1593. Split a String Into the Max Number of Unique Substrings

Given a string `s`, return *the maximum number of unique substrings that the given string can be split into*.

You can split string `s` into any list of **non-empty substrings**, where the concatenation of the substrings forms the original string. However, you must split the substrings such that all of them are **unique**.

A **substring** is a contiguous sequence of characters within a string.

Example 1:

```
Input: s = "ababccc"
Output: 5
Explanation: One way to split maximally is ['a', 'b', 'ab', 'c', 'cc']. Splitting like ['a', 'b', 'a', 'b', 'c', 'cc'] is not valid as you have 'a' and 'b' multiple times.
```

Example 2:

```
Input: s = "aba"
Output: 2
Explanation: One way to split maximally is ['a', 'ba'].
```

Example 3:

```
Input: s = "aa"
Output: 1
Explanation: It is impossible to split the string any further.
```

Constraints:

- `1 <= s.length <= 16`
- `s` contains only lower case English letters.

1593. Split a String Into the Max Number of Unique Substrings

```
/*
    Greedy: Doesn't work
    Counter example: "wwwzfvedwfvhsww"
*/
class Solution {
    public:
        int maxUniqueSplit(std::string s){
            int n=s.size();
            std::string sub="";
            int i=0;
            std::unordered_map<std::string,bool> list;
            while(i<n){
                sub+=s[i];
                if(list.find(sub)==list.end()) list[sub]=true,i++;
                else{
                    int j=i+1;
                    while(j<n && list.find(sub)!=list.end()){
                        sub+=s[j];
                        j++;
                    }
                    list[sub]=true;
                    i=j;
                }
                sub="";
            }

            return list.size();
        }
};
```

1593. Split a String Into the Max Number of Unique Substrings

```
/*
    Backtracking
    Time complexity: O(n.2^n)
    Space complexity: O(n)
*/
class Solution {
public:
    int maxUniqueSplit(std::string s){
        int n=s.size();
        std::unordered_map<std::string,bool> list;

        auto solve=[&](int start,auto& self)->int{
            if(start==n) return 0;
            int ans=0;
            for(int end=start+1;end<=n;++end){
                std::string sub=s.substr(start,end-start);
                if(!list[sub]){
                    list[sub]=true;
                    ans=std::max(ans,1+self(end,self));
                    list[sub]=false;
                }
            }
            return ans;
        };

        return solve(0,solve);
    }
};
```

1593. Split a String Into the Max Number of Unique Substrings

```
/*
   Backtracking+pruning
   Time complexity: O(n.2^n)
   Space complexity: O(n)
*/
class Solution {
   public:
      int maxUniqueSplit(std::string s){
         int n=s.size();
         std::unordered_map<std::string,bool> list;
         int ans=0;

         auto solve=[&](int start,int count,auto& self)->void{
            // Prune: If the current count plus remaining characters can't exceed
            // No need to go further
            if(count+(n-start)<=ans) return;

            if(start==n) ans=std::max(ans,count);

            for(int end=start+1;end<=n;++end){
               std::string sub=s.substr(start,end-start);
               if(!list[sub]){
                  list[sub]=true;
                  self(end,count+1,self);
                  list[sub]=false;
               }
            }
         };

         solve(0,0,solve);
         return ans;
      }
};
```