DIRECTED GRAPHS

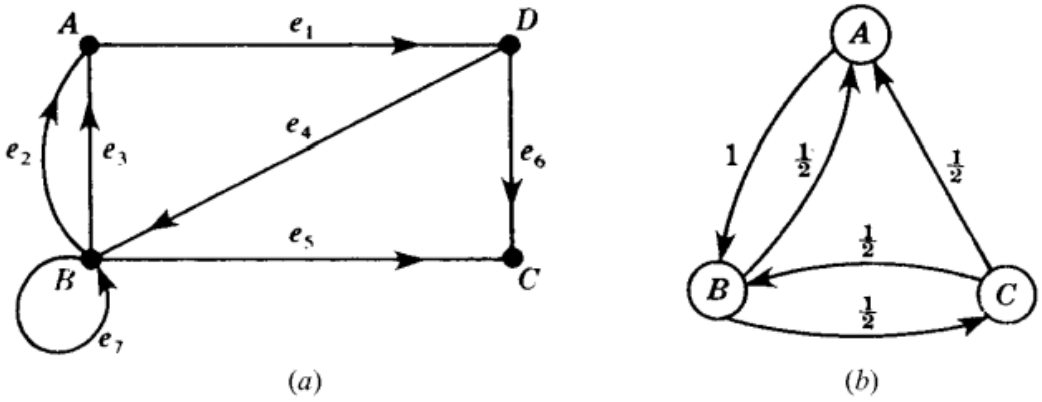
Directed graflar kenarlarının bir yönü olan graflardır. Bu tür graflar bilgisayarlar ve akış sistemleri gibi çeşitli dinamik sistemlerde çoğunlukla daha fazla kullanılan faydalı graflardır. Ancak bu ek özellikler grafların belirli özelliklerini tanımlamayı daha da zorlaştırır. Yani, bu tür graflar üzerinde işlem yapmak bir şehirde tek yönde gidişi olan sokaklarda dolaşmak zorunda olmaya benzetilebilir.

Directed graflar, yönledirilmemiş graflarda olduğu gibi G harfi ile gösterilir ve bir graf köşeler (V) ve kenarlar (E) olmak üzere iki bileşenden oluşur. V bir küme olmak üzere; V = V (G) biçimindeki gösterim, bir G grafındaki köşeleri (vertices), noktaları (points) ya da düğümleri (nodes) ihtiva eder/kapsar. E bir küme olmak üzere; E = E(G) biçimindeki gösterim, G grafının kenar (edge) olarak adlandırılan ve farklı noktaların sırasız çiftlerini oluşturan bir kümedir. Directed graflarda kenar e = (u, v), u düğümünden başlayıp v düğümünde sona erer. Eğer, u=v ise bu durumda kenar loop (döngü) olarak adlandırılır.

Örnek:

Aşağıdaki (a) grafını göz önüne alalım. Bu grafta dört tane düğüm A, B, C, D, yani V (G) = {A, B, C, D} ve yedi tane kenar: E(G) = {e1, e2,...,e7}={(A, D), (B, A), (B, A), (D, B), (B, C), (D, C), (B, B)} vardır. e2 ve e3 kenarlarına paralel kenar denir çünkü her ikisi de B düğümünde başlayıp A düğümünde bitmektedir.

Üç çocuk (A,B,C) bir topu birbirlerine attıklarını farz edin. A sürekli olarak topu B'ye attığını, fakat B ve C, A'dan farklı olarak topu birbirlerine atmaktadır. Bu dinamik sistem aşağıdaki (b) grafıyla temsil edilebilir. Kenarlar üzerindeki değerler topun tutulma olasılıklarını göstermektedir. A topu B ye 1 olasılıkla atmakta, B topu A ve C'ye ½ olasılıkla atmaktadır. C ise A ve B'ye ½ olasılıkla atmakta.



G = G(V , E) nin bir directed graf ve V' de G grafının V düğümlerinin bir alt kümesi olsun. E', E kenarlar kümesinin bir alt seti olsun öyle ki, E kenarının sonladığı düğüm V düğümler kümesinin bir elamnı olduğunu farz edelim. Bu durumda H^* (V , E) ye G grafının alt grafı olan bir directed graftır denir. Örneğin:

G = G(V , E) grafı için H (V^* , E^*) ye V = {B, C, D} düğümler kümesi tarafından tanımlanan G grafının alt grafı denir ve kenarlar E = (e4, e5, e6, e7) = {(D, B),(B, C),(D, C),(B, B)} biçiminde gösterilir.

Degrees

G bir directed graf olmak üzere, G grafının v düğümünün outdegre (çıkış derecesi) outdeg(v) biçiminde yazılır ve v düğümünden çıkan kenar sayısını, indegree (giriş derecesi) indegree(v) yazılır ve v düğümünde sonlanan kenar sayısı ifade eder. Bir kenarın bir düğümden başlayıp bir düğümde bitmesi aşağıdaki teoriyi oluşturur.

Teorem 9.1: Bir G grafına ait düğümlerin çıkış derecelerinin toplamı, aynı G grafının düğümlerinin giriş derecelerinin toplamına eşittir. Bu sayıda G grafının kenar sayısına eşittir.

Örnek: Yukarıda verilen şekildeki (a) grafını göz önüne alalım. Bu G grafındaki düğümlerin giriş ve çıkış dereceleri

outdeg (A) = 1, outdeg (B) = 4, outdeg (C) = 0, outdeg (D) = 2, Toplam=7

indeg (A) = 2, indeg (B) = 2, indeg (C) = 2, indeg (D) = 1. Toplam=7

Dolayısıyla elde edilen toplam değeri ele alınan G grafının kenarlarının sayısına eşittir. C düğümüne sink düğümü denir, çünkü bu düğümden çıkan hiçbir kenar yoktur, düğüme sadece giren kenarlar vardır.

DİRECTED GRAFLARIN GÖSTERİMİ-TEMSİL EDİLMESİ

Directed graflar bilgisayar belleklerinde iki biçimde gösterilir. Birincisi bitişiklik matrisi A ile grafların ardaşık gösterimi. İkincisi G grafının bağlı listeler ile temsil edilmesi

Directed Graflar ve İlişkiler, Bitişiklik Matrisi

G(V, E) grafı basit bir directed graf yani paralel kenarları olmayan bir graftır. E, $V \times V$ Kartezyen çarpımın bir alt kümesidir ve E ile V düğümler kümesi arasında bir ilişki vardır. Diğer taraftan, eğer R, V kümesi üzerinde bir ilişki ise o zaman G(V, R) basit bir directed graftır.

G grafının m düğümlü bir graf olduğunu ve G grafının düğümlerinin v1, v2,…,vm biçiminde sıralı olduğunu kabul edelim. Bu durumda G grafının $m \times m$ boyutundaki bitişiklik matrisi A = [aij] aşağıdaki biçimde tanımlanır.

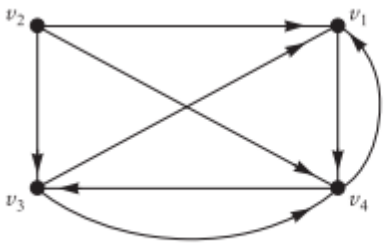
$$a_{ij} = \begin{cases} 1 & \text{if there is an edge } (v_i, v_j) \\ 0 & \text{otherwise} \end{cases}$$

Böyle bir matris 1 ve 0 elemanlarından oluşur.

G nin bitişiklik matrisi A, G grafının düğümlerinin sırasına bağlıdır. Fakat iki farklı sıralamadan elde edilen matrisler birbirleriyle yakından ilişkilidir. Bir kimse sadece satır ve sütunları değiştirerek diğer matrisi elde edebilir. Aksi belirtilmediği müddetçe biz örneklerimizde düğüm sırasının değişmediğini farz edeceğiz.

ÖRNEK:

Aşağıda verilen ve v1, v2, v3, v4 düğümlerinden oluşan directed G grafını göz önüne alalım. G grafının bitişiklik matrisi A aşağıda verilmiştir. Bitişiklik matrisi A'da yer alan 1'lerin sayısı kenarların sayısına eşit olduğunu dikkat ediniz.



(a)

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

(b)

G grafının bitişiklik matrisi A = [aij]'nın A, A^2, A^3 kuvvetlerini (gücünü) düşünelim. $a_K(i, j)$, matris A^K daki ij girişi temsil eder. $a_1(i, j) = a_{ij}$, v_i düğümünden v_j düğümüne olan yolun uzunluğunun 1 olduğunu ifade

eder. Benze biçimde bir kimse $a_2(i, j)$ v_i düğümünden v_j düğümüne olan yolun uzunluğunun 2 olduğunu gösterebilir. Bu durumu gösteren aşağıdaki örneği yapalım.

Örnek:

Tekrar yukarıdaki G grafını ve onun bitişiklik matrisini göz önüne alalım. Bu matrisin A^2, A^3, ve A^4 kuvveti aşağıdaki gibidir:

$$A^2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 2 & 0 & 1 & 2 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 2 \end{bmatrix}, \quad A^3 = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 3 & 0 & 2 & 3 \\ 2 & 0 & 1 & 2 \\ 2 & 0 & 2 & 1 \end{bmatrix}, \quad A^4 = \begin{bmatrix} 2 & 0 & 2 & 1 \\ 5 & 0 & 3 & 5 \\ 3 & 0 & 2 & 3 \\ 3 & 0 & 1 & 4 \end{bmatrix}$$

a_2 (4, 1) = 1 dir. Bu yüzden v4 düğümünden v1 düğümüne 2 birim uzunluğunda bir yol vardır. Ayrıca $a_3(2, 3) = 2$, bu yüzden v2 den v3'e 3 birim uzunluğunda iki yol vardır. a4(2, 4) = 5, v2 düğümünden v4 düğümüne 4 birim uzunluğunda 5 yol vardır.

Açıklama: A, G grafının bir bitişiklik matrisi olsun ve B_r de

$$B_r == A + A^2 + A^3 + \cdots + A^r$$

biçiminde tanımlanan bir matris olsun. Böylece B_r matrisinin ij girişi (elemanı), vi düğümünden vj düğümüne olan r uzunluğundaki yolların sayısını verir.

Yol Matrisi

G = G(V , E), m düğümlü (v1, v2,..., vm) basit bir yönlendirilmiş graf olsun. Yol matrisi ya da G'nin erişilebilirlik matrisi P aşağıdaki gibi tanımlanır.

$$p_{ij} = \begin{cases} 1 & \text{if there is a path from to } v_i \text{ to } v_j \\ 0 & \text{otherwise} \end{cases}$$

m düğümlü G grafında vi düğümünden vj düğümüne bir yol var olduğunu kabul edelim. vi≠vj olmadığı durumda vi den vj ye basit bir yol vardır ya da i=vj olduğunda bir döngü vardır. G grafı m düğümü olduğu için, böle bir basit yolun uzunluğu m-1 ya da daha olamlıdır. Veya bir döngü var ise bu döngünün uzunluğu m ya da daha kısa (az) olmalıdır. Bu şu anlama gelmektedir; yukarı da tanımlanan Bm matrisindeki ij girişlerinde sıfır olmayan elaman vardır. Benzer biçimde yol matrisi P ve Bm aynı sıfır olmayan elemanlara sahiptir/vardır. Bu ifadeleri formül olarak aşağıdaki biçimde gösteririz.

Önerme (9.5):

A, m düğümlü bir G grafının bitişiklik matrisi olsun. Sonra yol matrisi ve Bm matrisi aynı sıfır olmayan elemanlara sahiptir. Burada

$Bm = A + A^2 + A^3 + \cdots + A^m$

Hatırlayalım, herhangi iki çift u ve v düğümü arasında u dan v'ye ve v'den u'ya bir yol var ise böyle bir directed G grafına strongly (kuvvetli bir şekilde) bağlı graf denir. Benzer biçimde G kuvvetli bir şekilde bağlı graftır eğeryol G'nin yol matrisi P sıfır değerli girişlere sahip değilse. Bu gerçek aşağıdaki önermeyi ortaya çıkarır.

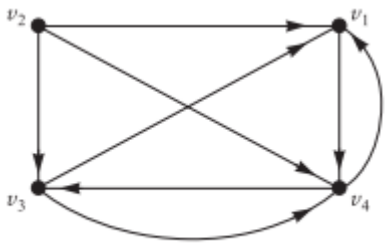
Önerme 9.6:

A, m düğümlü bir G grafının bitişklik matrisi olsun. G grafı kuvvetli şekilde bağlı bir graftır ancak ve ancak eğer Bm sıfır değerli elemana sahiğ değilse. Bu önermede

$Bm = A + A^2 + A^3 + \cdots + A^m$

Örnek 9.8:

Tekrar aşağıda verilen grafı ve bitişiklik matrisini göz önüne alalım.



(a)

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

(b)

Burada G grafı m=4 düğüme sahiptir. Örnek 9.7'deki A matrisi ve A^2, A^3, A^4 matrislerini toplarsak, izleyen B4 matrisini ve ayrıca B4 deki değeri sıfır olmayan girişleri 1 ile değiştirirsek yol matrisi (erişilebilirlik) P'yi elde ederiz.

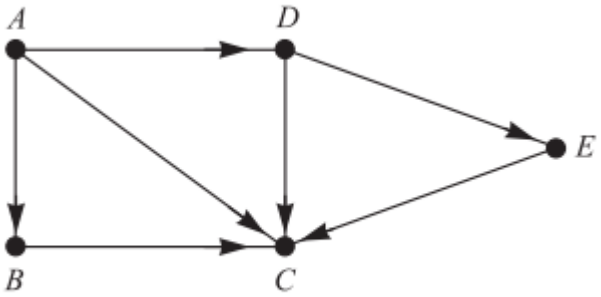
$$B_4 = \begin{bmatrix} 4 & 0 & 3 & 4 \\ 11 & 0 & 7 & 11 \\ 7 & 0 & 4 & 7 \\ 7 & 0 & 4 & 7 \end{bmatrix} \quad \text{and} \quad P = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

B4 ve P matrislerini incelediğimizde, sıfır değerli girişler görürüz, bu yüzden G kuvvetli bağlı bir (graf) değildir. Özellikle v2 düğümüne, diğer düğümlerden erişilemez.

YÖNLENDİRİLMİŞ GRAFLARIN BAĞLI LİSTELERLE TEMSİL EDİLMESİ.

G, m düğümlü bir directed matris olsun. G grafının bitişiklik matrisi A birçok sıfır değer ihtiva edebilir; bu yüzden çok bellek alanı kaybı olur. Buna bağlı olarak, G grafı seyrek olduğunda G grafı genellikle bellekte bağlı listelerle temsil edilir ayrıca bu gösterime bitişiklik yapısı denir, bu yapı aşağıda bir örnekle gösterilecektir.

ÖRNEK



(a) Graph G

Vertex	Adjacency list
A	B, C, D
B	C
C	$\varnothing$
D	C, E
E	C

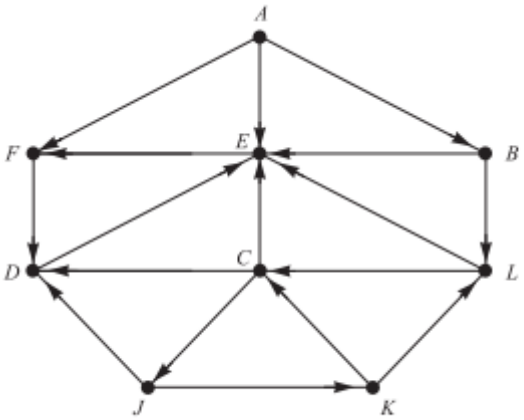
(b) Adjacency lists of G

(a) da gözüken directed grafı göz önüne alalım. Bu G grafı eşdeğer olarak (b) de gözüktüğü gibi bir tablo ile gösterilebilir. Bu tablo G grafının her düğümünü ve o düğümün bağlı listesini gösterir. Buna o düğümün komşuları da denir. Tablodaki Φ işareti boş listeyi temsil eder. Tabloda dikkat edilirse G grafının her kenarı bağlı listedeki bir düğümü bağlıdır. Burada G grafının bitişiklik listesinde yedi düğümü ve yedi kenarı vardır. Bu tablo ayrıca aşağıdaki gibi kompakt formda da gösterilebilir bu gösterimde ":" bir düğümü onun komşu listesinden ";" ise farklı listeleri ayırır.

$$G = [A : B, C, D; \quad B : C; \quad C : \varnothing; \quad D : C, E; \quad E : C]$$

DIRECTED GRAFLARDA: DEPTH-FIRST VE BREADTH-FIRST ALGORITMALARI

Directed graflarda bilgisayar belleklerinde iki şekilde saklanabilir. Burada grafların bilgisayar belleklerinde bitişiklik yapısına göre saklandığı kabul edilmektedir. Çözümler ona göre yapılacaktır. Bunun için aşağıda verilen grafı göz önüne alalım.



(a)

Adjacency lists
A: *B*, *E*, *F*
B: *E*, *L*
C: *D*, *E*, *J*
D: *E*
E: *F*
F: *D*
J: *D*, *K*
K: *C*, *L*
L: *C*, *E*

(b)

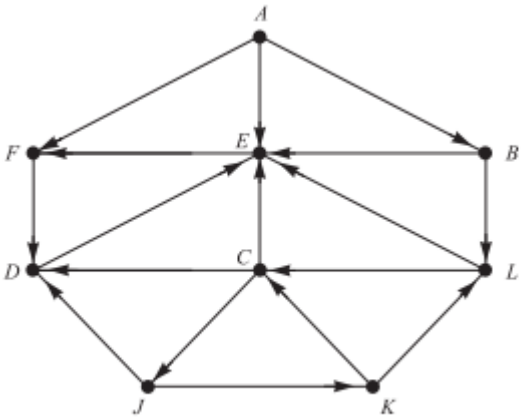
Graflar üzerinde yer alan düğüm ve kenarları işlemek için sistematik yaklaşımlara gereksinim vardır. Bunun için iki standart yol vardır. Birincisi depth-firsth-serach ve ikincisi breath-firsth-search algoritmalarıdır. Bu algoritmaların çalışması aynı yönsüz graflarda olduğu gibidir. Yani bu algoritmaları işlerken/çalıştırırken G grafının her bir düğümü (N) düğümün statüsü olarak adlandırılan üç durumdan birinde olur.

STATUS=1: (Hazır durum-Ready state), düğümün başlangıçtaki durumu

STATUS=2: (Bekleme durumu-waiting state), Bekleme listesinde bulunan N düğümü, prosese girmek için

STATUS=3: (İşlem durumunda) İşlem gören N düğümü

ÖRNEK: Yukarıdaki G grafını göz önüne alalım. Öncelikle G grafının bitişiklik listesini hazırlamamız lazım. Bu G grafının J düğümünden (J düğümü dahil olmak üzere) başlayarak erişilebilecek düğümlerin listesini bulmamız istenmektedir. Bu işlemi yapmak için depth-first-search algoritması ile çözelim/yapalım.



(a)

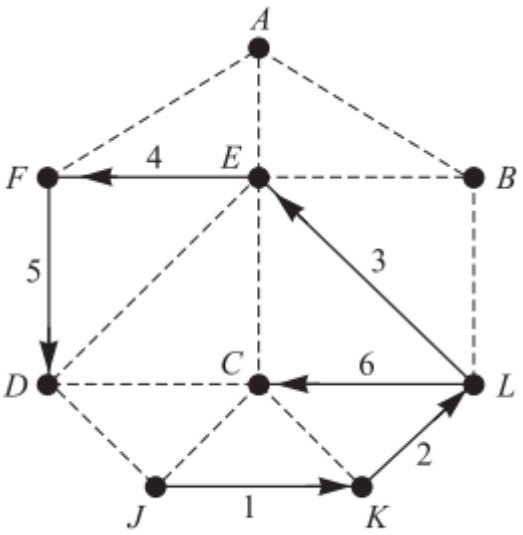
Adjacency lists
A: *B*, *E*, *F*
B: *E*, *L*
C: *D*, *E*, *J*
D: *E*
E: *F*
F: *D*
J: *D*, *K*
K: *C*, *L*
L: *C*, *E*

(b)

DFS algoritması STACK esasına göre düğümleri işleme alır. Buna göre bitişiklik matrisi/listesindeki düğümler işleme alınacaktır. (/) lı düğümler listeden silindiği anlamına gelmektedir.

STACK	Vertex
J	J
K_J, D_J	K_J
L_K, C_K, D_J	L_K
$E_L, C_L, \not{C}_K, D_J$	E_L
F_E, C_L, D_J	F_E
$D_F, C_L, \not{D}_J$	D_F
C_L	C_L
$\varnothing$	

(a)



(b)

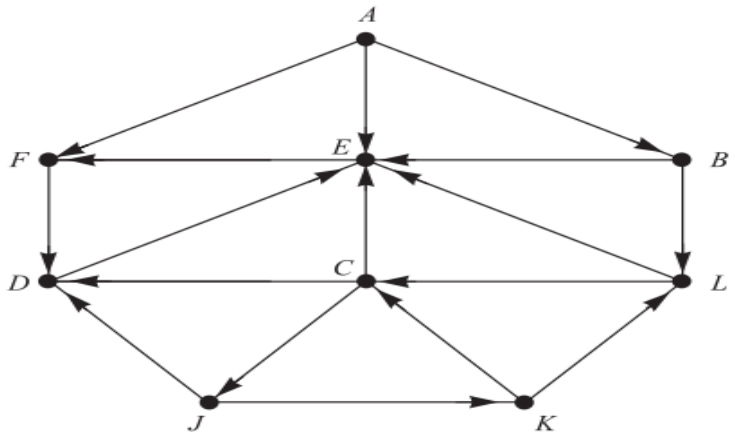
Her bir düğüm bitişiklik listesinden gelmektedir ve bu yüzden grafın tek bir kenarının son/nihai düğümüdür. Burada kenarların gösteriminde sonlandığı düğüm numarasını gösteren bir indis numarası kullanılır. D_j bu gösterim şu anlama gelmektedir. D'nin J'nin bitişiklik listesinde bulunduğunu ifade etmektedir. Böylece, J ile başlayan bir kenarın sonlama düğümüdür. Şekil (b) deki graf (kenarlar) kökü J olan bir root ağaç oluşturur. Ağaç üzerindeki numaralar ağaca eklenen kenarların sırasını göstermektedir. Bu ağaç J düğümünden erişilebilir düğümlerin bir grafı olan, G grafının bir alt grafını oluşturur.

BERTH-FIRST-SEARCH

Başlangıç düğümü A olmak üzere berth-first-algoritmasının arkasındaki genel fikir/yaklaşım aşağıdaki gibidir. İlk olarak başlangıç düğümü A işlenir. Sonra A düğümünün tüm komşuları işlenir. Daha sonra A düğümünün komşularının komşuları işlenir ve işlem böylece devam eder. Doğal olarak bir düğümün komşularının izlerini takip etmemize ve bir düğümün iki kez işlenmemesini garanti etmemize gerek var. Bu işlem, düğümlerin işlem görmek için bekleme listesinde tutulduğu kuyruk yapısını kullanarak başarılabilir. Bu algoritmada düğümün mevcut durumunu gösteren statü alanı (bellekte) tutularak başarılabilir.

ÖRNEK: Yine aşağıdaki G grafını göz önüne alalım. Bu grafın günlük şehirler arası uçak uçuşlarını temsil ettiğini ve A şehrinden J şehrine en az duruş ile uçmak istediğimizi kabul edelim. Burada tüm yolların

aynı ağırlıkta olduğunu kabul ediyoruz. Bu problemi çözmenin bir yolu A düğümünden başlayıp, J düğümüne erişildiğinde duran BFS algoritmasını kullanmaktır.



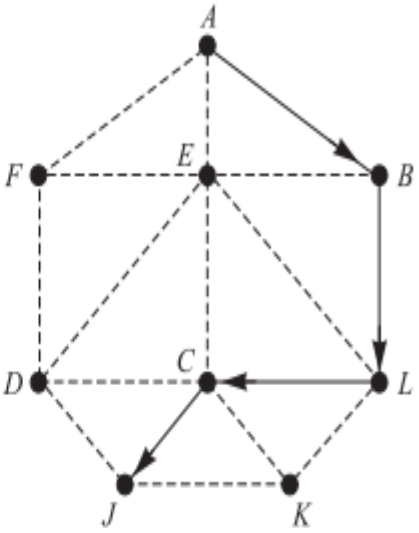
(a)

Adjacency lists
A: *B*, *E*, *F*
B: *E*, *L*
C: *D*, *E*, *J*
D: *E*
E: *F*
F: *D*
J: *D*, *K*
K: *C*, *L*
L: *C*, *E*

(b)

QUEUE	Vertex
A	A
F_A, E_A, B_A	B_A
L_B, F_A, E_A	E_A
L_B, F_A	F_A
D_F, L_B	L_B
C_L, D_F	D_F
C_L	C_L
J_C	J_C

(a)



(b)

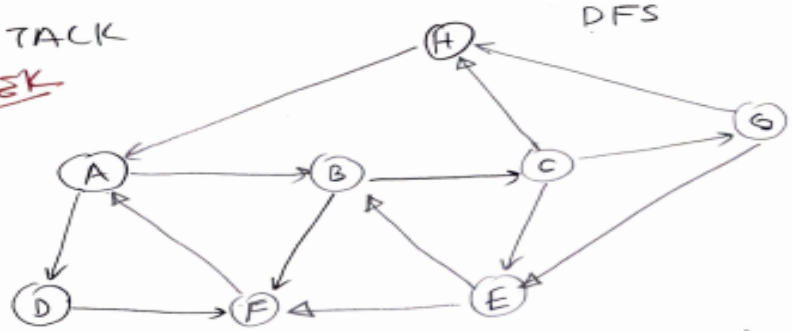
ÖRNEK-2

STACK

DFS

ÖRNEK

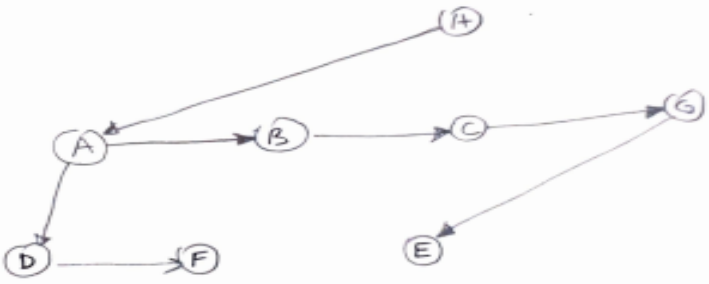
1



Vertex	
A	B, D
B	C, F
C	E, G, H
D	F
E	B, F
F	A
G	E, H
H	A

SORU: H'den ulaşılabilecek düğümler nelerdir?

STACK	Vertex
H	H
A_H	A_H
D_A , B_A	D_A
F_D B_A	F_D
~~A_F~~ B_A	B_A
~~F_B~~ , C_B	C_B
~~H~~E , ~~G~~E , E_C	G_C
~~H~~G , E_G , ~~E~~C	E_G
~~F~~E , ~~B~~E	
~~$\emptyset$~~	



2

QUEUE

BFS

SORU: A'dan E şehrine gidişlerinin yollarını bulunuz?

QUEUE	Vertex
A	A
D_A , B_A	B_A
F_B , C_B D_A	D_A
~~F_D~~ , F_B , C_B	C_B
H_C , G_C , E_C , F_B	F_B
~~A_E~~ , H_C , G_C , E_C	E_C
~~F_E~~ , ~~B_E~~ , H_C , G_C	G_C
~~H_G~~ , ~~E_G~~ , H_C	H_C
~~A_H~~	
$\emptyset$