trie based solution in Java

Given a dictionary of words and an input string, find the longest prefix of the string which is also a word in dictionary.

Examples:

```
Let the dictionary contains the following words:
{are, area, base, cat, cater, children, basement}

Below are some input/output examples:
---------------------------------------
Input String            Output
---------------------------------------
caterer                 cater
basemexy                base
child                   < Empty >
```

Solution

We build a Trie of all dictionary words. Once the Trie is built, traverse through it using characters of input string. If prefix matches a dictionary word, store current length and look for a longer match. Finally, return the longest match.
Following is Java implementation of the above solution based.

```
import java.util.HashMap;

// Trie Node, which stores a character and the children in a HashMap
class TrieNode {
```


Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion

```
    public TrieNode(char ch)  {
        value = ch;
        children = new HashMap<>();
        bIsEnd = false;
    }
    public HashMap<Character,TrieNode> getChildren() {   return childr
    public char getValue()                           {   return value;
    public void setIsEnd(boolean val)                {   bIsEnd = val;
    public boolean isEnd()                           {   return bIsEnd

    private char value;
    private HashMap<Character,TrieNode> children;
    private boolean bIsEnd;
}

// Implements the actual Trie
class Trie {
    // Constructor
    public Trie()   {     root = new TrieNode((char)0);       }

    // Method to insert a new word to Trie
    public void insert(String word)  {

        // Find length of the given word
        int length = word.length();
        TrieNode crawl = root;

        // Traverse through all characters of given word
        for( int level = 0; level < length; level++)
        {
            HashMap<Character,TrieNode> child = crawl.getChildren();
            char ch = word.charAt(level);

            // If there is already a child for current character of gi
            if( child.containsKey(ch))
                crawl = child.get(ch);
            else   // Else create a child
            {
                TrieNode temp = new TrieNode(ch);
                child.put( ch, temp );
                crawl = temp;
            }
        }

        // Set bIsEnd true for last character
        crawl.setIsEnd(true);
    }
```

Geometric Algorithms

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
    // The main method that finds out the longest string 'input'
    public String getMatchingPrefix(String input)  {
        String result = ""; // Initialize resultant string
        int length = input.length();  // Find length of the input stri

        // Initialize reference to traverse through Trie
        TrieNode crawl = root;

        // Iterate through all characters of input string 'str' and tr
        // down the Trie
        int level, prevMatch = 0;
        for( level = 0 ; level < length; level++ )
        {
            // Find current character of str
            char ch = input.charAt(level);

            // HashMap of current Trie node to traverse down
            HashMap<Character,TrieNode> child = crawl.getChildren();

            // See if there is a Trie edge for the current character
            if( child.containsKey(ch) )
            {
               result += ch;          //Update result
               crawl = child.get(ch); //Update crawl to move down in T

               // If this is end of a word, then update prevMatch
               if( crawl.isEnd() )
                    prevMatch = level + 1;
            }
            else  break;
        }

        // If the last processed character did not match end of a word
        // return the previously matching prefix
        if( !crawl.isEnd() )
               return result.substring(0, prevMatch);

        else return result;
    }

    private TrieNode root;
}

// Testing class
public class Test {
   public static void main(String[] args) {
```

```
        Trie dict = new Trie();
        dict.insert("are");
        dict.insert("area");
        dict.insert("base");
        dict.insert("cat");
        dict.insert("cater");
        dict.insert("basement");

        String input = "caterer";
        System.out.print(input + ":   ");
        System.out.println(dict.getMatchingPrefix(input));

        input = "basement";
        System.out.print(input + ":   ");
        System.out.println(dict.getMatchingPrefix(input));

        input = "are";
        System.out.print(input + ":   ");
        System.out.println(dict.getMatchingPrefix(input));

        input = "arex";
        System.out.print(input + ":   ");
        System.out.println(dict.getMatchingPrefix(input));

        input = "basemexz";
        System.out.print(input + ":   ");
        System.out.println(dict.getMatchingPrefix(input));

        input = "xyz";
        System.out.print(input + ":   ");
        System.out.println(dict.getMatchingPrefix(input));
    }
}
```

Output:

```
caterer:   cater
basement:   basement
are:   are
arex:   are
basemexz:   base
xyz:
```

Time Complexity: Time complexity of finding the longest prefix is O(n) where n is length of the

input string. Refer this for time complexity of building the Trie.

This article is compiled by **Ravi Chandra Enaganti**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Related Tpoics:

- Print a Binary Tree in Vertical Order | Set 2 (Hashmap based Method)
- Print Right View of a Binary Tree
- Red-Black Tree | Set 3 (Delete)
- Construct a tree from Inorder and Level order traversals
- Print all nodes at distance k from a given node
- Print a Binary Tree in Vertical Order | Set 1
- Interval Tree
- Check if a given Binary Tree is height balanced like a Red-Black Tree

Writing code in comment? Please use **ideone.com** and share the link here.

22 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Yogesh • 23 days ago

Hi Ravi,

As per the problem statement it will fail for some test cases.

e.g. input "a" it will print ans "a" which is wrong as "a" is not in the dictionary.

I think , isEnd() method can be properly used. Save the prev matched Node

as use it before printing the result.Chaged the getPrefix method a bit.

// The main method that finds out the longest string 'input'

public String getMatchingPrefix(String input) {

String result = ""; // Initialize resultant string

int length = input.length(); // Find length of the input string

// Initialize reference to traverse through Trie

TrieNode crawl = root;

see more

• Reply • Share ›

wasseypuriyan • 7 months ago

```
//Trie
#include<iostream>
```

```
#include<cstdlib>
#include<ctime>
#include<stack>

using namespace std;

struct node{
    char data;
    node *child[128];
};


class trie{

    private:
    node *root;
```

see more

• Reply • Share ›

illuminati • 10 months ago

#include
#include
using namespace std;
struct node{
bool isLeaf;
node* alp[26];
}*root=NULL;
void trie_insert(string str)
{
int i,j;
node *ptr;

```
int len=str.size();
if(root==NULL)
{
root=new node;
root->isLeaf=false;
for(i=0;ialp[i]=NULL;
}
```

see more

• Reply • Share ›

pritybhudolia • 10 months ago

C Implementation

```
 #include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define ARRAY_SIZE(a) sizeof(a)/sizeof(a[0])
#define ALPHABET_SIZE (26)
#define CHAR_TO_INDEX(c) ((int)c - 97)

// trie node
typedef struct trie_node trienode;
struct trie_node
{
    int value;
    trienode *children[ALPHABET_SIZE];
};
```

see more

aman1234 • 11 months ago

```
 string substr(string s)
{
       if(!root)
       return "";

       static string ans="";
       string temp="";
       int level=s.size();
       node * c=root;

       for(int i=0;i<level;i++)
       {
            c=c->point[s[i]-'a'];

            if(c==NULL)
            return ans;
           temp+=s[i];
```

see more

1 ^ v • Reply • Share ›

Sudipto • 11 months ago

@geeksforgeeks : What is the use of the 'value' field in a 'TrieNode'?

```
/* Paste your code here (You may delete these lines if not writing c
```

1 ^ v • Reply • Share ›

Nitendra Kumar • a year ago

```
 import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Test1{
        public static void main(String[] args) {
                List<String> list = new ArrayList<String>();
                list.add("are");
                list.add("area");
                list.add("base");
                list.add("cat");
                list.add("cater");
                list.add("children");
                list.add("basement");
                Map<Integer, String> map = new HashMap<Integer, Strin

                String input = "caterer";
```

see more

• Reply • Share ›

Shaik Zakir Hussain • a year ago

vishu, ye kya ho gaya bhai tujhe ?

1 • Reply • Share ›

Vishwanath Pratap Singh • a year ago

#include<stdio.h>
#include<string.h>
#include<stdlib.h>

typedef struct trie{.

int words;
int prefixes;
struct trie* next[26];.
}trie;

trie* initialize(trie *node){.

node=(trie *)malloc(sizeof(trie));
node->words=0;
node->prefixes=0;
int i=0;
for(i=0;i<26;i++)

node->next[i]=NULL;.

see more

• Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

1 • Reply • Share ›

Kumar Gautam • a year ago

Code without using HashMap.The idea is to use references as we use pointer

```
 package Java;
import java.io.*;

class TNode{
        final int MAXCHAR = 26;
        private boolean bIsEnd;
        private TNode children[];
```

```
        bIsEnd = false;
        children = new TNode[MAXCHAR];
        for(int i = 0; i < MAXCHAR; i++)
                children[i] = null;
}

public void setIsEnd(boolean val){
```

see more

Reply • Share ›

Manish Untwal • a year ago

check the constructor of trienode for hashmap initialization, do declare the typ

Reply • Share ›

spicavigo • a year ago

Here is a solution in Python in O(n).

Run the code here - http://codebunk.com/bunk#-ltYw...

[sourcecode language="Python"]
class Node(object):
def __init__(self, ch):
self.ch = ch
self.children = {}
self.words = []

def add(self, sword, word):
if not sword:
self.words.append(word)
return
chnode = self.children.get(sword[0], Node(sword[0]))

```
chnode.add(sword[1:], word)
self.children[sword[0]] = chnode
```

see more

• Reply • Share ›

spicavigo • a year ago

A solution in O(n) in Python
http://codebunk.com/bunk#-ltYw...

• Reply • Share ›

Prateek Sharma • a year ago

Python Code.....

```
[sourcecode language="Python"]
import numpy as np
def longestPrefixString(a,string):
newList =[]
newListLength= []
lenOfString = len(string)
liststr = list(string)
for i in range(len(a)):
length = len(a[i])
n =0
while(n<lenOfString and n<length):
if ord(a[i][n]) == ord(liststr[n]):
n +=1
continue
else:
break
```

see more

anishp2012 • a year ago

Please have a look at my code below.Have used a simple HashSet<string> to

Kindly give comments if there is some problem with the code

```
/**
 *
 */
package com.abb.java.trainings;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.HashSet;
import java.util.Set;

/**
 * @author INANPRA4
 * @date Apr 27, 2013
 * @time 11:20:21 PM
```

see more

kartik ➦ anishp2012 • a year ago

@anishp2012: Thanks for suggesting another method. The code looks
cases. However, time complexity of this looks more that that of the Trie
all prefixes and them look them in hashTable. The worst case complex
method suggested above takes O(n) time. Please correct me if i am w

anishp2012 ➚ kartik • a year ago

@kartik : Thanks for your reply.There needs to be a correction
"The prefixes generated which you see in my code, will be com
entered(Initial input provided by the user) and if not found in the
entered string to check for the word in the dictionary again untill
had been depicted in the initial code provided by Ravi.Hence,I a
checking for string matches which has a longer prefix String in
For retrieval from the dictionary,I have used a HashSet<string>
for a HashSet<e>, for retrieval complexity is O(1).Hence the wo
and not O(n^2)."
Would love to provide more documentation if you still face any
which I've provided.
Kindly correct me if you feel I'm not correct

Regards
Anish

^ | v • Reply • Share ›

kartik ➚ anishp2012 • a year ago

@anishp2012: To the best of my knowledge, the worst
HashSet is O(n). You may refer following link.

http://stackoverflow.com/quest...

1 ^ | v • Reply • Share ›

Aashish ➚ kartik • a year ago

The TRIE DS will always have a win-win factor because
as a prefix(thereby saving space).

1 ^ | v • Reply • Share ›

Ravi Chandra ➚ kartik • a year ago

I second kartik. No doubt, the solution provided by Anish

solution. But the worst case time complexity of this solu
dictionary with many words, the space complexity for Tr
be better.

anishp2012 kartik • a year ago

@kartik : Its typically O(1) and even referring to the link
can see that
"Yes, but it's really the worst case: if all the elements in
hash code (or a hash code leading to the same bucket)
hashCode and a normally distributed key sample, a look
Still,I feel we can improve by using TreeSet<string> inst
would like my words in the dictionary to be in the sorted
you agree to the same
Thanks
Anish