En Kısa Yol Algoritmaları Nedir? Dijkstra ve Bellman-Ford

Çalışma Kağıdı

En kısa yol algoritmaları, ağırlıklı bir grafikte, bir köşe (düğüm) ile diğer köşeler arasındaki en az maliyetli (en kısa) yolu hesaplar. Dijkstra tek yönlü negatif olmayan kenar ağırlıkları için, Bellman-Ford ise negatif kenar ağırlıklarını da içeren durumlar için kullanılır.

Sorular

1. Aşağıdaki algoritmalardan hangisi negatif kenar ağırlıklarını işleyebilir?

A) Dijkstra Algoritması
B) Bellman-Ford Algoritması
C) Prim Algoritması
D) Kruskal Algoritması

2. Dijkstra Algoritması'nın temel kısıtlaması nedir?

A) Sadece yönlü graflarda çalışır.
B) Negatif kenar ağırlıklarını işleyemez.
C) Çok büyük graflarda yavaştır.
D) Sadece tek bir hedef nokta için çalışır.

3. Bir ağ yönlendirme protokolünde en kısa yolu bulmak için hangi algoritma daha uygundur?

A) Dijkstra Algoritması
B) Bellman-Ford Algoritması
C) BFS (Genişlik Öncelikli Arama)
D) DFS (Derinlik Öncelikli Arama)

4. Dijkstra Algoritması ile Bir Şehirdeki En Kısa Rota Bulma

5. Bellman-Ford Algoritması ile Negatif Döngü Tespiti

6. Tanımla: En Kısa Yol Algoritması Nedir?

7. Tanımla: Dijkstra Algoritması Hangi Durumda Kullanılır?

8. Tanımla: Bellman-Ford Algoritması Hangi Durumda Kullanılır?

Cevap Anahtarı

1. B) Bellman-Ford Algoritması — Bellman-Ford algoritması, negatif kenar ağırlıklarını doğru bir şekilde işleyebilir ve ayrıca grafikte negatif döngülerin varlığını tespit edebilir.

2. B) Negatif kenar ağırlıklarını işleyemez. — Dijkstra algoritması, negatif kenar ağırlıklarına sahip graflarda doğru sonuçlar vermez. Bu tür durumlar için Bellman-Ford algoritması tercih edilir.

3. A) Dijkstra Algoritması — Ağ yönlendirme protokollerinde genellikle kenar maliyetleri (gecikme, bant genişliği vb.) negatif olmadığı için Dijkstra Algoritması yaygın olarak kullanılır.

4. 1. Başlangıç şehrini seçin ve mesafesini 0 olarak ayarlayın, diğer tüm şehirlerin mesafesini sonsuz yapın. 2. Ziyaret edilmemiş şehirler kümesini oluşturun. 3. Mevcut şehirden komşu şehirlere olan mesafeleri güncelleyin. 4. Mevcut şehri ziyaret edildi olarak işaretleyin ve listeden çıkarın. 5. Ziyaret edilmemiş şehirler kümesinden en küçük mesafeye sahip şehri seçin ve mevcut şehir yapın. 6. Hedef şehre ulaşıldığında veya ziyaret edilmemiş şehir kalmadığında durun.

5. 1. Grafikteki tüm kenarlar için V-1 kez gevşetme işlemi yapın (V: köşe sayısı). 2. Gevşetme işlemi sonrasında tekrar bir gevşetme işlemi yapıldığında herhangi bir mesafede azalma oluyorsa, grafikte negatif döngü var demektir. 3. Negatif döngü yoksa, algoritma en kısa yolları bulmuştur.

6. Bir grafikte, bir başlangıç noktasından diğer noktalara giden en az maliyetli yolu bulan algoritmadır.

7. Grafikte negatif olmayan kenar ağırlıkları olduğunda kullanılır.

8. Grafikte negatif kenar ağırlıkları olduğunda ve negatif döngü olup olmadığını tespit etmek için kullanılır.