

CENG 213 Veri Yapıları 9: Çizge(Graph)

Öğr.Gör. Şevket Umut ÇAKIR

Pamukkale Üniversitesi

Hafta 9

1 Graph Tanımı ve Özellikleri

- Tanımlar

2 Dolaşma İşlemleri/Traversal

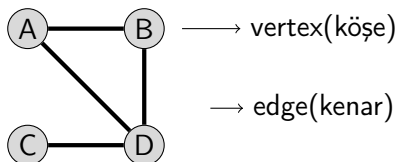
- Derinine Arama/Depth First Search
- Enine Arama(Breadth First Search)
- Topolojik Sıralama/Topological Sorting

3 Çizge Temsilleri

- Komşuluk Matrisi/Adjacency Matrix
- Komşuluk Listesi/Adjacency List

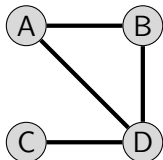
Tanım

Çizge(Graph) veri yapısı köşe(vertex) ve kenarlardan(edge) oluşan bir yapıdır. $G = (V, E)$ Kenarlar iki köşe arasındaki (v, w) yollardır. $v, w \in V$

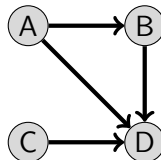


Tanım

Çizgenin kenarları sıralı ise ya da bir düğümden başka bir düğüme doğru ise buna **yönlü çizge(directed graph)** aksi takdirde **yönsüz çizge(undirected graph)** adı verilir.



Şekil: Yönsüz Çizge



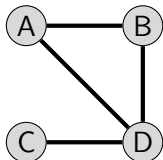
Şekil: Yönlü Çizge

En çok kenar sayısı

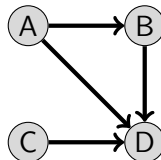
- Yönsüz çizge için:

Tanım

Çizgenin kenarları sıralı ise ya da bir düğümden başka bir düğüme doğru ise buna **yönlü çizge(directed graph)** aksi takdirde **yönsüz çizge(undirected graph)** adı verilir.



Şekil: Yönsüz Çizge



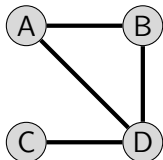
Şekil: Yönlü Çizge

En çok kenar sayısı

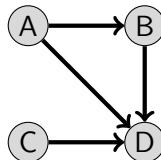
- Yönsüz çizge için: $\frac{n(n-1)}{2}$
- Yönlü çizge için:

Tanım

Çizgenin kenarları sıralı ise ya da bir düğümden başka bir düğüme doğru ise buna **yönlü çizge(directed graph)** aksi takdirde **yönsüz çizge(undirected graph)** adı verilir.



Şekil: Yönsüz Çizge



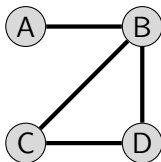
Şekil: Yönlü Çizge

En çok kenar sayısı

- Yönsüz çizge için: $\frac{n(n-1)}{2}$
- Yönlü çizge için: $n(n-1)$

Düğüm Derecesi/Vertex Degree

Yönsüz çizgeler için her bir düğümün kenar sayısıdır.

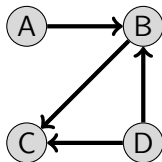


Tablo: Yönsüz Çizge için Düğüm Dereceleri

	A	B	C	D
Düğüm Derecesi	1	3	2	2

Düğüm Derecesi/Vertex Degree

Yönlü çizgeler için her bir düğüme gelen(Indegree) ve düğümden çıkan(Outdegree) kenar sayısıdır.

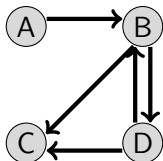


Tablo: Yönlü Çizge için Düğüm Dereceleri

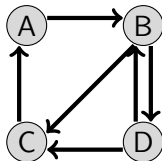
	A	B	C	D
Indegree	0	2	2	0
Outdegree	1	1	0	2

Tanım

Bir düğümden başka bir düğüme ulaşılabiliriyorsa, bu iki düğüm arasında **yol(path)** vardır. $A, X_1, X_2, \dots, X_n, C$ kenarları mevcutsa A'dan C'ye yol vardır.

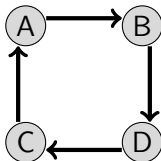


- A, B, C
- A, B, D, C
- A, B, D, B, C



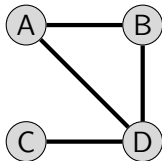
- **Basit Yol(Simple Path):** Aradaki düğümlere sadece 1 defa uğranıyorsa basit yol vardır
 - A, B, C
 - A, B, D, C
 - A, B, C, A
 - A, B, D, C, A
- Aksi takdirde yol basit değildir
 - A, B, D, B, C

Bir yol başladığı yerde bitiyorsa çizgede döngü(cycle) vardır.

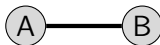


Tanım

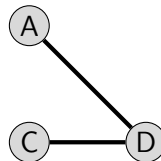
Bir çizgenin hernagi bir parçasına **alt çizge(subgraph)** denir.



Şekil: Örnek Çizge



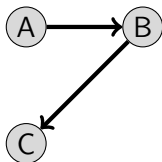
Şekil: Alt Çizge 1



Şekil: Alt Çizge 2

Tanım

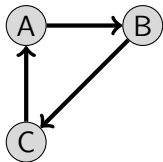
V_i ile V_j düğümleri arasında yol varsa bu iki düğüm **bağlıdır**(connected). Herhangi iki düğüm arasında bir yol varsa bu çizgeye **bağlı çizge** (connected graph) adı verilir.



Şekil: Bağlı Olmayan Çizge

- A, B
- B, C
- ~~B, A~~
- ~~C, A~~
- ~~C, B~~

Bağlı Çizgeler/Connected Graphs



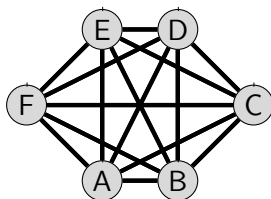
- A, B
- B, C
- B, A
- C, A
- C, B

Şekil: Bağlı Çizge(Connected Graph)

Tam Bağlı Çizge/Completely Connected Graphs

Tanım

Herhangi iki düğüm arasında bir kenar varsa buna **tam bağlı çizge**(completely connected graph) adı verilir.

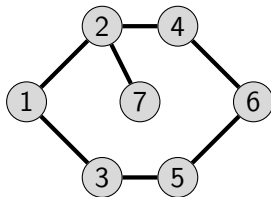


Şekil: Tam Bağlı Çizge

Derinine Arama/Depth First Search

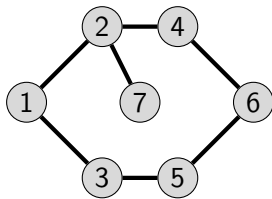
Tanım

Dolaşma işlemi bir düğümden başlar. Gidilebilecek mümkün komşulardan bir tanesini seçer ve ilerler. Gidilebilecek diğer düğümlere bakmadan önce yeni gidilen düğümün komşuları incelenir ve bir tanesine gidilir. Gidilebilecek komşu kalmayınca geri sarılarak mümkün komşular aranır. Tek biçimlilik sağlamak için mümkün komşulardan alfabetik olarak küçük olan önce seçilir.



Şekil: DFS Örneği(1 düğümünden başlansın)

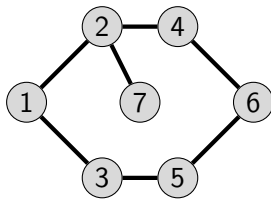
Derinine Arama/Depth First Search



Şekil: DFS Örneği(1 düğümünden başlansın)

1,

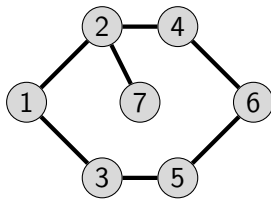
Derinine Arama/Depth First Search



Şekil: DFS Örneği(1 düğümünden başlansın)

1, 2,

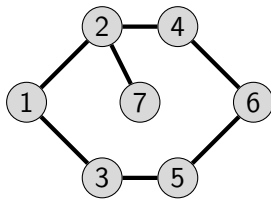
Derinine Arama/Depth First Search



Şekil: DFS Örneği(1 düğümünden başlansın)

1, 2, 4,

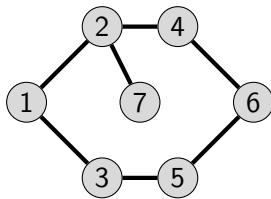
Derinine Arama/Depth First Search



Şekil: DFS Örneği(1 düğümünden başlansın)

1, 2, 4, 6,

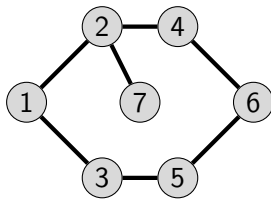
Derinine Arama/Depth First Search



Şekil: DFS Örneği(1 düğümünden başlansın)

1, 2, 4, 6, 5,

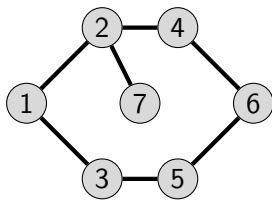
Derinine Arama/Depth First Search



Şekil: DFS Örneği(1 düğümünden başlansın)

1, 2, 4, 6, 5, 3,

Derinine Arama/Depth First Search



Şekil: DFS Örneği(1 düğümünden başlansın)

1, 2, 4, 6, 5, 3, 7

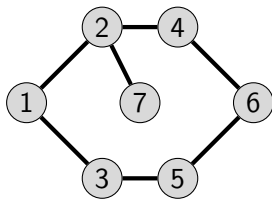
Function DFS(*kaynak*):

```
s ← yeni yığıt(new stack);  
visited ← {} // boş küme  
s.push(kaynak);  
while s boş olmadığı sürece do  
    | mevcut ← s.pop();  
    | if mevcut, visited kümesi içindeyse then  
    | | continue;  
    | end  
    | visited.add(mevcut);  
    | foreach (mevcut, v) kenarına sahip bütün v köşeleri için do  
    | | s.push(v);  
    | end  
end
```


Enine Arama(Breadth First Search)

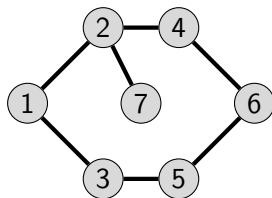
Tanım

Dolaşma işlemi bir düğümden başlar. Gidilebilecek mümkün komşularhepsini sırayla seçer ve ilerler. Gidilebilecek bütün komşular gezildikten sonra ilk komşunun komşuları seçilir. Tek biçimlilik sağlamak için mümkün komşulardan alfabetik olarak küçük olan önce seçilir.



Şekil: BFS Örneği(1 düğümünden başlansın)

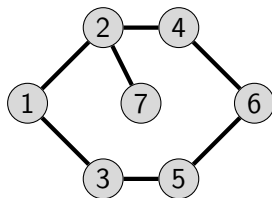
Enine Arama(Breadth First Search)



Şekil: BFS Örneği(1 düğümünden başlansın)

1,

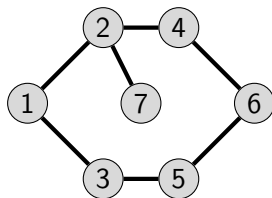
Enine Arama(Breadth First Search)



Şekil: BFS Örneği(1 düğümünden başlansın)

1, 2,

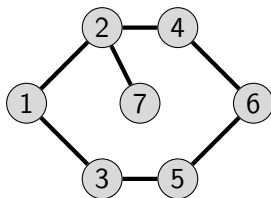
Enine Arama(Breadth First Search)



Şekil: BFS Örneği(1 düğümünden başlansın)

1, 2, 3,

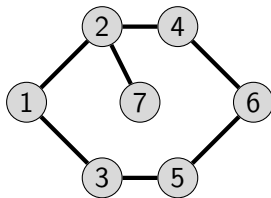
Enine Arama(Breadth First Search)



Şekil: BFS Örneği(1 düğümünden başlansın)

1, 2, 3, 4,

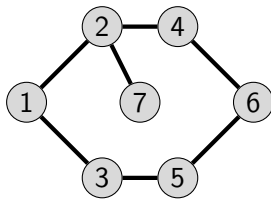
Enine Arama(Breadth First Search)



Şekil: BFS Örneği(1 düğümünden başlansın)

1, 2, 3, 4, 7,

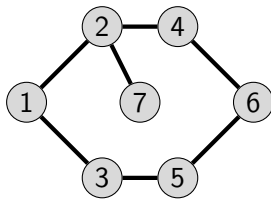
Enine Arama(Breadth First Search)



Şekil: BFS Örneği(1 düğümünden başlansın)

1, 2, 3, 4, 7, 5,

Enine Arama(Breadth First Search)



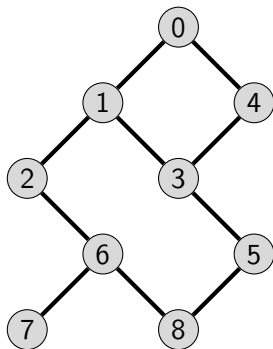
Şekil: BFS Örneği(1 düğümünden başlansın)

1, 2, 3, 4, 7, 5, 6

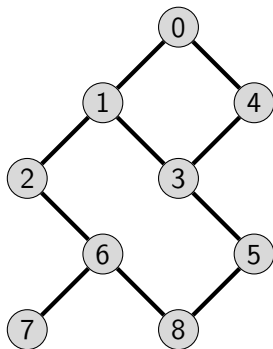
Function BFS(*kaynak*):

```
q ← yeni kuyruk(new queue);  
visited ← {} // boş küme  
q.enqueue(kaynak);  
while q boş olmadığı sürece do  
    | mevcut ← s.dequeue();  
    | if mevcut, visited kümesi içindeyse then  
    | | continue;  
    | end  
    | visited.add(mevcut);  
    | foreach (mevcut, v) kenarına sahip bütün v köşeleri için do  
    | | q.enqueue(v);  
    | end  
end
```

Aşağıdaki çizgede 0 düğümünden başlayarak derinine(DFS) ve enine(BFS) arama gerçekleştirin.



Aşağıdaki çizgede 0 düğümünden başlayarak derinine(DFS) ve enine(BFS) arama gerçekleştirin.



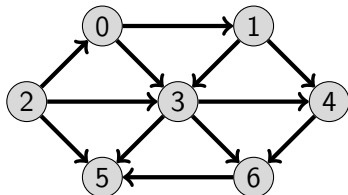
DFS: 0, 1, 2, 6, 7, 8, 5, 3, 4

BFS: 0, 1, 4, 2, 3, 6, 5, 7, 8

Topolojik Sıralama/Topological Sorting

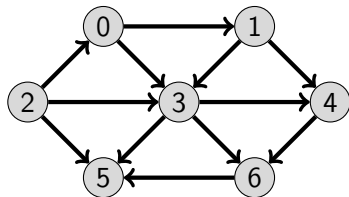
Tanım

Topolojik sıralama yönlü döngü içermeyen çizgelerde(directed acyclic graphs(DAG)) çalışan bir sıralama çeşididir.



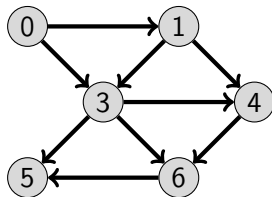
- Çizgede düğüm olduğu sürece giriş derecesi(Indegree) 0 olan bir düğüm seç ve çizgeden çıkar

Topolojik Sıralama Örneği



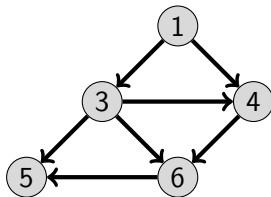
Sıralama:

Topolojik Sıralama Örneği



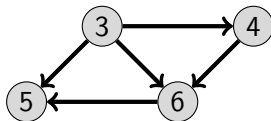
Sıralama: 2

Topolojik Sıralama Örneği



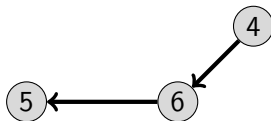
Sıralama: 2, 0

Topolojik Sıralama Örneği



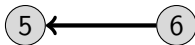
Sıralama: 2, 0, 1

Topolojik Sıralama Örneği



Sıralama: 2, 0, 1, 3

Topolojik Sıralama Örneği



Sıralama: 2, 0, 1, 3, 4

Topolojik Sıralama Örneği

5

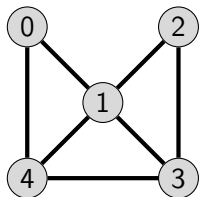
Sıralama: 2, 0, 1, 3, 4, 6,

Topolojik Sıralama Örneği

5

Sıralama: 2, 0, 1, 3, 4, 6, 5

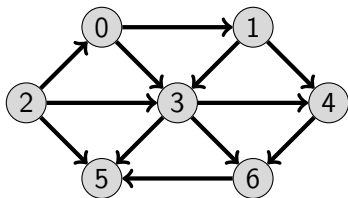
Komşuluk Matrisi/Adjacency Matrix



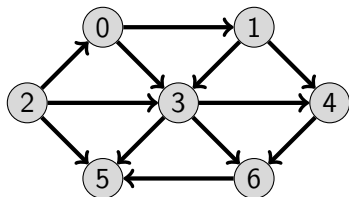
Tablo: Komşuluk Matrisi

	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0

Aşağıdaki çizgenin komşuluk matrsini oluşturun.



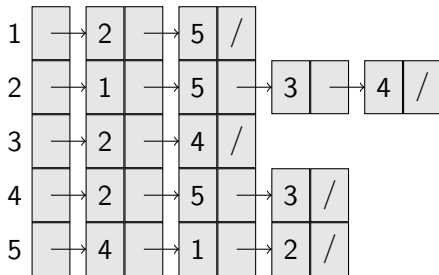
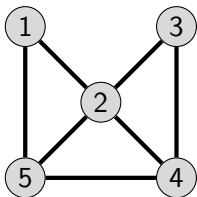
Aşağıdaki çizgenin komşuluk matrsini oluşturun.



Tablo: Komşuluk Matrisi

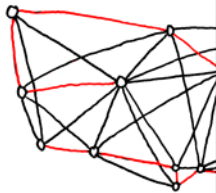
	0	1	2	3	4	5	6
0	0	1	0	1	0	0	0
1	0	0	0	1	1	0	0
2	1	0	0	1	0	1	0
3	0	0	0	0	1	1	1
4	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0
6	0	0	0	0	0	1	0

Komşuluk Listesi/Adjacency List



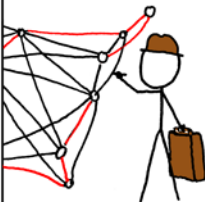
BRUTE-FORCE
SOLUTION:

$$O(n!)$$



DYNAMIC
PROGRAMMING
ALGORITHMS:

$$O(n^2 2^n)$$



SELLING ON EBAY:
 $O(1)$

STILL WORKING
ON YOUR ROUTE?

SHUT THE
HELL UP.

