Subset of String - Quick Revision Notes

Logic of the Given Code:

- This program generates all subsequences (subsets) of a given string using recursion.
- A subsequence is a sequence that can be derived from another string by deleting some or no elements without changing the order.
- The function subs(ans, original) works as follows:
  1. If the original string becomes empty, it prints the current subsequence and returns.
  2. Otherwise, the first character of original is extracted.
  3. Two recursive calls are made:
     - One excluding the current character.
     - One including the current character.

Code with Comments:

```
#include<iostream>
using namespace std;

// Function to generate subsequences of a string using recursion
void subs(string ans, string original){
    // Base case: If original string is empty, print the current subsequence
    if(original == ""){
        cout << ans << "\n";
        return;
    }

    char ch = original[0]; // Extract the first character

    // Recursive call excluding the current character
    subs(ans, original.substr(1));

    // Recursive call including the current character
    subs(ans + ch, original.substr(1));
}

int main(){
    string str = "abc";
    subs("", str); // Generate and print all subsequences of "abc"
}
```

Dry Run of the Code (For "abc")

Function Calls Breakdown:

```
subs("", "abc")
  subs("", "bc")  // Exclude 'a'
     subs("", "c")  // Exclude 'b'
        subs("", "")  prints ""
        subs("c", "")  prints "c"
     subs("b", "c")  // Include 'b'
        subs("b", "")  prints "b"
        subs("bc", "")  prints "bc"
  subs("a", "bc")  // Include 'a'
     subs("a", "c")  // Exclude 'b'
        subs("a", "")  prints "a"
        subs("ac", "")  prints "ac"
     subs("ab", "c")  // Include 'b'
        subs("ab", "")  prints "ab"
        subs("abc", "")  prints "abc"
```

Output:

```
""
"c"
"b"
"bc"
"a"
"ac"
"ab"
"abc"
```

Time Complexity Analysis:

- Each character has two choices (either include or exclude).
- If the input string has n characters, we generate 2^n subsequences.
- The time complexity is O(2^n), which is exponential.

Key Takeaways:

- The recursive function explores all possible subsequences.

- Uses two recursive calls for each character: one excluding and one including it.
- Time complexity is O(2^n) because each character has two choices.
- The base case ensures that the function stops when the original string is empty.