

```
public static int compute(String s) {  
    if(s.equals("0")){  
        return 0;  
    }  
    if(s.equals("1")){  
        return 1;  
    }  
    if (s.charAt(s.length()-1) == '0'){  
        return 2 * compute(s.substring(0,s.length()-1));  
    }  
    if (s.charAt(s.length()-1) == '1'){  
        return 1 + 2 * compute(s.substring(0,s.length()-1));  
    }  
    return -1;  
}
```

The diagram illustrates the recursive call tree for the `compute` function. Nodes are colored purple, red, and yellow, and connected by lines of the same color. The tree shows the sequence of recursive calls and returns for a given input string `s`.

- Purple nodes:** Represent the initial call to `compute(s)` and the return values from the base cases (`0` and `1`).
- Red nodes:** Represent the recursive calls to `compute(s.substring(0, s.length()-1))` when the last character is `'0'`.
- Yellow nodes:** Represent the recursive calls to `compute(s.substring(0, s.length()-1))` when the last character is `'1'`.

The tree structure shows how the function calls itself with a substring of length `s.length()-1` until it reaches the base cases, and then returns the result back up the call stack.