Date: 22/02/19
Page No.

# String permutation Algorithm:

str → input string  aux → for each permutation

① take ith (i = 0 to < size - 1)

aux + str[i];

② Now cut new string with out char i

str = str.substr(0, i) + str.substr(i+1, size);

| | str | aux | ~~aux~~ | ~~str~~ |
|---|---|---|---|---|
| eg: | abc | " | | |
| | ~~a~~bc | a | | |

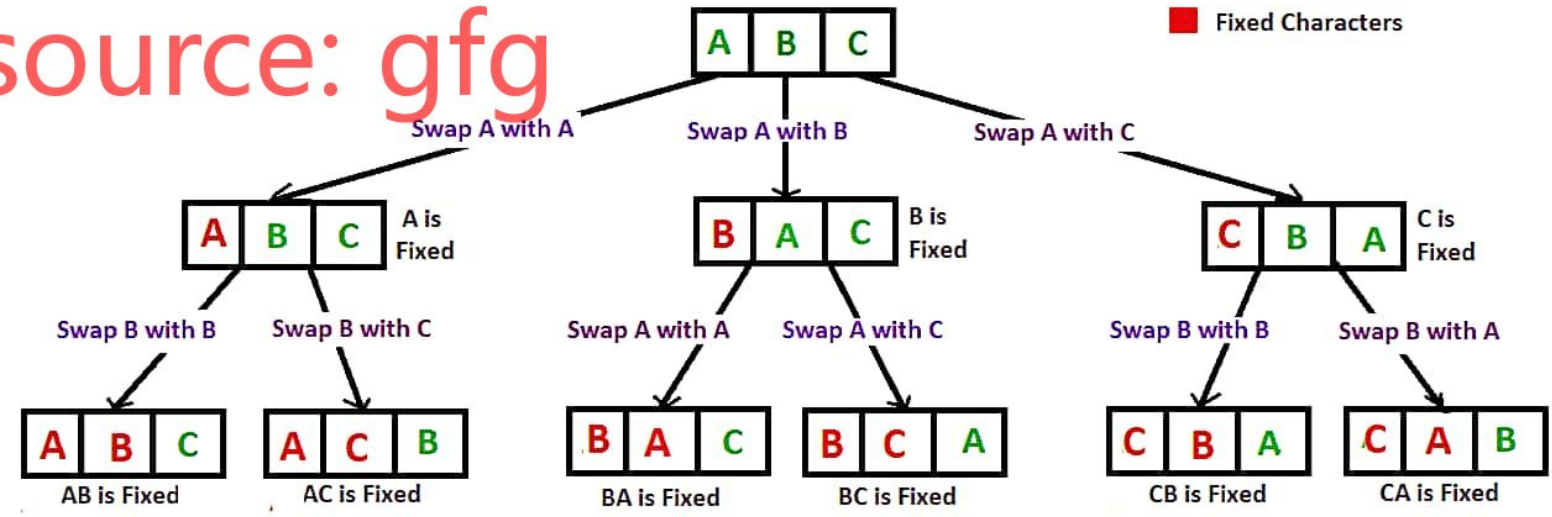
| eg: | aux | string | i |
|---|---|---|---|
| | " " | abc | 0 |
| | a | bc | 0 |
| | ab | c | 0 |
| | abc | " " | 0 |

Now i = 1

| | | |
|---|---|---|
| b | ac | |
| ba | c | |
| bac | " " | |

if str.length() == 0
    result.insert(aux);

source: gfg




**Recursion Tree for Permutations of String "ABC"**

Date :
Page No.

## List permutation :

```
listpermutation (list l, list aux, itr, result){
    if( itr == l.size() -1 ){
        result.insert(l);
        return;
    }
    list aux = l;
    for (i = itr; i<l.size(); i++){
        swap(aux[i], aux[itr]);
        listpermutation (l, aux, itr+1, result);
    }
}
```