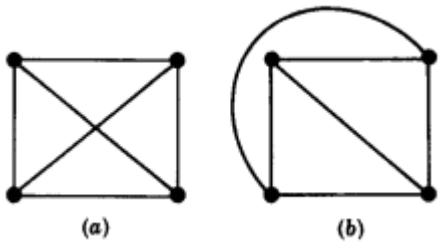
Planar Graflar

Düz bir zemine çizilebilen ve kenarları birbirini kesmeyen bir graf ya da çoklu graf planar bir graftır. Dört düğümlü bir komple bağlı genellikle kenarları kesişen aşağıdaki şekil (a)'da gözüktüğü gibi çizilmesine rağmen, bu graf şekil(b)'de gözüktüğü gibi kenarları kesişmeden de çizilebilir. Böyle bir graf planar graf olarak adlandırılır. Üç graf, planar gafların önemli bir sınıfını oluşturur. Bu bölümde bu üç önemli planar graf tanıtılacaktır.



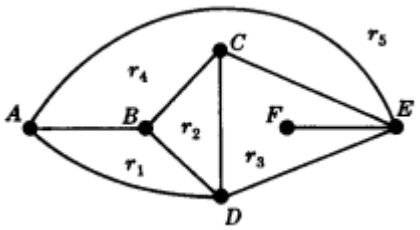
HARİTALAR BÖLGELER

Sınırlı bir düzlemsel çoklu grafın belirli bir düzlemde temsili harita olarak adlandırılır. Eğer birçoklu graf bağlı ise (connected) harita da bağlı olduğunu söyleriz. Verilen bir harita düzlemi çeşitli bölgelere böler. Örneğin. Aşağıdaki şekilde altı köşe ve dokuz kenar düzlemi beş bölgeye böler. Bölgelerin dört tanesi çevreli olduğuna dikkat edin, fakat beşinci bölge diyagramın dışında olup çevrelenmemiştir. Eğer haritamız tüm düzlemden ziyade bazı büyük dikdörtgenler ihtiva eder (kapsarsa) , böylece bölgelerin miktarının sayısında genel olarak kayıp olmaz.

Bir haritanın her bir bölgesinin sınırları kenarlardan oluştuğuna dikkat edin. Bazen kenarlar döngü oluşturur, fakat bazen de oluşturmaz. Örneğin: aşağıda verilen şekilde, r3 dışındaki tüm bölgelerin sınırları döngü oluşturur. Fakat, eğer r3 bölgesinde C köşesinden başlayarak saat yönünde hareket edersek aşağıdaki kapalı yolu elde ederiz.

(C, D, E, F , E, C)

Burada {E,F} kenarı iki kez oluşur. Bir r bölgesinin derecesi deg(r) olarak yazılır ve döngünün boyu ya da kapalı olan yürüme sınırlarını ifade eder/anlamına gelir. Her kenar ya iki bölgeyi sınırlar ya da bir bölgede ihtiva edilir (içinde yer alır) ve bölgenin sınırları boyunca herhangi bir yürüyüşte iki kez oluşur. Böylece daha önce düğümler için verilen teoremimizi analoji yaparak bölgeler için bir teoreme dönüştürebiliriz.



Teorem: Bir haritanın bölgelerinin derecelerinin toplamı kenar sayılarının iki katına eşittir. Yukarıda verilen şekildeki bölgelerin dereceleri:

deg(r1) = 3, deg(r2) = 3, deg(r3) = 5, deg(r4) = 4, deg(r5) = 3 dir.

Derecelerin toplamı 18 dir ve bu değer kenar sayılarının iki katına eşittir.

Notasyonel gösterim için bir haritanın düğümleri nokta ya da daire ile resmedilir ya da düzlemdeki hatların ya da çizgilerin her kesişim yerlerini düğüm olarak kabul ederiz.

Eulear Formulu

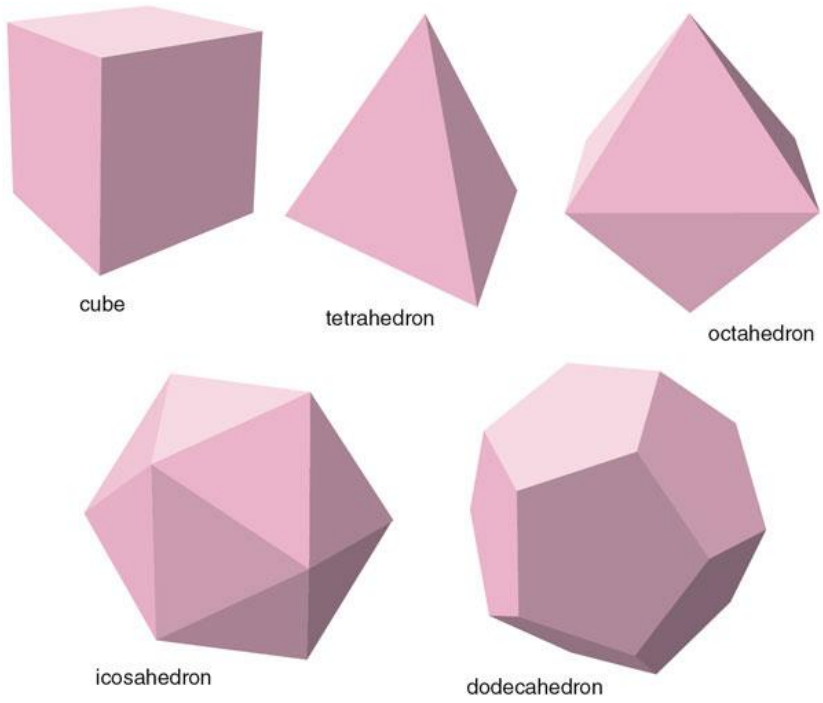
Herhangi bir G grafında V düğüm, E kenar ve R de bölge sayısını göstermek üzere Eular formülü aşağıdaki gibi yazılır.

V − E + R = 2

Yukarıdaki şekli göz önüne aldığınızda V=6, E=9 ve R=5 olmak üzere Eular formülü daima aşağıdaki sonucu sağlar.

V − E + R = 6 − 9 + 5 = 2

Eulear Formulu Uygulama Örneği



Dodecahedron; oniki yüzlü, 12 yüzü olan normal bir katıdır. Her yüzü bir düzenli beşgenden oluşur. Bu oniki yüzlü beşgende kaç tane kenar ve kaç tane köşe vardır? **(İpucu: kenarları saymak için kombinatorik ve köşeleri saymak için Euler formülünü kullanın)**

ÇÖZÜM:

e, v, kenarların ve köşelerin sayısını göstersin. Her kenar iki beşgen yüz tarafından paylaşılır ve her yüzün 5 kenarı vardır. Böylece, her yüz için kenar sayısını toplarsak, toplam $5 \cdot 12 = 60$ elde ederiz. Ancak bu toplamda, her kenar iki kez sayıldı (fazla sayma), yani $2e = 60$ ve dolayısıyla $e = 30$.

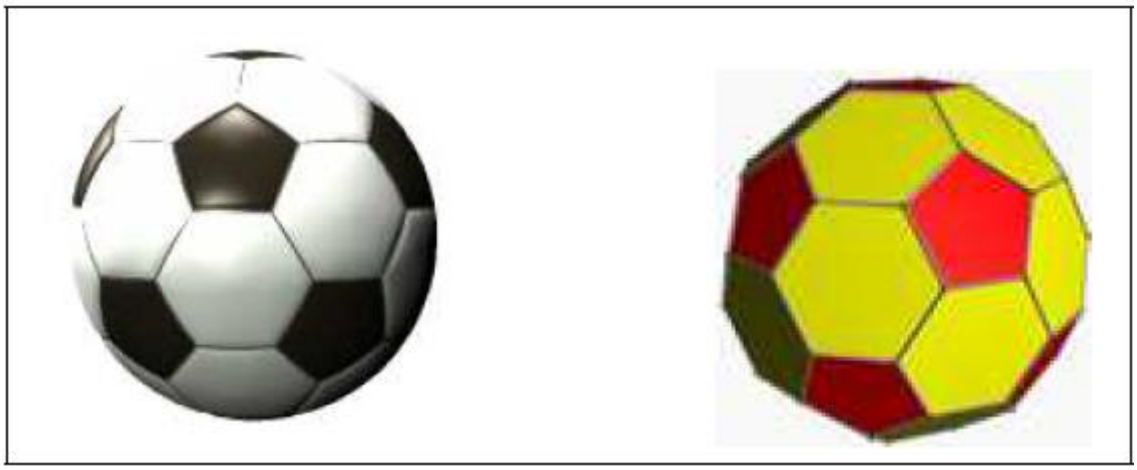
Şimdi Euler formülü bize şunu söylüyor:

$v - e + R = 2$.

$E = 30$ ve $R = 12$ olduğundan, $v = 20$ olduğu sonucuna varıyoruz.

ÖRNEK:

Bir futbol topunun şekli, adı verilen bir çokyüzlü ile temsil edilebilir. yüzleri 12 normal beşgen olan kesik dodecahedron ve aynı kenar uzunluğuna sahip 20 normal altıgen. Her beşgen 5 altıgen ile çevrili, oysa her altıgen 3 beşgen ve 3 altıgen ile çevrilidir (bkz. Şekil 1).



Şekil 1: Bir futbol topu ve karşılık gelen çokyüzlü parçalar (polyhedron).

Bu futbol topunda kaç kenar ve kaç köşe vardır? (**İpucu**: kenarları saymak için kombinatorik ve köşeleri saymak için Euler formülünü kullanın)

Çözüm.

Her bir taraf, iki altıgen veya bir altıgen ve bir beşgen. Her biri 5 kenarlı 12 beşgen ve her biri 6 kenarlı 20 altıgen vardır: bu nedenle, her yüzün kenarlarının sayısını toplarsak, toplam

$12 \cdot 5 + 20 \cdot 6 = 180$

Kenar var. Ancak her kenar iki kez sayılır (fazla sayılır), bu nedenle gerçek kenar sayısı e = 180/2 = 90'dır. Şimdi, Euler formülünü kullanarak düğüm (köşe) sayısını bulalım.

v - e + R = 2.

E = 90 ve R = 12 + 20 = 32 olduğundan, v = 60 köşe olduğu sonucuna varıyoruz.

GRAF RENKLENDİRME

Bir G grafını göz önüne alalım. Düğüm renklendirme ya da kısaca Graf renklendirme, G grafının komşu olan düğümlerine farklı renk atama işlemi olarak bilinir. Eğer n renkle renklendirilebilir bir G grafı var ise bu G grafına, n-renklenebilir (renkli) bir graf denir. Minimum sayıda renkle düğümleri renklendirmek istenir/gerekir, bu sayıya G grafının chromatic sayısı (değeri) denir. Graflarda düğümleri renklendirmek için geliştirilmiş algoritmalar var. Bunlardan en bilineni Welch and Powell algoritmasıdır. Bu algoritma her zaman minimum chromatic değerini vermeyebilir.

Algoritma (Welch-Powell): G grafını girdi olarak al.

Adım 1. G grafının düğümlerinin derecelerini azalacak biçimde sırala

Adım 2. İlk düğüme birinci rengi (C1) ata ve sonra bir önceki düğüme bitişik olmayan sıradaki düğümlere aynı rengi (C1) ata.

Adım 3. İkinci adımı, ikinci bir renk ile (C2) renk ataması yapılmamış düğümlere sırasıyla tekrar et

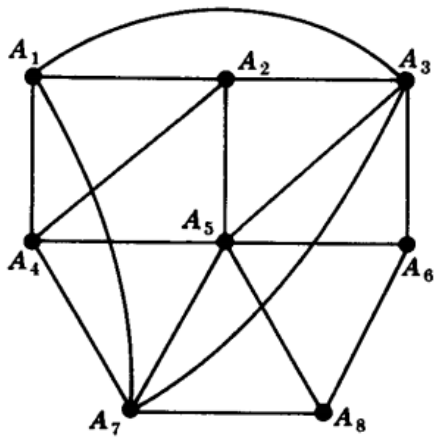
Adım 4. Üçüncü adımı, üçüncü bir renk ile (C3), daha sonra dördüncü bir renk ile (C4) tüm düğümler renklenene kadar tekrar et.

Adım 5. Çıkış.

ÖRNEK:

Aşağıdaki şekilde verilen G grafını göz önüne alın. Welch-Powell algoritmasını kullanan G grafını renklendiriniz. Grafın düğüm derecelerini azalan sırada sıralanması ile aşağıdaki sıra elde edilir.

A5, A3, A7, A1, A2, A4, A6, A8



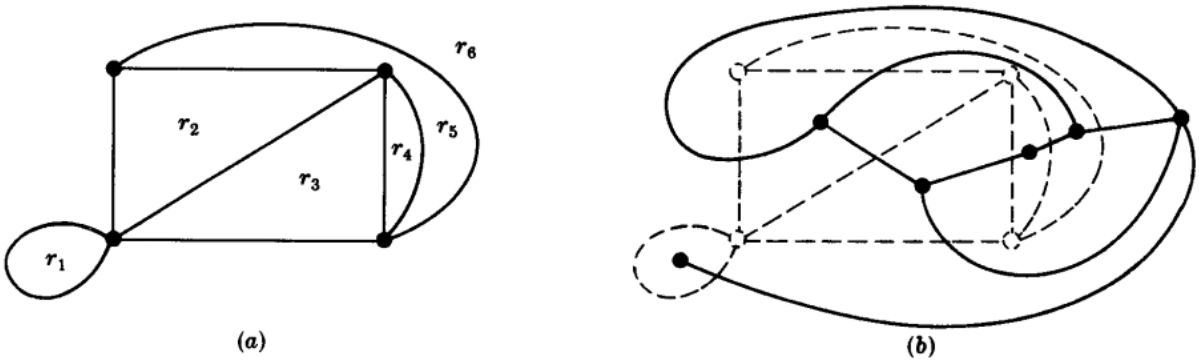
İlk renk düğüm A5 ve A1'e atanır. İkinci renk düğüm A3, A4 ve A8'e atanır. Üçüncü renk düğüm A7, A2 ve A6'ya atanır. Her düğüme bir renk atanmış oldu ve G grafının renklenebilir (colorable) değeri $\chi(G) = 3$ olarak elde edilir.

Dual Maps and the Four Color Theorem (Eş Ağlar ve Dört Renk Teoremi)

Aşağıdaki şekildeki M haritasını göz önüne alalım. Başka bir deyişle, M bir düzlemsel çoklu grafın güzlemsel temsili olan bir M haritasıdır. M'nin iki bölgesi eğer iki ortak bir kenara sahip ise bu iki

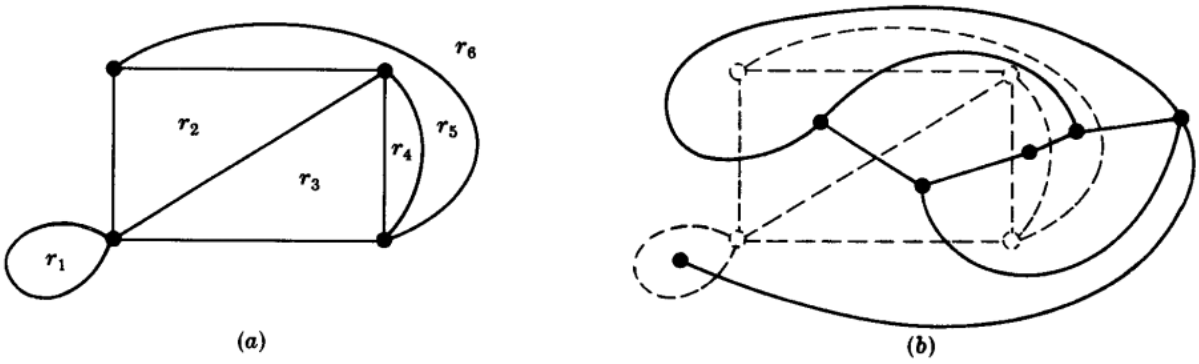
bölge bitişiktir/komşudur denir. Bu yüzden, aşağıdaki şekildeki r2 ve r5 bitişiktir fakat r3 ve r5 bitişik değildir. M yi renklendirmek, M'nin her bir bölgesine bir renk atanması demektir öyleki bitişik olan bölgeler farklı renkli olur. n-renkle renklendirilebilen haritaya n-colorable harita denir. Böylece aşağıdaki şekildeki harita 3-colorable bir haritadır çünkü bölgelere aşağıdaki renkler atanabilir.

r1=red, r2=white, r3 =red, r4 =white, r5 =red, r6 =blue



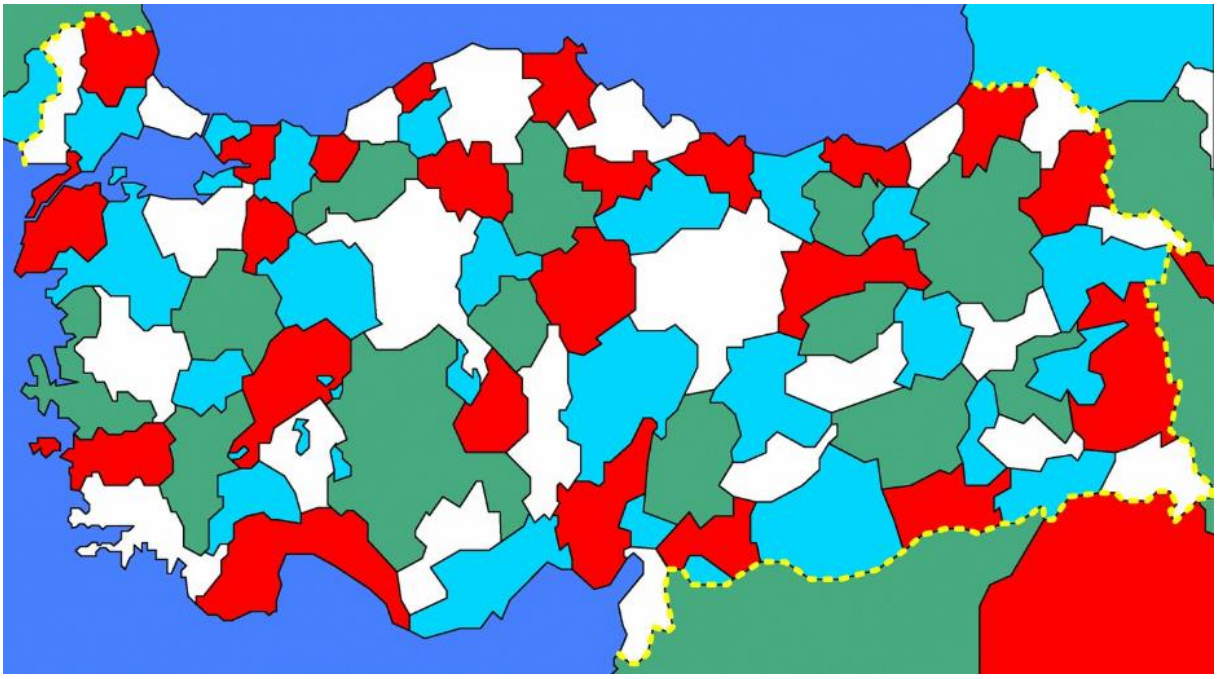
Görülebileceği gibi harita renklendirme ile bir önceki bölümdeki graf renklendirme arasında benzerlikler vardır. Gerçekte, aşağıda tanımlanan dual map (eş harita) kavramını kullanarak, bir haritayı renklendirme planar (düzlemsel) bir grafın düğümlerinin renklendirmesi ile eşdeğer olduğu gösterilebilir.

M gibi bir harita göz önüne alın. M'nin her bölgesinde bir nokta seçelim ve eğer iki bölge ortak bir kenara sahipse o zaman belirlediğimiz noktaları ortak kenar boyunca bir eğri (çizgi) ile birleştiririz. Bu eğriler birbirini kesmeyecek şekilde çizilir. Böylece M'nin duali olarak adlandırılan M^* gibi yeni bir harita elde ederiz öyleki M^*'nin her düğümü bir bölgeye karşılık gelir. Yukarıdaki Şekil (b), Şekil (a) daki haritanın dualini göstermektedir. M^* nin tam olarak M'nin bir düğümünü kapsayacağı ve M^* nin her kenarı tam olarak M'nin bir kenarı ile kesişeceği ispatlanabilir (ispatlanmıştır, gösterilmiştir). Böylece M, M^* haritasının duali olur.



Dört Renk Teoremi (Appel ve Haken)

Eğer bir M haritasının bölgelerini, komşu bölgeler farklı renkte olacak şekilde renklendirilirse dört renkten fazla renge gerek olmaz. Yukarıdaki teoremin ispatı için bilgisayar kullanılır. Ancak Appel ve Haken ilk defa bu teoremin doğru olmadığını söyleyenlerden yaklaşık 2000 farklı tipte planar (düzlemsel) harita arasından karşı bir örnek göstermelerini isteyerek dört renk teoreminin yanlış olmadığını gösterdiler. Bu teoremi esas alan yaklaşımla renklendirilen Türkiye haritasının bir örneği aşağıda verilmiştir.



Şekil-Dört renk kuralına göre renklendirilen Türkiye haritası

GRAFLARIN BİLGİSAYAR BELLEĞİNDE TEMSİL EDİLMESİ

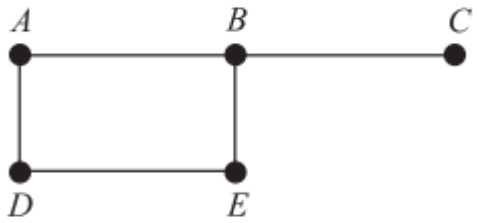
Grafların bilgisayar belleğinde saklanmasının iki standart yolu vardır. Birinci yol, G grafının ardışık temsili olarak adlandırılan bitişiklik matrisleri vasıtasıyla gösterilmesi. Diğer yol, komşuların bağlı listelerinin kullanılarak G grafının temsil edilmesi. Matris çözümleri genellikle G grafı yoğun olduğunda kullanılır, bağlı listeler ise genellikle G grafı seyrek olduğunda kullanılır. G grafının bellekte saklanma yoluna bakılmaksızın, G grafı normal olarak bilgisayara onun formal yani, düğümlerin ve düğüm bağlantılarını oluşturan kenar çiftlerinin toplamının tanımı ile girilir.

Bitişiklik Matrisi

m düğümlü bir G grafını ve bu grafın düğümlerinin v1, v2,...,vm biçiminde sıralı olduğunu farz edelim. Sonra, m × m boyutundaki bu G grafının bitişiklik matrisi A = [aij] aşağıdaki gibi tanımlanır.

$$aa_{ij} = \begin{cases} 1 \text{ eğer } v_i, \quad v_j \text{ ile komşu ise, bitişik ise} \\ 0 \qquad\qquad\qquad\qquad\qquad \text{değilse} \end{cases}$$

Aşağıdaki şekil (b), şekil (a) daki G grafının bitişiklik matrisini ihtiva ediyor/gösteriyor, ki bu matriste düğümler A,B,C,D,E ile gösterilmiştir. G'nin her vi, vj kenarı aij = 1 ve aj i = 1 olmak üzere iki kez temsil edilmiştir. Böylece, bitişiklik matrisi simetrik bir matristir.



(a)

	A	B	C	D	E
A	0	1	0	1	0
B	1	0	1	0	1
C	0	1	0	0	0
D	1	0	0	0	1
E	0	1	0	1	0

(b)

G grafının bitişiklik matrisi A, G grafının düğümlerinin sırasına bağlıdır, yani farklı bir düğüm sırası farklı bir bitişiklik matrisi sonucu doğurur. Fakat, herhangi iki böyle bir bitişiklik matrisleri yakından ilişkilidir, başka bir ifade ile birisi diğerinden elde edilebilir sadece satır ve sütunları yer değiştirerek. Diğer taraftan, bitişiklik matrisi bilgisayara girilen kenarların sırasına bağlı değildir. Yukardaki gösterimlerde çeşitli variyasyonlar vardır. Eğer G bir çoklu graf ise, o zaman genellikle aij, {vi, vj } kenar sırasını belirtir. Ayrıca, eğer G ağırlıklı bir graf ise, o zaman aij, {vi, vj } kenarının ağırlığını temsil eder.

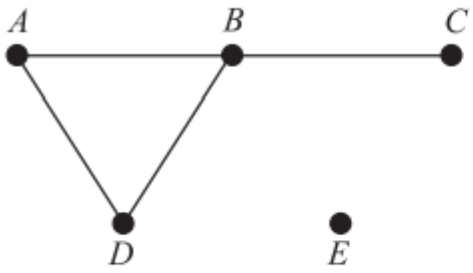
Bağlı listelerle G grafının temsil edilmesi.

G, m düğümlü bir graf olsun. G grafının bellekte bitişiklik matrisi A ile temsil edilmesinin bazı dezavantajları vardır. Birincisi, G grafına bir düğüm eklemek ve çıkarmak zor olabilir/zordur. Bunun sebebi, A matrisinin boyutu değişmesine ve düğümlerin tekrar sıralanmasına gereksinim olur. Bundan dolayı matris A da birçok değişiklik olabilir.

G grafı seyrek olduğunda, matris A birçok sıfır ihtiva eder, böylece çok büyük miktarda belek israfı oluşur. Buna bağlı olarak, G seyrek olduğunda, G grafı genellikle bitişiklik yapısı olarak adlandırılan bazı bağlı listelerle bellekte temsil edilir. Bu yapı aşağıda bir örnekle tanımlanmıştır.

Aşağıda verilen şekil (a) daki G grafını göz önüne alalım. G grafı şekil (b) görüldüğü gibi eşdeğerli olarak satırların düğümleri gösterdiği ve her bir düğümün karşısında bitişiklik listesi yer alan bir tablo ile tanımlanabilir. Burada Φ boş listeyi gösterir. U tablo daha kompakt formda aşağıdaki gibi de gösterilebilir.

G= [A:B,D; B:A, C, D; C:B; D:A, B; E: Φ]



(a)

Vertex	Adjacency list
A	*B, D*
B	*A, C, D*
C	*B*
D	*A, B*
E	∅

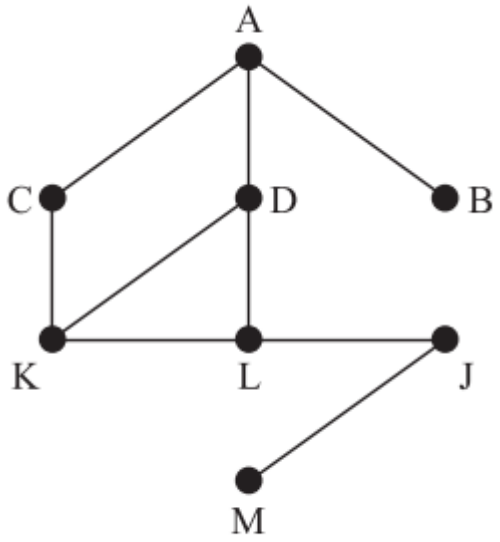
(b)

Bağlı listelerle temsil edilen bir G garfı ki bu graf bilgisayar belleğinde onun bağlı listesi ile saklanır, normal olarak düğüm dosyası ve diğeri de kenar dosyası olmak üzere iki dosya ihtiva eder.

GRAF ALGORİTMALARI

Bu bölümde G grafının düğüm ve kenarlarını sistematik olarak incelemek için iki önemli graf algoritması incelenecektir. Birincisi depth-first-search (DFS) algoritması ve diğeri ise breath-first-search (BFS) algoritmasıdır. Diğer graf algoritmaları bir sonraki bölümde yönlendirilmiş graflarda

tartışılacaktır. Herhangi bir belirli graf algoritması G grafının bellekte saklama yöntemine bağlıdır. Burada G grafının bellekte onun bitişiklik matrisi şeklinde saklandığını kabul ediyoruz. Örnek olarak alacağımız G grafının bitişiklik matrisinin yapısı aşağıdaki şekilde verilmiştir. Bu şekilde yer alan düğümlerin alfabetik sıraya göre sıralandığını kabul ediyoruz.



(a)

Vertex	Adjacency list
A	B, C, D
B	A
C	A, K
D	A, K, L
J	L, M
K	C, D, L
L	D, J, K
M	J

(b)

Algoritmaları çalıştırırken, G grafının her bir düğümü status olarak adlandırılan üç durumdan biri (birinde) olur.

Status=1: (hazır durumu), N düğümünün başlangıç durumu

Status=2: (bekleme durumu), N düğümünün, işlem görmek için listede (bekleme) olması durumu

Status=3: (İşlem görme durumu), N düğümünün işlem görmesi durumu

Depth-First-Search algoritması için (DBS) bekleme listesi STACK olacak, oysa Beardth-First-Search algortiması için bekleme listesi QUEUE olacaktır.

Depth-First-Serach Algoritması

Depth-First-Search Algoritması'ında başlangıç düğümü olarak A'nın alınmasının arkasındaki genel fikir aşağıdaki gibidir. İlk olarak A başlangıç düğümünü işliyoruz. Sonra, A 'dan başlayarak P yolu boyunca yer alan her N düğümünü işliyoruz. Yani A düğümünün komşusunu işliyoruz, daha sonra diğer komşusunu işliyoruz ve işlem yol üzerindeki düğümler bitene kadar devam eder. Sona geldikten sonra, yani işlenmemiş düğüm kalmayıncaya kadar, başka bir P′ yol devam edebilene kadar P yolunu geriye döneriz. Ve buna benzer. Gelecek olası yolların başlangıç düğümlerini tutmak için geriye

dönüşlerde STACK kullanarak başarılır. Ayrıca herhangi bir düğümün mevcut durumunu bize söyleyen bir alan STATUS'ne ihtiyacımız var öyle ki hiçbir düğüm birden fazla kez işlem görmez.

DFS algoritmasının adımları aşağıdaki şekilde gözükmektedir. Algoritma sadece A başlangıç düğümüne bağlı düğümleri, yani A düğümüne bağlı elemanları işleyecek. Bir kimse G grafındaki tüm düğümleri işlemek istediğini farz edelim. Sonra, başka bir düğüm ile (B düğümü olarak adlandırdığımız) ki bu düğüm hala hazır durumda, STATE=1 başlaması amacıyla algoritma değiştirilmeli. Bu B düğümü dolaşılabilir düğümler listesi vasıtasıyla elde edilebilir.

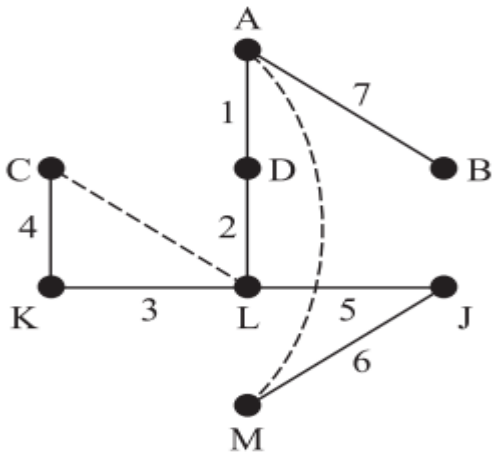
Örnek:

DFS algoritmasının aşağıdaki grafa uygulandığını farz edelim. Bu durumda işlenecek düğüm sırasını DFS algoritması ile bulunuz.

Öncelikle bitişiklik matrisini göz önüne almalıyız Buna göre DFS algoritmasını bu listeye (matrise) uygularız. Aşağıdaki şekil (a) işlenecek düğüm sırasını ve bekleme listesi STACK'taki sırayı gösterir.

STACK	Vertex
A	A
D, C, B	D
L, K, C, B	L
K, J, ~~K~~, C, B	K
C, J, ~~C~~, B	C
J, B	J
M, B	M
B	B
∅	

(a)



(b)

STACK	VERTEX
A	A
D, C, B	D
L, K, A, C, B	L
K, J, D, K, C, B	K
L, D,C, J, C, B	C
K,A,J,B	J
M,L,B	M
J,B	B
Φ,A	

Not: Bu tabloda kırmızı renkli olan düğümler işlenmiş olan düğümleri göstermektedir. Bu düğümler göz önüne alınmaz. Yeşil olan düğümler ise (stack) yapısında tekrar olan düğümleri gösteriyor. Dolayısıyla tekrar eden bu düğümler göz önüne alınmıyor.

Breadth-First-Search

Breadth-First-Search Algoritması'ında başlangıç düğümü olarak A'nın alınmasının arkasındaki genel fikir aşağıdaki gibidir. İlk olarak A düğümünü işleriz. Sonra, A düğümünün tüm komşu düğümlerini işleriz. Ve bu şekilde devam ederiz. Doğal olarak bir düğümün komşularının izini tutmamıza ve ayrıca bir düğümün iki kez işlenmediğini garanti etmemize ihtiyacımız var. Bu işlem, işlenmek için bekleyen düğümleri bir kuyrukta (QUEUE) tutarak ve düğümün mevcut durumunu bize söyleyen bellekteki bir STATUS alanı ile başarılır/üstesinden gelinir.

Breadth-First-Search Algoritması yalnızca başlangıç düğümü A'ya bağlı yani A düğümünün bağlı komponetleri olan düğümleri işleyecek. Bir kimsenin G grafındaki tüm düğümleri işlemek istediğini kabul edelim. Sonra algoritma B düğümü olarak adlandırdığımız yani STATUS=1 durumunda olan diğer bir düğüm ile başlayarak kendini değiştirir. Bu B düğümü, düğümlerin bulunduğu liste dolaşılarak elde edilir.

Örnek:

Şimdide aynı G garfını Breadth-First-Search Algoritması kullanarak işleyelim. Burada da grafın bitişiklik matrisini (listesini) kullanıyoruz.

QUEUE	VERTEX
A	A
D, C, B	B_A
D, C	C_A
D	D_A
L,K	K_D
L	L_D
J	J_L
M	M_J
Φ	