Dynamic Programming Lecture 2

Tuesday, 10 September 2024 6:08 AM

https://www.geeksforgeeks.org/problems/palindromic-patitioning4845/1

Palindrome checker:- (using dp)

```
function substringPalindromeCheck(s, n):
    P = matrix of size n x n
    for (i: 0 → n-1):                      } length 1 substrings
        P[i][i] = TRUE
    for (i: 0 → n-2):                      } length 2 substrings
        P[i][i+1] = (s[i] == s[i+1])
    for (length: 3 → n):                   } length 3 to n.
        for (i: 0 → n-length):
            j = i + length - 1
            P[i][j] = (s[i] == s[j] && P[i+1][j-1])
    return P
```

```
function minPalindromeCuts(s, n):
    P = substringPalindromeCheck(s, n)
    dp = array of size n

    for (i: 0 → n-1)
        if P[0][i]:
            dp[i] = 0
        else:
            dp[i] = ∞
            for (j: 0 → i-1)
                if P[j+1][i]:
                    dp[i] = min(dp[i], dp[j]+1)
    return dp[n-1]
```

```python
#User function Template for python3

class Solution:
    def palindromicPartition(self, s):
        n = len(s)
        p = [[False]*n for _ in range(n)] # n x n matrix
        for i in range(n): # Length 1 substrings
            p[i][i] = True
        for i in range(n-1): # Length 2 substrings
            p[i][i+1] = (s[i]==s[i+1])
        for l in range(3, n+1): # Length 3 to n
            for i in range(n-l+1):
                j = i+l-1
                p[i][j] = (s[i]==s[j] and p[i+1][j-1])

        dp = []
        for i in range(n):
            if p[0][i]:
                dp.append(0)
            else:
                ans = 5000
                for j in range(i):
                    if p[j+1][i]:
                        ans = min(ans, dp[j]+1)
                dp.append(ans)
        return dp[-1]


# } Driver Code Ends
```

Time $O(n^2)$, Space $O(n^2)$

Time $O(n^2)$, Space $O(n)$

Time :- $O(n^2)$

Space :- $O(n^2)$

https://www.hackerrank.com/challenges/sam-and-substrings/

n = 42

2, 4, 42 Sum = 2+4+42 = 48

n = 421

1, 2, 4, 42, 21, 421 Sum = 1+2+4+42+21+421 = 491

12 4 (2)

n = 4

ending at 4

(4)

Sum 4

not ending at 4

Sum 0

n = 42

ending at 2

(2) (42)

Sum (44)

int = (2)

not ending at 2

4

Sum (4)

$n = 421$

ending at 1

1, 21, 421

Sum = 443

int = (3)

not ending at 1

2, 42, 4

Sum = 48

$1 + 21 + 421$

$= 1 + (2 \times 10 + 1) + (42 \times 10 + 1)$

$= 1 + 1 + 1 + (2 + 42) \times 10$

$= 1 + 2 + 44 \times 10$

$= 443$

$n = 4217$

ending at 7

(7), 17, 217, 4217

4458

snb = 4

not ending at 7

1, 21, 421, 2, 42, 4

$443 + 48 = 491$

Sum (ending at 7)

$= 7 + 7 \times (3) + 443 \times 10$

$= 7 + 21 + 4430$

$= 4458$

State:- Sum including last character : si

substrings including last character : ni

Sum Not including last character : sni

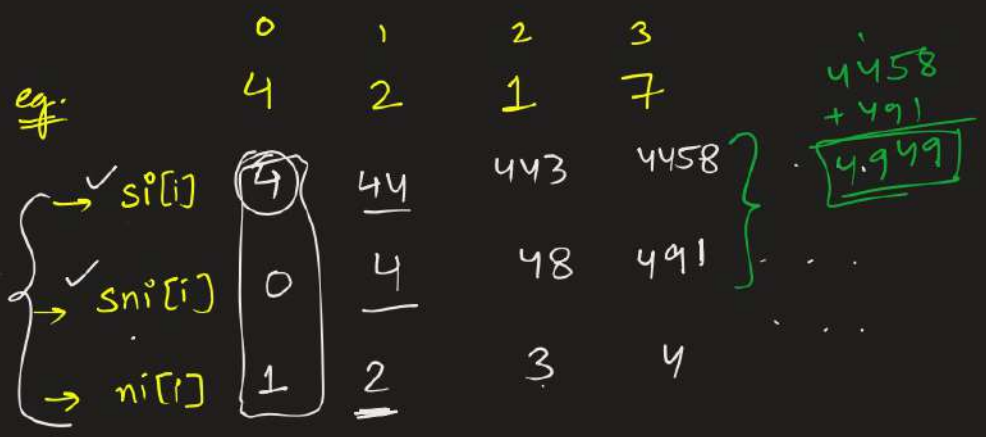
$$
\begin{cases}
si[i] = (s[i] - \text{asci}(0')) * (1 + ni[i-1]) + si[i-1] * 10 \\
sni[i] = si[i-1] + sni[i-1] \\
ni[i] = ni[i-1] + 1
\end{cases}
$$

$si[i] = 4 \times 10 + 1 \times 2 + 2$

$= 4 \times 10 + 2 \times (1+1)$

Substrings ending at i including i

1, 21, 421



1, 21, 421

not including i

... 4, 2, 42

41

$443 * 10 + 7 \times (4)$

$= 4430 + 28 = 4458$

$1 * (1+2) + 44 \times 10$

$= 440 + 3 = 443$

Base Case:- if i == 0

$$\begin{cases} si[i] = s[i] - \text{ascii}('0') \\ sni[i] = 0 \\ ni[i] = 1 \end{cases}$$

Py :

```python
def substrings(n):
    MOD = int(1e9+7)
    si = ord(n[0])-ord('0')
    sni = 0
    ni = 1
    for i in range(1, len(n)):
        sni = (sni + si) % MOD
        si = ((si*10 + (ord(n[i])-ord('0'))*(ni+1))) % MOD
        ni += 1
    return (sni + si) % MOD
```

$T = O(c)$

↓

chars in n

$S = O(1)$

C++ :

```cpp
int substrings(string n) {
    long long MOD = 1e9+7, si = n[0]-'0', sni=0, ni=1;
    for(int i=1; i<n.length(); i++) {
        sni = (sni + si) % MOD;
        si = (si*10 + (n[i]-'0')*(ni+1)) % MOD;
        ni++;
    }
    return (sni + si) % MOD;
}
```

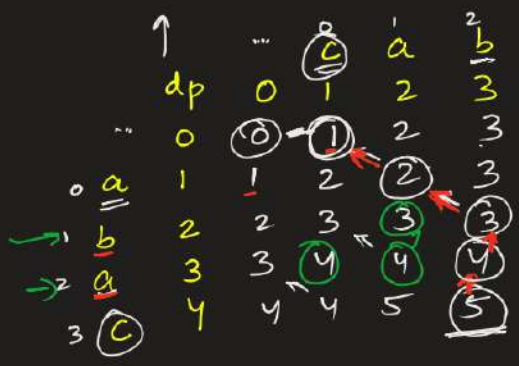
https://leetcode.com/problems/shortest-common-supersequence/

m = 4 n = 3

s1 = abac s2 = cab

s = cabac

dp[i][j] → length of SCS for s1[0...i-1] s2[0...j-1]



dp[i][j] = 0, i = 0 & j = 0

dp[i][j] = 0, i = 0 or j = 0

$$dp[i][j] = \begin{cases} dp[i-1][j-1] + 1, & s1[i-1] == s2[j-1] \\ 1 + \min(dp[i-1][j], dp[i][j-1]), & \text{otherwise} \end{cases}$$

```cpp
class Solution {
public:
    string shortestCommonSupersequence(string str1, string str2) {
        int m = str1.length(), n = str2.length();
        int dp[m+1][n+1];
        dp[0][0] = 0;
        for(int i=1; i<=m; i++) dp[i][0] = i;
        for(int i=1; i<=n; i++) dp[0][i] = i;
        for(int i=1; i<=m; i++) {
            for(int j=1; j<=n; j++) {
                if(str1[i-1] == str2[j-1])
                    dp[i][j] = 1+dp[i-1][j-1];
                else
                    dp[i][j] = 1+min(dp[i][j-1], dp[i-1][j]);
            }
        }
        cout << dp[m][n];
        int i=m, j=n;
        string ans = "";
        while(i!=0 && j!=0) {
            if(str1[i-1] == str2[j-1]) {
                ans += str1[i-1];
                i--;
                j--;
            }
            else if(dp[i-1][j] < dp[i][j-1]) {
                ans += str1[i-1];
                i--;
            }
            else {
                ans += str2[j-1];
                j--;
            }
        }
        while(i!=0) {
            ans += str1[i-1];
            i--;
        }
        while(j!=0) {
            ans += str2[j-1];
            j--;
        }
        reverse(ans.begin(), ans.end());
        return ans;
    }
};
```

TC = O(m * n)

SC = O(m * n)

↓

can be reduced to O(n)

https://leetcode.com/problems/wildcard-matching/

s: a b a c d b c e

p: a b ? * e match

s: a b a c d b c e

p: a b ? * ? c NOT a match ×

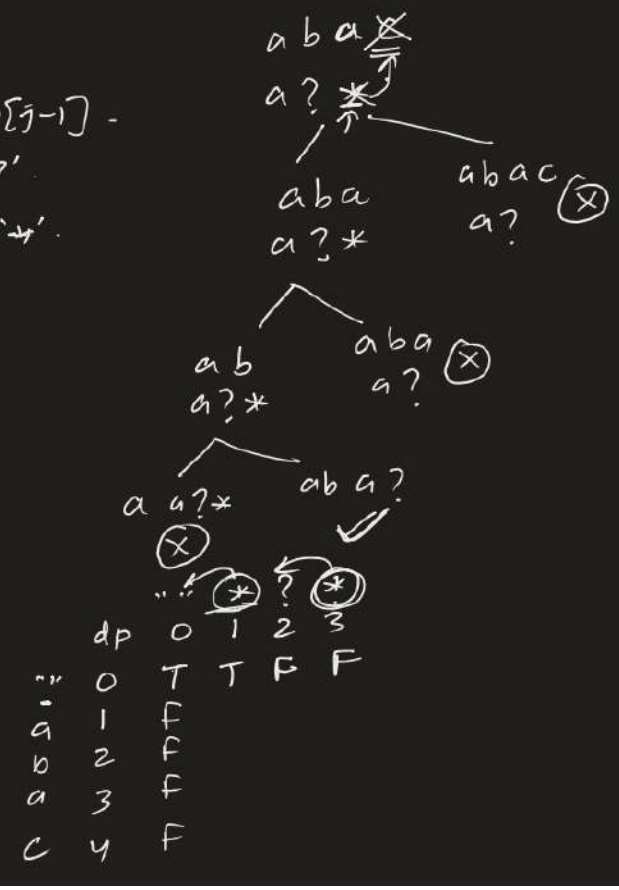
$$dp[i][j] = \begin{cases} dp[i-1][j-1], & s[i-1] == p[j-1] \\ dp[i-1][j-1], & p[j-1] = '?' \\ dp[i-1][j] \text{ or } dp[i][j-1], & p[j-1] = '*' \\ \text{False}, & \text{otherwise} \end{cases}$$

$s[0 \cdots i-1]$

$p[0 \cdots j-1]$

base cases

$$\begin{cases} dp[0][0] = \text{True} \\ dp[i][0] = \text{False}, & i > 0 \\ dp[0][i] = \begin{cases} \text{False}, & p[i-1] \neq '*' \\ dp[0][i-1], & p[i-1] = '*' \end{cases} \end{cases}$$



Ans:- dp[m][n]

```cpp
class Solution {
public:
    bool isMatch(string s, string p) {
        int m = s.length(), n = p.length();
        bool dp[m+1][n+1];
        dp[0][0] = true;
        for(int i=1; i<=m; i++) dp[i][0] = false;
        for(int i=1; i<=n; i++) {
            if(p[i-1] == '*') dp[0][i] = dp[0][i-1];
            else dp[0][i] = false;
        }
        for(int i=1; i<=m; i++) {
            for(int j=1; j<=n; j++) {
                if(s[i-1] == p[j-1] || p[j-1] == '?')
                    dp[i][j] = dp[i-1][j-1];
                else if(p[j-1] == '*')
                    dp[i][j] = dp[i-1][j] || dp[i][j-1];
                else
                    dp[i][j] = false;
            }
        }
        return dp[m][n];
    }
};
```

$T = O(m * n)$

$S = O(m * n)$

↳ reducible to $O(n)$