Graph Yapılarında Çokgen Analizi

Öğrenci Adı: Burak Atalay

Öğrenci Numarası: 22011641

Dersin Eğitmeni: Göksel Biricik

Video Linki: https://youtu.be/0HGNzI8rMgo

1- Problem Tanımı:

Kenar bilgileri dosyadan okunan grafın içindeki tüm çokgenlerin bulunup yazdırılması ve bunun için DFS algoritmasından faydalanılması isteniyor.

2- Problemin Çözümü:

Kullanıcıdan graftaki node sayısı isteniyor. Bu işlem ile graph structındaki adjList için allocation işlemi gerçekleştiriliyor. Ardından kenarların bulunduğu dosya adı istenip okuma işlemi gerçekleştiriliyor ve kenarlar graftaki komşuluk listesine ekleniyor. Sonrasında main'de findAllCycles fonksiyonu çağrılıyor. Bu fonksiyonda DFS fonksiyonu sırayla her bir node için çağrılıyor. Buradaki amaç önce ilk node'un dahil olduğu tüm şekilleri bulmak, sonrasında ilk node için olan çağrı bitince sonraki çağrılarda ilk node için bulunan şekilleri tekrardan bulmamak için isVisited dizisinde ilk node işaretleniyor. İkinci node için yapılan çağrılarda, ikinci node'un dahil olup ilk node'un dahil olmadığı şekillerin hepsi bulunuyor. Bu şekilde devam edip tüm şekiller bulunuyor. DFS çağrılarının içerisinde ise şunlar gerçekleşiyor: önce kendisi için DFS çağrılan node, bulunabilecek şekillerin yollarının tutulduğu stack'e atılıyor. isVisited dizisinde ise bu node işaretleniyor ki kendisinin sebep olacağı çocuk çağrılardan tekrar kendisi çağrılmasın. Aslında bu bir cycle, yani şekil bulunmuş olduğunu gösterir ancak implemente edilen DFS'de cycle bulma ancak target node'a denk gelinirse gerçekleşir. target ise findAllCycles'ta kendisi için DFS çağrılan node olarak DFS'ye parametre olarak verilir. Bu şekilde yalnızca target ile başlayıp target ile biten şekiller bulunur. DFS'de komşuları gezerken ise eğer komşu, komşuları gezilen node'un parent'ı değil ve target ise şekil bulunmuş olur. İki node'a sahip bir şekil bulmayı engellemek için parent parametre olarak alınıyor. Ve stack'teki path, isDuplicateCycle fonksiyonuna verilir. Bu fonksiyona olan ihtiyaç şu sebeple oluştu: graf yönsüz olduğundan bir şekil hem saat yönünde hem de saat yönün tersine şeklinde iki kez bulunacak. Bu sebeple yeni bir şekil bulunduğunda, bulunan şekil daha önceden bulunanlardan birinin tersi yönünde mi diye bu fonksiyon yardımıyla kontrol edilir. Eğer duplicate değilse şekillerin tutulduğu listeye bu şekil eklenir. Şekiller ise graf için tutulan komşuluk listesine benzer mantıkta tutulur. POLYGON structında şeklin yolu, uzunluğu ve kaçgen olduğu tutulur. POLYGONS structında ise polygonsList şekline adjList mantığında bir liste tutulur. Bu listenin üçüncü indeksinde üçgenler, dördüncüde dörtgenler vs tutulur. Bu hem çıktı olarak sıralı şekilde önce üçgenler sonra dörtgenler şeklinde istendiğinden sıralama maliyetinden kurtulmak için hem de isDuplicateCycle'a yeni bulunan şekil kaçgen ise sadece onların gönderilmesi (polygons->polygonsList[stack->top]) için yapıldı. Listeye şekil ekleme ise adjList'te yeni node head'e eklenirken bu şekiller için alfabetik sırayı bozacağından tail'e ekleme yapılır. Bunun içinde polygons'ta her bir listenin tail'leri tutulur. Her bir şeklin sayısı için ise counter dizisi tutulur. Eğer ki komşular gezilirken bu komşu target değilse ve isVisited'ı 1 değilse bu komşu için DFS çağrılır. Şeklin çevre uzunluğu ise bu iki koşula da girersek komşu ve parent arasındaki kenarın ağırlığı fonksiyona adresiyle verilen totalWeight değişkenine eklenerek halledilir. Çağrıdan sonra veya şekil listeye eklendikten sonra bu kenar ağırlığı totalWeightten çıkarılır çünkü bu komşu da ileride bulunabilecek şekiller için yolun tutulduğu stack'ten çıkarılır. Tüm koşular gezildikten sonra komşuları gezilen node stack'ten çıkarılır ve isVisited2ı 0 yapılır çünkü target'ın dahil olduğu ve bu node'a ulaşılan farklı bir yola sahip bir şekil var olabilir. Mesela A-B-C-D-A yoluna sahip bir şekil bulundu ve şu anda C için yapılan DFS içerisinde bulunuluyor. Eğer ki C'nin isVisited'ı DFS sonunda sıfırlanmaz ise, A ile C arasındaki muhtemel bir komşuluk ihtimali için A'dan C çağırılamaz ve A-C-D-A yoluna sahip muhtemel bir şekil bulunamamış olur. Bu şekilde her node için findAllCycles fonksiyonunda DFS çağrısı yapılır. Fonksiyon bitiminde

main'de tüm şekillerin listesine sahip polygons değişkeni printPolygons fonksiyonuna verilir ve şekiller istenen output biçiminde yazdırılır.

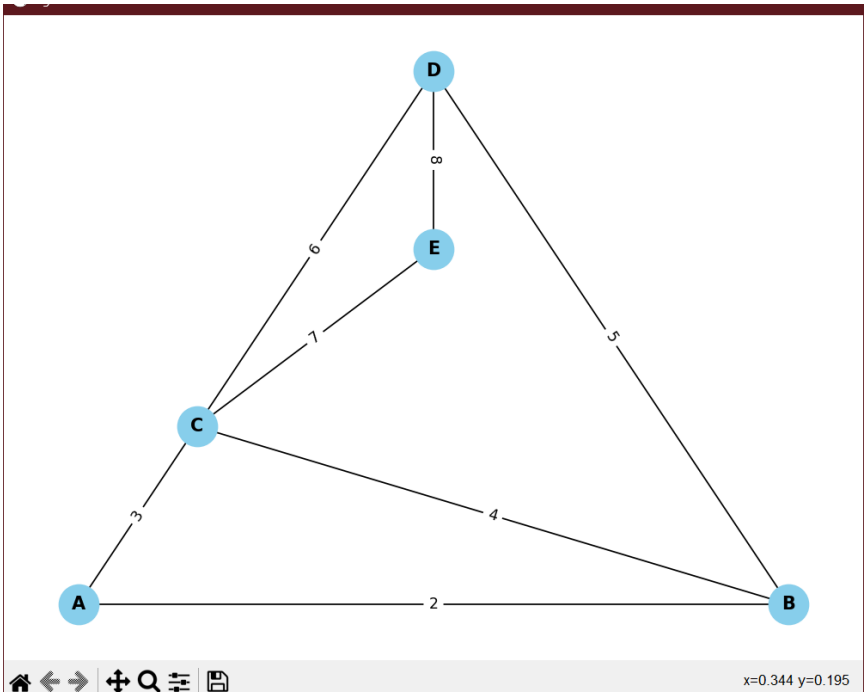
3- Karşılaşılan Sorunlar:

Programın ilk aşamasında bir şekil yeniden bulunduğunda her zaman tam anlamıyla ilk bulunanın tersi yönde olacağını farkedememiştim. Bu sebeple isDuplicateCycle fonksiyonu karmaşık bir hal almıştı. Sonrasında tersi dışında farklı bir şekil bulamayacağını farkettiğimde sadece aynı büyüklüğe sahip şekiller arasında tersi var mı yok mu diye kontrol edilecek şekilde değiştirdim. Ayrıca şekilleri önce üçgenler sonra dörtgenler şeklinde yazdırma kısmından classroom duyurusu ile haberim oldu. Bunu duyunca sıralama maliyetinden kurtulmak için graftaki komşuluk listesi mantığına benzer şekilde şekilleri tutmaya başladım.

4- Ekran Çıktıları:

Senaryo 1-

N = 5
A B 2
A C 3
B C 4
B D 5
C D 6
C E 7
D E 8



```
the number of nodes in the graph: 5
enter the filename: sample.txt

sekil sayisi: 6
3'gen sayisi: 3
4'gen sayisi: 2
5'gen sayisi: 1

3'gen: A B C A  uzunluk: 9
3'gen: B C D B  uzunluk: 15
3'gen: C D E C  uzunluk: 21

4'gen: A B D C A  uzunluk: 16
4'gen: B C E D B  uzunluk: 24

5'gen: A B D E C A  uzunluk: 25


--------------------------------
Process exited after 8.049 seconds with return value 0
Press any key to continue . . .
```

Senaryo 2-

N = 7

A B 2

D B 5

E F 6

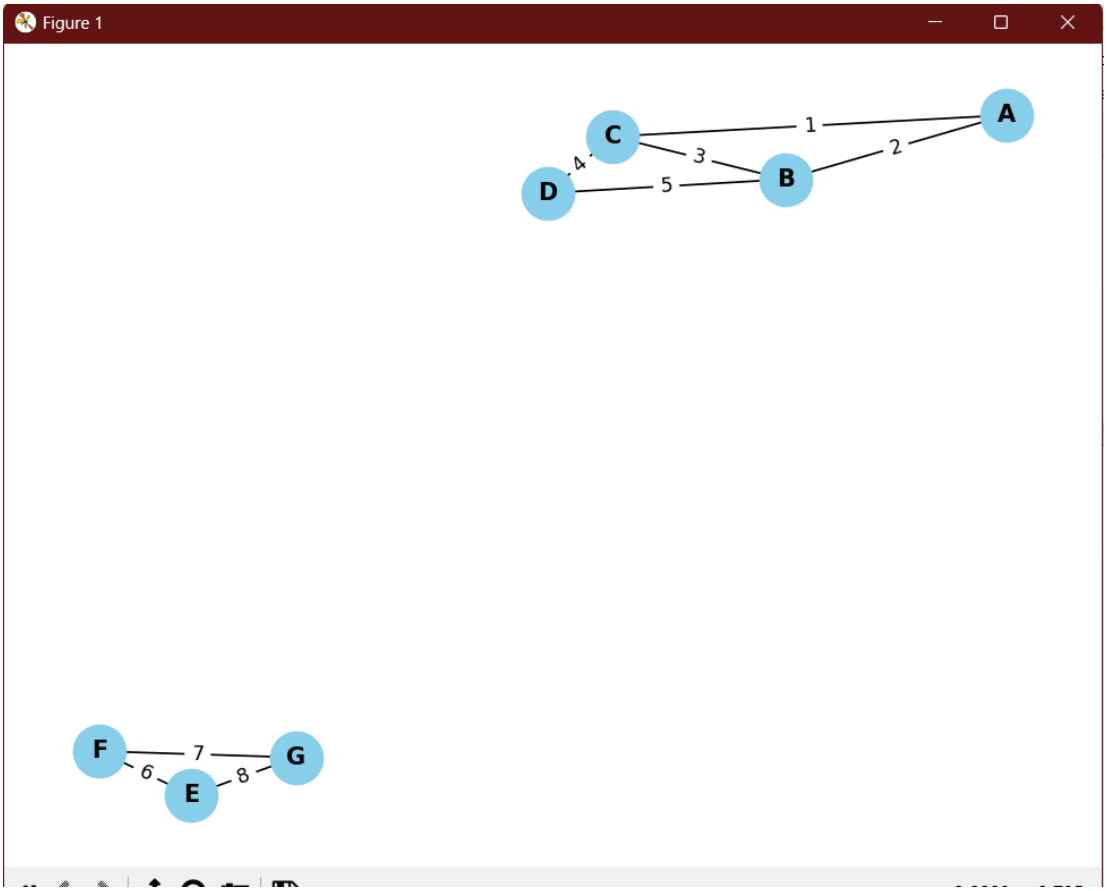
G F 7

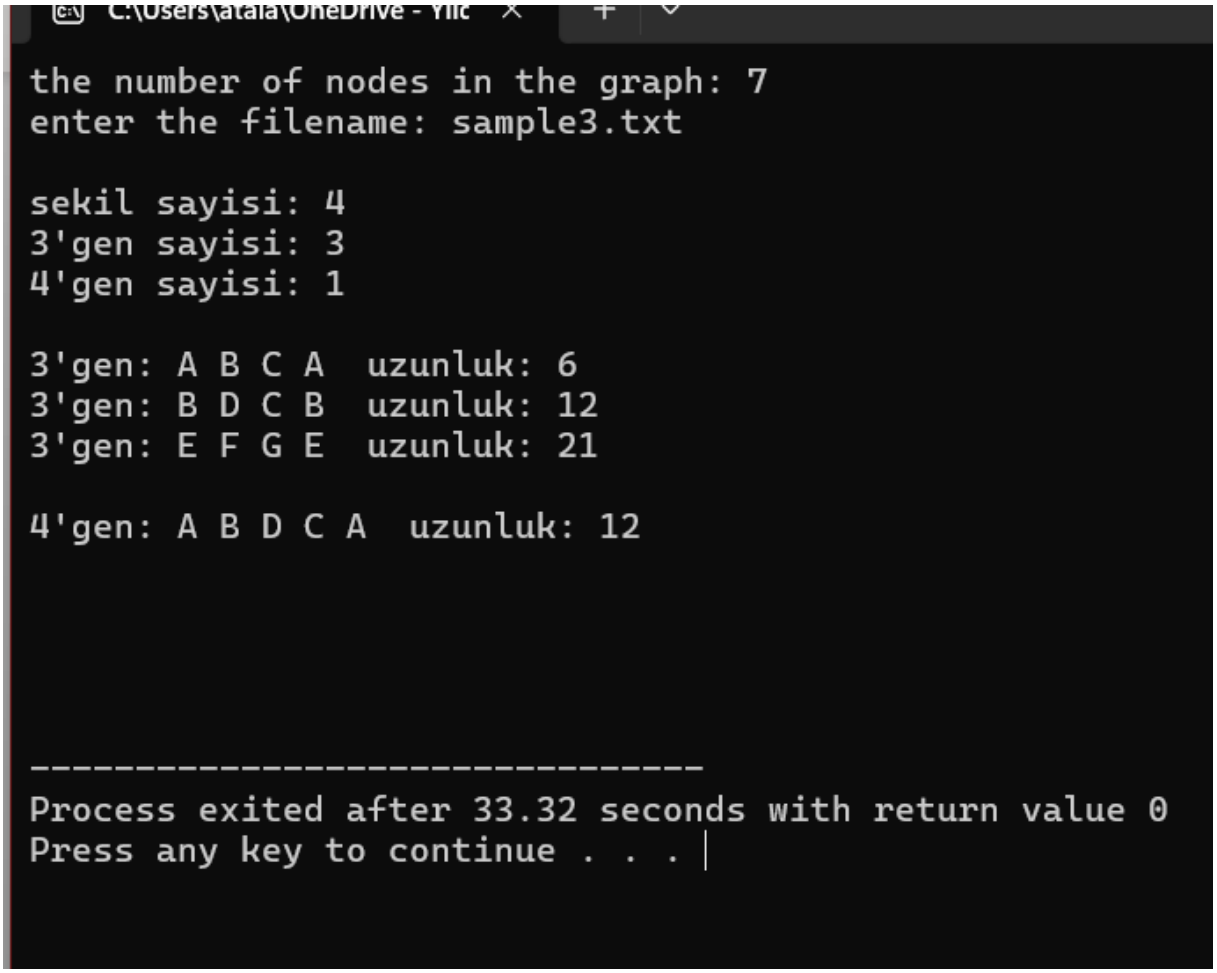
E G 8

C A 1

C B 3

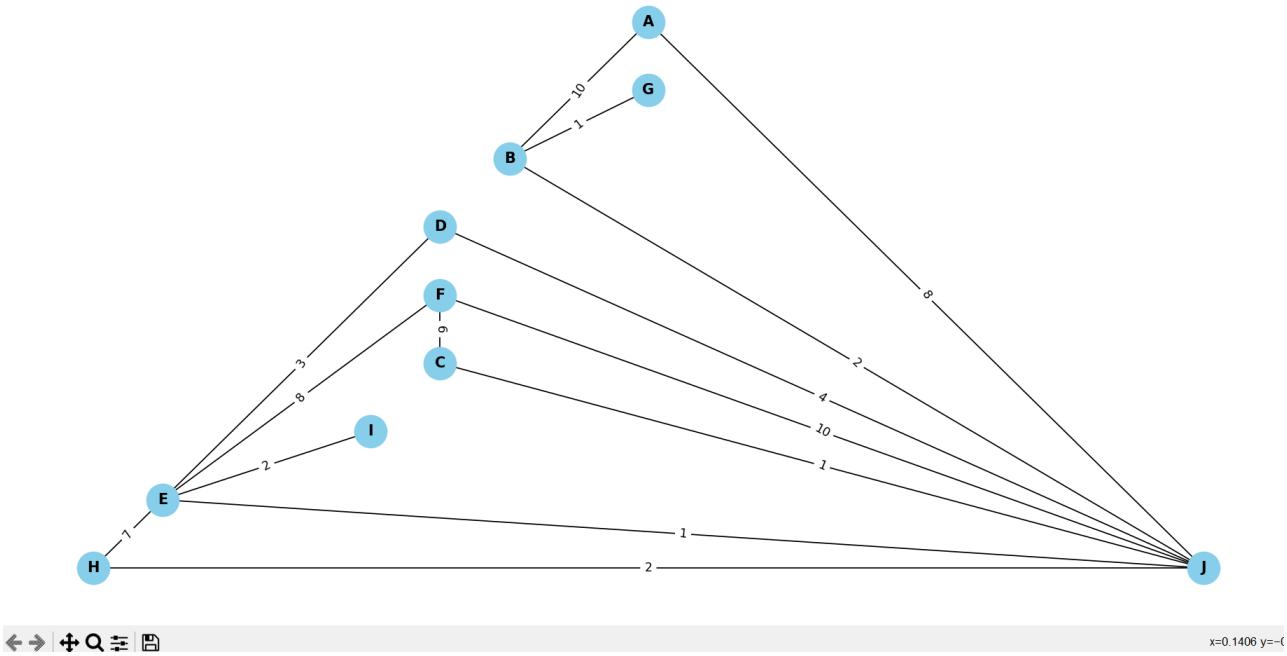
C D 4

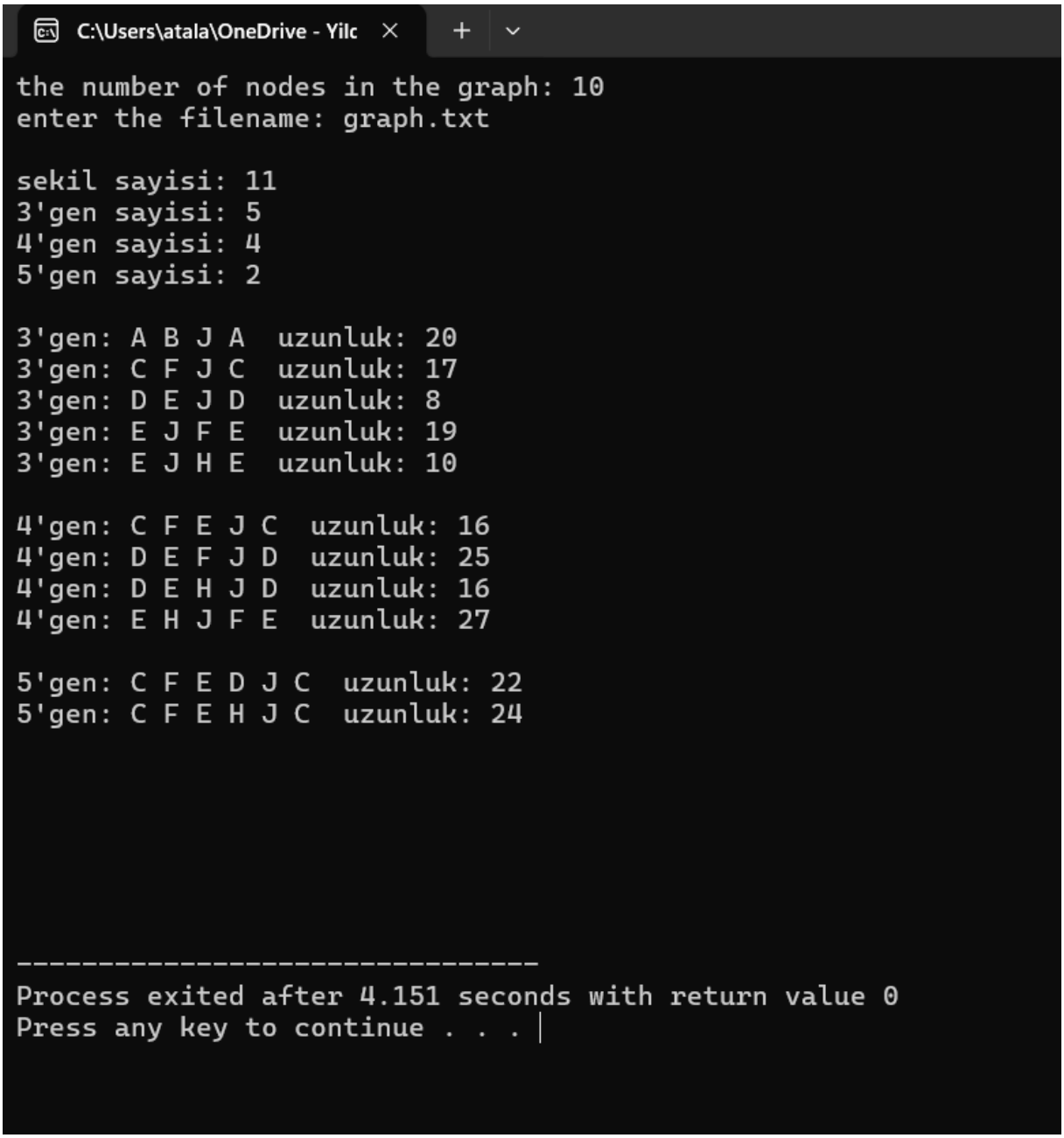



C:\Users\atala\OneDrive - Yıld
the number of nodes in the graph: 7
enter the filename: sample3.txt
sekil sayisi: 4
3'gen sayisi: 3
4'gen sayisi: 1
3'gen: A B C A uzunluk: 6
3'gen: B D C B uzunluk: 12
3'gen: E F G E uzunluk: 21
4'gen: A B D C A uzunluk: 12
Process exited after 33.32 seconds with return value 0
Press any key to continue . . .

Senaryo 3-

N = 10
H J 2
A B 10
E J 1
J F 10
E I 9
E H 7
E D 3
G B 1
F E 8
B J 2
A J 8
D J 4
I E 2
C F 6
C J 1




C:\Users\atala\OneDrive - Yilc
the number of nodes in the graph: 10
enter the filename: graph.txt
sekil sayisi: 11
3'gen sayisi: 5
4'gen sayisi: 4
5'gen sayisi: 2
3'gen: A B J A uzunluk: 20
3'gen: C F J C uzunluk: 17
3'gen: D E J D uzunluk: 8
3'gen: E J F E uzunluk: 19
3'gen: E J H E uzunluk: 10
4'gen: C F E J C uzunluk: 16
4'gen: D E F J D uzunluk: 25
4'gen: D E H J D uzunluk: 16
4'gen: E H J F E uzunluk: 27
5'gen: C F E D J C uzunluk: 22
5'gen: C F E H J C uzunluk: 24
Process exited after 4.151 seconds with return value 0
Press any key to continue . . .

Senaryo 4-

N = 9

A B 4

A H 8

B C 8

B H 11

C D 7

C F 4

C I 2

D E 9

D F 14

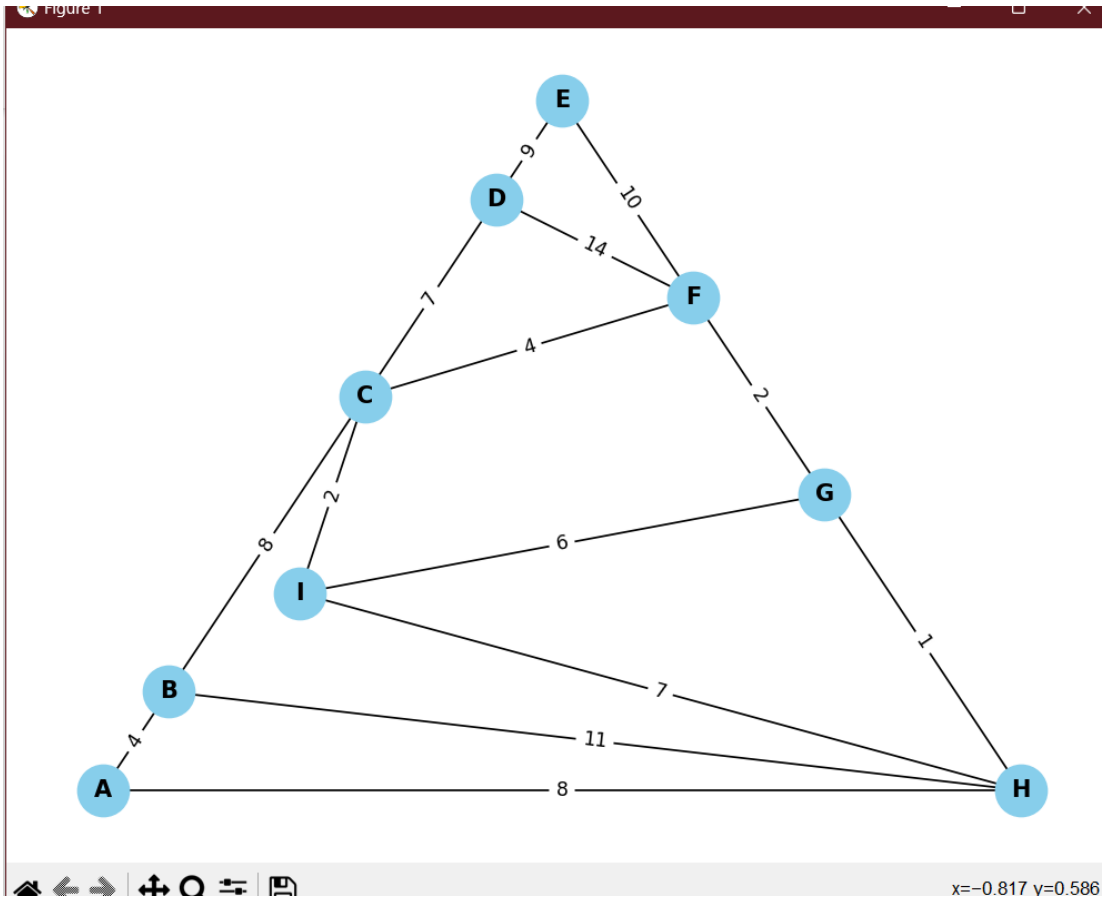
E F 10

F G 2

G I 6

G H 1

H I 7



```
the number of nodes in the graph: 9
enter the filename: sample6.txt

sekil sayisi: 27
3'gen sayisi: 4
4'gen sayisi: 3
5'gen sayisi: 5
6'gen sayisi: 6
7'gen sayisi: 5
8'gen sayisi: 3
9'gen sayisi: 1

3'gen: A B H A  uzunluk: 23
3'gen: C D F C  uzunluk: 25
3'gen: D E F D  uzunluk: 33
3'gen: G I H G  uzunluk: 14

4'gen: B C I H B  uzunluk: 28
4'gen: C F G I C  uzunluk: 14
4'gen: C D E F C  uzunluk: 30

5'gen: A B C I H A  uzunluk: 29
5'gen: B C I G H B  uzunluk: 28
5'gen: B C F G H B  uzunluk: 26
5'gen: C F G H I C  uzunluk: 16
5'gen: C D F G I C  uzunluk: 31

6'gen: A B C I G H A  uzunluk: 29
6'gen: A B C F G H A  uzunluk: 27
6'gen: B C F G I H B  uzunluk: 38
6'gen: B C D F G H B  uzunluk: 43
6'gen: C D F G H I C  uzunluk: 33
6'gen: C D E F G I C  uzunluk: 36

7'gen: A B C F G I H A  uzunluk: 39
7'gen: A B C D F G H A  uzunluk: 44
7'gen: B C D F G I H B  uzunluk: 55
7'gen: B C D E F G H B  uzunluk: 48
7'gen: C D E F G H I C  uzunluk: 38

8'gen: A B C D F G I H A  uzunluk: 56
8'gen: A B C D E F G H A  uzunluk: 49
8'gen: B C D E F G I H B  uzunluk: 60

9'gen: A B C D E F G I H A  uzunluk: 61


--------------------------------
Process exited after 4.65 seconds with return value 0
Press any key to continue . . .
```