[SERKAN DOLDUR](javascript:;)

E235013188

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ ANABİLİM DALI

BİLİŞİM TEKNOLOJİLERİ PR. (YL) (UZAKTAN EĞİTİM)

EBT 512 VERİ YAPILARI VE ALGORİTMALAR 1.ÖDEV

**a) Dizi Veri Yapısı ile Dijkstra Algoritması:**

Dizi veri yapısıyla Dijkstra algoritmasını uygularken her düğümün **en kısa yol uzunluğunu ve önceki düğümü** tutacak bir dizi kullanırız.

**Algoritma Adımları:**

1. Başlangıç düğümünü seç ve uzaklık dizisini güncelle.

2. Başlangıç düğümüne komşu olan düğümleri gez, uzaklık dizisini güncelle.

3. Henüz işlenmemiş en kısa mesafedeki düğümü seç.

4. Seçilen düğüme komşu olan düğümleri gez, uzaklık dizisini güncelle.

5. İşlenmiş düğümleri işaretle.

6. Hedef düğümüne ulaşana kadar 3-5. adımları tekrarla.

**Zaman Karmaşıklığı:** Her adımda tüm düğümleri dolaşıp uzaklık dizisini güncelleme işlemi yapılır.

Bu işlem O(V^2) zaman karmaşıklığına sahiptir.

**Bellek Karmaşıklığı:** Uzaklık dizisi ve önceki düğüm dizisi tutulur. Her düğüm için 2 değer tutulduğundan bellek karmaşıklığı O(V) olacaktır.

**Örnek:**

A --5-- B

/ \ / \

1 4 2 C

\ / \ /

D --3—E

**Adımlar:**

1.Başlangıç düğümünü seçiyoruz. Başlangıç düğümümüz A olsun.

2.A'dan A'ya gitmek için 0 adım gerektiğini ve önceki düğümünün olmadığını belirtiyoruz. Diğer düğümlere ise sonsuz uzaklık ve önceki düğüm olarak Null değerlerini atıyoruz.

**Adım 0:**

|  |  |  |
| --- | --- | --- |
| **Düğüm** | **Uzaklık** | **Önceki Düğüm** |
| A | 0 | Null |
| B | ∞ | Null |
| C | ∞ | Null |
| D | ∞ | Null |
| E | ∞ | Null |

3.A'nın komşularını ve onlara olan uzaklıklarını güncelliyoruz.

B: A'dan B'ye 5 birim.

D: A'dan D'ye 1 birim.

**Adım 1:**

|  |  |  |
| --- | --- | --- |
| **Düğüm** | **Uzaklık** | **Önceki Düğüm** |
| A | 0 | Null |
| B | 5 | A |
| C | ∞ | Null |
| D | 1 | A |
| E | ∞ | Null |

4.Henüz işlenmemiş en kısa mesafedeki düğümü seçiyoruz. Şu an D düğümü en kısa mesafede.

5.D'nin komşularını ve onlara olan uzaklıklarını güncelliyoruz.

B: D'den B'ye 4 birim.

E: D'den E'ye 3 birim.

**Adım 2:**

|  |  |  |
| --- | --- | --- |
| **Düğüm** | **Uzaklık** | **Önceki Düğüm** |
| A | 0 | Null |
| B | 5 | A |
| C | ∞ | Null |
| D | 1 | A |
| E | ∞ | D |

6. Şu anki en kısa mesafedeki düğüm B. B'nin komşularını kontrol ediyoruz.

A zaten işlendiği için kontrol etmiyoruz.

C: B'den C'ye 2 birim.

E: B'den E'ye 2 birim.

**Adım 3:**

|  |  |  |
| --- | --- | --- |
| **Düğüm** | **Uzaklık** | **Önceki Düğüm** |
| A | 0 | Null |
| B | 5 | A |
| C | 7 | B |
| D | 1 | A |
| E | 4 | D |

7.Şu anki en kısa mesafedeki düğüm E. E'nin komşularını kontrol ediyoruz.

B ve D zaten işlendiği için kontrol etmiyoruz.

C: E'den C'ye 2 birim.

**Adım 4:**

|  |  |  |
| --- | --- | --- |
| **Düğüm** | **Uzaklık** | **Önceki Düğüm** |
| A | 0 | Null |
| B | 5 | A |
| C | 6 | E |
| D | 1 | A |
| E | 4 | D |

8.En kısa yollar hesaplanarak algoritmamız tamamlanmış oldu.

**b) Bağlı Liste Veri Yapısı ile Dijkstra Algoritması:**

**Bağlı liste veri yapısı**, Dijkstra algoritmasında kullanılabilir ancak genellikle **performans açısından dizilere tercih edilmez.**

**Algoritma Adımları:** Adımlar aynıdır, sadece veri yapısı farklıdır.

**Zaman Karmaşıklığı:** her adımda tüm düğümleri dolaşıp uzaklık dizisini güncelleme işlemi yine O(V^2) zaman karmaşıklığına sahiptir.

**Bellek Karmaşıklığı:** Bağlı liste her düğüm için bir diziye göre daha fazla bellek tüketir. Her bir düğüm için en az iki işaretçi gerektiğinden bellek karmaşıklığı O(V) olacaktır.

**Örnek:**

A --4-- B

/ \ / \

1 2 3 C

\ / \ /

D --5—E

**Adımlar:**

1.Başlangıç düğümünü seçiyoruz. Başlangıç düğümümüz A olsun.

2.A'dan A'ya gitmek için 0 adım gerektiğini ve önceki düğümünün olmadığını belirtiyoruz. Diğer düğümlere ise sonsuz uzaklık ve önceki düğüm olarak Null değerlerini atıyoruz.

**Adım 0:**

|  |  |  |
| --- | --- | --- |
| **Düğüm** | **Uzaklık** | **Önceki Düğüm** |
| A | 0 | Null |
| B | ∞ | Null |
| C | ∞ | Null |
| D | ∞ | Null |
| E | ∞ | Null |

3.A'nın komşularını ve onlara olan uzaklıklarını bağlı listeyle temsil ediyoruz.

B: A'dan B'ye 4 birim.

D: A'dan D'ye 1 birim.

**Adım 1:**

A -> B:4 -> D:1 -> Null

4.Henüz işlenmemiş en kısa mesafedeki düğümü seçiyoruz. Şu an D düğümü en kısa mesafede.

5.D'nin komşularını ve onlara olan uzaklıklarını güncelliyoruz.

B: D'den B'ye 5 birim.

E: D'den E'ye 5 birim.

**Adım 2:**

A -> D:1 -> B:5 -> E:5 -> Null

6.Şu anki en kısa mesafedeki düğüm B. B'nin komşularını kontrol ediyoruz.

A zaten işlendiği için kontrol etmiyoruz.

C: B'den C'ye 3 birim.

**Adım 3:**

A -> D:1 -> B:5 -> E:5 -> C:8 -> Null

7.Şu anki en kısa mesafedeki düğüm E. E'nin komşularını kontrol ediyoruz.

D ve B zaten işlendiği için kontrol etmiyoruz.

8. En kısa yollar hesaplanarak algoritmamız tamamlanmış oldu.

**c) Öncelikli Kuyruk Veri Yapısı ile Dijkstra Algoritması:**

Öncelikli kuyruk (priority queue) kullanarak Dijkstra algoritmasını gerçekleştirmek, **genellikle en etkili yöntemdir.** Her adımda en küçük uzaklığa sahip düğümü seçmek için kullanılır.

**Algoritma Adımları:** Adımlar yine aynıdır ancak en kısa uzaklığa sahip düğümü seçmek için öncelikli kuyruk kullanılır.

**Zaman Karmaşıklığı:** En küçük uzaklığa sahip düğümü bulmak öncelikli kuyruk ile O(log V) karmaşıklığında yapılır. Toplam karmaşıklık O(V log V + E log V) olur ki, E kenar sayısıdır.

**Bellek Karmaşıklığı:** Öncelikli kuyruk da her düğüm için bir öncelik ve işaretçi gerektirir. Bellek karmaşıklığı O(V) olacaktır.

**Sonuç:**

Dizi veri yapısı, bellek açısından en az yer kaplar ancak zaman karmaşıklığı yüksektir.

Bağlı liste veri yapısı, genelde dizilere göre daha fazla bellek tüketir ve zaman karmaşıklığı benzerdir.

Öncelikli kuyruk veri yapısı hem zaman hem de bellek açısından daha etkilidir. **En iyi performansı sağlar.**

Öncelikli kuyruk, Dijkstra algoritması için en uygun veri yapısıdır.

**Örnek**

A --3-- B

/ \ / \

1 4 2 C

\ / \ /

D --5—E

**Adımlar:**

1.Başlangıç düğümünü seçiyoruz. Başlangıç düğümümüz A olsun.

2.A'dan A'ya gitmek için 0 adım gerektiğini ve önceki düğümünün olmadığını belirtiyoruz. Diğer düğümlere ise sonsuz uzaklık ve önceki düğüm olarak Null değerlerini atıyoruz.

**Adım 0:**

|  |  |  |
| --- | --- | --- |
| **Öncelikli Kuyruk: (A, 0)** | | |
| **Düğüm** | **Uzaklık** | **Önceki Düğüm** |
| A | 0 | Null |
| B | ∞ | Null |
| C | ∞ | Null |
| D | ∞ | Null |
| E | ∞ | Null |

3.Öncelikli kuyruktan en kısa mesafedeki düğümü seçiyoruz, şu an A.

4.A'nın komşularını ve onlara olan uzaklıklarını güncelliyoruz.

B: A'dan B'ye 3 birim.

C: A'dan C'ye 1 birim.

**Adım 1:**

|  |  |  |
| --- | --- | --- |
| **Öncelikli Kuyruk: (C,1),(B,3)** | | |
| **Düğüm** | **Uzaklık** | **Önceki Düğüm** |
| A | 0 | Null |
| B | 3 | A |
| C | 1 | A |
| D | ∞ | Null |
| E | ∞ | Null |

5.Öncelikli kuyruktan en kısa mesafedeki düğümü seçiyoruz, şu an C.

6.C'nin komşularını ve onlara olan uzaklıklarını güncelliyoruz.

B: C'den B'ye 2 birim.

E: C'den E'ye 2 birim.

**Adım 2:**

|  |  |  |
| --- | --- | --- |
| **Öncelikli Kuyruk: (D, 4), (B, 3), (E, 3)** | | |
| **Düğüm** | **Uzaklık** | **Önceki Düğüm** |
| A | 0 | Null |
| B | 3 | A |
| C | 1 | A |
| D | ∞ | Null |
| E | 2 | C |

7.Öncelikli kuyruktan en kısa mesafedeki düğümü seçiyoruz, şu an E.

8.E'nin komşularını ve onlara olan uzaklıklarını güncelliyoruz.

D: E'den D'ye 5 birim.

**Adım 3:**

|  |  |  |
| --- | --- | --- |
| **Öncelikli Kuyruk: (D, 4), (B, 3), (D, 7)** | | |
| **Düğüm** | **Uzaklık** | **Önceki Düğüm** |
| A | 0 | Null |
| B | 3 | A |
| C | 1 | A |
| D | ∞ | Null |
| E | 2 | C |

9.Öncelikli kuyruktan en kısa mesafedeki düğümü seçiyoruz, şu an D.

10.D'nin komşularını ve onlara olan uzaklıklarını güncelliyoruz.

B: D'den B'ye 4 birim.

**Adım 4:**

|  |  |  |
| --- | --- | --- |
| **Öncelikli Kuyruk: (D, 7), (B, 3)** | | |
| **Düğüm** | **Uzaklık** | **Önceki Düğüm** |
| A | 0 | Null |
| B | 3 | A |
| C | 1 | A |
| D | 4 | E |
| E | 2 | C |

11.Öncelikli kuyruktan en kısa mesafedeki düğümü seçiyoruz, şu an B.

12.B'nin komşularını kontrol ediyoruz, ancak zaten işlendiğinden geçiyoruz.

**Adım 5:**

|  |  |  |
| --- | --- | --- |
| **Öncelikli Kuyruk: (D, 7)** | | |
| **Düğüm** | **Uzaklık** | **Önceki Düğüm** |
| A | 0 | Null |
| B | 3 | A |
| C | 1 | A |
| D | 4 | E |
| E | 2 | C |

13.Son adıma geldik, öncelikli kuyruktan en kısa mesafedeki düğümü seçiyoruz, şu an D.

14.D'nin komşularını kontrol ediyoruz, ancak zaten işlendiğinden geçiyoruz.

**Adım 6:**

|  |  |  |
| --- | --- | --- |
| **Öncelikli Kuyruk: Boş** | | |
| **Düğüm** | **Uzaklık** | **Önceki Düğüm** |
| A | 0 | Null |
| B | 3 | A |
| C | 1 | A |
| D | 4 | E |
| E | 2 | C |

15. En kısa yollar hesaplanarak algoritmamız tamamlanmış oldu.

Sonuç: Bu örnekte A'dan diğer düğümlere olan en kısa yol uzunlukları bu şekilde hesaplanmış oldu.