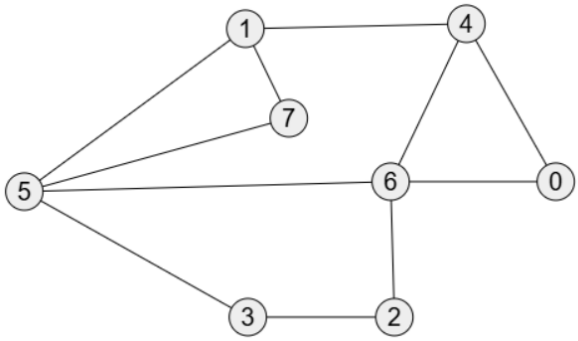
GİRESUN ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
ALGORİTMALAR DERSİ FİNAL SINAVI

Adı – Soyadı – Numarası:

Soru 1: Aşağıda verilen yönsüz çizge için 0 numaralı düğümden başlayarak (a) BFS ve DFS algoritmaları ile gezildiğinde oluşan yolu yazınız. (b) İki algoritmanın çalışma mantığını, kullandığı veri yapısını ve algoritma karmaşıklığını açıklayınız. Not: Komşu düğümlerden birini seçerken numarası düşük olana öncelik veriniz.



BFS: [0, 4, 6, 1, 2, 5, 7, 3]

DFS: [0, 4, 1, 5, 3, 2, 6, 7]

BFS Süreci: (Düğümler seviye seviye ziyaret edilir) O(V + E)

- 0 düğümünden başla, bir kuyruğa ekle.
- 0'ı ziyaret edildi olarak işaretle ve yola ekle.
- 0'ı kuyruktan çıkar, komşularını düğüm numaralarına göre artan sırayla incele.
- Ziyaret edilmemiş her komşu için, işaretle, kuyruğa ekle ve yola ekle.
- Kuyruk boşalana kadar devam et.

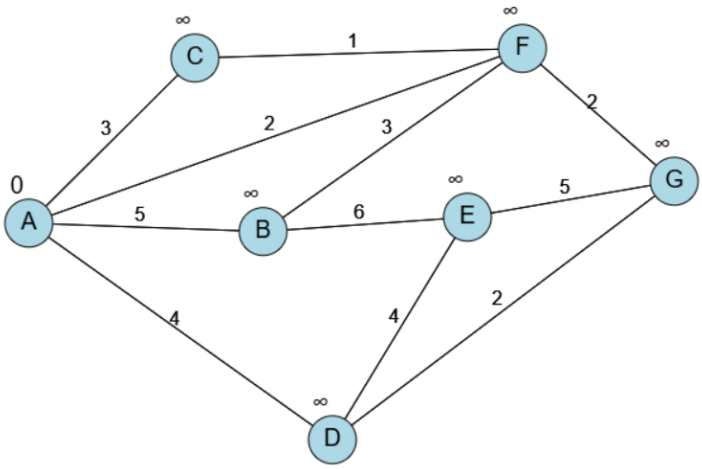
DFS Süreci: (Her düğüm mümkün olduğunca derinlemesine keşfedilir) O(V + E)

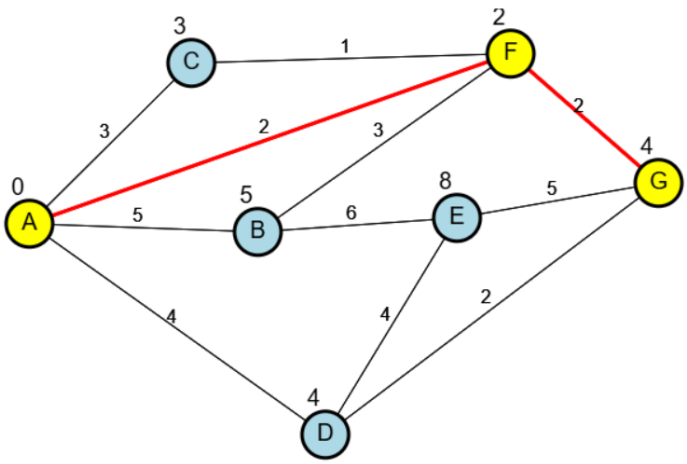
- 0 düğümünden başla, bir yığına ekle
- 0'ı ziyaret edildi olarak işaretle ve yola ekle.
- 0'ın komşularını düğüm numaralarına göre artan sırayla incele.
- İlk ziyaret edilmemiş komşuya özyinelemeli olarak git (veya yığına ekle) ve işlemi tekrarla.
- Ziyaret edilmemiş komşu kalmadığında geri dön.

Soru 2: Aşağıda verilen yönsüz çizge için A düğümünden G düğümüne olan en kısa yolu (a) Dijkstra algoritması ile çalışma mantığını anlatarak bulunuz. (b) Bellman Ford algoritması ile çalışma mantığını anlatarak bulunuz.

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 26.05.2025

GİRESUN ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
ALGORİTMALAR DERSİ FİNAL SINAVI





Dijkstra O((V + E) log V)

- C'ye olan mesafe güncelleniyor: 3
- D'ye olan mesafe güncelleniyor: 4
- B'ye olan mesafe güncelleniyor: 5
- F'ye olan mesafe güncelleniyor: 2
- G'ye olan mesafe güncelleniyor: 4
- E'ye olan mesafe güncelleniyor: 8
- G düğümüne olan en kısa yol: A -> F -> G, Mesafe: 4

Bellman-Ford O(V * E)

- A -> C kenarı gevşetiliyor: C'ye yeni mesafe = 3
- A -> D kenarı gevşetiliyor: D'ye yeni mesafe = 4
- A -> B kenarı gevşetiliyor: B'ye yeni mesafe = 5
- A -> F kenarı gevşetiliyor: F'ye yeni mesafe = 2
- B -> E kenarı gevşetiliyor: E'ye yeni mesafe = 11
- D -> E kenarı gevşetiliyor: E'ye yeni mesafe = 8
- D -> G kenarı gevşetiliyor: G'ye yeni mesafe = 6
- F -> G kenarı gevşetiliyor: G'ye yeni mesafe = 4
- G düğümüne olan en kısa yol: A -> F -> G, Mesafe: 4

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 26.05.2025

GİRESUN ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
ALGORİTMALAR DERSİ FİNAL SINAVI

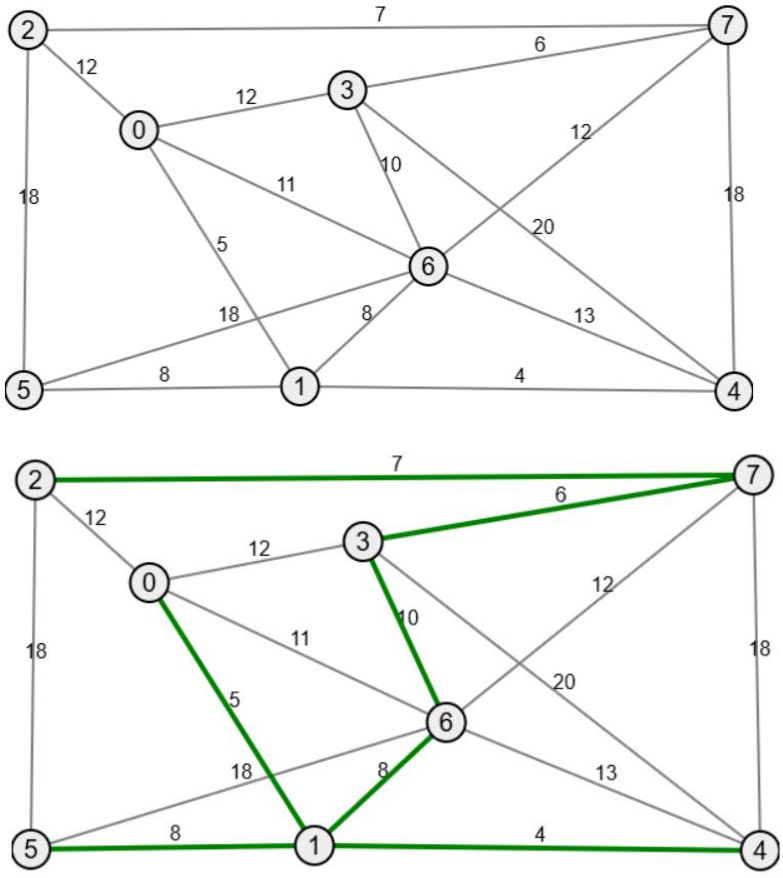
Dijkstra

- Mesafeleri başlat: mesafe[A] = 0, diğerleri = sonsuz.
- En küçük mesafeli düğümü seçmek için öncelik kuyruğu kullan.
- Seçilen düğüm için komşularına mesafeleri güncelle:
  - mesafe[komşu] = min(mesafe[komşu], mesafe[mevcut] + kenar_ağırlığı).
- Hedef (G) ulaşılana veya tüm düğümler işlenene kadar devam et.

Bellman-Ford

- Mesafe[A] = 0, diğerleri = sonsuz.
- Tüm kenarları V-1 kez gevşet: her kenar (u, v) için,
  - mesafe[v] = min(mesafe[v], mesafe[u] + kenar_ağırlığı).
- Negatif döngü kontrolü için bir kez daha gevşet.

Soru 3: Aşağıda verilen yönsüz çizgenin minimum kapsayan ağacını (MST) (a) Prim algoritması ile bulunuz. (b) Kruskal algoritması ile bulunuz. (c) İki algoritmanın çalışma mantığını karşılaştırınız. (d) İki algoritma hangi durumlarda aynı, hangi durumlarda farklı sonuç verir?



Prim O((V + E) log V)

- Herhangi bir düğümden (ör. 0) başla, MST'ye ekle.
- MST'yi ağaç dışındaki düğümlere bağlayan kenarları öncelik kuyruğunda tut.
- En düşük ağırlıklı kenarı seç, hedef düğümü MST'ye ekle, yeni kenarları kuyruğa ekle.

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 26.05.2025

- Tüm düğümler eklenene kadar devam et.

Kruskal Kenar sıralama için O(E log E), Union-Find için O(E α(V))

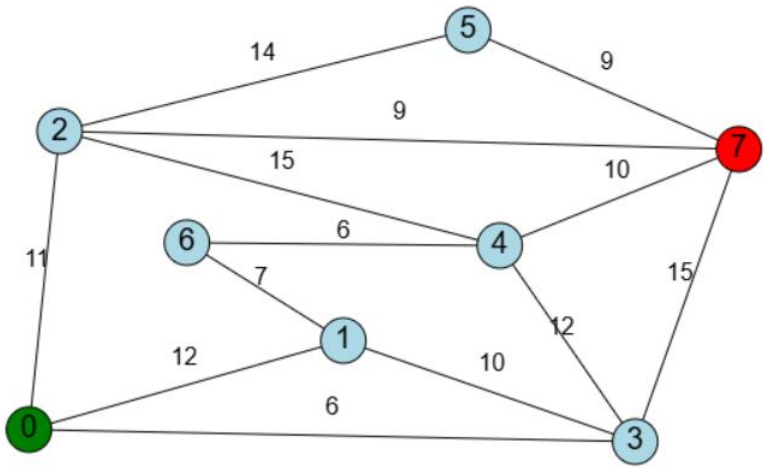
- Tüm kenarları artan ağırlığa göre sırala.
- Bağlı bileşenleri izlemek için Union-Find veri yapısı kullan.
- Her kenarı, farklı bileşenleri birleştiriyorsa MST'ye ekle (bileşenleri birleştir).
- V-1 kenar eklenince dur.

Prim: MST'yi tek bir düğümden büyütür, her seferinde bir düğüm ekler. Kruskal: Tüm kenarları ele alır, döngü oluşturmayan en küçük kenarı ekler.

Prim: Kenar seçimi için öncelik kuyruğu. Kruskal: Döngü tespiti için Union-Find.

Her iki algoritma, verilen çizge için aynı toplam ağırlığa sahip MST üretir. Eğer çizge tek bir MST'ye sahipse (tüm kenar ağırlıkları farklıysa), aynı ağacı üretirler. Birden fazla MST varsa (bazı kenar ağırlıkları eşitse), Prim ve Kruskal farklı kenarlar seçebilir, ancak toplam ağırlık aynı kalır. Örneğin, A-B ve C-D kenarları ağırlık 5 ise, Prim A-B'yi, Kruskal C-D'yi seçebilir.

Soru 4: Aşağıda verilen çizgede 0 numaralı düğümden 7 numaralı düğüme olan maksimum akışı (network flow) (a) ford-fulkerson algoritması ile bulunuz. (b) Edmond karp algoritması ile bulunuz. (c) İki algoritmayı kullandıkları yaklaşım açısından karşılaştırınız.

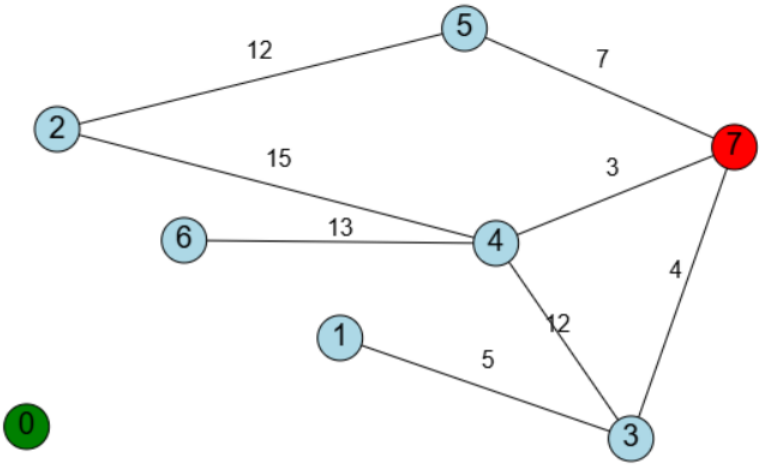


Ford-Fulkerson O(E * |f_max|)

- Artırıcı yol bulundu: 0-3, 3-7 kapasite 6
- Yol artırıldı. Mevcut maksimum akış: 6
- Artırıcı yol bulundu: 0-2, 2-7 kapasite 9
- Yol artırıldı. Mevcut maksimum akış: 15
- Artırıcı yol bulundu: 0-2, 2-5, 5-7 kapasite 2
- Yol artırıldı. Mevcut maksimum akış: 17
- Artırıcı yol bulundu: 0-1, 1-6, 6-4, 4-7 kapasite 6
- Yol artırıldı. Mevcut maksimum akış: 23
- Artırıcı yol bulundu: 0-1, 1-6, 6-4, 4-7 kapasite 1
- Yol artırıldı. Mevcut maksimum akış: 24
- Artırıcı yol bulundu: 0-1, 1-3, 3-7 kapasite 5

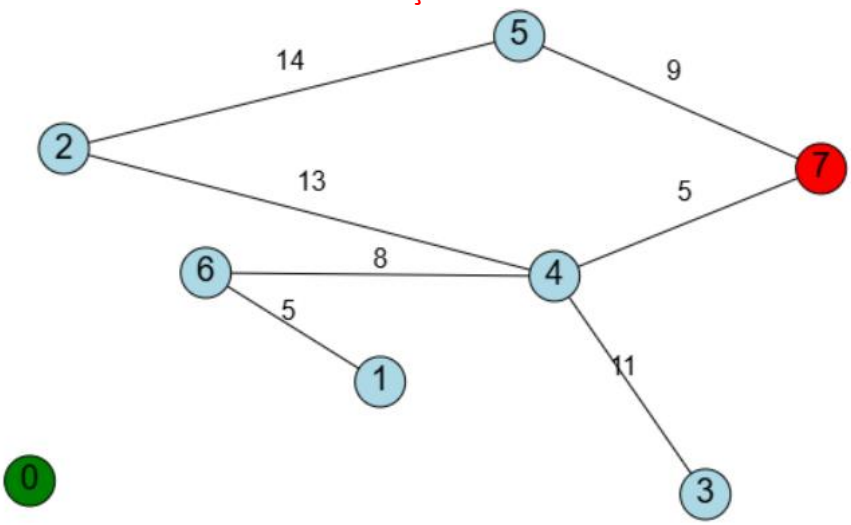
Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 26.05.2025

- Yol artırıldı. Mevcut maksimum akış: 29



Edmonds-Karp O(V * E^2)

- Artırıcı yol bulundu: 0-2, 2-7 kapasite 9
- Yol artırıldı. Mevcut maksimum akış: 9
- Artırıcı yol bulundu: 0-3, 3-7 kapasite 6
- Yol artırıldı. Mevcut maksimum akış: 15
- Artırıcı yol bulundu: 0-1, 1-3, 3-7 kapasite 9
- Yol artırıldı. Mevcut maksimum akış: 24
- Artırıcı yol bulundu: 0-2, 2-4, 4-7 kapasite 2
- Yol artırıldı. Mevcut maksimum akış: 26
- Artırıcı yol bulundu: 0-1, 1-3, 3-4, 4-7 kapasite 1
- Yol artırıldı. Mevcut maksimum akış: 27
- Artırıcı yol bulundu: 0-1, 1-6, 6-4, 4-7 kapasite 2
- Yol artırıldı. Mevcut maksimum akış: 29



Ford-Fulkerson: DFS kullanır, herhangi bir artırıcı yolu seçebilir, büyük kapasitelerde daha fazla iterasyon yapabilir. Edmonds-Karp: BFS kullanır, en az kenarlı yolu seçer, polinomik süreyi garanti eder.

Başarılar.. Dr. Öğr. Üyesi Sercan KÜLCÜ 26.05.2025