Assignment on Backtracking

Submitted by

Khondaker Tasnia Hoque

BSSE 1205

Batch : 12th

IIT, DU

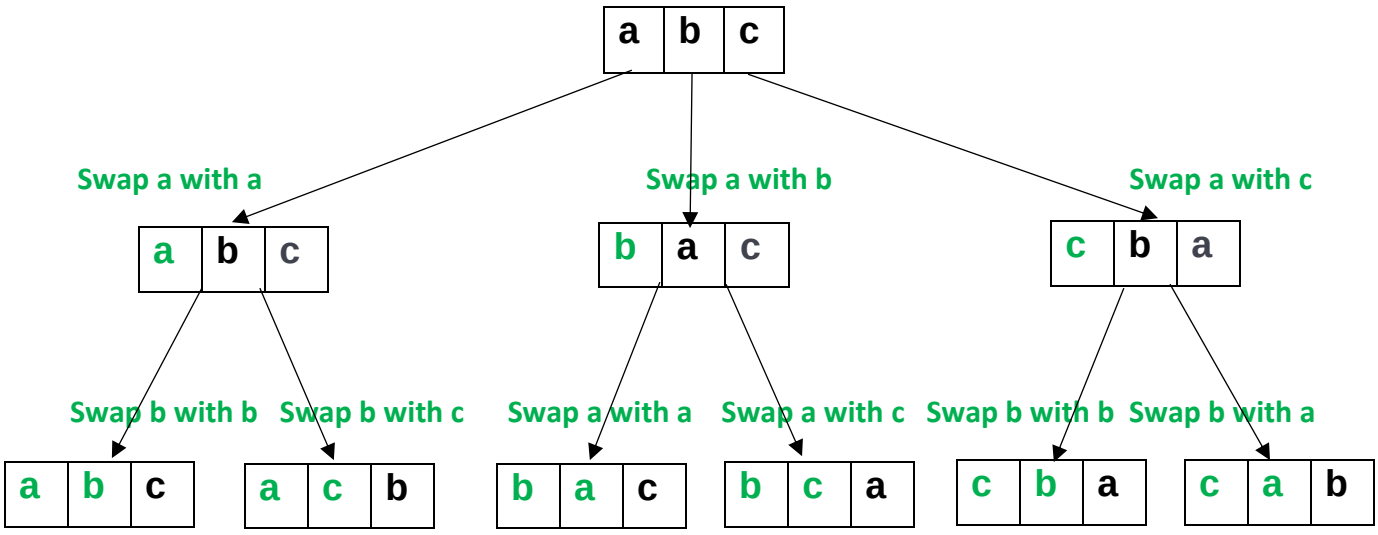
Task – 1 (Algorithm to print all permutations of a given string)

Step-1: Start
Step-2: Take a set of string . set<string>s
Step-3: Input string str
Step-4: n → str. size( )
Step-5: permute(str, 0 ,n-1)
Step-6: For i=s.begin() to i!=s.end() do step 7
Step-7: Cout<<*i
Step-8: End

permute(str, l ,r)

Step-1: l → start
Step-2: r → end
Step-3: If (l=r) do step 4
Step-4: s.insert(str)
Step-5: Else for i=0 to r do step 6 to 8
Step-6: swap(str[l],str[i])
Step-7: permute(str, l+1, r)
Step-8: swap(str[l],str[i])
Step-9: End

Working Methodology Diagram :



At first a string type variable str will take a string as input. Then a user defined function permute is called. According to the algorithm we have to pass the string , the starting index and the size of the string or the last index of the string as parameter in step 5. In permute

function we have to run a for loop from the starting index to last and swap the first index with itself at first in step 6 then we have called permute function again in step 7. Here second index is swapped with second index. Then again second index is swapped with third index in step 8 and so on. When the first and last index will be same then we have pushed the new string after permutation in set to avoid duplicate string.

Task -2 (Algorithm to print all combination of a string having k elements from a string having n elements)

Step-1: Start

Step-2: Take a set of string . set<string>s

Step-3: Input string str

Step-4: $n \rightarrow str.size( )$

Step-5: Input integer k

Step-6: combination(str, data, 0 , n-1 , 0 , k)

Step-7: For i=s.begin( ) to i!=s.end( ) do step 8

Step-8: Cout<<*i

Step-9: End

combination(str,data,start,end,index,k)

Step-1: If (index=k) do step 2 to 5
Step-2: Take an empty string S1
Step-3: For j=0 to k
S1=S1+data[j]
Step-4: sort(S1.begin( ),S1.end( ))
Step-5: s.insert(S1)
Step-6: For i=start to end && end-i+1>=k-index
do step 7 to 8
Step-7: swap(data[index],str[i])
Step-8: combination(str,data,i+1,end,index+1,k)
Step-9: End

Modifications:

Comparing with the algorithm in task 1 , in first part of second algorithm there is one extra step that is step 5 where we have taken the integer k . In step 6 , there are 3 extra parameters we have inserted in combination function those are an extra string “data” where we will store k characters selecting from a string of n characters, the first index of data and the last index of data.

In part 2, combination function, we have brought a modification that is instead of inserting the string in set directly , we have first store the modified string in an empty string and have sorted.

Sorting is because of avoiding duplicate combinations. Then we have inserted the modified string into set like the first algorithm.

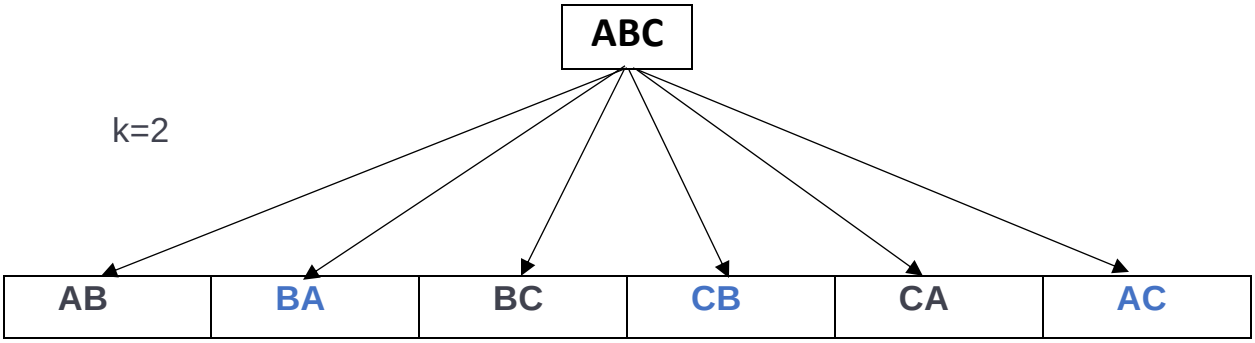
In step 6 there is a change compared to step 5 of algorithm 1, that is i<=end && end-i+1>=k-index, it is because , so that we can create a new string of length k.

The last modification is, in algorithm 1 , we have swapped twice in the user defined function where in algorithm 2 we have swapped once.

In step 4 of combination function , we have sorted the string of length k while we have not sort any string in algorithm 1. Here, the diagram for algorithm 2 will clear the matter of sorting.

Let, a given string is ABC and we have to chose k letters from the string. Let the value of k is 2.

Now according to the algorithm 2, we will get :



To avoid the duplicate case like AB and BA, we have sorted the string and made both string as AB and AB. Then inserted both into set and thus finally we have got the unique combinations

AB, BC, AC.

--------END--------