Memoization Nedir?

Çalışma Kağıdı

Memoization, fonksiyon çağrılarının sonuçlarını bir veri yapısında (genellikle bir harita veya dizi) saklayarak, aynı girdilerle tekrar çağrıldığında hesaplama yapmak yerine önbellekteki sonucu döndüren bir tekniktir.

Sorular

1. Aşağıdakilerden hangisi memoization'ın sağladığı ana faydadır?
A) Daha fazla bellek kullanımı
B) Daha az kod satırı
C) Daha hızlı çalışma süresi
D) Daha karmaşık hata ayıklama

2. Bir fonksiyonun sonucunu saklama işlemine ne ad verilir?
A) Önbelleğe Alma (Caching)
B) Memoization
C) Lazy Evaluation
D) JIT Derleme

3. Memoization genellikle hangi programlama paradigmasıyla ilişkilendirilir?
A) Nesne Yönelimli Programlama
B) Fonksiyonel Programlama ve Dinamik Programlama
C) Prosedürel Programlama
D) Olay Güdümlü Programlama

4. Fibonacci serisi hesaplamasında memoization kullanımı?

5. Dinamik programlama problemlerinde memoization nasıl uygulanır?

6. Tanımla: Memoization'ın temel amacı nedir?

7. Tanımla: Memoization'da hangi veri yapısı sıklıkla kullanılır?

8. Tanımla: Memoization hangi tür algoritmalarda etkilidir?

Cevap Anahtarı

1. C) Daha hızlı çalışma süresi — Memoization, tekrarlayan hesaplamaları önleyerek algoritmaların çalışma süresini önemli ölçüde azaltır.

2. B) Memoization — Fonksiyonun hesaplama sonuçlarını saklayarak tekrar kullanılmasını sağlayan teknik memoization'dır.

3. B) Fonksiyonel Programlama ve Dinamik Programlama — Memoization, özellikle özyinelemeli fonksiyonların verimli hale getirilmesinde ve dinamik programlama problemlerinin çözümünde yaygın olarak kullanılır.

4. 1. Bir harita (veya dizi) oluşturarak hesaplanmış Fibonacci değerlerini saklayın. 2. Fibonacci fonksiyonu çağrıldığında, önce değerin haritada olup olmadığını kontrol edin. 3. Eğer varsa, saklanan değeri döndürün. 4. Yoksa, değeri hesaplayın, haritaya kaydedin ve sonra döndürün.

5. 1. Problemin alt problemlerini tanımlayın. 2. Alt problemlerin çözümlerini saklamak için bir tablo (dizi veya harita) kullanın. 3. Bir alt problemi çözmeden önce, çözümünün tabloda olup olmadığını kontrol edin. 4. Eğer varsa, tablo değerini kullanın. 5. Yoksa, alt problemi çözün, sonucu tabloya kaydedin ve kullanın.

6. Tekrarlayan hesaplamaları önleyerek performansı artırmak.

7. Harita (Map) veya Dizi (Array).

8. Özyinelemeli fonksiyonlar ve dinamik programlama algoritmaları.