

Print all permutations in sorted (lexicographic) order

Given a string, print all permutations of it in sorted order. For example, if the input string is “ABC”, then output should be “ABC, ACB, BAC, BCA, CAB, CBA”.

We have discussed a program to print all permutations in [this](#) post, but here we must print the permutations in increasing order.

Following are the steps to print the permutations lexicographic-ally

1. Sort the given string in non-decreasing order and print it. The first permutation is always the string sorted in non-decreasing order.
2. Start generating next higher permutation. Do it until next higher permutation is not possible. If we reach a permutation where all characters are sorted in non-increasing order, then that permutation is the last permutation.

Steps to generate the next higher permutation:

1. Take the previously printed permutation and find the rightmost character in it, which is smaller than its next character. Let us call this character as ‘first character’.
2. Now find the ceiling of the ‘first character’. Ceiling is the smallest character on right of ‘first character’, which is greater than ‘first character’. Let us call the ceil character as ‘second character’.
3. Swap the two characters found in above 2 steps.
4. Sort the substring (in non-decreasing order) after the original index of ‘first character’.

Let us consider the string “ABCDEF”. Let previously printed permutation be “DCFEB A”. The next

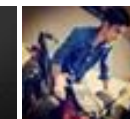
Google™ Custom Search



GeeksforGeeks



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



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

[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

permutation in sorted order should be "DEABCF". Let us understand above steps to find next permutation. The 'first character' will be 'C'. The 'second character' will be 'E'. After swapping these two, we get "DEFCBA". The final step is to sort the substring after the character original index of 'first character'. Finally, we get "DEABCF".

Following is C++ implementation of the algorithm.

```
// Program to print all permutations of a string in sorted order.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* 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 swap two characters a and b
void swap (char* a, char* b)
{
    char t = *a;
    *a = *b;
    *b = t;
}

// This function finds the index of the smallest character
// which is greater than 'first' and is present in str[l..h]
int findCeil (char str[], char first, int l, int h)
{
    // initialize index of ceiling element
    int ceilIndex = l;

    // Now iterate through rest of the elements and find
    // the smallest character greater than 'first'
    for (int i = l+1; i <= h; i++)
        if (str[i] > first && str[i] < str[ceilIndex])
            ceilIndex = i;

    return ceilIndex;
}

// Print all permutations of str in sorted order
void sortedPermutations ( char str[] )
{
    // Get size of string
    int size = strlen(str);
```

```

// Sort the string in increasing order
qsort( str, size, sizeof( str[0] ), compare );

// Print permutations one by one
bool isFinished = false;
while ( ! isFinished )
{
    // print this permutation
    printf ( "%s \n", str );

    // Find the rightmost character which is smaller than its next
    // character. Let us call it 'first char'
    int i;
    for ( i = size - 2; i >= 0; --i )
        if ( str[i] < str[i+1] )
            break;

    // If there is no such character, all are sorted in decreasing order
    // means we just printed the last permutation and we are done.
    if ( i == -1 )
        isFinished = true;
    else
    {
        // Find the ceil of 'first char' in right of first character
        // Ceil of a character is the smallest character greater than it
        int ceilIndex = findCeil( str, str[i], i + 1, size - 1 );

        // Swap first and second characters
        swap( &str[i], &str[ceilIndex] );

        // Sort the string on right of 'first char'
        qsort( str + i + 1, size - i - 1, sizeof( str[0] ), compare );
    }
}

```

```

// Driver program to test above function
int main()
{
    char str[] = "ABCD";
    sortedPermutations( str );
    return 0;
}

```

Output:



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 to C++](#)

[► String Java](#)

[► String Function](#)

ABCD
ABDC
....
....
DCAB
DCBA

The upper bound on time complexity of the above program is $O(n^2 \times n!)$. We can optimize step 4 of the above algorithm for finding next permutation. Instead of sorting the subarray after the 'first character', we can reverse the subarray, because the subarray we get after swapping is always sorted in non-increasing order. This optimization makes the time complexity as $O(n \times n!)$. See following optimized code.

```
// An optimized version that uses reverse instead of sort for
// finding the next permutation
```

```
// A utility function to reverse a string str[l..h]
```

```
void reverse(char str[], int l, int h)
```

```
{
    while (l < h)
    {
        swap(&str[l], &str[h]);
        l++;
        h--;
    }
}
```

```
// Print all permutations of str in sorted order
```

```
void sortedPermutations ( char str[] )
```

```
{
    // Get size of string
    int size = strlen(str);

    // Sort the string in increasing order
    qsort( str, size, sizeof( str[0] ), compare );

    // Print permutations one by one
    bool isFinished = false;
    while ( ! isFinished )
    {
        // print this permutation
        printf ("%s \n", str);
    }
}
```

```


// Find the rightmost character which is smaller than its next
// character. Let us call it 'first char'
int i;
for ( i = size - 2; i >= 0; --i )
    if (str[i] < str[i+1])
        break;

// If there is no such character, all are sorted in decreasing order
// means we just printed the last permutation and we are done.
if ( i == -1 )
    isFinished = true;
else
{
    // Find the ceil of 'first char' in right of first character
    // Ceil of a character is the smallest character greater than it
    int ceilIndex = findCeil( str, str[i], i + 1, size - 1 );

    // Swap first and second characters
    swap( &str[i], &str[ceilIndex] );

    // reverse the string on right of 'first char'
    reverse( str, i + 1, size - 1 );
}
}
}


```

AdChoices 

► [Math String](#)

► [String C](#)

► [String Original](#)

AdChoices 

► [String String](#)

► [New String](#)

► [For String](#)

The above programs print duplicate permutation when characters are repeated. We can avoid it by keeping track of the previous permutation. While printing, if the current permutation is same as previous permutation, we won't print it.

This article is compiled by **Aashish Barnwal** and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



Better Than Hadoop.

HPCC Systems is Big Data Processing and Analytics
Open Source. Proven. Trusted.

 LexisNexis® [Learn More](#) 

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 \(Longest Common Substring\)](#)



8



Tweet

0



1

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

@geeksforgeeks, **Some rights reserved**

Contact Us!

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team