Given a sequence of words, print all anagrams together | Set 1

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”.

A **simple method** is to create a Hash Table. Calculate the hash value of each word in such a way that all anagrams have the same hash value. Populate the Hash Table with these hash values. Finally, print those words together with same hash values. A simple hashing mechanism can be modulo sum of all characters. With modulo sum, two non-anagram words may have same hash value. This can be handled by matching individual characters.

Following is **another method** to print all anagrams together. Take two auxiliary arrays, index array and word array. Populate the word array with the given sequence of words. Sort each individual word of the word array. Finally, sort the word array and keep track of the corresponding indices. After sorting, all the anagrams cluster together. Use the index array to print the strings from the original array of strings.

Let us understand the steps with following input Sequence of Words:

```
"cat", "dog", "tac", "god", "act"
```

1) Create two auxiliary arrays index[] and words[]. Copy all given words to words[] and store the original indexes in index[]

```
index[]:  0   1   2   3   4
words[]: cat dog tac god act
```

2) Sort individual words in words[]. Index array doesn’t change.

```
index[]:   0    1    2    3    4
words[]:  act  dgo  act  dgo  act
```

3) Sort the words array. Compare individual words using strcmp() to sort

```
index:     0    2    4    1    3
words[]:  act  act  act  dgo  dgo
```

4) All anagrams come together. But words are changed in words array. To print the original words, take index from the index array and use it in the original array. We get

```
"cat tac act dog god"
```

Following is C implementation of the above algorithm. In the following program, an array of structure “Word” is used to store both index and word arrays. DupArray is another structure that stores array of structure “Word”.

```
// A program to print all anagarms together
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// structure for each word of duplicate array
struct Word
{
    char* str;  // to store word itself
    int index; // index of the word in the original array
};

// structure to represent duplicate array.
struct DupArray
{
    struct Word* array;  // Array of words
    int size;   // Size of array
};

// Create a DupArray object that contains an array of Words
struct DupArray* createDupArray(char* str[], int size)
{
    // Allocate memory for dupArray and all members of it
    struct DupArray* dupArray =
              (struct DupArray*) malloc( sizeof(struct DupArray) );
    dupArray->size = size;
    dupArray->array =
```


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

```
                (struct Word*) malloc( dupArray->size * sizeof(struct Wo

    // One by one copy words from the given wordArray to dupArray
    int i;
    for (i = 0; i < size; ++i)
    {
        dupArray->array[i].index = i;
        dupArray->array[i].str = (char*) malloc( strlen(str[i]) + 1 );
        strcpy(dupArray->array[i].str, str[i] );
    }

    return dupArray;
}

// Compare two characters. Used in qsort() for sorting an array of cha
int compChar(const void* a, const void* b)
{
    return *(char*)a - *(char*)b;
}

// Compare two words. Used in qsort() for sorting an array of words
int compStr(const void* a, const void* b)
{
    struct Word* a1 = (struct Word *)a;
    struct Word* b1 = (struct Word *)b;
    return strcmp(a1->str, b1->str);
}

// Given a list of words in wordArr[],
void printAnagramsTogether(char* wordArr[], int size)
{
    // Step 1: Create a copy of all words present in given wordArr.
    // The copy will also have orignal indexes of words
    struct DupArray* dupArray = createDupArray(wordArr, size);

    // Step 2: Iterate through all words in dupArray and sort individu
    int i;
    for (i = 0; i < size; ++i)
       qsort(dupArray->array[i].str,
             strlen(dupArray->array[i].str), sizeof(char), compChar);

    // Step 3: Now sort the array of words in dupArray
    qsort(dupArray->array, size, sizeof(dupArray->array[0]), compStr);

    // Step 4: Now all words in dupArray are together, but these words
    // changed. Use the index member of word struct to get the corresp
    // original word
```

```
    for (i = 0; i < size; ++i)
        printf("%s ", wordArr[dupArray->array[i].index]);
}

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

Output:

```
act tac cat god dog
```

Time Complexity: Let there be N words and each word has maximum M characters. The upper bound is O(NMLogM + MNLogN).
Step 2 takes O(NMLogM) time. Sorting a word takes maximum O(MLogM) time. So sorting N words takes O(NMLogM) time. step 3 takes O(MNLogN) Sorting array of words takes NLogN comparisons. A comparison may take maximum O(M) time. So time to sort array of words will be O(MNLogN).

We will soon be publishing more efficient methods to solve this problem. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recent Comments

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)

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

15 Comments GeeksforGeeks

Sort by Newest

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

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

guest • 2 months ago

modulo sum method as suggested initially wouldn't be right because the sum well got by adding any other set of numbers.. i

  • Reply • Share ›

kanhaiya • 3 months ago

@geeksforgeeks why this code is giving error while compiling it as c++ code i DupArray* but why

  • Reply • Share ›

Rajnish Garg • 3 months ago

```
import java.util.*;

public class PrintAllAnagramsTogether {
public HashMap<string,arraylist<string>> words = new HashMap<string, array

public void seqWords(String[] arr) {
for(String word: arr) {
char[] c = word.toCharArray();
Arrays.sort(c);
String Sortedword = new String(c);
if(words.containsKey(Sortedword)) {
words.get(Sortedword).add(word);
}
else {
ArrayList<string> ar = new ArrayList<string>();
ar.add(word);
words.put(Sortedword, ar);
}
```

see more

Reply • Share ›

Siddhartha Banerjee • 6 months ago

You can sort a word of characters in O(M) time using counting sort (instead of O(NM + MNlogN)

2 Reply • Share ›

Sumit Monga • 7 months ago

Another solution using CountSort and then RadixSort to sort the individual cha

```
#include<stdio.h>

#include<string.h>

#include<iostream>

#include<stdlib.h>

using namespace std;

#define RANGE 255

typedef struct Word

{

char * str;

int ind;
```

see more

Reply • Share ›

Kalyani • 8 months ago

I didn't get hw this qsort will work exactly.. I got the overall idea of the logic., bu of function qsort()

Reply • Share ›

gargsanjay • 10 months ago

array is a pointer to word struct still we are using array[i].word
isn't this wrong??

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

Reply • Share ›

pritybhudolia • 11 months ago

The idea that I have used here is similar to method 1.The sum of each letter in have same sumvalue.

```
/*
#include<stdio.h>
#include<conio.h>
void printAnagramsTogether(char* wordArr[], int size)
{
     int i=0,j=0,k=0,temp,index;
    int sum[10][10]={0};
    //here sum is a 2d array that contains the  letter in each word a
    for(i=0;i<size;i++)
    {

        for(j=0;wordArr[i][j]!='&#92&#48';j++)
         {

                                        sum[k][0]=sum[k][0]+(int)word
```

• Reply • Share ›

Amit Agnihotri ➦ pritybhudolia • 8 months ago

not necessarily
ex: "aabc" and "bbc" will have same sum.
similary "abc" and "aad".....

1 • Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

• Reply • Share ›

atul • 2 years ago

another approach:-
1) sort the input
2) insert sorted input in trie
3) "at the end of word" node of trie..maintain linked list which contain actual wo
4) if already present then append to the linked list maintained.

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

1 • Reply • Share ›

Code_Addict ➦ atul • 4 months ago

This method is discussed in Set 2 of same above problem.

http://www.geeksforgeeks.org/g...

• Reply • Share ›

Code_Addict ➦ atul • 4 months ago

Great solution with another DS. Thanks

monika ➦ atul • a year ago

really...a very good solution!!

• Reply • Share ›

atul • 2 years ago

another approach:-
1) sort the input
2) insert sorted input in trie
3) "at the end of word" node of trie..maintain linked list which contain actual wo

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

• Reply • Share ›