Back tracking

Wednesday, 22 May 2024 12:21 PM

All 08/10/25 2:33 PM problems are composed by these three steps: choose, explore, unchoose.
https://leetcode.com/problems/palindrome-partitioning/
So for each problem, you need to know:

1. choose what? For this problem, we choose each substring.
2. how to explore? For this problem, we do the same thing to the remained string.
3. unchoose Do the opposite operation of choose.

Let's take this problem as an example:
1.Define helper(): Usually we need a helper funcition in backtracking problem, to accept more parameters.
2.Parameters: Usually we need the following parameters
1. The object you are working on: For this problem is String s.
2. A start index or an end index which indicate which part you are working on: For this problem, we use substring to indicate the start index.
3. A step result, to remember current choose and then do unchoose : For this problem, we use List<String> step.
4. A final result, to remember the final result. Usually when we add, we use 'result.add(new ArrayList<>(step))' instead of 'result.add(step)', since step is reference passed. We will modify step later, so we need to copy it and add the copy to the result;
3.Base case: The base case defines when to add step into result, and when to return.
4.Use for-loop: Usually we need a for loop to iterate though the input String s, so that we can choose all the options.
5.Choose: In this problem, if the substring of s is palindrome, we add it into the step, which means we choose this substring.
6.Explore: In this problem, we want to do the same thing to the remaining substring. So we recursively call our function.
7.Un-Choose: We draw back, remove the chosen substring, in order to try other options.

1. Create a helper method by passing additional parameters

```
int size;
public int maxUniqueSplit(String s) {
    size = 0;
    Set<String> set = new HashSet<>();
    helper(s, 0, set);
    return size;
}
```

2. Define the end result computation as if condition inside that method
3. Iterate through the length of the given input
4. Create a temp variable to store the provided range
5. Make a decision using if condition when the condition is not satisfied
6. Now choose
7. Compute
8. Un choose

```
private void helper(String s, int start, Set<String> set) {
    if(s==null || s.length()==0){
        size = Math.max(size, set.size());
    }
    for(int i=1; i<=s.length(); i++){
        String tmp = s.substring(0, i);
        if(set.contains(tmp)){
            continue;
        }
        set.add(tmp);
        helper(s.substring(i), i, set);
        set.remove(tmp);
    }
}
```

https://leetcode.com/problems/split-a-string-into-the-max-number-of-unique-substrings/?envType=daily-question&envId=2024-10-21