Bölüm 15: Soru Cevap

Algoritmalar

Soru

- V düğüm ve E kenarı olan bir çizgenin komşuluk listesi temsili üzerinde BFS (Breadth-First Search) algoritmasının bellek karmaşıklığı nedir?

- A) O(V)
- B) O(E)
- C) O(V + E)
- D) $O(V^2)$

Cevap

- A) O(V)

- BFS algoritması, bir çizgenin komşuluk listesi temsilinde V düğüm olduğunda, bir düğüm ziyaret edildiği zaman bellekte sabit bir miktarda yer işgal eder. Bu nedenle, BFS algoritmasının bellek karmaşıklığı *O(V)* olur. Çünkü BFS, ziyaret edilen düğümleri bir kuyrukta saklar ve her düğüm yalnızca bir kez kuyruğa eklenir ve kuyruktan çıkarılır.

Soru

- Dijkstra algoritmasının zaman karmaşıklığı nedir?

- A) O(E log V)
- B) $O(V^2)$
- C) O(V log V)
- D) O(E + V)

Cevap

- A) O(E log V)

- Dijkstra algoritması, en kısa yol problemi için kullanılır ve zaman karmaşıklığı *O(E log V)* olarak ifade edilir. Burada E, kenar sayısını ve V, düğüm sayısını temsil eder. Dijkstra algoritması, öncelikli kuyruk veri yapısını kullanarak minimum mesafeli düğümleri seçerken her bir düğümün en fazla bir kez ziyaret edildiği için E kenarının logaritmik bir faktörle çarpılması gerekir.

Soru

- Floyd-Warshall algoritmasının, V düğümlü ağırlıklı bir çizge üzerinde tüm çiftler arası en kısa yol bulma işlemi zaman karmaşıklığı nedir?

- A) O(V log V)
- B) $O(V^3)$
- C) O(E log V)
- D) O(E + V)

Cevap

- B) $O(V^3)$

- Floyd-Warshall algoritması, tüm çiftler arası en kısa yol problemini çözmek için kullanılır ve $O(V^3)$ zaman karmaşıklığına sahiptir. Burada V, düğüm sayısını temsil eder. Floyd-Warshall algoritması, dinamik programlama yaklaşımını kullanır ve tüm düğümler arasındaki en kısa yolları bulmak için üçlü iç içe döngüler kullanır.

Soru

- Derinlik-Öncelikli Arama (DFS) algoritmasının, V düğüm ve E kenarı olan bir çizgeyi gezme işlemi için bellek karmaşıklığı nedir?

- A) O(V)
- B) O(E)
- C) O(V + E)
- D) $O(V^2)$

Cevap

- A) O(V)

- Derinlik-Öncelikli Arama (DFS) algoritması, bir çizgeyi gezme işlemi sırasında yığın (stack) veri yapısını kullanır. Her bir düğümü ziyaret ettiğinde, bu düğümün bilgisini yığına ekler ve işlem tamamlandığında yığından çıkarır. Bu nedenle, DFS algoritmasının bellek karmaşıklığı, en fazla V düğümünü saklamak için gereken bellek miktarıdır.

Soru

- Ağırlıksız çizgenin iki düğümü arasındaki en kısa yolu bulmak için hangi arama algoritması kullanılır?

- A) İkili Arama (Binary Search)
- B) Derinlik-Öncelikli Arama (DFS)
- C) Genişlik-Öncelikli Arama (BFS)
- D) Doğrusal Arama (Linear Search)

Cevap

- C) Genişlik-Öncelikli Arama (BFS)

- Genişlik-Öncelikli Arama (BFS), ağırlıksız bir çizge üzerinde iki düğüm arasındaki en kısa yolu bulmak için kullanılır. BFS, başlangıç düğümünden başlayarak tüm komşu düğümleri keşfeder ve ardından bu düğümlerin komşularını keşfeder. Bu şekilde, BFS her adımda belirli bir uzaklığa (katman) sahip olan düğümleri keşfeder ve en kısa yolu bulur.

Soru

- V düğüm ve E kenardan oluşan bir çizgeyi gezme işlemi için Genişlik-Öncelikli Arama (BFS) algoritmasının zaman karmaşıklığı nedir?

- A) O(V)
- B) O(E)
- C) O(V + E)
- D) $O(V^2)$

Cevap

- C) O(V + E)

- Genişlik-Öncelikli Arama (BFS) algoritması, çizgeyi gezerken kuyruk (queue) veri yapısı kullanır. Her bir düğümü yalnızca bir kez ziyaret ettiği için, BFS algoritmasının zaman karmaşıklığı, her düğümü bir kez işlediği V düğümlerinin sayısına bağlıdır. Ayrıca, her kenarı en fazla bir kez işlediği için, BFS algoritmasının zaman karmaşıklığı aynı zamanda E kenar sayısına da bağlıdır.

Soru

- Rabin-Karp dize eşleştirme algoritmasının en kötü durum zaman karmaşıklığı nedir?

- A) O(n)
- B) O(n + m)
- C) O(nm)
- D) O(n log m)

Cevap

- C) O(nm)

- Rabin-Karp dize eşleştirme algoritmasının en kötü durum zaman karmaşıklığı, *O(nm)* dir. Bu, metin dizesinin her alt dizesinde örüntünün karmaşıklığını yeniden hesaplamak zorunda olmasından kaynaklanır. Her bir alt dize $n-m+1$ kere kontrol edilmelidir ve her bir kontrolde örüntünün karmaşıklığı *O(m)* dır. Bu nedenle, toplam zaman karmaşıklığı $O((n-m+1)\cdot m)$ veya *O(nm)* dir.

Soru

- Dinamik programlama kullanarak En Uzun Ortak Alt Dizi (LCS) algoritmasının zaman karmaşıklığı nedir?

- A) O(n)
- B) $O(n^2)$
- C) O(nm)
- D) $O(2^n)$

Cevap

- C) O(nm)

- En Uzun Ortak Alt Dizi (LCS) algoritması, iki dize arasındaki en uzun ortak alt diziyi bulmak için dinamik programlama kullanır. İlk dize uzunluğu *n*, ikinci dize uzunluğu *m* olsun. LCS algoritması, iki dizeyi karşılaştırırken bir matris oluşturur ve bu matrisi doldurur. Matrisin boyutu *n*×*m* olur ve her bir hücreyi doldurmak için sabit zaman gerektirir.

Soru

- Bir metin içinde bir örüntünün tüm tekrarlarını bulmak için kullanışlı olan ve zaman karmaşıklığı O(n + m) olan hangi dize algoritmasıdır?

- A) Rabin-Karp
- B) Knuth-Morris-Pratt (KMP)
- C) Boyer-Moore
- D) Z Algoritması

Cevap

- B) Knuth-Morris-Pratt (KMP)

- Knuth-Morris-Pratt (KMP) dize eşleştirme algoritması, bir örüntünün bir metin içindeki tüm tekrarlarını bulmak için kullanılabilir ve zaman karmaşıklığı $O(n+m)$ olarak ifade edilir. Bu algoritma, metin içindeki örüntünün tüm tekrarlarını etkili bir şekilde bulmak için özel olarak tasarlanmıştır.

Soru

- Yönlü bir çizge içinde güçlü bağlı bileşenleri bulmak için hangi algoritma kullanılır?

- A) Prim algoritması
- B) Floyd-Warshall algoritması
- C) Dijkstra algoritması
- D) Kosaraju algoritması

Cevap

- D) Kosaraju algoritması

- Güçlü bağlı bileşenleri bulmak için Kosaraju algoritması kullanılır. Bu algoritma, yönlü çizge içindeki güçlü bağlı bileşenleri etkili bir şekilde tanımlamak için kullanılır. Kosaraju algoritması, çizgenin transpozunu alarak ve derinlik-öncelikli arama (DFS) yöntemini kullanarak güçlü bağlı bileşenleri tanımlar.

Soru

- Bellman-Ford algoritmasının, negatif kenar ağırlıklarına sahip ağırlıklı çizgede en kısa yolu bulmak için zaman karmaşıklığı nedir?

- A) O(V)
- B) O(E)
- C) O(V log V)
- D) O(VE)

Cevap

- D) O(VE)

- Bellman-Ford algoritması, bir ağırlıklı çizgenin en kısa yolunu bulmak için kullanılır ve negatif kenar ağırlıklarını kabul eder. Algoritmanın zaman karmaşıklığı, tüm kenarları *V x E* kez inceleme yaparak *O(VE)* olur. Her bir kenarı inceleme işlemi, algoritmanın tüm düğümleri için gerçekleşir, bu da toplam karmaşıklığı *O(VE)* yapar.

Soru

- Hem yönlü hem yönsüz bir çizge içinde döngü tespit etmek için hangi algoritma kullanılır?

- A) Genişlik-Öncelikli Arama (BFS)
- B) Derinlik-Öncelikli Arama (DFS)
- C) Dijkstra algoritması
- D) Floyd-Warshall algoritması

Cevap

- B) Derinlik-Öncelikli Arama (DFS)

- Derinlik-Öncelikli Arama (DFS) algoritması, bir çizge içinde döngü tespit etmek için kullanılabilir ve hem yönlü hem de yönsüz çizgelerle başa çıkabilir. DFS, çizgenin her düğümünü keşfederken geri izleme yapar ve ziyaret edilen düğümleri bir yığında tutar. Eğer DFS, zaten ziyaret edilmiş bir düğümü tekrar ziyaret ederse, bu durum bir döngü olduğunu gösterir.

Soru

- Kruskal algoritmasının, V düğümü ve E kenarı olan bir çizgenin minimum kapsayan ağacını bulmak için zaman karmaşıklığı nedir?

- A) O(V)
- B) O(E)
- C) O(V log V)
- D) O(E log V)

Cevap

- D) O(E log V)

- Kruskal algoritması, bir çizgenin minimum kapsayan ağacını bulmak için kullanılır. Algoritmanın zaman karmaşıklığı, kenarların ağırlıklarına göre sıralanması ve bir birleştirme-bölme veri yapısı kullanılarak *O(ElogV)* olur.

SON