هوالعلیم

پاسخ تمرین چهارم الگوریتم

مریم سعیدمهر

9629373

22 فروردین 1398

پاسخ سوال یک :

```
# W is the length of the wood
# n is the number of parts
# val is the value of each part
# wt is the length of each part

def function(W, n, val, wt):

        # MAX[i] is going to store maximum
        # value with wood length i.
        MAX = [0 for i in range(W + 1)]

        # Fill MAX[] using above recursive formula
        for i in range(W + 1):
                for j in range(n):
                        if (wt[j] <= i):
                                MAX[i] = max(MAX[i], MAX[i - wt[j]] + val[j])

        return MAX[W]
```

در این جا چون تعداد قطعات به اندازه طول چوب است یعنی n = W پس واضح است که اوردر این تابع $O(n^2)$ است

پاسخ سوال دو :

```
# a is the array
# size is the array size

def function(a, size):

        max_so_far = 0
        max_ending_here = 0

        for i in range(size):
                max_ending_here = max_ending_here + a[i]
                if max_ending_here < 0:
                        max_ending_here = 0

                # Do not compare for all elements. Compare only
                # when max_ending_here > 0
                elif (max_so_far < max_ending_here):
                        max_so_far = max_ending_here

        return max_so_far
```

واضح است که اوردر این تابع $O(n)$ است

سوال سه :

در ترکیبیات دوره دبیرستان آموختیم که ((: تعداد مسیر خواسته شده در سوال برابر است با :

C(m+n , m) که برابر است با C(m+n , n)

برای یافتن این مقدار میتوان کدی نوشت که در O(n) انجام میشود!

```
def factor(s,n) :
        res = 1
        for i in range(s, n+1):
                res = res*i
        return res

def function(m , n):
        x = factor(2,min(m,n))
        y = factor(max(m,n)+1,m+n)
        return x/y
```

بدیهی است که اوردر این سوال O(n) است.

پاسخ سوال چهار :

```
# M is the matrix

def function(M):
        R = len(M) # no. of rows in M[][]
        C = len(M[0]) # no. of columns in M[][]

        S = [[0 for k in range(C)] for l in range(R)]
        # here we have set the first row and column of S[][]

        # Construct other entries
        for i in range(1, R):
                for j in range(1, C):
                        if (M[i][j] == 1):
                                S[i][j] = min(S[i][j-1], S[i-1][j],
                                                S[i-1][j-1]) + 1
                        else:
                                S[i][j] = 0

        # Find the maximum entry and
        # indices of maximum entry in S[][]
        max_of_s = S[0][0]
        max_i = 0
        max_j = 0
        for i in range(R):
                for j in range(C):
                        if (max_of_s < S[i][j]):
                                max_of_s = S[i][j]
                                max_i = i
                                max_j = j

        return max_of_s
```

واضح است که اوردر این تابع O(R*C) است (R : Row و C : Column)

پاسخ سوال پنج :

```
def function(st) :
        # table[i][j] will be false if substring
        # str[i..j] is not palindrome. Else
        # table[i][j] will be true
        n = len(st)
        table = [[0 for x in range(n)] for y in range(n)]
        maxLength = 1
        for i in range(n):
                table[i][i] = True
        # check for sub-string of length 2.
        start = 0
        for  i  in range(n-1):
                if (st[i] == st[i + 1]) :
                        table[i][i + 1] = True
                        start = i
                        maxLength = 2
        # Check for lengths greater than 2.
        # k is length of substring
        for k in range(3, n + 1):
                # Fix the starting index
                for i  in range(n – k + 1):
                        # Get the ending index of
                        # substring from starting
                        # index i and length k
                        j = i + k - 1
                        # checking for sub-string from
                        # ith index to jth index iff
                        # st[i+1] to st[(j-1)] is a
                        # palindrome
                        if (table[i + 1][j - 1] and st[i] == st[j]) :
                                table[i][j] = True
                                if (k > maxLength) :
                                        start = i
                                        maxLength = k

        return maxLength
```

واضح است که اوردر این تابع همان مطلوب سوال یعنی $O(n^2)$ است

* سوالات امتیازی * :)

* سوال اول *

```
# size is the size of the array

def function(size, array):
    stack = []
    max_area = 0
    i = 0
    while i < size:
        if (len(stack) == 0) or (array[stack[len(stack)-1]] <= array[i]):
            stack.append(i)
            i += 1

        else:
            top_val = array[stack.pop()]
            area = top_val * i

            if len(stack):
                area = top_val * (i - stack[len(stack)-1] - 1)
            max_area = max(area, max_area)

    while len(stack):
        top_val = array[stack.pop()]
        area = top_val * i
        if len(stack):
            area = top_val * (i - stack[len(stack)-1] - 1)
        max_area = max(area, max_area)

    return max_area
```

بدیهی است که اوردر این سوال $O(n)$ است.

* سوال دوم *

```
def max_Rectangle(R, C, array):
    result = function(C, array[0])  # “function” is the one implemented in page  7
    for i in range(1, R):
        for j in range(C):
            if array[i][j] == 1:
                array[i][j] += array[i - 1][j]

        result = max(result, function(C, array[i]))

    return result
```

واضح است که اوردر این سوال O(R*C) است (R : Row و C : Column)