Permutation of string

→ (length of string)!

String → 'abc'

Permutation →

1st < abc, acb

bac

bca (2nd)

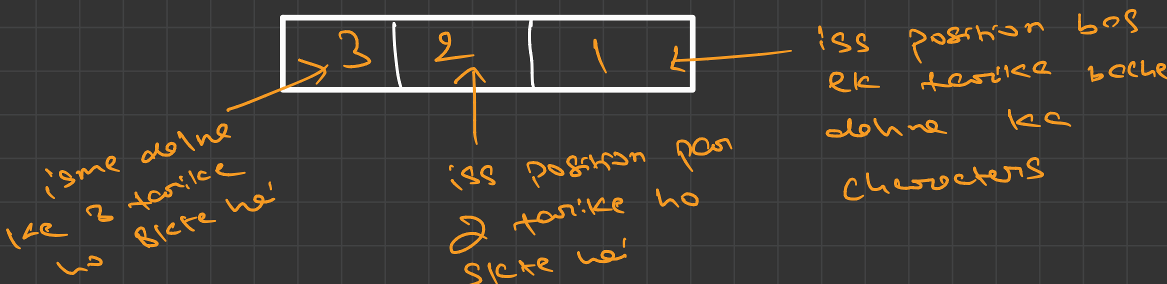
cab

cba (3rd)

also solve with Stl → next_permutation

=> har ek character ko har ek position par rakhna hai

String = "abc"

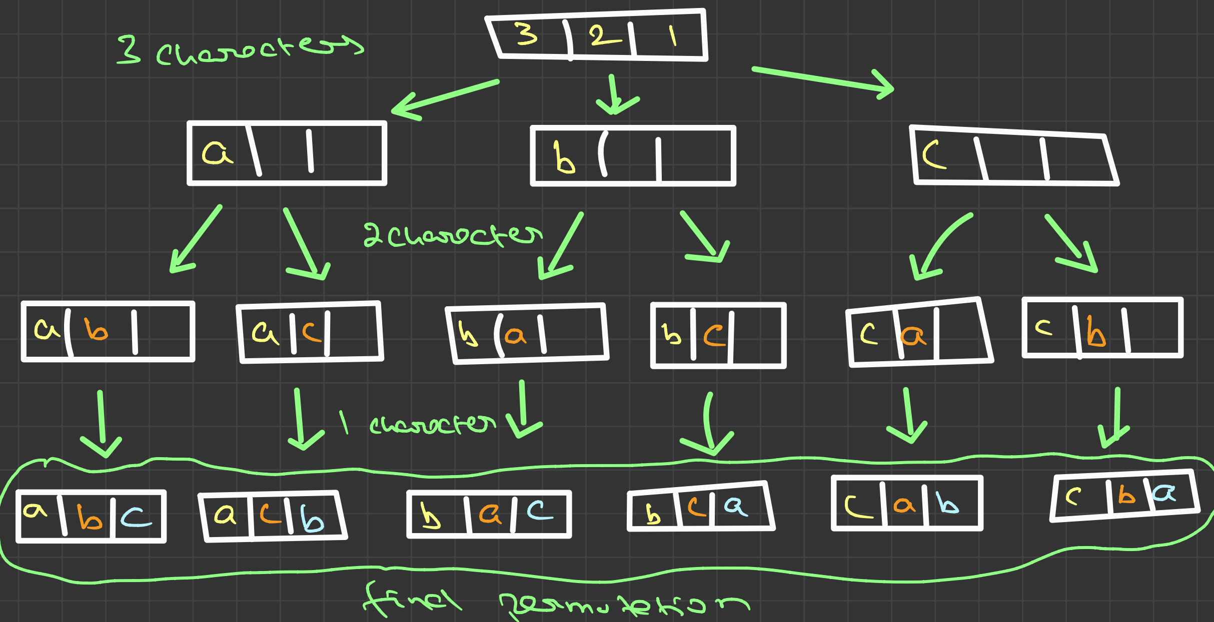


1 box → 3 characters

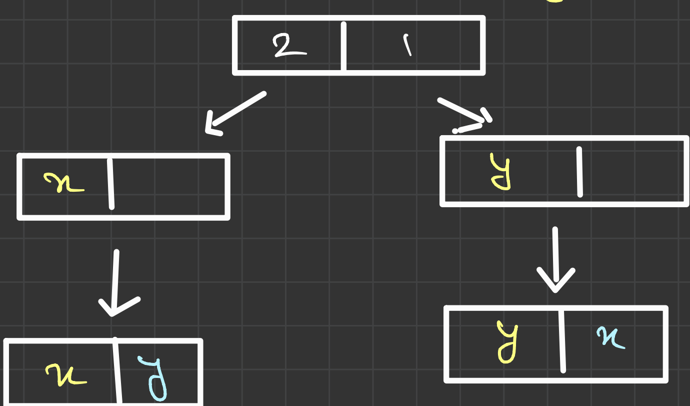
2 box → 2 characters

3 box → 1 character

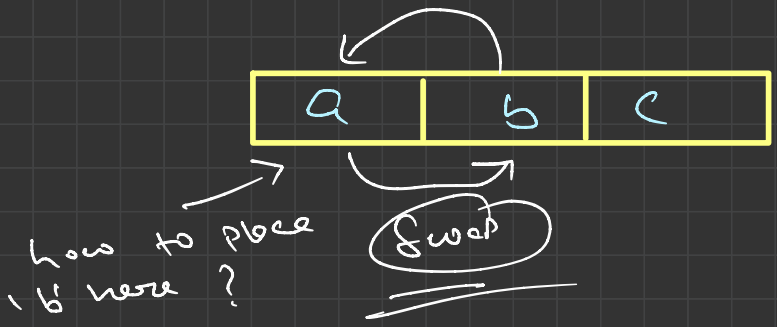
$3 \times 2 \times 1 = 6$ possibilities



String xy → 2! = 2



har ek position par har ek character ko rakhne ka try kar rahe hai



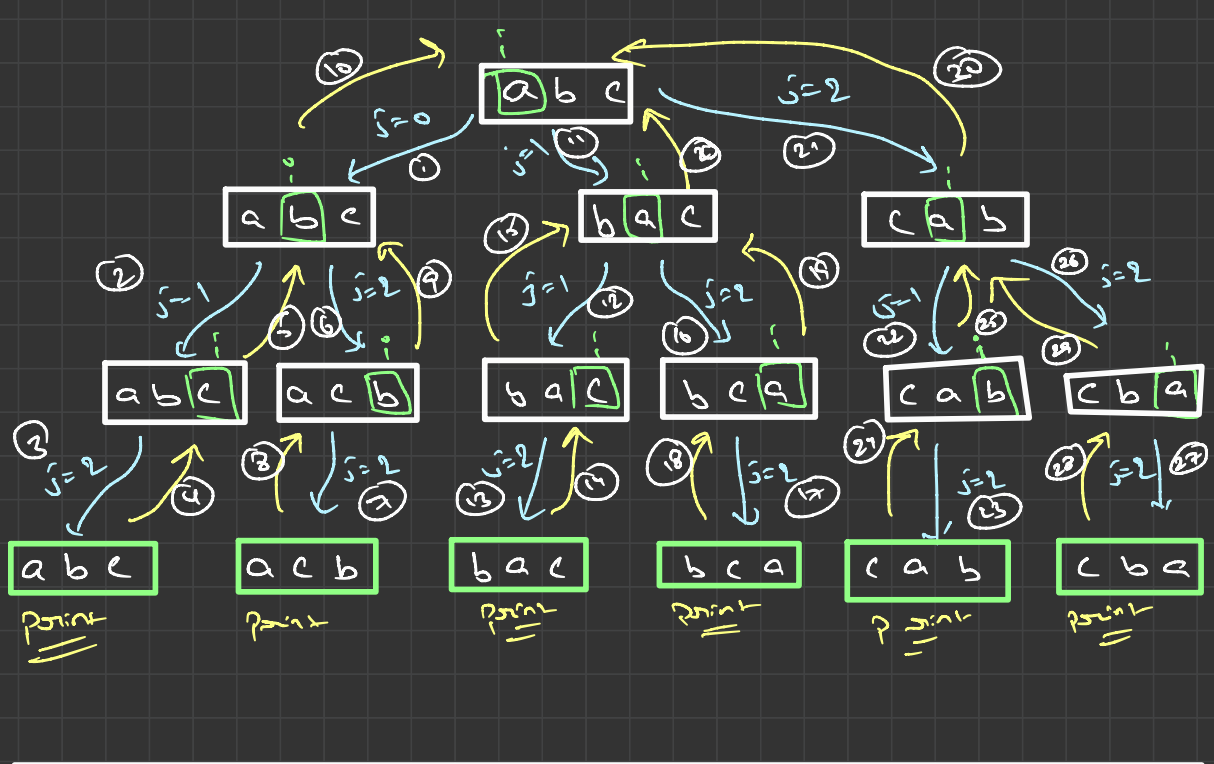
Solve with recursion

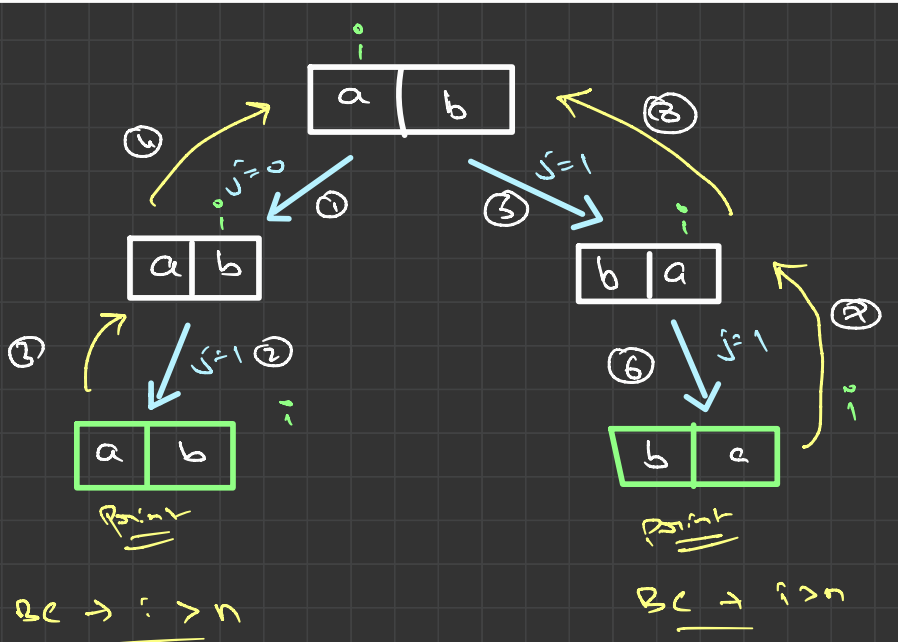
↓

using 2 pointer approach

i → iterate string

j → loop → i < n


a b c
j=0
j=1
j=2
a b c
b a c
c a b
a b c
a c b
b a c
b c a
c a b
c b a
a b c
a c b
b a c
b c a
c a b
c b a
Point
Point
Point
Point
Point
Point


a b
j=0
j=1
a b
b a
a b
b a
Point
Point
Bc → i > n
Bc → i > n

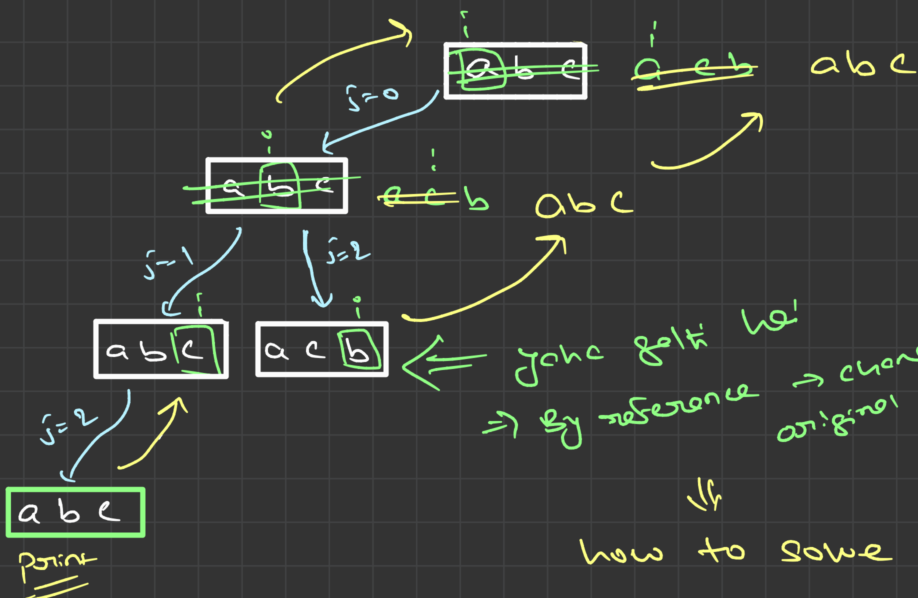
```
void printPermu( String str , int i )
{
    // Base case
    if ( i >= str.length() -1 )
    {
        cout << str << "  " ;
        return;
    }

    for( int j = i ; j < str.length() ; j++ )
    {
        swap( str[i] , str[j] );

        // rec call
        printPermu( str , i+1 );
    }
}

int main()
{
    String str = " a b c " ;
    int i = 0 ;
    printPermu ( str , i ) ;
}
```

By calling by reference Str it prints wrong ones



⇐ yahi galti hai

⇒ By reference → change original string

⇓

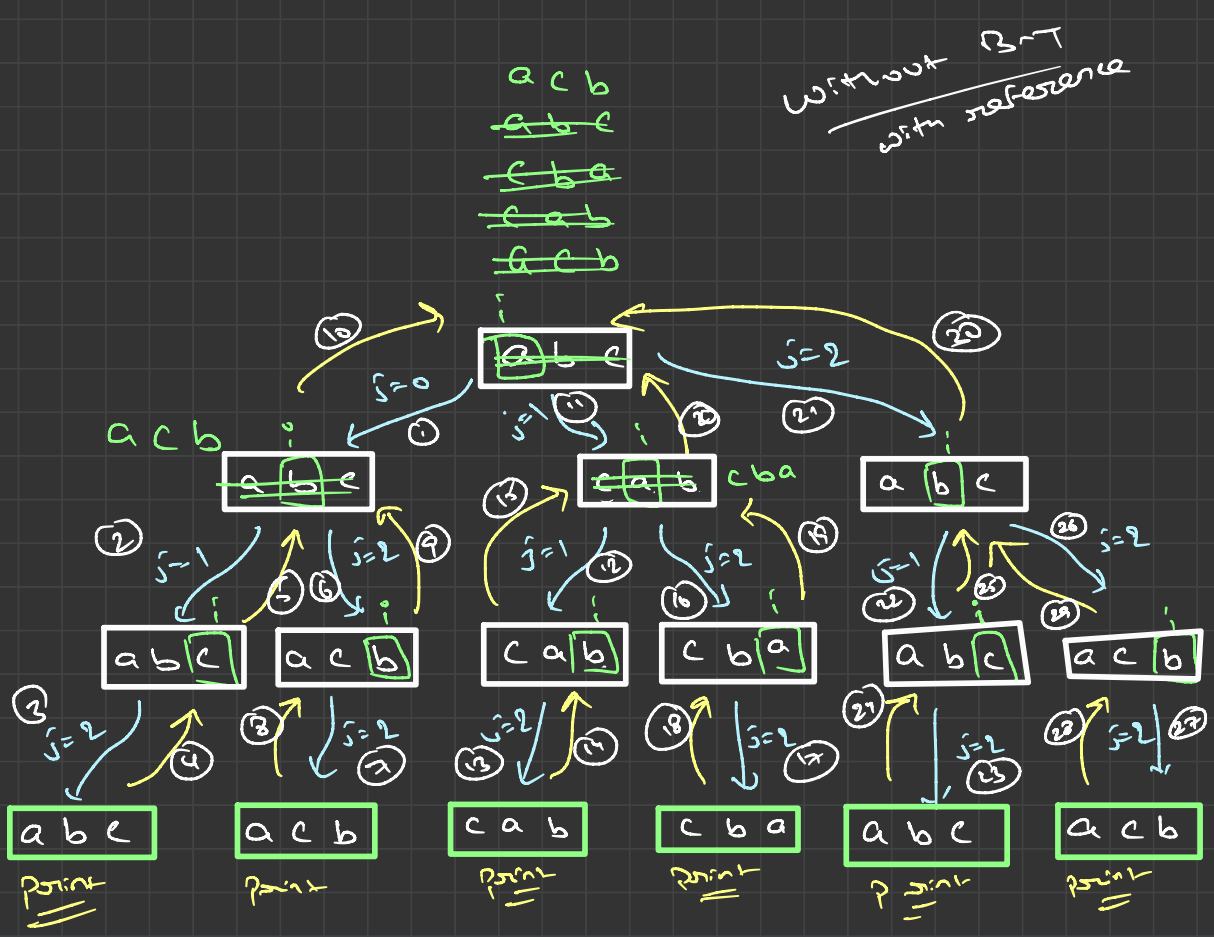
how to solve ?

⇓

when we backtrack → swap again

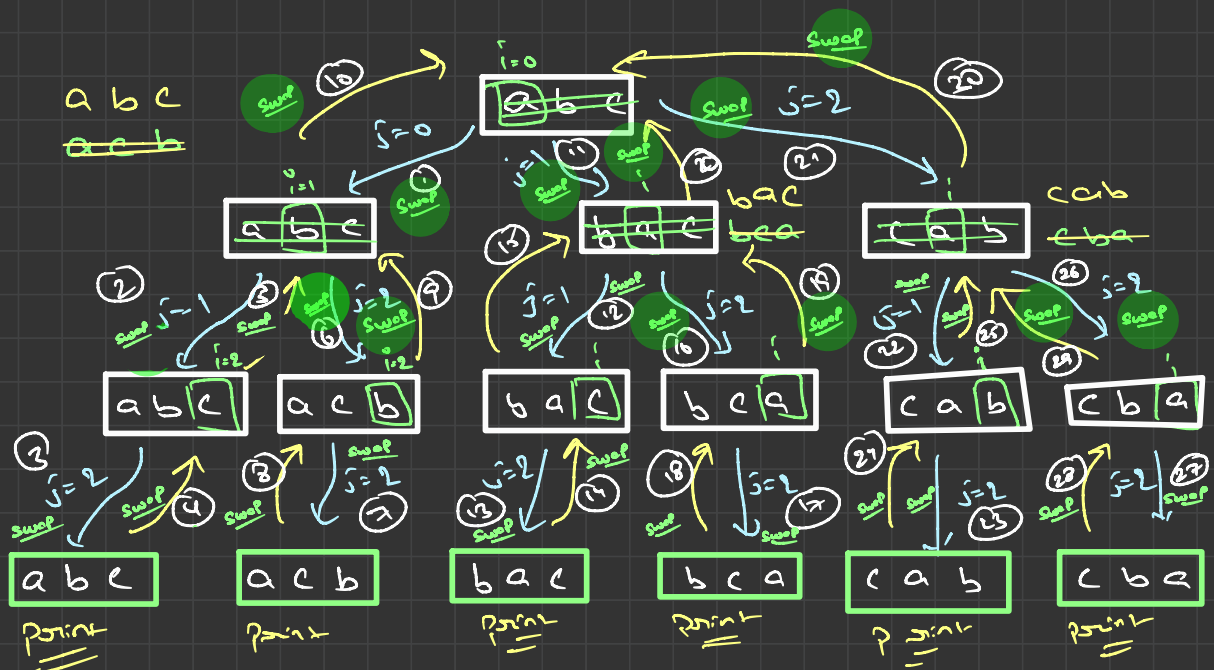
swap(str[i], str[j])

backtracking → nothing → simple recursion → rec call se wapas aate hue kux additional operation perform karna lege


Without BT
with reference
a c b
a b c
j=0
j=1
j=2
a c b
c b a
a b c
a c b
c a b
c b a
a b c
a c b
a b c
a c b
c a b
c b a
a b c
a c b
Print
Print
Print
Print
Print
Print

With B-T
With reference

~~abc~~ → ~~cba~~ → abc
~~bac~~ → ~~bca~~ → ~~bac~~ → ~~abc~~
~~acb~~ → ~~abc~~



Solve with STL → next_permutation

next_permutation (s.begin() , s.end())

① abc

↳ acb

↳ bac

↳ bca

↳ cba

↳ cab

↳ a

```
=> s = "abc";
cout << s << endl;
while (next_permutation (s.begin(), s.end()))
{
    cout << s;
}
```