Sivas Cumhuriyet Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü 2024-2025 Eğitim Öğretim Yılı Bahar Algoritma Analizi ve Tasarımı Course AAD321 Quiz 3

1. Bir fonksiyonun asimptotu neyi ifade eder?

A) Fonksiyonun başlangıç değerini.

B) Fonksiyonun sonlu bir noktadaki değerini

C) Fonksiyonun sonsuz büyüklükteki değerlerde yaklaştığı doğruyu veya eğriyi.

D) Fonksiyonun türevini.

2. Big Oh (O) notasyonu aşağıdakilerden hangisini ifade eder?

A) Fonksiyonun en iyi durumdaki sınırını.

B) Fonksiyonun en kötü durumdaki üst sınırını.

C) Fonksiyonun tam büyüme oranını.

D) Fonksiyonun alt sınırını.

3. Bir algoritmanın temel hedeflerinden biri aşağıdakilerden hangisidir?

A) Sonsuz döngüye girerek sürekli çalışmak.

B) Belirli girdilerle, sonlu bir zaman içinde istenilen çıktıyı elde etmek.

C) Rastgele sonuçlar üretmek.

D) Girdileri değiştirmeden olduğu gibi bırakmak.

4. Rekürsif algoritmalarda, her çağrıda sabit sayıda işlem yapılırsa algoritmanın çalışma süresi neye bağlıdır?

A) Girdi büyüklüğüne.

B) Rekürsif çağrı sayısına.

C) Temel işlemin süresine.

D) Döngü değişkeninin değerine.

5. İkiliSistem algoritmasında $n \leftarrow \lfloor n/2 \rfloor$ işlemi neden yapılır?

A) Sayının basamak sayısını artırmak için.

B) Sayıyı ikilik sisteme çevirmek için.

C) Sayıyı küçültüp döngünün sonlanmasını sağlamak için.

D) Sayının negatif olup olmadığını kontrol etmek için.

6. İkiliSistem algoritmasında, döngü en fazla kaç kez çalışır?

A) n kere.

B) $\log_2(n)$ kere.

C) n^2 kere.

D) 2n kere.

7. Fibonacci algoritmasında, F(0) ve F(1) neden tanımlanır?

A) Döngünün sonlanması için.

B) Rekürsif çağrının durması için taban durum olarak.

C) Daha hızlı sonuç almak için.

D) Algoritmanın daha kısa sürmesi için.

8. Temel işlem gerçekleştirme sayısının formülü toplam formüle dönüştürülmeden önce hangi faktör göz önünde bulundurulmalıdır?

A) Girdi büyüklüğü.

B) Döngü sayısı.

C) En iyi, en kötü ve ortalama durum analizleri.

D) Zaman verimliliği.

9. Bir algoritmanın zaman verimliliğini analiz ederken, büyüme derecesi nasıl gösterilir?

A) Toplam işlem süresi ile.

B) Döngü içindeki işlem sayısı ile.

C) Toplam formülün büyüme derecesine göre.

D) Rekürsif çağrı sayısına göre.

10. Limit yöntemi kullanılarak fonksiyonların büyüme oranları nasıl karşılaştırılır?

A) Sadece fonksiyonların başlangıç değerlerine bakılarak..

B) Fonksiyonların en büyük değerlerini karşılaştırarak.

C) Fonksiyonların sonsuza giderken büyüme oranlarına bakılarak.

D) Fonksiyonların sabit katsayılarıyla.

11. Aşağıdaki algoritma, verilen bir n tam sayısının ikilik (binary) sistemdeki toplam basamak sayısını hesaplar. Ancak bazı kısımlar eksiktir. Boşlukları uygun ifadelerle doldurunuz.

```
// Input: n uzunluğunda küçükten büyüğe
sıralı bir Adizisi ve bir K değeri
// Output: K’nın olduğu A’nın indisi
alt ← ___(1)___
üst ← ___(2)___
while alt ___(3)___ üst
orta ← ⌊ (alt + üst) / ___(4)___ ⌋
if A[orta] = ___(5)___
return ___(6)___
else if A[orta] ___(7)___ K
üst ← ___(8)___
else
alt ← ___(9)___
return ___(10)___
```

çözüm:

```
binarySearch(A[0, ……. ,n-1], K)
// Girdi: n uzunluğunda küçükten büyüğe
sıralı bir A dizisi ve bir K değeri
// Çıktı: K’nın olduğu A’nın indisi
veya False (bulunamazsa)

alt ← 0
üst ← n - 1
while alt ≤ üst do
    orta ← ⌊(alt + üst) / 2⌋
    if A[orta] = K then
        return orta
    else if A[orta] > K then
        üst ← orta - 1
    else
        alt ← orta + 1

return False
```

12. Verilen bir sayının asal olup olmadığını özyineleme (rekürsif) kullanarak belirleyen bir algoritma yazın.

Cevap:

```
ALGORITHM isPrime(n, i)
//Input:n: A positive integer.
//Output:True if n is a prime number
and False if n is not a prime number.
    if n ≤ 1 then
        return False
    if i ≥ sqrt(n) then
        return True
    if n mod i = 0 then
        return False
    return isPrime(n, i + 1)
```

13. Teoremi ispatlayınız:

Teorem:

$t(n) \in O(g(n)) \Leftrightarrow g(n) \in \Omega(t(n))$Yani, bir fonksiyon g(n), fonksiyon t(n) için bir üst sınırsa, o fonksiyon t(n) de fonksiyon g(n) için bir alt sınırdır.

Kanıt: $\Rightarrow$) $t(n) \in O(g(n))$ olsun. Bu durumda bir $n0>0$ ve bir $c>0$ için ve her $n \geq n0$ için $t(n) \leq c.g(n)$ olur

Her iki taraf pozitif c’ye bölünürse $g(n) \geq 1/c\ .t(n)$ olur.

Bu da $g(n) \in \Omega(t(n))$ demektir.

$\Leftarrow$$g(n) \in \Omega(t(n))$ olsun. Bu durumda bir $n0>0$’dan büyük her n ve bir $c>0$ için

$g(n) \geq c.t(n)$ olur. Her iki taraf c’ye bölünürse $1/c.\ g(n) \geq t(n)$ olur. (yani $t(n) \leq 1/c\ .\ g(n)$) Bu durumda $t(n) \in O(g(n))$ olur..

14. Büyüme derece soruları:

A) n! ve (n+1)! fonksiyonlarının büyüme derecelerini karşılaştırınız

B) 2^n ve n^3'ün fonksiyonlarının büyüme derecelerini karşılaştırınız

C) $\log_2(n)$ ve $\sqrt{n}$ fonksiyonlarının büyüme derecelerini karşılaştırınız

A) Biliyoruz ki: (n+1)!=(n+1)·n!

Bu ifade,n! fonksiyonuna göre sabit bir katsayıyla çarpıldığı için, büyüme derecesi açısından büyük-O notasyonunda şu sonucu elde ederiz:

(n+1)!=Θ(n!)

Bu,(n+1)! fonksiyonlarının asemptotik olarak aynı büyüklükte olduğunu gösterir.

Ama cevap böyle değil!!!

Cevap böyle olacak:

$$\lim_{n\to\infty} \frac{n!}{(n+1)!} = \lim_{n\to\infty} \frac{n!}{(n+1)\cdot n!}$$

Bu ifadeyi sadeleştirirsek:

$$\lim_{n\to\infty} \frac{1}{n+1}$$

Buradan açıkça görüyoruz ki, n sonsuza giderken

$\frac{1}{n+1}$ sıfıra yaklaşır yani: $\lim_{n\to\infty} \frac{n!}{(n+1)!} = 0$

n! büyüme derecesi daha küçüktür

O halde (n+1)! fonksiyonu, n! fonksiyonundan daha büyük diyebiliriz

---

B) 2^n Daha hızlı büyümektedir çünkü:

$$\lim_{n\to\infty} \frac{2^n}{n^3} = \lim_{n\to\infty} \frac{2^n \ln 2}{3n^2} = \lim_{n\to\infty} \frac{2^n (\ln 2)^2}{6n} = \lim_{n\to\infty} \frac{2^n (\ln 2)^3}{6} = \infty$$

$$n^3 \in O(2^n), \quad 2^n \in \Omega(n^3)$$

C)

$$\lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}} = \lim_{n\to\infty} \frac{(\log_2 n)'}{(\sqrt{n})'} = \lim_{n\to\infty} \frac{(\log_2 e)\frac{1}{n}}{\frac{1}{2\sqrt{n}}} = 2\log_2 e \lim_{n\to\infty} \frac{1}{\sqrt{n}} = 0$$

$\log_2(n)$ büyüme derecesi daha küçüktür

O halde √n fonksiyonu, $\log_2(n)$ fonksiyonundan daha büyük diyebiliriz.

15. Verilen bir sayının mükemmel sayı olup olmadığını kontrol eden bir algoritma yazın. Mükemmel sayı, kendisi hariç bölenlerinin toplamı kendisine eşit olan sayıdır.

```
ALGORITHM isPerfect(n)
//Input: A positive integer n
//Output: True if n is a perfect
number, otherwise False.
    sum ← 0
    for i ← 1 to n/2 do
        if n mod i = 0 then
            sum ← sum + i
    return (sum = n)
```

Süre 55dk Başarılar dilerim...

Eng: Abdulrahman Hamdi