

```
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 calls for the function `compute` with the input string `"1010"`. The nodes represent the state of the recursive process, and the edges represent the sequence of calls and returns.

- Nodes:**
 - Orange nodes: Represent the initial call `compute("1010")` and the recursive call `compute("101")`.
 - Red nodes: Represent the recursive call `compute("10")` and the base case `compute("1")`.
 - Yellow nodes: Represent the recursive call `compute("0")` and the base case `compute("0")`.
 - Purple nodes: Represent the recursive call `compute("1010")` and the base case `compute("1")`.
- Edges:**
 - Orange edges: Connect the initial call `compute("1010")` to `compute("101")` and back.
 - Red edges: Connect `compute("101")` to `compute("10")` and back, and `compute("10")` to `compute("1")` and back.
 - Yellow edges: Connect `compute("10")` to `compute("0")` and back.
 - Purple edges: Connect `compute("1010")` to `compute("101")` and back, and `compute("101")` to `compute("10")` and back.