Subsequences of strings

You are given a string 'STR' containing lowercase English letters from a to z inclusive. Your task is to find all non-empty possible subsequences of 'STR'.

A Subsequence of a string is the one which is generated by deleting 0 or more letters from the string and keeping the rest of the letters in the same order.

Detailed explanation (Input/output format, Notes, Images)

Constraints:

1 <= T <= 10
1 <= |STR| <= 16

Where |STR| represents the length of the string 'STR'.

Time Limit: 1 sec

Sample Input 1:

1
abc

Sample Output 1:

a ab abc ac b bc c

Explanation of sample input 1:

All possible subsequences of abc are :
“a” , “b” , “c” , “ab” , “bc” , “ac”, “abc”

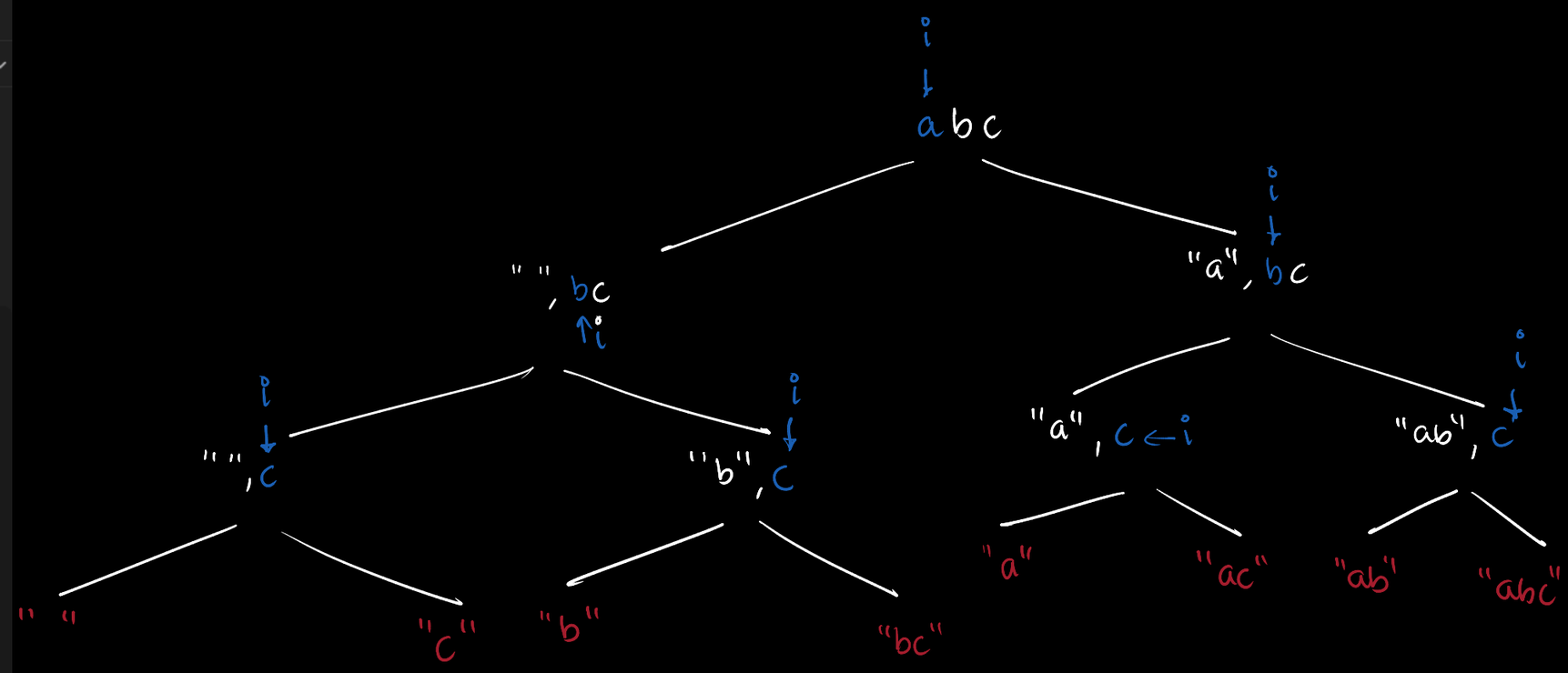
Sample Input 2:

1
bbb

Sample Output 2:

b b b bb bb bb bbb

→ Using recursion :-



→ The base case will be as soon as i >= n, we will push the string to our vector of string.

n → size of str.

```
#include <bits/stdc++.h>

void getTheSubsequence(vector<string> &ans, string str, int i, int n, string s) {
    if(i >= n) {
        if(s != "")
        ans.push_back(s);
        return;
    }

    getTheSubsequence(ans, str, i + 1, n, s);
    s = s + str[i];
    getTheSubsequence(ans, str, i + 1, n, s);
}

vector<string> subsequences(string str){
    vector<string> ans;
    string s = "";
    int i = 0, n = str.length();
    getTheSubsequence(ans, str, i, n, s);

    return ans;
}
```

Time Complexity	O(2^N)
Space Complexity	O(2^N)

→ Bitwise

000 → ""
001 → "c"
010 → "b"
011 → "bc"
100 → "a"
101 → "ac"
110 → "ab"
111 → "abc"

```
#include <bits/stdc++.h>
vector<string> subsequences(string str) {

  vector<string> ans;

  int length = str.length();

  int endIndex = 1 << length;

  for (int i = 1; i < endIndex; i++) {
    int j = length - 1, no = i;

    string output = "";

    while (no) {
      if ((no & 1)) {
        output = str[j] + output;
      }
      j--;
      no >>= 1;
    }
    if (output != "") {
      ans.push_back(output);
    }
  }

  return ans;
}
```

→ 2^N

→ $\log 2^N = N * \log 2$

Time Complexity	$O(N * 2^N)$
Space Complexity	$O(N)$