Split A String Into The Max Number Of Unique Substrings

solved by	Senan
Platform	LeetCode
difficulty	Medium
# Serial	1593
tags	Backtracking String Manipulation
language	C++
solved on	@21/10/2024
link	https://leetcode.com/problems/split-a-string-into-the-max-number-of-unique-substrings/description/
Completion	☑

Intuition

The problem involves splitting a string into the maximum number of unique substrings. We aim to explore different ways of dividing the string and ensure that each substring is distinct. By using recursion and backtracking, we can attempt to find the maximum number of unique splits possible by exploring all valid substrings and keeping track of the substrings we've already used.

Approach

1. Use recursion with backtracking to explore all possible ways to split the string.
2. At each recursive step, iterate through the remaining part of the string, generate substrings, and check if they are unique.
3. If a substring is unique (i.e., not in the set of already used substrings), add it to the set and recursively attempt to split the remaining string.
4. Keep track of the maximum number of unique substrings encountered during this process.
5. Use backtracking to revert the state by removing the substring from the set once the recursion for that branch is complete.

Complexity

Time Complexity:

The time complexity is **O(n2^n)**, where `n` is the length of the string `s`. This is because, in the worst case, we are exploring all possible ways of splitting the string into substrings.

Space Complexity:

The space complexity is **O(n)** due to the recursion depth and the storage required for the set of substrings (which could potentially hold up to `n` substrings if each character is considered separately).

Code

```
class Solution {
    int count(int j, string& s, unordered_set<string>& mySet) {
        if (j >= s.size())
            return 0;
        int maxCount = 0;
        for (int i = j; i < s.size(); i++) {
            string temp = s.substr(j, i - j + 1);
            if (mySet.find(temp) == mySet.end()) {
                mySet.insert(temp);
                maxCount = max(maxCount, 1 + count(i + 1, s, mySet));
                mySet.erase(temp);
            }
        }
        return maxCount;
    }

public:
    int maxUniqueSplit(string s) {
        unordered_set<string> mySet;
        return count(0, s, mySet);
    }
};
```