Given a sequence of words, print all anagrams together | Set 2

Given an array of words, print all anagrams together. For example, if the given array is {"cat", "dog", "tac", "god", "act"}, then output may be "cat tac act dog god".

We have discussed two different methods in the previous post. In this post, a more efficient solution is discussed.

Trie data structure can be used for a more efficient solution. Insert the sorted order of each word in the trie. Since all the anagrams will end at the same leaf node. We can start a linked list at the leaf nodes where each node represents the index of the original array of words. Finally, traverse the Trie. While traversing the Trie, traverse each linked list one line at a time. Following are the detailed steps.

1) Create an empty Trie
2) One by one take all words of input sequence. Do following for each word
…**a)** Copy the word to a buffer.
…**b)** Sort the buffer
…**c)** Insert the sorted buffer and index of this word to Trie. Each leaf node of Trie is head of a Index list. The Index list stores index of words in original sequence. If sorted buffe is already present, we insert index of this word to the index list.
3) Traverse Trie. While traversing, if you reach a leaf node, traverse the index list. And print all words using the index obtained from Index list.

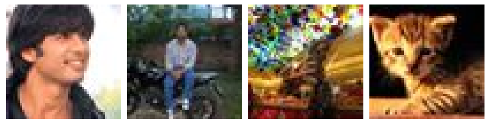
```
// An efficient program to print all anagrams together
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Google™ Custom Search

GeeksforGeeks

52,731 people like GeeksforGeeks.

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



```
#include <ctype.h>

#define NO_OF_CHARS 26

// Structure to represent list node for indexes of words in
// the given sequence. The list nodes are used to connect
// anagrams at leaf nodes of Trie
struct IndexNode
{
    int index;
    struct IndexNode* next;
};

// Structure to represent a Trie Node
struct TrieNode
{
    bool isEnd;  // indicates end of word
    struct TrieNode* child[NO_OF_CHARS]; // 26 slots each for 'a' to '
    struct IndexNode* head; // head of the index list
};


// A utility function to create a new Trie node
struct TrieNode* newTrieNode()
{
    struct TrieNode* temp = new TrieNode;
    temp->isEnd = 0;
    temp->head = NULL;
    for (int i = 0; i < NO_OF_CHARS; ++i)
        temp->child[i] = NULL;
    return temp;
}

/* Following function is needed for library function qsort(). Refer
   http://www.cplusplus.com/reference/clibrary/cstdlib/qsort/ */
int compare(const void* a, const void* b)
{  return *(char*)a - *(char*)b; }

/* A utility function to create a new linked list node */
struct IndexNode* newIndexNode(int index)
{
    struct IndexNode* temp = new IndexNode;
    temp->index = index;
    temp->next = NULL;
    return temp;
}
```

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

```c
// A utility function to insert a word to Trie
void insert(struct TrieNode** root, char* word, int index)
{
    // Base case
    if (*root == NULL)
        *root = newTrieNode();

    if (*word != '\0')
        insert( &( (*root)->child[tolower(*word) - 'a'] ), word+1, ind
    else  // If end of the word reached
    {
        // Insert index of this word to end of index linked list
        if ((*root)->isEnd)
        {
            IndexNode* pCrawl = (*root)->head;
            while( pCrawl->next )
                pCrawl = pCrawl->next;
            pCrawl->next = newIndexNode(index);
        }
        else  // If Index list is empty
        {
            (*root)->isEnd = 1;
            (*root)->head = newIndexNode(index);
        }
    }
}

// This function traverses the built trie. When a leaf node is reached
// all words connected at that leaf node are anagrams. So it traverses
// the list at leaf node and uses stored index to print original words
void printAnagramsUtil(struct TrieNode* root, char *wordArr[])
{
    if (root == NULL)
        return;

    // If a lead node is reached, print all anagrams using the indexes
    // stored in index linked list
    if (root->isEnd)
    {
        // traverse the list
        IndexNode* pCrawl = root->head;
        while (pCrawl != NULL)
        {
            printf( "%s \n", wordArr[ pCrawl->index ] );
            pCrawl = pCrawl->next;
        }
    }
    for (int i = 0; i < NO_OF_CHARS; ++i)
        printAnagramsUtil(root->child[i], wordArr);
}

// The main function that prints all anagrams together. wordArr[] is i
// sequence of words.
void printAnagramsTogether(char* wordArr[], int size)
{
    // Create an empty Trie
    struct TrieNode* root = NULL;

    // Iterate through all input words
    for (int i = 0; i < size; ++i)
    {
        // Create a buffer for this word and copy the word to buffer
        int len = strlen(wordArr[i]);
        char *buffer = new char[len+1];
        strcpy(buffer, wordArr[i]);

        // Sort the buffer
        qsort( (void*)buffer, strlen(buffer), sizeof(char), compare );

        // Insert the sorted buffer and its original index to Trie
        insert(&root,  buffer, i);
    }

    // Traverse the built Trie and print all anagrms together
    printAnagramsUtil(root, wordArr);
}


// Driver program to test above functions
int main()
{
    char* wordArr[] = {"cat", "dog", "tac", "god", "act", "gdo"};
    int size = sizeof(wordArr) / sizeof(wordArr[0]);
    printAnagramsTogether(wordArr, size);
    return 0;
}
```

Output:

```
cat
tac
act
```

Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...

Backtracking | Set 7 (Sudoku) · 16 minutes ago

RVM Can someone please elaborate this Qs from above...

Flipkart Interview | Set 6 · 35 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...

Software Engineering Lab, Samsung Interview | Set 2 · 36 minutes ago

@meya Working solution for question 2 of 4f2f round....

Amazon Interview | Set 53 (For SDE-1) · 1 hour ago

sandeep void rearrange(struct node *head) {...

Given a linked list, reverse alternate nodes and append at the end · 2 hours ago

Neha I think that is what it should return as, in...

Find depth of the deepest odd level leaf node · 2 hours ago

```
dog
god
gdo
```

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

Related Tpoics:

- Printing Longest Common Subsequence
- Suffix Array | Set 2 (nLogn Algorithm)
- Rearrange a string so that all same characters become d distance away
- Recursively remove all adjacent duplicates
- Find the first non-repeating character from a stream of characters
- Dynamic Programming | Set 33 (Find if a string is interleaved of two other strings)
- Remove “b” and “ac” from a given string
- Dynamic Programming | Set 29 (Longest Common Substring)

2 Tweet 0 0

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

15 Comments GeeksforGeeks

Sort by Newest ▾

Join the discussion…

Tarzan • a month ago

Trie is such a complex solution. Here is a much easier Java solution
Create a HashMap of the type <string, list<string="">>

The key string here will be the sorted string and list will contain all the strings f
to the sorted key. For example:
cat and tac will have the same key as act. So its hashmap entry will be
act : cat -> tac

At the end, just traverse over the entire keyset of the hashmap.

^ | ⌄ • Reply • Share ›

alien • 2 months ago

What is time complexity for this solution?

^ | ⌄ • Reply • Share ›

wgpshashank • 5 months ago

Just writing some quick points , let me know if anything wrong here .

1.keeping index is not necessary since you are already sort the work and then
that word (O(k) k is constant here if that exist in trie just add it linked list which
while printing anagram you just need print all such linked list which contains ar
thus extra overhead can be removed.

2. Lets talk about complexity , is it better then hashmap approach ?

we are sorting n words which O(nmlogm) m is length of string , n is no of strir
insertion in O(n) in trie. Overall O(nmlogm) which is same as HashTable so w
should have been instead .

Reply • Share ›

Mukul Rawal • 7 months ago

Assign each character a unique number. Let's say a prime number. For eg: a=
Now, for a string add each character's corresponding unique number, For eg.
Now we will have all the anagrams with same number as the sum of their cha
We can create a bucket linked list with these numbers and print the correspor
linked list.

Reply • Share ›

Vu Duc Minh → Mukul Rawal • 4 months ago

Hashing with the summation of irrational number like sqrt(2), sqrt(3) ...
overflow.

Reply • Share ›

Shubhankar Srivastava → Mukul Rawal • 6 months ago

by that logic "ef (11 + 13)" is same as "ch (5 + 19)". I think we can grou
and same sum, and then check for anagrams by calculating the freque
comparing it with other strings to check if they are anagrams or not.

Reply • Share ›

Satya • 9 months ago

What will be the space and time complexity of this code??

2 • Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

Reply • Share ›

sreeram • a year ago
i think instead of hash tables we can have a int num in the trie node which can

lets say the 5th string is inserted and the num will be 5
lets say we get the anagram of the string at 7 th string we can update num to l

in the end we can finally split the number to print them
1 ^ | v • Reply • Share ›

sreeram ➦ sreeram • a year ago
I meant instead of linked lists
^ | v • Reply • Share ›

Aashish ➦ sreeram • a year ago
The idea seems good. But, it is limited to addressing the anagr
1 ^ | v • Reply • Share ›

memo • a year ago
Could you please give complexity analysis of above program?

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

^ | v • Reply • Share ›

Palash • a year ago
I think, trie of hash-table would be more efficient.
^ | v • Reply • Share ›

R • 2 years ago
I am not able to visualize trie DS. Can you please add one diagram as well to i

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

Reply • Share ›

Jeet • 2 years ago

Nice

Reply • Share ›

Subscribe   Add Disqus to your site