T.C.

ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ALGORİTMA ANALİZİ DERSİ

FİNAL ÖDEVİ - 1

Soru 1:

Parametre olarak gelen m, n değerleri ile m x n boyutunda, içi rastgele belirlenen harflerle dolu olan bir matris oluşturan ve onu geri döndüren fonksiyonu oluşturun ve karmaşıklığını hesaplayın.

words = ['q','w','e','r','t','y','u','o','p','a','s','d','f','g','h','j','k','l','i','z','x','c','v','b','n','m']

g\_matrix = [] #global matris

def new\_matrix(m,n): ###3n+4n^2

global g\_matrix #1

n\_matrix = [] #2

for i in range(m): #n\*(2+4\*n) = 2n+4n^2

n\_matrix.append([])

for j in range(n):

loc = random.randint(0,len(words)-1)

n\_matrix[i].append(words[loc])

g\_matrix = n\_matrix #1

print\_matrix(g\_matrix) #n

def print\_matrix(m):

for i in m: #n

print(i)

kodun karmaşıklığı O(3n+4n^2) = O(n+n^2) = O(n^2)

Soru 2:

Parametre olarak satır, sütun, yön ve kelime değerlerini alıp, 1. soruda oluşturulan matrise, satır, sütun değerlerinin belirttiği konumdan itibaren, parametre olarak gelen yön değerinin belirttiği yönde kelimenin yazılmasını sağlayan fonksiyon yazın ve karmaşıklığını hesaplayın.

#0: soldan saga, 1: sagdan sola, 2: asagidan yukariya, 3: yukaridan asagiya

def add\_word(i,j,orient,word):

global g\_matrix #1

if(len(g\_matrix) > i and len(g\_matrix[0]) > j): #4

#kodlar burada

if(orient == 0):#1

#soldan saga

while(word != ""): #n

c = word[0] #2

word = word[1:] #2

if (j == len(g\_matrix[0])): #3

j = 0 #1

i += 1 #1

if (i == len(g\_matrix)): #3

i = 0 #1

g\_matrix[i][j] = c #3

j += 1 #2

elif(orient == 1):

#sagdan sola

...

elif(orient == 2):

#asagidan yukariya

...

elif(orient == 3):

#yukaridan asagiya

...

else:

print("orientation out of range")

else:

print("out of range")

return

print\_matrix(g\_matrix)

(Yukarıdaki kodda ilk if benzer işlemleri yapan fonksiyonlar bulunduğu için ve karmaşıklığa etkileri, ilk if ile aynı olduğu için ayrıntıları eklenmemiştir, fakat kod kısmında hepsi mevcuttur.)

3. satırda bulunan if(len(g\_matrix) > i and len(g\_matrix[0]) > j) sorgusu ve altında yapılan işlemlerin karmaşıklığı: O(n)’dir.

Son satırda bulunan print\_matrix(g\_matrix) çağrısının karmaşıklığı: O(n)’dir.

Toplam karmaşıklık O(2n) = O(n)’dir.

Soru 3:

Elemanları harf olan bir matrisin uzunluğu en az 10 olan kelimelerden oluşan palindrom kelimeleri bulan fonksiyonu yazınız ve karmaşıklığını hesaplayınız.

def myReverse(v): # O(n)

v2=[]

for i in v[::-1]:

v2.append(i)

return v2

def palindrom\_list():

global g\_matrix

if(len(g\_matrix[0]) < 10): # O(1)

return []

elif(len(g\_matrix[0]) == 10): # O(n)

pol\_list = []

for i in range(len(g\_matrix)):

if(g\_matrix[i] == myReverse(g\_matrix[i])):

if(g\_matrix[i] not in pol\_list):

pol\_list.append(g\_matrix[i])

return pol\_list

else: # O(n^3)

pol\_list = []

a = g\_matrix[0]

count = len(a)-9

count = count\*(count+1)/2

count = int(count)

for i in range(len(g\_matrix)): # O((n^3+n^2)/2) = O(n^3)

l,k = 10,0

for j in range(count): # O((n^2+n)/2) = O(n^2)

v = g\_matrix[i][k:k+l]

if(v == myReverse(v)): # O(n)

if(v not in pol\_list):

pol\_list.append(v)

l+=1

if(k+l == len(g\_matrix[0])+1):

l = 10

k+=1

return pol\_list

Bu cevapta 3 farklı karmaşıklık vardır, bu da matristen matrise değişmektedir.

Eğer matrisin satırlarının uzunluğu, 10’dan küçük ise o zaman O(1)’dir.

Eğer matrisin satırlarının uzunluğu, 10’a eşit ise o zaman O(n)’dir.

Eğer matrisin satırlarının uzunluğu, 10’dan büyük ise o zaman O(n^3)’dür.

Soru 4:

1, 2 ve 3. sorularda yazılan fonksiyonları test eden bir fonksiyon yazınız ve karmaşıklığını hesaplayınız.

def tester():

#palindrom test

print(new\_matrix(15,15)

print(add\_word(0,0,0,"tenettenettenet"))

print\_matrix(palindrom\_list())

Tester fonksiyonunun karmaşıklığı, içinde çağrılan fonksiyonların karmaşıklığına göre belirlenmektedir. Buna göre Tester fonksiyonunun karmaşıklığı: O(n^3)’dür. Bunun nedeni ise, testi yapılan matrisin satırlarının boyutu 15’dir, yani 10’dan büyüktür.