T.C.

ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ALGORİTMA ANALİZİ DERSİ

FİNAL ÖDEVİ – 2

Soru 1:

Parametre olarak n değerini alıp, n boyutlu, değerleri int olan bir vektör oluşturan fonksiyon yazınız ve karmaşıklığını hesaplayınız.

def cr\_vector(n):

vec = []

for i in range(n): #O(n)

vec.append(random.randint(0,10))

return vec

Bu fonksiyonun karmaşıklığı: O(n)’dir.

Soru 2:

İki vektörü parametre olarak alan ve bunları skaler olarak çarpan fonksiyonu yazın ve karmaşıklığını hesaplayın.

def sc\_mul\_vectors(v1,v2):

\_len = len(v1)

if(\_len > len(v2)):

\_len = len(v2)

mul = 0

for i in range(\_len): #O(n)

mul += v1[i]\*v2[i]

return mul

Bu fonksiyonun karmaşıklığı: O(n)’dir.

Soru 3:

Parametre olarak m, n alan ve m x n boyutunda bir matris döndüren fonksiyonu yazın ve karmaşıklığını hesaplayın.

def cr\_matrix(m,n):

mat = []

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

mat.append(cr\_vector(n)) #O(n)

print\_matrix(mat)

return mat

Bu fonksiyonun karmaşıklığı: O(n^2)’dir.

Soru 4:

Parametre olarak gelen iki matrisi çarpan ve sonucu geri döndüren fonksiyonu yazın ve karmaşıklığını hesaplayın.

def mul\_matrix(m1,m2):# O(n^3) > O(n^2) === O(n^3)

if(len(m1[0]) == len(m2)):

rev\_m2 = rev\_matrix(m2) # O(n^2)

mul\_mat = []

for i in range(len(m1)): #O(n^3)

mul\_mat.append([])

for j in range(len(rev\_m2)): #O(n^2)

mul\_mat[i].append(sc\_mul\_vectors(m1[i],rev\_m2[j])) #O(n)

return mul\_mat

else:

return None

def rev\_matrix(m): #O(n^2)

rev = []

for i in range(len(m[0])):

rev.append([])

for j in range(len(m)):

rev[i].append(m[j][i])

return rev

rev\_matrix fonksiyonunun karmaşıklığı O(n^2)’dir, mul\_matrix fonksiyonunun karmaşıklığı O(n^3)’dür, fakat eğer iki matris çarpma işlemi için uygun değil ise karmaşıklık O(1)’dir.

Soru 5:

Daha önce yazılan fonksiyonları kullanarak, 5 adet matrisi çarpan fonksiyonu yazınız ve karmaşıklığını hesaplayınız.

def mul\_matrix\_list(m\_list):

res = m\_list[0]

for i in range(1,len(m\_list)): #O(n^4) veya O(n)

res = mul\_matrix(res,m\_list[i])# O(n^3) veya O(1)

if(res == None):

return None

return res

(Soruda 5 adet matris dendiği için hesaplamalar 5 adet matris varmış gibi yapılmaktadır.)

Çarpılan matrislerin birbirleri ile çarpılmaları için uygun olmaları durumunda karmaşıklık O(n^4), fakat karmaşıklık olmadığı durumlarda ise değişken durumlar sergilenmektedir.

Soru 6:

Soru 5’e verilen cevabın değişken olduğunu tartışınız ve bunun nasıl mümkün olduğunu anlatınız.

Karmaşıklığın değişken olmasının nedeni, verilen matrislerin çarpma işlemine uygun olup olmamasına bağlı olarak değişmektedir. Çünkü ilk çarpım uygun değil ise karmaşıklık O(1) olur, ikinci çarpım uygun değil ise karmaşıklık O(n^3) olur, fakat diğer çarpımlarda sorun çıkması veya çıkmaması durumunda ise karmaşıklık O(n^4) olur

Soru 7:

Yukarıda yazılan fonksiyonları test eden bir fonksiyon yazın ve karmaşıklığını bulun.

def tester():

## test case 1 -> fail

m\_list = []

for i in range(5):

m\_list.append(cr\_matrix(3,6))

res = mul\_matrix\_list(m\_list)

if(res != None):

print\_matrix(res)

else:

print("\*\*\*\*\*")

print("carpilamaz")

## test case 2 -> succeed

print("-----")

m\_list = []

for i in range(5):

m\_list.append(cr\_matrix(3,3))

res = mul\_matrix\_list(m\_list)

if(res != None):

print\_matrix(res)

else:

print("\*\*\*\*\*")

print("carpilamaz")

Yukarıdaki fonksiyonda test case 1 için karmaşıklık O(1)’dir, fakat test case 2 için karmaşıklık O(n^4)’dür. Buna göre tester fonksiyonunun karmaşıklığı O(n^4) olur.