Sivas Cumhuriyet Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü
2024-2025 Eğitim Öğretim Yılı Bahar Algoritma Analizi ve Tasarımı Course AAD321 Vize Sınavı 1

1- Algoritma Tasarım ve Analiz sürecinin aşamaları nelerdir?

1- Problemin Anlaşılması

2- Sahip olunan / mevcut olan donanımı anlamak

3- Probleme yaklaşık ya da tam çözüm geliştirmek

4- dizaynlardan / stratejilerden biri seçilir

5- Algoritma doğruluğunun test edilmesi

6- Algoritma Analizi

2- Bir dizide toplamı belirli bir S değerine eşit olan iki eleman olup olmadığını kontrol eden algoritmanın **C*worst*(*n*)** yani en kötü durum zaman karmaşıklığı nedir?

```
toplamVarMi(A[0,…,n-1], S)
   // Girdi: n elemanlı A dizisi ve S tamsayısı
   // Çıktı: Eğer A içinde toplamı S olan iki eleman varsa true, yoksa false

   for i = 0'dan n-2'ye
      for j = i+1'den n-1'e
         if A[i] + A[j] == S
            return true
   return false
```

Algoritmanın toplam çalışma süresini analiz edelim:

$$\sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$$

İç toplamı açarsak:

$$\sum_{i=0}^{n-2} (n-1-i)$$

Bu bir **aritmetik seri** olup toplamı şu şekilde hesaplanır:

$$(n-1) + (n-2) + \ldots + 1 = \frac{(n-1) \cdot n}{2}$$

$$C_{\text{worst}}(n) = O(n^2)$$

i=0 için j=1, 2, …. , n-1 değerini alır. Toplamda n-1 adet kıyaslama olur.
i=1 için j=2, …. , n-1 değerini alır. Toplamda n-2 adet kıyaslama olur.
i=2 için j=3, 4, …. , n-1 değerini alır. Toplamda n-3 adet kıyaslama olur.
.
.
i=n-2 için j=n-1 değerini alır. 1 adet kıyaslama yapılır.
Tüm bu kıyaslama sayılarını toplarsak:

$$C_{worst}(n) = (n-1) + (n-2) + \cdots + 1 = \frac{(n-1).n}{2}$$

3- Aşağıdaki fonksiyonların zaman karmaşıklıklarını Big O notasyonu ile ifade edin.

$T(n)$	Big O
$T(n) = 4n^4 + 3n + \log n$	$O(n^4)$
$T(n) = 2^{n+1} + n^3 + 5$	$O(2^n)$
$T(n) = 5n + \sqrt{n} + \log^3 n$	$O(n)$
$T(n) = n^3 + 2n \log n + 100$	$O(n^3)$

4- Bir A dizisi verildiğinde, dizideki herhangi iki farklı eleman arasındaki maksimum mesafeyi bulan bir algoritma yazınız.

(Not: Sadece pseudo kod yazmak yeterlidir.)

```
ALGORITHM MaxDistance(A[0..n − 1])
Input: Array A[0..n−1] of numbers
Output: Maximum distance between two of its elements
dmax ← −∞
for i ← 0 to n − 1 do
    for j ← 0 to n − 1 do
        if i ≠ j and |A[i] − A[j]| > dmax then
            dmax ← |A[i] − A[j]|
return dmax
```

5- Aşağıda verilen algoritma iki sayının ebob'ün ve ekok'ün pseudo kodları bulmaktadır.

Ancak bazı kısımlar eksiktir. Eksik yerleri tamamlayınız.

a)

ALGORITHM gcd(m, n)

// Input: Two nonnegative, not-both-zero integers m and n

// Output: Greatest common divisor of m and n

while n ≠ 0 do

    r ← m mod n

    m ← n

    n ← r

return m

b) Ekok (En Küçük Ortak Kat) hesaplamak için EBOB (En Büyük Ortak Bölen) algoritmasını kullanabiliriz. İki sayının EKOK'u şu formülle bulunur:

$$\text{EKOK}(a, b) = \frac{|a \cdot b|}{\text{EBOB}(a, b)}$$

ALGORITHM lcm(a, b)

// Input: Two positive integers a and b

// Output: Least common multiple (EKOK) of a and b

ebob ← gcd(a, b)

ekok ← (a * b) / ebob

return ekok

Her soru eşit puanlıdır. Süre 55dk Başarılar dilerim...

Eng: Abdulrahman Hamdi