CPT206 CW1 Week 6 Report - 2252377

05/04/2025

1 Code Explanation

1.1 Source code

```java
public class Main {
    public static void main(String[] args) {
        System.out.println(countOccurrences("I have a bananananana",
            "na"));//The result should be 4.
    }

    public static int countOccurrences(String input, String sub) {
        //First handle the boundary condition.
        if (sub == null || sub.isEmpty() || input.length() < sub.length()) {
            return 0;
        }

        //Using recursive techniques to count the times of repetition.
        int CountRepe = 0;
        if (input.startsWith(sub)) {
            CountRepe = 1;
        }
        return CountRepe + countOccurrences(input.substring(1), sub);
    }
}
```

1.2 Key features:

- Boundary condition: If the length of total input is less than that of the sub or null input, return 0.
- Using recursion to count the total repetition times. Continuously extract the first character and compare it with the first character of the remaining string. Return 1 if they are the same, otherwise return 0.
  Recursion sums up all these returned values to get the total number of repetitions.This method is quite similar to the 'factorial calculation'.
  Visualization(example): 1+(1+(0+(1+(1+(0+(0+...

- Initializing CountRepe to 0: recursive flow will add up values from all levels of recursion. If the condition is not met, adding 0 (the default value) will not affect the final result. This ensures the summation logic remains correct.

2 Development & Testing

2.1 Development Process

2.1.1 First programming

I set the original boundary condition as: $input.length() < sub.length()$

2.1.2 Improvement(assisted with Deepseek-R1)

AI added two conditions to address cases where the substring is empty and deleted the 'equal' sign of my original condition. To simplify the boundary conditions, I combined the three together.

```
public static int countOccurrences(String input, String sub) {
    // 处理子串为空的情况
    if (sub == null || sub.isEmpty()) {
        return 0; // 或 throw new IllegalArgumentException()
    }

    // 修正边界条件：当剩余长度不足时终止递归
    if (input.length() < sub.length()) {
        return 0;
    }
```

Figure 1: AI-assisted Code Improving

2.2 Testing

input	sub	Expected	Actual
"hello all!"	"ll"	2	2
"an input string"	"not a sub-string"	0	0
"I have a bananananana"	"na"	4	4
"I have a pencil"	""	0	0
"aaaaaaaaaa"	"aa"	9	9

3. Personal Reflection

- The complexity of this method is O(n), which performs very well(each character checked only once).
- When using this method, there is a potential risk of integer overflow. However, for most string tests, the int type is generally sufficient. To ensure robustness and prevent overflow in extreme cases, it might be safer to replace int with long for the return type and counter variables.