Veri Yapıları ve Algoritmalar Dönem Projesi

Graflarda Çokgen Tespiti

Öğrenci Adı: Mehmet Ali Duran

Öğrenci Numarası: 21011090

Dersin Eğitmeni: Mehmet Amaç Güvensan

Video Linki: https://youtu.be/8aDgCDphtoE

İçindekiler

1. Problem Tanımı

Bu projede, graflarda çokgen tespiti yapılması istenmektedir. N adet düğüm ve M adet kenardan oluşan yönsüz ağırlıklandırılmış bir graf adjacency list olarak verilmektedir. Bu graftaki bağlantıların oluşturduğu çokgenleri (üçgen, dörtgen, beşgen, altıgen vb.) bulup sahip olduğu düğümleri ve şekillerin her birinin çevre uzunluklarını yazdıran algoritmanın tasarlanması gerekmektedir. Tasarlanan algoritmanın, bulunan çokgenlerin sayısını, bu çokgenleri oluşturan düğümleri ve çokgenlerin çevre uzunluklarını ayrı ayrı ekrana basması istenmiştir.

2. Problemin Çözümü

İstenen problemin çözümü için, ilk olarak adjacency list şeklinde verilen grafın programda temsil edilmesi gerekmektedir. Bunun için struct yapısı kullanılmıştır. Bu struct, kenar bilgilerini tutmaktadır; kenarın ağırlığı ve bağlı olduğu düğümlerden biri karşılıklı tutulmamaktadır. Çünkü ileride kenarlar eklenirken hem kaynak hem de hedef listeye bu kenar eklendiği için zaten iki taraflı tutulmuş olmaktadır. Grafı tamamen oluşturabilmek için, bu kenarları tutan bir struct yapısı daha oluşturulmuştur. Bu yapıya "AdjList" adı verilmiş olup, bir "Edges" listesi ve boyutunu bilmek için bir "size" değişkeni içermektedir.

Ödev dosyasında düğüm ve kenar sayısının N ve M olarak kullanıcıdan alınması gerektiği belirtilmiştir. Ancak daha sonra format değişip dosyadan okunması istendiği için bu değerler alınmamıştır. İstenen değerler dosyadan okuma sırasında hesaplanmakta ve program içinde bu şekilde kullanılmaktadır. Graf elde edildikten sonra çokgenleri tespit etmek için DFS (Depth-First Search) yaklaşımı kullanılarak döngü olup olmadığı tespit edilmelidir. DFS algoritması şu şekilde özetlenebilir:

- Gezinti için her düğüm için bir "visited" dizisi tutulur.

- İlk başlatılan düğüm için bu değer 1 yapılır yani "ziyaret edildi" demektir ve burada bir başlangıç indisi tutulur.

- Daha sonra bu düğümün komşularına gidilir, eğer ziyaret edilmemişse rekürsif olarak bu düğüm için tekrar fonksiyon çağrısı yapılır.

- Ziyaret edilmiş ve başlangıç noktasından itibaren 3 döngü geçilmiş ise bir döngü bulunmuş demektir. Döngü bulunana kadar geçilen her kenardaki ağırlık bir "cost" değişkeninde tutulur.

- Çevre tespit edildikten sonra çokgeni oluşturan kenarlar bulunmalıdır. Bunun için bu döngünün yolu bir "path" listesine eklenir ve bu işlem bütün liste için yapılır.

- Bütün döngüler tespit edildikten sonra yazdırma aşamasına geçilir.

- Buradaki sorun aynı çokgenlerin birden fazla kez tespit edilmesi ve bunların hepsinin yazdırılmasıdır. Bunu önlemek için bir eşsiz döngü kontrolü yapılır.

- Eğer ele alınan çokgen daha önce yazdırılmadı ise ekrana basılır, muadili bir çokgen basıldıysa yazdırılmaz.

- Örnek olarak A, B, C, A düğümlerinden oluşan bir üçgen için B, C, A, B döngüsü aynı üçgeni ifade eder; bunlardan sadece bir tanesi yazdırılmalıdır.

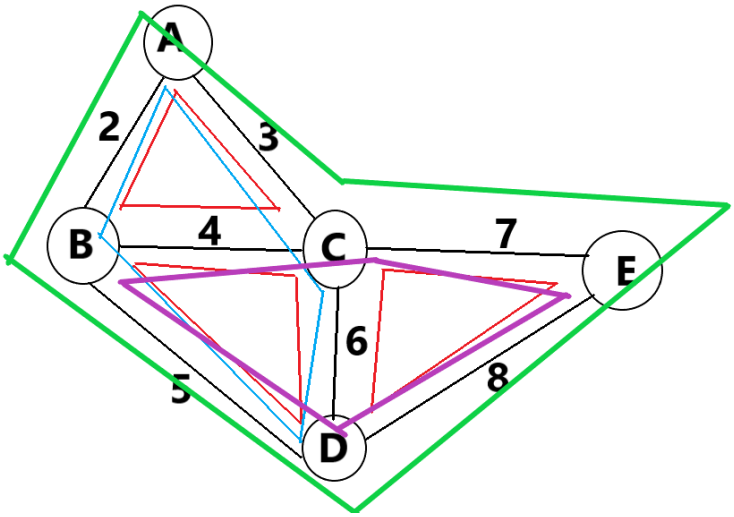
3. Karşılaşılan Sorunlar

Parent Node Gelince Çokgen Bulduğunu Zannetmesi: Rekürsif bir çağrı yapıldığı için ve yönsüz graflarla çalışıldığı için herhangi bir düğümü oluşturan listedeki düğümler de bu düğüm komşu olarak bulunduğu için DFS sırasında çokgen olduğunu zannedilmektedir. Bunu önlemek için bir uzunluk değişkeni tutulmuş ve 3'ten küçükse bulunan çevrim bir çokgen olmadığı tespit edilmiştir.

Aynı Çokgenlerin Tekrar Yazdırılması: Bu sorunu çözmek için bulunan döngüler "eşsiz mi" isimli fonksiyondan geçirilir. Eğer daha önce bulunan bir döngü değilse çokgenler listesine eklenir. Kontrol işlemi şu şekilde yapılır: Başlangıçtan itibaren gezilen düğümler bir "path" dizisinde tutulur. Şu anda gezilen düğümün komşularından olan yeni düğüm daha önce gezilen bir düğüm ise ve path'in başındaki ilk elemana eşitse çokgen tespit edilmiş demektir. Çokgen bulunduktan sonra daha önce bulunan çokgenlerin olduğu liste ile eleman eleman karşılaştırılır. Birebir aynı döngü var ise eklenmez.

4. Ekran Çıktıları:

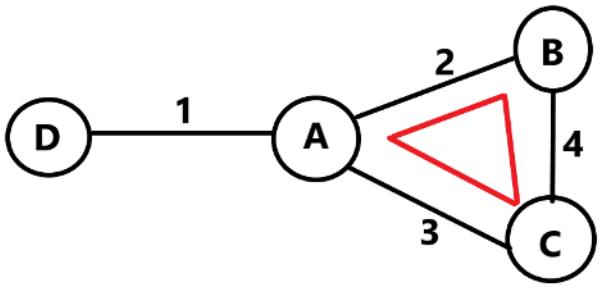
Verilen Sample.txt dosyası için nodelar yaklaşık olarak şu şekilde gözükmektedir.



Burada kırmızı çizgiler ile çizilen 3 adet üçgen, mavi ve mor çizgi ile çizilen 2 adet dörtgen ve yeşil çizgi ile çizilen 1 adet beşgen bulunmaktadır. Bu girdi için programın verdiği çıktı şu şekilde olmaktadır.

```
C:\Code Practices\Data Struct
3'gen sayisi: 3
1. 3'gen: A B C A cevresi:  9
2. 3'gen: B C D B cevresi:  15
3. 3'gen: C D E C cevresi:  21

4'gen sayisi: 2
1. 4'gen: B C E D B cevresi:  24
2. 4'gen: A B D C A cevresi:  16

5'gen sayisi: 1
1. 5'gen: A B D E C A cevresi:  25

Toplam yazdirilan cokgen sayisi: 6

--------------------------------
Process exited after 1.076 seconds with return value 0
Press any key to continue . . .
```

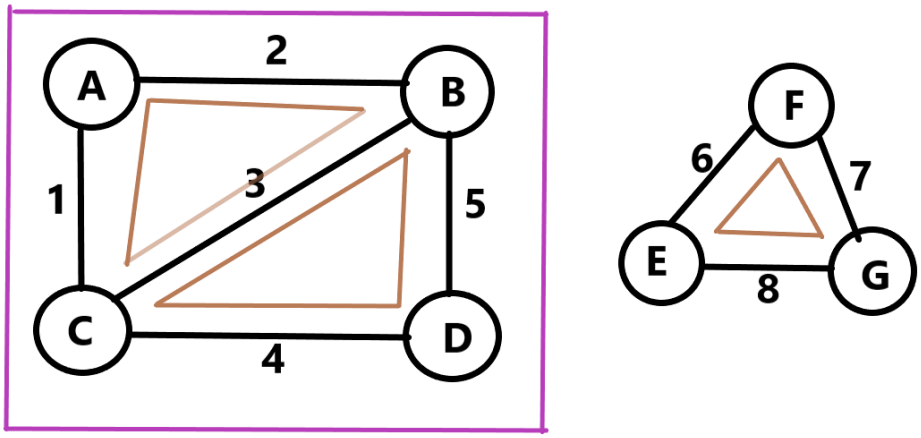
İkinci örnek olan Sample2.txt dosyası için nodelar yaklaşık olarak şöyle gözükmektedir:



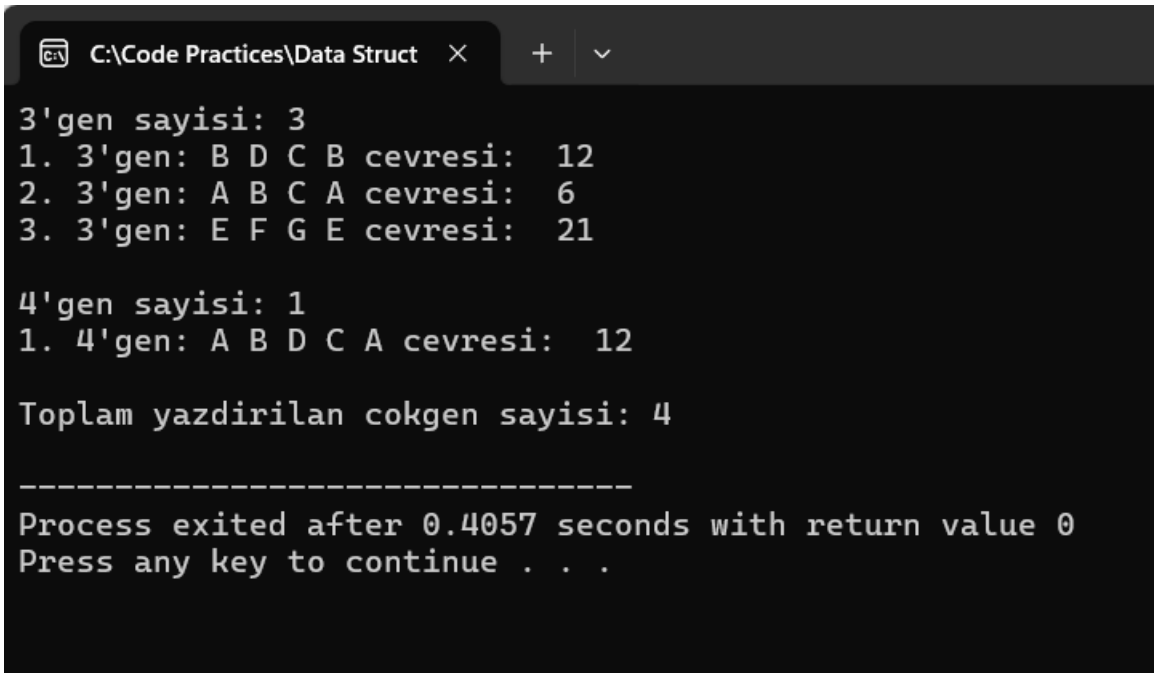
Programın çıktısı ise şu şekilde olmaktadır:

```
C:\Code Practices\Data Struct
3'gen sayisi: 1
1. 3'gen: A C B A cevresi:  9

Toplam yazdirilan cokgen sayisi: 1

--------------------------------
Process exited after 0.6497 seconds with return value 0
Press any key to continue . . .
```

3. örnek Sample3.txt yaklaşık olarak şu şekilde gözükmektedir:

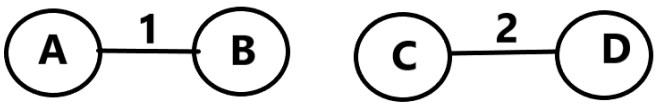


Bu girdiler için program çıktısı şu şekilde olmaktadır:



Görüldüğü gibi birbirinden bağımsız şekiller içinde doğru sonuç vermektedir. Bunu yapabilmesinin sebebi bütün düğümler için DFS algoritması çalıştırıldığı için bağlantısız graflar olsa bile diğer grafa geçebilmektedir.

4. ve son örnek olarak Sample4.txt dosyası yaklaşık olarak şöyle gözükmektedir:



Bu girdiler için çıktı ise şu şekilde olmaktadır: