Özyineleme ve Kuyruk Özyinelemesi Nedir?

Çalışma Kağıdı

Özyineleme, bir problemi daha küçük alt problemlere ayırarak çözen bir tekniktir. Kuyruk özyinelemesi, özyinelemeli çağrının fonksiyonun son işlemi olduğu ve bu sayede yığın taşması riskinin azaldığı bir özyineleme biçimidir.

Sorular

1. Aşağıdakilerden hangisi özyinelemeli bir fonksiyonun temel bileşenlerinden biri DEĞİLDİR?
A) Taban durumu
B) Özyinelemeli adım
C) Döngüsel kontrol
D) Fonksiyonun kendi kendisini çağırması

2. Kuyruk özyinelemesinde, özyinelemeli çağrı hangi konumda olmalıdır?
A) Fonksiyonun başında
B) Fonksiyonun ortasında
C) Fonksiyonun sonunda (kuyruk pozisyonunda)
D) Herhangi bir konumda olabilir

3. Özyineleme kullanımının potansiyel bir riski nedir?
A) Bellek sızıntısı
B) Yığın taşması (Stack Overflow)
C) Aşırı CPU kullanımı
D) Veri kaybı

4. Faktöriyel hesaplamasında özyineleme nasıl kullanılır?

5. Fibonacci serisi hesaplamasında özyineleme nasıl gösterilir?

6. Kuyruk özyinelemesi ile faktöriyel hesaplama örneği nedir?

7. Tanımla: Özyineleme nedir?

8. Tanımla: Kuyruk özyinelemesi nedir?

9. Tanımla: Özyinelemenin avantajları nelerdir?

Cevap Anahtarı

1. C) Döngüsel kontrol — Özyinelemeli fonksiyonlarda taban durumu ve özyinelemeli adım bulunur. Döngüsel kontrol yerine fonksiyonun kendi kendini çağırması esastır.

2. C) Fonksiyonun sonunda (kuyruk pozisyonunda) — Kuyruk özyinelemesinde, özyinelemeli çağrı fonksiyonun gerçekleştirdiği son işlem olmalıdır.

3. B) Yığın taşması (Stack Overflow) — Özyinelemeli çağrılar yığına eklenir ve çok derine inerse yığın taşması meydana gelebilir.

4. 1. Taban durumunu tanımla: faktöriyel(0) = 1. 2. Özyinelemeli adımı tanımla: faktöriyel(n) = n * faktöriyel(n-1). 3. Fonksiyonu çağır: faktöriyel(5) hesaplanır.

5. 1. Taban durumları: fibonacci(0) = 0, fibonacci(1) = 1. 2. Özyinelemeli adım: fibonacci(n) = fibonacci(n-1) + fibonacci(n-2). 3. Fonksiyon çağrısı: fibonacci(6) hesaplanır.

6. 1. Yardımcı fonksiyon tanımla: factorial_tail(n, accumulator). 2. Taban durumu: n == 0 ise accumulator'ı döndür. 3. Özyinelemeli adım: factorial_tail(n-1, n * accumulator). 4. Başlangıç çağrısı: factorial_tail(5, 1).

7. Bir fonksiyonun kendi kendisini çağırmasıdır.

8. Özyinelemeli çağrının fonksiyonun son işlemi olduğu ve kuyruk pozisyonunda bulunduğu özyineleme türüdür.

9. Kodun okunabilirliğini artırır, karmaşık problemleri daha basit hale getirir.