T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ ANABİLİM DALI
BİLİŞİM TEKNOLOJİLERİ PR. (YL) (UZAKTAN EĞİTİM)

VERİ YAPILARI VE ALGORİTMALAR 1.ÖDEV
DIJKSTRA ALGORİTMASI

Hazırlayan
SEDAT ÖZTÜRK
E235013168

Öğretim Üyesi
Prof. Dr. NEJAT YUMUŞAK

MAYIS 2024

Dijkstra Algoritması

Dijkstra algoritması ismini algoritmanın geliştiricisi olan Hollandalı bilgisayar bilimci Edsger Dijkstra'dan almaktadır. Dijkstra algoritması, en kısa yol problemi için kullanılan bir graf algoritmasıdır. Bu algoritmanın amacı, çıkış noktası olarak belirlediği noktadan farklı düğümler aracılığıyla en kısa yol ile hedefe ulaşmaktır.

Günümüzde de internet trafiğinin yönlendirilmesinde, oyun programlamada, Google ve Yandex haritaların kullandığı algoritmalardan birisidir. Endüstri Mühendisliği alanında da üretim, lojistik, bir hizmetin en kısa yolu göz önünde bulundurarak kullanıcıya ulaştırmak gibi konular başta olmak üzere birçok optimizasyon probleminin çözümünde faydalanılır.

Dijkstra Algoritması genel olarak, amaçlanan en kısa mesafeye varana kadar doğru mesafenin yaklaşık olarak hesaplanan değerinin daha uygun değerler ile yer değiştirmesi olayını kullanır. Bu olaya "Gevşeme ilkesi" denir.

Algoritma kullanmamız için bazı kuralları vardır. Grafiğimiz ağırlık ve yöne sahip olmalıdır. Kenarların ağırlıkları pozitif değerlerde iken doğru çalışmaktadır. Bunun nedeni, en kısa yol bulunurken kenarların ağırlıklarının eklenmesidir. Yani bir grafta kenarlardan herhangi biri negatif tanımlanırsa algoritma istenilen sonucu bulamayacaktır. Yine de bazı algoritmalar (Bellman-Ford gibi) ile bu problem aşılabilir.

Dizi Veri Yapısı ile incelendiğinde;

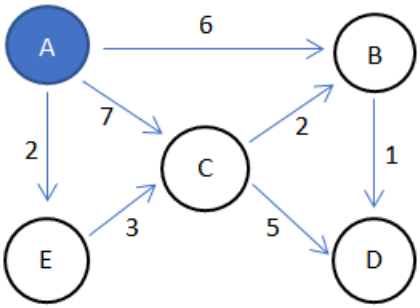
Her düğümün başlangıç düğümüne en yakın mesafe ve önceki düğüm bilgisi tutacak şekilde bir dizi kullanırız.

Çalışma Mantığı:

- Tüm düğümlerin mesafesini sonsuz olarak ayarla, başlangıç düğümünün mesafesini 0 olarak ayarla.
- Ziyaret edilmemiş düğümler arasında en kısa mesafeye sahip olanı seç.
- Seçilen düğümün komşularını ziyaret et ve mesafelerini güncelle
- Seçilen düğümü ziyaret edilmiş olarak işaretle.
- Tüm düğümler ziyaret edilene kadar işlemi tekrarla.

Adımlar:

1. **Adım:**
   - **a.** Başlangıç düğümümüz A olsun.
   - **b.** A'dan A ya gitmek için mesafe 0 birim mesafedir ve önceki düğümleri olmadığı için boş geçiyoruz. Diğer düğümlere sonsuz uzaklıkta olduğunu belirtiyoruz.

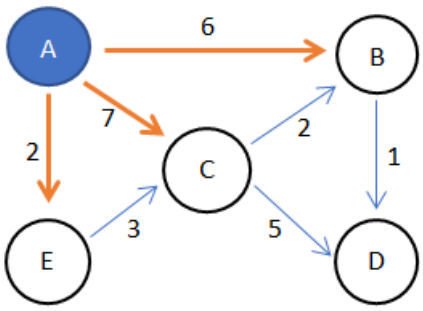


Düğüm	Mesafe	Önceki Düğüm
A	0	
B	∞	
C	∞	
D	∞	
E	∞	

2. **Adım:**
   - **a.** A ya komşu B ve E düğümlerine olan uzaklıkları yazıyoruz.

b. A dan B ye 6 birim mesafe bulunmaktadır. A dan E ye 2 birim mesafe bulunmaktadır. A dan C ye 7 birim mesafe bulunmaktadır.

c. A düğümü ziyaret edilenler listesine alınır.



Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	7	A
D	∞	
E	2	A

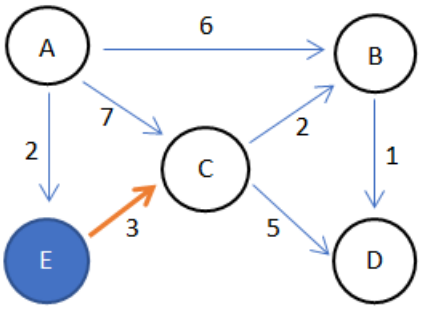
3. Adım:

a. Elde edilen uzaklıklardan en küçük olan düğümü seçiyoruz. En küçük mesafe E düğümüdür.

b. E düğümünün komşu düğümlerini güncelliyoruz.

c. E den sadece C ye komşuluğu bulunmaktadır ve mesafesi 3 birimdir. Bu durumda A dan E olan mesafe ile toplandığında C düğümünün yeni mesafesi 5 oluyor. Bir önceki adımda C düğümü mesafesi 7 birimdi. Yeni oluşan mesafe 7 birimden küçük olduğu için yeni mesafe bilgisi ile önceki düğüm C için güncellenmelidir.

d. E düğümü ziyaret edilenler listesine alınır.



Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	5	E
D	∞	
E	2	A

4. Adım:

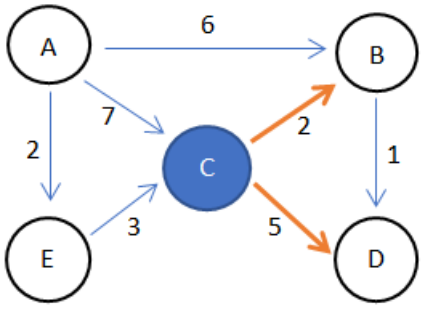
a. Elde edilen uzaklıklardan en küçük olan düğümü seçiyoruz. En küçük mesafe C düğümüdür.

b. C düğümünün komşu B ve D düğümlerine olan uzaklıklarını yazıyoruz.

c. A dan C ye olan 5 birim mesafe ile C düğümü B düğümüne olan mesafesi 2 birim toplandığında 7 birim mesafe elde edilir. B düğümünün şu anki mesafesi 6 birimden büyük olduğu için değiştirilmez.

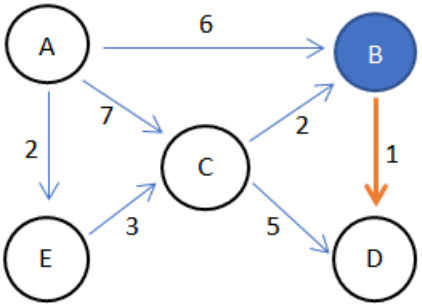
d. A dan C ye olan 5 birim mesafe ile C düğümü D düğümüne olan mesafesi 5 birim toplandığında 10 birim mesafe elde edilir.

e. C düğümü ziyaret edilenler listesine alınır.



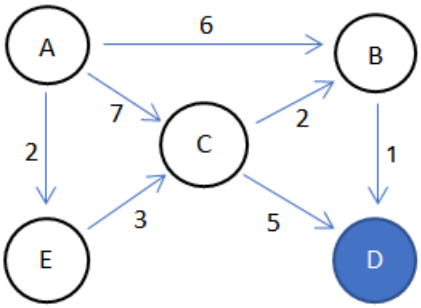
Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	5	E
D	10	C
E	2	A

5. **Adım:**
   a. Elde edilen uzaklıklardan en küçük olan düğümü seçiyoruz. En küçük mesafe B düğümüdür.
   b. B düğümü sadece D ye komşuluğu bulunmaktadır ve mesafesi 1 birimdir. Bu durumda A dan B ye olan mesafe ile toplandığında 7 birim mesafe elde edilir. Yeni oluşan mesafe 10 birimden küçük olduğu için yeni mesafe bilgisi ile önceki düğüm D için güncellenmelidir.
   c. B düğümü ziyaret edilenler listesine alınır.



Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	5	E
D	7	B
E	2	A

6. **Adım:**
   a. D düğümü en son düğüm olması ve hiçbir düğüme komşuluğu bulunmadığı için algoritma tamamlanmış oldu.



Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	5	E
D	7	B
E	2	A

Zaman Karmaşıklığı Analizi:

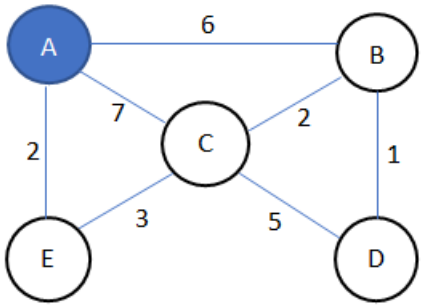
1. Algoritmanın çalışma süresini girdi boyutuna bağlıdır. Her adımda tüm düğümleri dolaşılıp uzaklık bilgisi güncellenir.
2. Bu işlem O(V2) zaman karmaşıklığına sahiptir. Burada V (Vertex) düğüm sayısı ifade eder.
3. Her bir düğüm için tüm komşularının ziyaret etmek ve en kısa mesafeyi bulmak için dizi üzerinden doğrusal arama yapılması gerektiği anlamına gelir.

Hafıza Karmaşıklığı Analizi:

1. Algoritmanın çalışması gereken hafıza miktarını ifade eder.
2. Dizi veri yapısında her düğüm için mesafeyi saklamak üzere bir dizi kullanılır ve bu da O(V) hafıza karmaşıklığına sahiptir.
3. Ayrıca her düğümün ziyaret edilip edilmediğini takip etmek için ek bir diziye ihtiyaç duyulur
4. Sonuç olarak toplam bellek karmaşıklığı O(V) olur.

Bağlı Liste Veri Yapısı ile incelendiğinde;

Her düğüm kendine ait bir veriyi ve sıradaki düğümün adresini tutar. Düğümlerin bağlı listelerde saklamak için aşağıdakine benzer bir dizi tanımlanabilir.



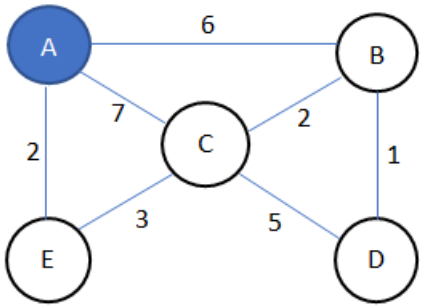
Düğüm	Bağlı Liste
A	[('B', 6), ('E', 2), ('C', 7)]
B	[('A', 6), ('C', 2), ('D', 1)]
C	[('B', 2), ('A', 7), ('D', 5), ('E', 3)]
D	[('B', 1), ('C', 5)]
E	[('A', 2), ('C', 3)]

Çalışma Mantığı:

- Tüm düğümlerin mesafesini sonsuz olarak ayarla, başlangıç düğümünün mesafesini 0 olarak ayarla.
- Ziyaret edilmemiş düğümler arasında en kısa mesafeye sahip olanı seç.
- Seçilen düğümün bağlı listesindeki komşularını ziyaret et ve mesafelerini güncelle
- Seçilen düğümü ziyaret edilmiş olarak işaretle.
- Tüm düğümler ziyaret edilene kadar işlemi tekrarla.

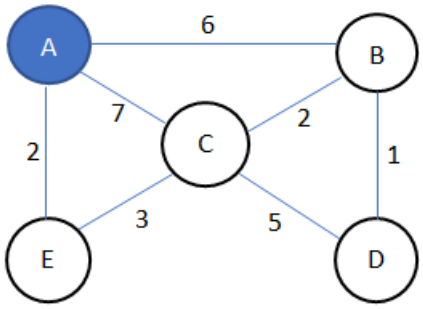
Adımlar:

1. **Adım:**
   - **a.** Başlangıç düğümümüz A olsun.



Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	∞	
C	∞	
D	∞	
E	∞	

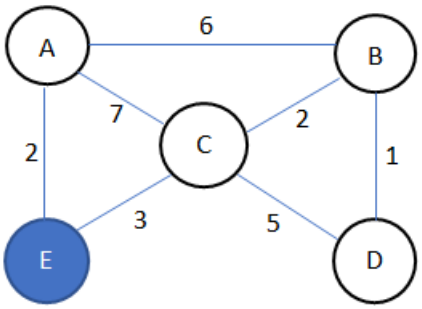
2. **Adım:**
   - **a.** Komşuların mesafelerini güncelleyin. A düğümünün bağlı listesi [('B', 6), ('E', 2), ('C', 7)]
   - **b.** B, C ve E düğümleri güncellendi.
   - **c.** A düğümü ziyaret edildi şeklinde işaretlenir.



Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	7	A
D	∞	
E	2	A

3. **Adım:**
   - **a.** En kısa mesafe E düğümü olması sebebiyle komşuların mesafelerini güncelleyin. E düğümünün bağlı listesi: [('A', 2), ('C', 3)]

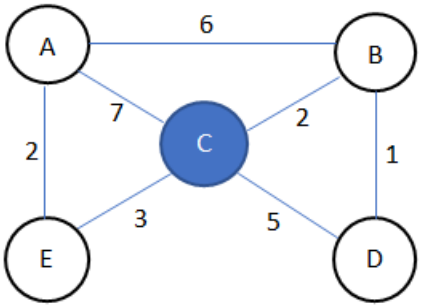
b. A düğümü öncesinde ziyaret edilmişti.
c. C düğümü E üzerinden 5 olarak güncellenir.
d. C düğümü ziyaret edildi şeklinde işaretlenir.



Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	5	E
D	∞	
E	2	A

4. Adım:

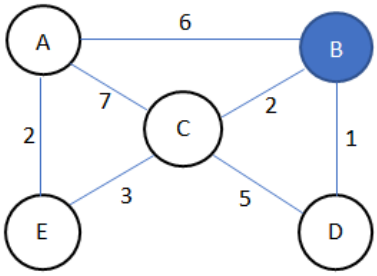
a. C düğümü bağlı komşuları mesafeleri güncelleyin. C düğümünün bağlı listesi: [('B', 2), ('A', 7), ('D', 5), ('E', 3)]
b. Komşuları A, B, D ve E düğümleridir.
c. A ve E düğümleri öncesinde ziyaret edilmişti.
d. D düğümü C düğümü üzerinden 10 olarak güncellenir.
e. B düğümü C üzerinden mesafesi 7 olarak hesaplanır. Ancak mesafesi zaten 6 olduğu için değiştirilmez.



Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	5	E
D	10	C
E	2	A

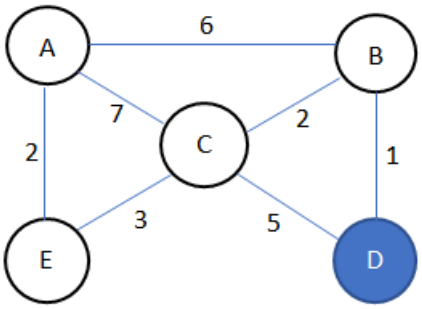
5. Adım:

a. En kısa mesafe B düğümüdür. B düğümünün bağlı listesi: [('A', 6), ('C', 2), ('D', 1)]
b. B düğümünün bağlı komşuları A, C ve D düğümleridir.
c. A ve C düğümleri öncesinde ziyaret edilmişti.
d. D düğümü B üzerinden 7 olarak güncellenir**.**



Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	5	E
D	7	B
E	2	A

6. **Adım:**
    a. D düğümünün bağlı listesi: [(‘B’, 1), (‘C’, 5)]
    b. B ve C düğümleri öncesinde ziyaret edilmişti.
    c. D düğümü ziyaret edildi şeklinde işaretlenir.



Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	5	E
D	7	B
E	2	A

Zaman Karmaşıklığı Analizi:

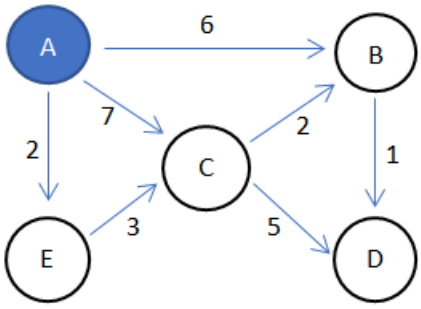
1. Her düğüm için komşularına erişmek ve mesafeleri güncellemek için O(V) zaman gerekebilir, bu toplam da O(V2) zaman karmaşıklığa yol açar.

Hafıza Karmaşıklığı Analizi:

1. Her düğüm ve kenar için gereken hafıza miktarı toplamda O(V + E) ile ifade edilir. V, düğüm sayısını ve E, kenar sayısını ifade eder.

Öncelikli Kuyruk Veri Yapısı ile incelendiğinde;

Öncelikli kuyruk veri yapısı ise, bu algoritmanın daha verimli çalışmasını sağlar. Öncelikli kuyruk, her düğümün önceliğini (genellikle en düşük toplam ağırlığa sahip olma) göz önünde bulundurarak, işlenmesi gereken sonraki düğümü seçer.



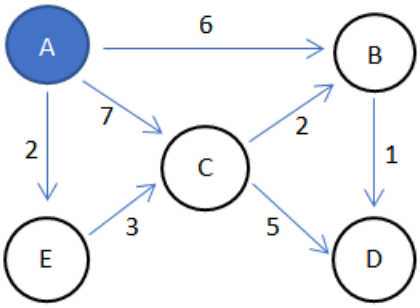
Çalışma Mantığı:

- Tüm düğüm, kuyrukta bir öncelik değerine sahip olur. Bu değer, başlangıç düğümünden o düğüme kadar olan toplam ağırlığı temsil eder.
- Algoritma başladığında, başlangıç düğümü öncelikli kuyruğa eklenir ve öncelik değeri 0 olarak ayarlanır.
- Öncelikli kuyruk boş olmadığı sürece, kuyruktan en düşük öncelik değerine sahip düğüm çıkarılır ve işlenir.

- İşlenen düğümün komşuları incelenir ve eğer bu komşulara ulaşmanın yeni ve daha kısa bir yolu bulunursa, bu komşular güncellenir ve öncelikli kuyruğa eklenir.
- Bu işlem, hedef düğüme ulaşana kadar veya kuyruk boşalana kadar devam eder.

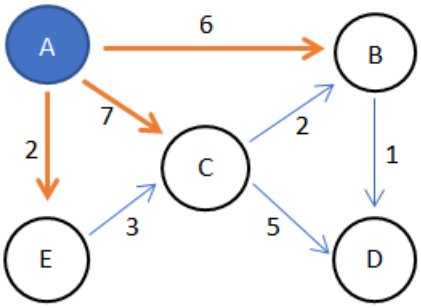
Adımlar:

1. **Adım:** Öncelikli Kuyruk (A,0)



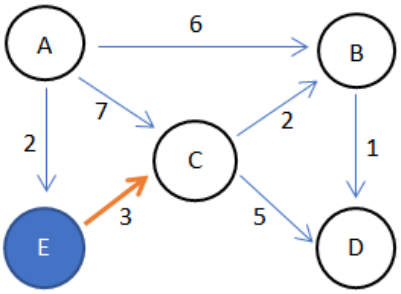
Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	∞	
C	∞	
D	∞	
E	∞	

2. **Adım:** Öncelikli Kuyruk (A,0)
   a. A düğümünün komşularına olan B, C, E düğümlerini güncelliyoruz.
   b. A düğümünden B düğümüne 6 birim, A düğümünden C düğümüne 7 birim, A düğümünden E düğümüne 2 birim



Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	7	A
D	∞	
E	2	A

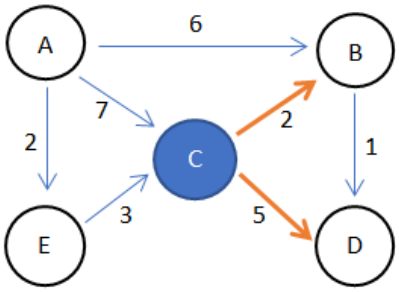
3. **Adım:** Öncelikli Kuyruk (B,6), (C,7), (E,2)
   a. Öncelikli kuyruktan en yakın mesafe E düğümüdür.
   b. E düğümünün komşusu olan C düğümünü güncelliyoruz.
   c. E düğümünden C düğümüne 3 birim



Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	7	A
D	∞	
E	2	A

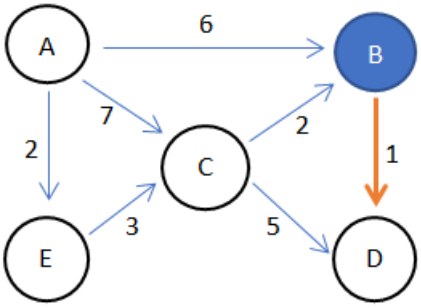
4. **Adım:** Öncelikli Kuyruk (C,5), (B,6), (C,7)
   a. Öncelikli kuyruktan en yakın mesafe C düğümüdür.
   b. C düğümünün komşusu olan B, D düğümünü güncelliyoruz.

c. C düğümünden B düğümüne 2 birim, C düğümünden D düğümüne 5 birim



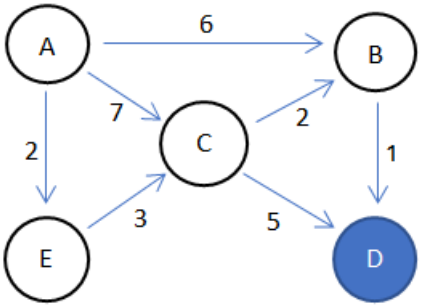
Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	5	E
D	∞	
E	2	A

5. **Adım:** Öncelikli Kuyruk (B,6), (B,7), (D, 10)
   - **a.** Öncelikli kuyruktan en yakın mesafe B düğümüdür.
   - **b.** B düğümünün komşusu D düğümünü güncelliyoruz.
   - **c.** B düğümünün D düğümüne 1 birim



Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	5	E
D	7	B
E	2	A

6. **Adım:** Öncelikli kuyruk (D,7), (D, 10)
   - **a.** Öncelikli kuyruk en yakın mesafe D düğümüdür
   - **b.** D düğümü son adımdır.



Düğüm	A ya en kısa mesafe	Önceki Düğüm
A	0	
B	6	A
C	5	E
D	7	B
E	2	A

Zaman Karmaşıklığı Analizi:

1. Öncelikli kuyruk kullanıldığında aman karmaşıklığı **O((V + E) log V) dir**

Hafıza Karmaşıklığı Analizi:

1. Öncelikli kuyruk yapısı da O(V+E) hafıza karmaşıklığına sahiptir.

Kısaltmalar:

- **O (Büyük O Notasyonu):** Algoritmanın zaman veya hafıza karmaşıklığını ifade eder. Bir algoritmanın çalışma hızını veya hafıza kullanımını analiz etmek için kullanılır. Örneğin, "O(n)" ifadesi, bir algoritmanın çalışma süresinin girdi boyutuna doğrusal olarak bağlı olduğunu ifade eder.
- **V (Düğüm Sayısı):** Graf içindeki düğüm sayısını temsil eder. Örneğin, bir ağaç yapısında V, ağaçtaki düğüm sayısını ifade eder.
- **E (Kenar Sayısı):** Graf içindeki kenar sayısını temsil eder. Örneğin, bir ağaç yapısında E, ağaçtaki kenar sayısını ifade eder.

Yorum:

En kısa yol problemlerini çözmek için kullanılan klasik bir algoritmadır ve farklı veri yapılarına kullanarak uygulanabilir. Bu veri yapılarını her birinin zaman ve bellek karmaşıklığı üzerinden önemli etkileri vardır. ***Dizi Veri Yapısı*** genellikle en basit yaklaşımdır. Ancak bu yöntemde en düşük maliyetli düğümü bulmak için tüm düğümleri tarayarak minimumu bulmak gerekir. ***Bağlı Liste Veri Yapısı*** uygulandığında, düğümler ve kenarları dinamik olarak saklayabilir. Ancak en düşük maliyetli düğümü bulma süreci verimsizdir ve zaman karmaşıklığı dizi veri ile benzerdir. ***Öncelikli Kuyruk Veri Yapısı*** en düşük maliyetli düğümü bulmayı çok daha verimli hale getirir ve büyük graflar üzerinden çalışırken zaman ve hafıza açısından en verimli seçenektir. Dizi ve bağlı liste kullanımı daha basit olabilir, ancak büyük veri setleri için önerilmez.