

Lexicographic rank of a string

Given a string, find its rank among all its permutations sorted lexicographically. For example, rank of "abc" is 1, rank of "acb" is 2, and rank of "cba" is 6.

For simplicity, let us assume that the string does not contain any duplicated characters.

One simple solution is to initialize rank as 1, **generate all permutations in lexicographic order**. After generating a permutation, check if the generated permutation is same as given string, if same, then return rank, if not, then increment the rank by 1. The time complexity of this solution will be exponential in worst case. Following is an efficient solution.

Let the given string be "STRING". In the input string, 'S' is the first character. There are total 6 characters and 4 of them are smaller than 'S'. So there can be $4 * 5!$ smaller strings where first character is smaller than 'S', like following

```
RXXXXX
IXXXXX
NXXXXX
GXXXXX
```

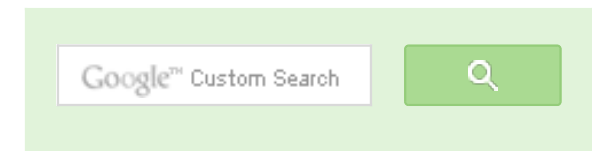
Now let us Fix S' and find the smaller strings starting with 'S'.

Repeat the same process for T, rank is $4*5! + 4*4! + \dots$

Now fix T and repeat the same process for R, rank is $4*5! + 4*4! + 3*3! + \dots$

Now fix R and repeat the same process for I, rank is $4*5! + 4*4! + 3*3! + 1*2! + \dots$

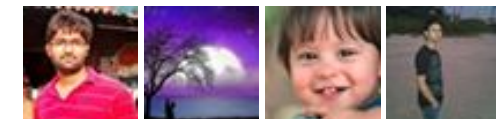
Now fix I and repeat the same process for N, rank is $4*5! + 4*4! + 3*3! + 1*2! + 1*1! + \dots$



GeeksforGeeks



52,731 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

[Geometric Algorithms](#)

Now fix N and repeat the same process for G, rank is $4*5! + 4*4 + 3*3! + 1*2! + 1*1! + 0*0!$

Rank = $4*5! + 4*4! + 3*3! + 1*2! + 1*1! + 0*0! = 597$

Since the value of rank starts from 1, the final rank = $1 + 597 = 598$

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

// A utility function to find factorial of n
int fact(int n)
{
    return (n <= 1)? 1 : n * fact(n-1);
}

// A utility function to count smaller characters on right
// of arr[low]
int findSmallerInRight(char* str, int low, int high)
{
    int countRight = 0, i;

    for (i = low+1; i <= high; ++i)
        if (str[i] < str[low])
            ++countRight;

    return countRight;
}

// A function to find rank of a string in all permutations
// of characters
int findRank (char* str)
{
    int len = strlen(str);
    int mul = fact(len);
    int rank = 1;
    int countRight;

    int i;
    for (i = 0; i < len; ++i)
    {
        mul /= len - i;

        // count number of chars smaller than str[i]
        // from str[i+1] to str[len-1]
        countRight = findSmallerInRight(str, i, len-1);
```



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

```

        rank += countRight * mul ;
    }

    return rank;
}

```

```

// Driver program to test above function
int main()
{
    char str[] = "string";
    printf ("%d", findRank(str));
    return 0;
}

```

Output

598

The time complexity of the above solution is $O(n^2)$. We can reduce the time complexity to $O(n)$ by creating an auxiliary array of size 256. See following code.

```

// A O(n) solution for finding rank of string
#include <stdio.h>
#include <string.h>
#define MAX_CHAR 256

// A utility function to find factorial of n
int fact(int n)
{
    return (n <= 1)? 1 :n * fact(n-1);
}

// Construct a count array where value at every index
// contains count of smaller characters in whole string
void populateAndIncreaseCount (int* count, char* str)
{
    int i;

    for( i = 0; str[i]; ++i )
        ++count[ str[i] ];

    for( i = 1; i < 256; ++i )
        count[i] += count[i-1];
}

// Removes a character ch from count[] array

```



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

AdChoices 

[▶ Java Array](#)

[▶ String Function](#)

```
// constructed by populateAndIncreaseCount()
void updatecount (int* count, char ch)
{
    int i;
    for( i = ch; i < MAX_CHAR; ++i )
        --count[i];
}

// A function to find rank of a string in all permutations
// of characters
int findRank (char* str)
{
    int len = strlen(str);
    int mul = fact(len);
    int rank = 1, i;
    int count[MAX_CHAR] = {0}; // all elements of count[] are initial

    // Populate the count array such that count[i] contains count of
    // characters which are present in str and are smaller than i
    populateAndIncreaseCount( count, str );

    for (i = 0; i < len; ++i)
    {
        mul /= len - i;

        // count number of chars smaller than str[i]
        // from str[i+1] to str[len-1]
        rank += count[ str[i] - 1 ] * mul;

        // Reduce count of characters greater than str[i]
        updatecount (count, str[i]);
    }

    return rank;
}

// Driver program to test above function
int main()
{
    char str[] = "string";
    printf ("%d", findRank(str));
    return 0;
}
```

The above programs don't work for duplicate characters. To make them work for duplicate characters, find all the characters that are smaller (include equal this time also), do the same as

above but, this time divide the rank so formed by $p!$ where p is the count of occurrences of the repeating character.

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



AdChoices

[► C# String Split](#)

[► Replace String](#)

[► String Java](#)

AdChoices

[► String C](#)

[► String String](#)

[► For String](#)

Related Topics:

- [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 \(Longest Common Substring\)](#)



4



Tweet

0



0

Writing code in comment? Please use [ideone.com](https://www.ideone.com) and share the link here.

