

Trie memory optimization using hash map

Difficulty Level : Easy • Last Updated : 26 Sep, 2017

We introduced and discussed an implementation in below post.

[Trie | \(Insert and Search\) – GeeksforGeeks](#)

The implementation used in above post uses an array of alphabet size with every node. It can be made memory efficient. One way to implementing Trie is linked set of nodes, where each node contains an array of child pointers, one for each symbol in the alphabet. This is not efficient in terms of time as we can't quickly find a particular child.

The efficient way is an implementation where we use hash map to store children of a node. Now we allocate memory only for alphabets in use, and don't waste space storing null pointers.

```
// A memory optimized CPP implementation of trie
// using unordered_map
#include <iostream>
#include <unordered_map>
using namespace std;

struct Trie {
    // isEndOfWord is true if the node
    // represents end of a word
    bool isEndOfWord;

    /* nodes store a map to child node */
    unordered_map<char, Trie*> map;
};

/*function to make a new trie*/
Trie* getNewTrieNode()
{
    Trie* node = new Trie;
    node->isEndOfWord = false;
    return node;
}

/*function to insert in trie*/
void insert(Trie*& root, const string& str)
{
    if (root == nullptr)
        root = getNewTrieNode();

    Trie* temp = root;
    for (int i = 0; i < str.length(); i++) {
        char x = str[i];

        /* make a new node if there is no path */
        if (temp->map.find(x) == temp->map.end())
            temp->map[x] = getNewTrieNode();

        temp = temp->map[x];
    }

    temp->isEndOfWord = true;
}

/*function to search in trie*/
bool search(Trie* root, const string& str)
{
    /*return false if Trie is empty*/
    if (root == nullptr)
        return false;

    Trie* temp = root;
    for (int i = 0; i < str.length(); i++) {

        /* go to next node*/
        temp = temp->map[str[i]];

        if (temp == nullptr)
            return false;
    }

    return temp->isEndOfWord;
}

/*Driver function*/
int main()
{
    Trie* root = nullptr;

    insert(root, "geeks");
    cout << search(root, "geeks") << " ";

    insert(root, "for");
    cout << search(root, "for") << " ";

    cout << search(root, "geekk") << " ";

    insert(root, "gee");
    cout << search(root, "gee") << " ";

    insert(root, "science");
    cout << search(root, "science") << endl;

    return 0;
}
```

Output:

```
1 1 0 1 1
```

Space used here with every node here is proportional to number of children which is much better than proportional to alphabet size, especially if alphabet is large.



This article is contributed by **Pranav**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](#) or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Attention reader! Don't stop learning now. Get hold of all the important DSA concepts with the **DSA Self Paced Course** at a student-friendly price and become industry ready. To complete your preparation from learning a language to DS Algo and many more, please refer **Complete Interview Preparation Course**.

 Like 0

< Previous

Program to find Circumference of a Circle

Next >

Working with tmux session

RECOMMENDED ARTICLES

Page : 1 2 3

- 01 Auto-complete feature using Trie
07, Jun 17
- 05 Count of distinct substrings of a string using Suffix Trie
30, Nov 16
- 02 Pattern Searching using a Trie of all Suffixes
29, Aug 14
- 06 Boggle | Set 2 (Using Trie)
12, Feb 17
- 03 Find shortest unique prefix for every word in a given list | Set 1 (Using Trie)
14, Oct 15
- 07 Sorting array of strings (or words) using Trie
10, Sep 17
- 04 Longest Common Prefix using Trie
23, Jun 16
- 08 Program for assigning usernames using Trie
28, Sep 18



Article Contributed By :



Vote for difficulty

Current difficulty : Easy

Easy

Normal

Medium

Hard

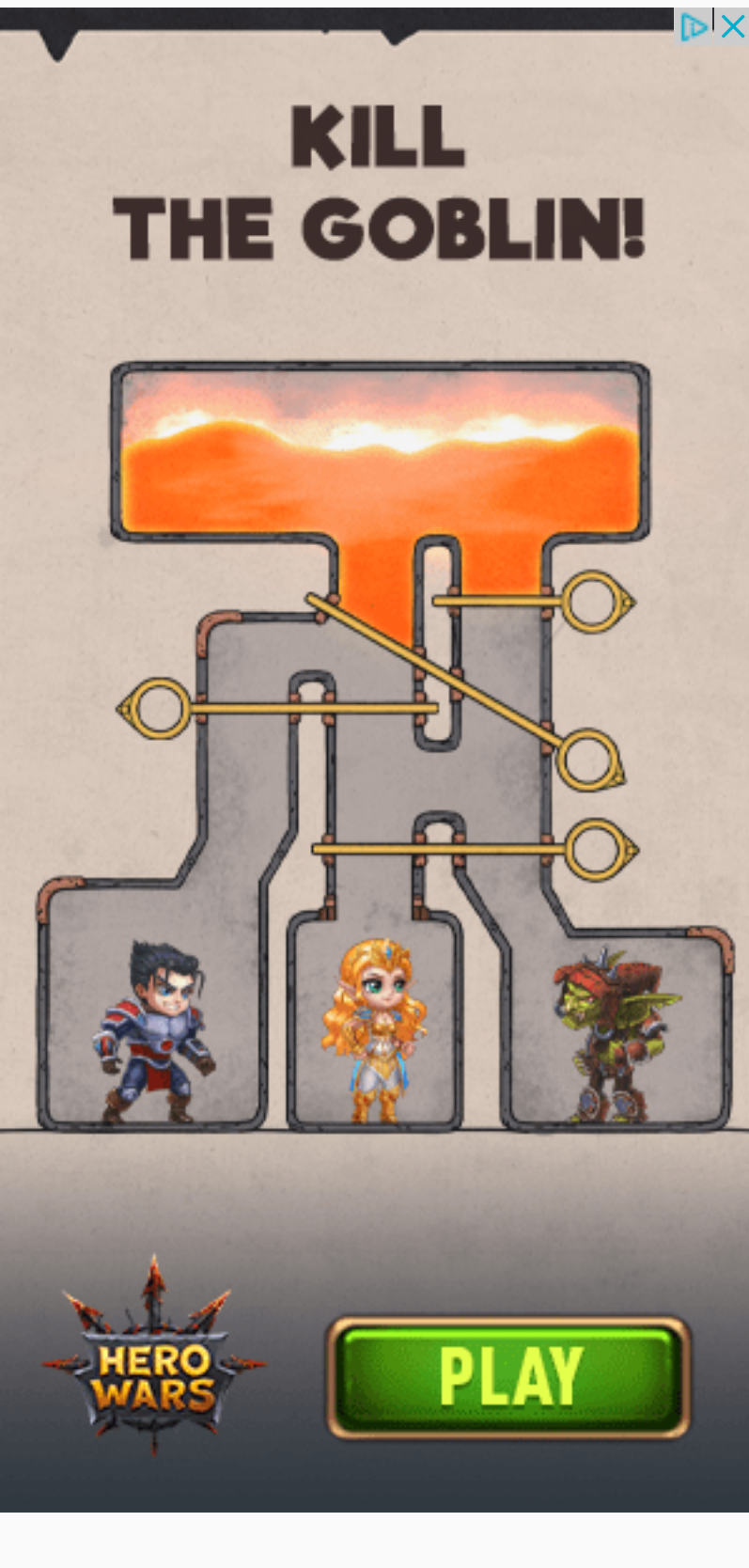
Expert

Article Tags : [cpp-unordered_map](#), [Trie](#), [Advanced Data Structure](#)

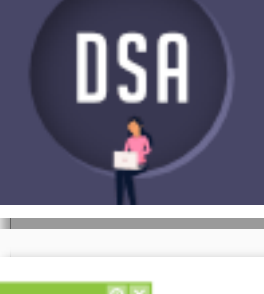
Practice Tags : [Trie](#)

Improve Article

Report Issue

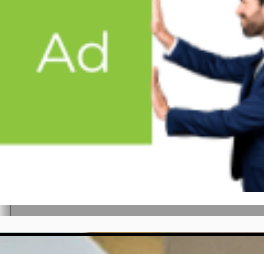


WHAT'S NEW



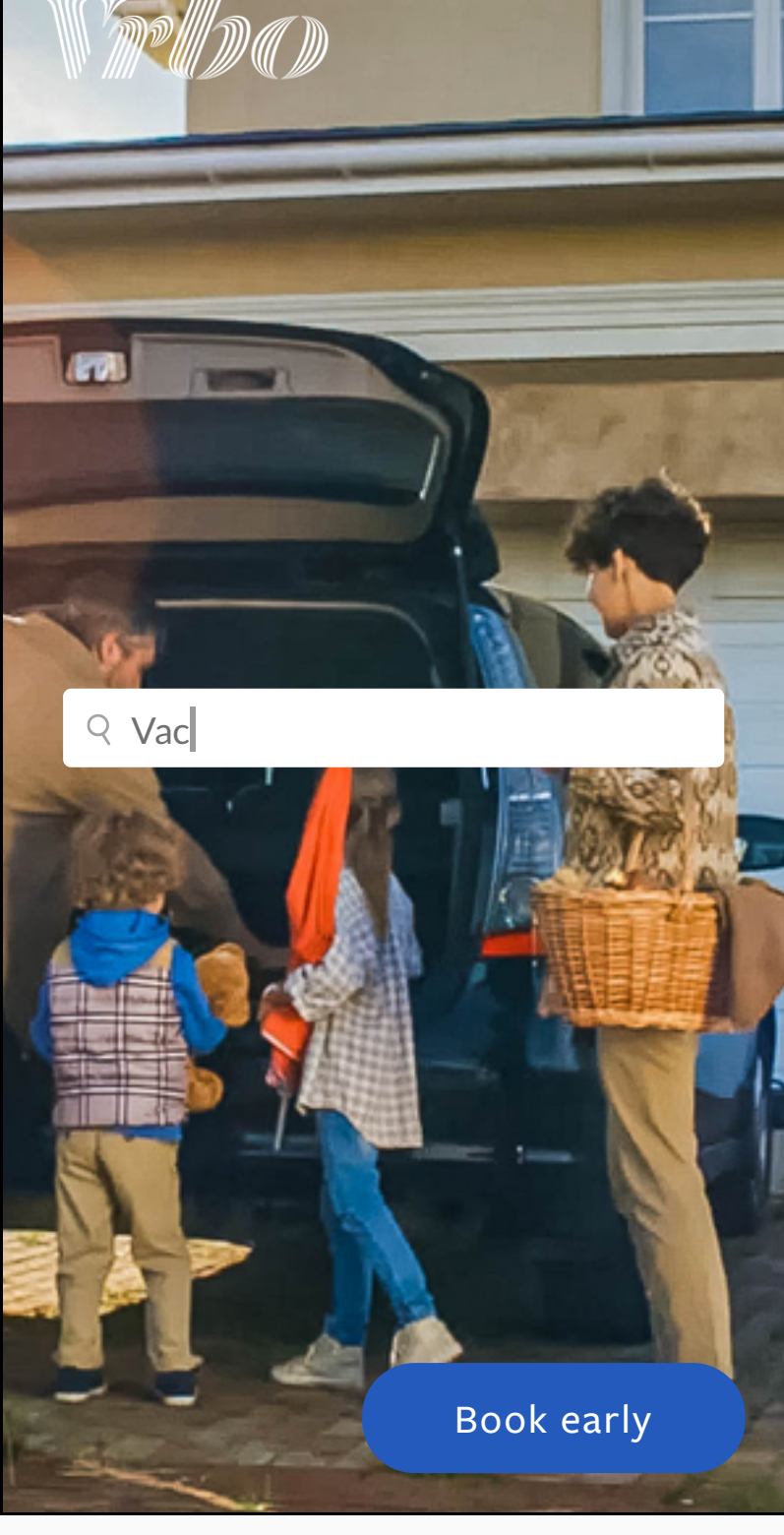
DSA Self Paced Course

[View Details](#)



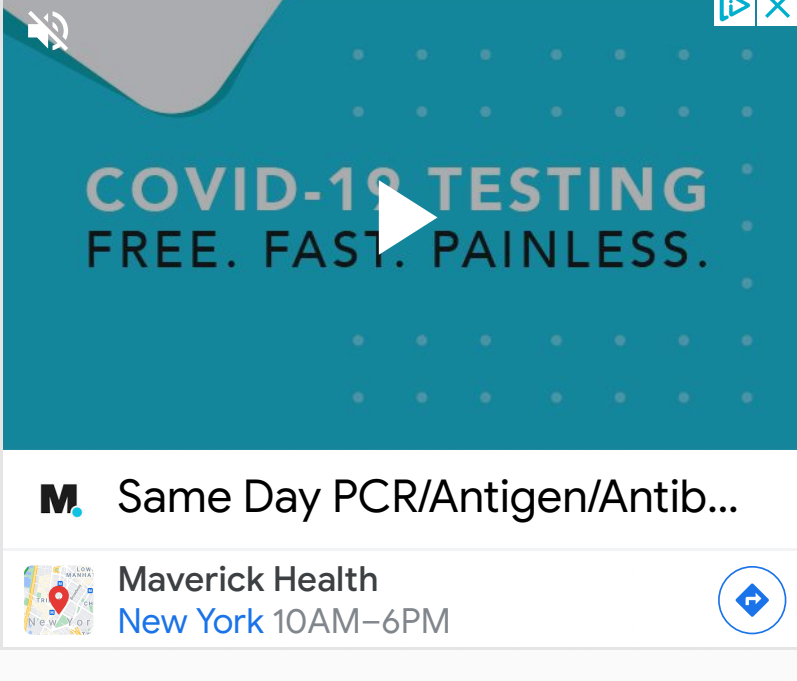
Ad free experience with GeeksforGeeks Premium

[View Details](#)



MOST POPULAR IN ADVANCED DATA STRUCTURE

- Maximum Occurrence in a Given Range
- Ackermann Function
- Difference between B tree and B+ tree
- Quad Tree
- 2-3 Trees | (Search and Insert)



MORE RELATED ARTICLES IN ADVANCED DATA STRUCTURE

- Some Basic Theorems on Trees
- Suffix Array | Set 1 (Introduction)
- Extendible Hashing (Dynamic approach to DBMS)
- Find maximum in a stack in O(1) time and O(1) extra space
- Skip List | Set 2 (Insertion)

