Design and Analysis of Algorithms

Assignment No. 5 *C2*

Group members:

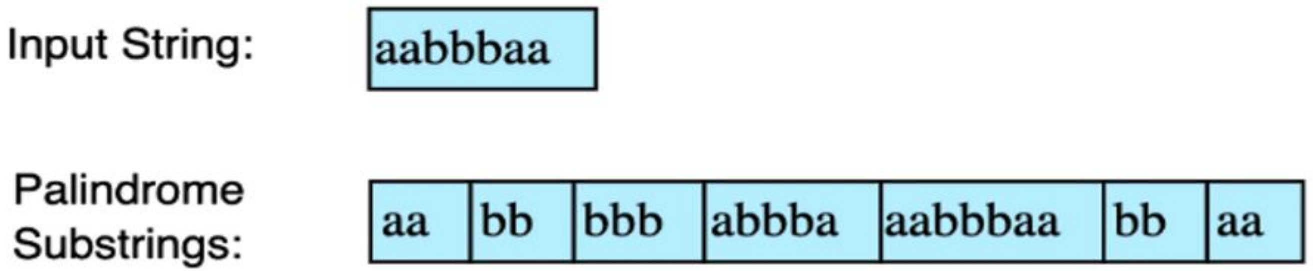
Harsh Meena	-IIT2019005
Asha Jyothi Donga	-IIT2019006
Sampada Kathar	-IIT2019007

Contents

Finding the number of non-empty substrings that are palindromes from a given string

Problem:

Given a string S, count the number of non-empty sub strings that are palindromes. A sub string is any continuous sequence of characters in the string. A string is said to be palindrome, if the reverse of the string is same as itself. Two sub strings are different if they occur at different positions in S. Solve using Dynamic programming



Input String: aabbbaa

Palindrome Substrings:						
aa	bb	bbb	abbba	aabbbaa	bb	aa

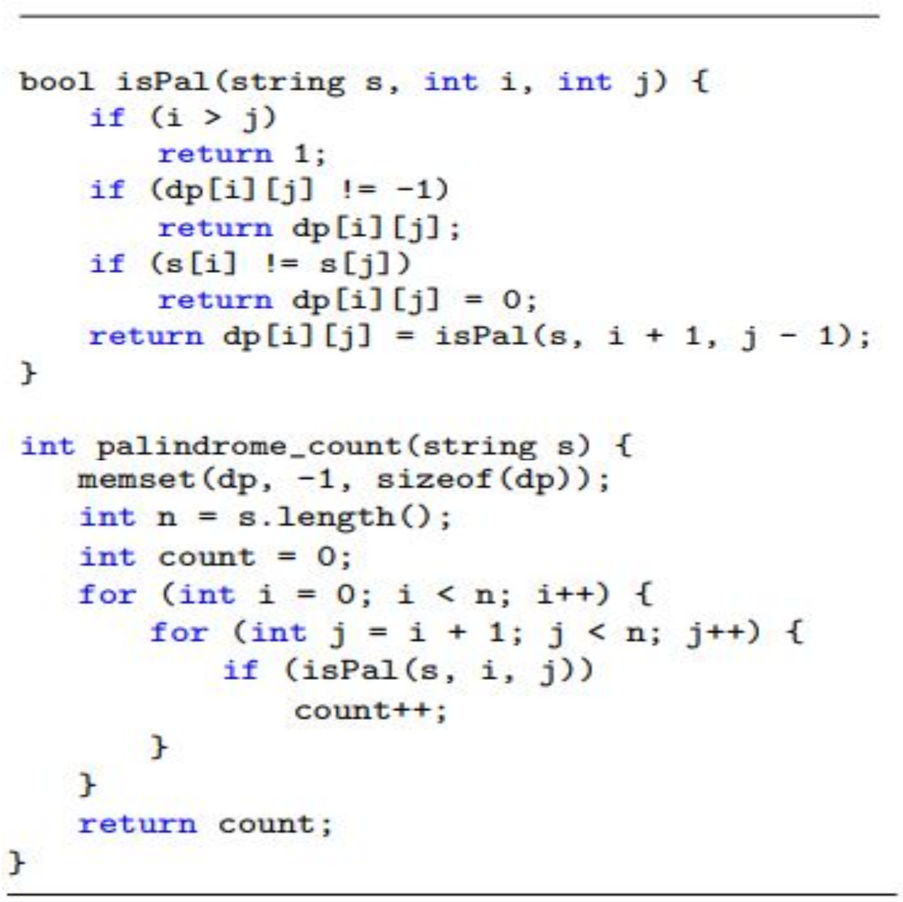
[4]

Algorithm

1. Our base condition is i > j. Initial values are i = 0, and j = n − 1 where n is the length of the substring. Return 1 if this condition is met.
2. Check whether the characters are equal for all i and j, if not then return 0.
3. Increment i and decrement j and repeat the process. Do this for all values of i and j

This approach uses Top Down DP i.e memoized version of recursion.

Pseudo Code :



```
bool isPal(string s, int i, int j) {
    if (i > j)
        return 1;
    if (dp[i][j] != -1)
        return dp[i][j];
    if (s[i] != s[j])
        return dp[i][j] = 0;
    return dp[i][j] = isPal(s, i + 1, j - 1);
}

int palindrome_count(string s) {
   memset(dp, -1, sizeof(dp));
   int n = s.length();
   int count = 0;
   for (int i = 0; i < n; i++) {
       for (int j = i + 1; j < n; j++) {
           if (isPal(s, i, j))
               count++;
       }
   }
   return count;
}
```

Time Complexity and Space Complexity Analysis

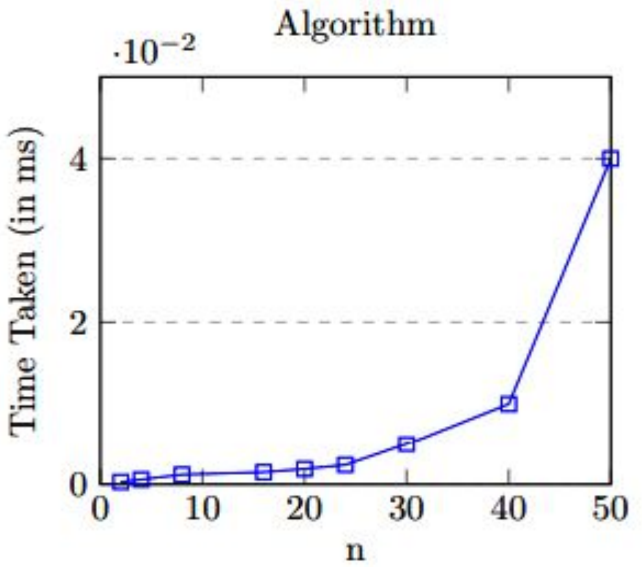
Time Complexity

Using this reccursive dynamic programming approach, the function is called a total of maximum n 2 +c (here c is constant) times to calculate each dp value.

- The worst case complexity of this algo will be O(n^2).
- The best case occurs when either n is equal to 1, i.e., when string is of length 1. Thus, the best case time complexity is Ω(1)

Space Complexity

This algorithm has a space complexity of O(n^2) for the dp table.



The above graph shows the variance of Time with respect to size of the second array, and follows our theoretically calculated TC of $O(n^2)$.

Conclusion

From this paper we can conclude that top down dynamic programming algorithms, which make use of memoization are quite efficient and should be used to solve recursion problems when there are repeating sub problems.

References:

[1]vaibhav29498, ’substring palindroms’, GeeksforGeeks, 2018.[Online] [Accessed: 1-Feb-2021]

[2]’Dynamic Programming’[Online][Accessed: 1-Feb-2021]

[3]GeeksforGeeks, ’reccursive dp — Set 1 (Introduction)’, GeeksforGeeks, 2018, GeeksforGeeks, 2018. [Online][Accessed: 1-Feb-2021]

[4]https://www.educative.io/m/find-all-palindrome-substrings