# Ayrık Matematik
# (Ayrık İşlemsel Yapılar)

Fırat İsmailoğlu, PhD

## Hafta 12:
## Graflar (Çizgeler) ve Ağaçlar - II

# Hafta 12
# Plan

1. Yollar – Bağlantı- Uzaklık
2. Bağlı Bileşenler
3. Genişlik Öncelikli Arama (Breadth First Search)
4. Derinlik Öncelikli Arama (Depth First Search)

# Yollar – Bağlantı- Uzaklık

Graflarla ilgili en önemli sorulardan biri graf uzerindeki bir düğümden başka bir düğüme kenarlari takip ederek gidip gidemeyecegimizdir.

Ilgili sorular:

- Mersin'den Sivas'a trenle gidebilir miyiz?
- Fatih Terim'den Fazıl Say'a bir arkadaslik zinciri var midir?
- Her defasında bir harf değiştirerek ve anlamlı kelimeler türeterek '*kayak*' kelimesinden '*sabun*' kelimesine varilabilir mi?

Bu gibi sorular bizi yol tanimina götürür.

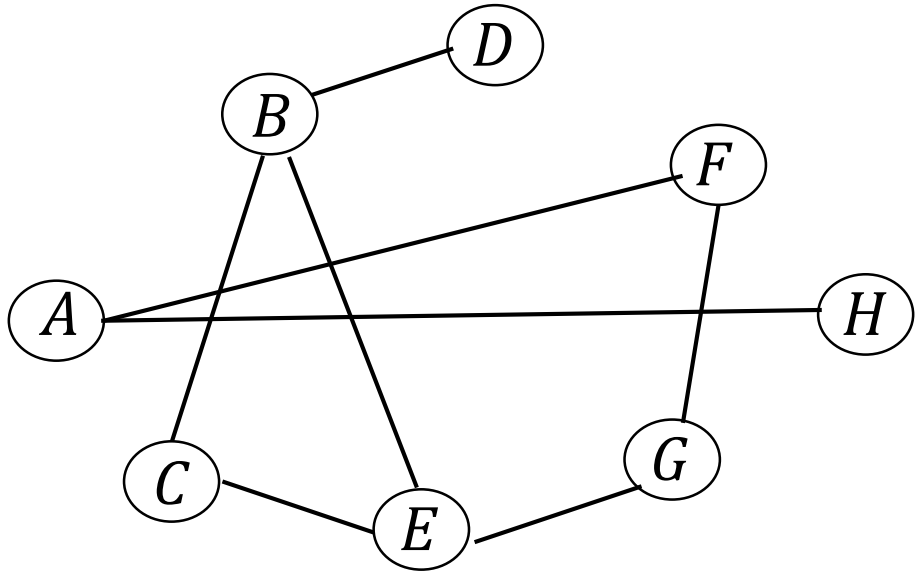
Yol (Path)

$G = (V, E)$ bir graf olsun. G icinde bir yol $(u_1, u_2, \dots, u_k)$ düğümler dizisidir öyleki bu dizideki her ardışık düğüm arasında bir kenar vardır:

$$(u_i, u_{i+1}) \in E \, , i \ \in \{1, \dots, k-1\}$$

# Yollar – Bağlantı- Uzaklık

ör.




1. $H$ düğümünden $E$ düğümüne bir yol var mıdır?
2. $D$ düğümünden $F$ düğümüne üç yol bulunuz.

Çözüm.

1. $H$'den $E$'ye yol: $(H, A, F, G, E)$
2. $D$'den $F$'ye bir yol: $(D, B, E, G, F)$.

   $D$'den $F$'ye bir baska yol: $(D, B, C, E, G, F)$.

$D$'den $F$'ye uçuncu yolu B-C-E düğümlerini birkaç kez ziyaret ederek (tur atarak) bulabiliriz: $D$'den $F$'ye üçüncü yol: $(D, B, C, E, B, C, E, G, F)$.

# Yollar – Bağlantı- Uzaklık

## Basit Yol (Simple Path)

Basit yol, yolu olşturan düğümlerin yalnızca bir kez ziyaret edildiği yoldur.

Bir onceki örnekteki $(H, A, F, G, E)$ ve $(D, B, E, G, F)$ birer basit yoldur fakat $B, C,$ ve $E$ düğümleri bir kac kez ziyaret edildiginden $(D, B, C, E, B, C, E, G, F)$ yolu basit değldir,

## Bağlı Düğümler (Connected Nodes)

Bir $G = (V, E)$ grafinda eğer $u \in V$ ve $v \in V$ düğümleri bağlı ise $u$ 'dan $v$'ye bir yol vardır.

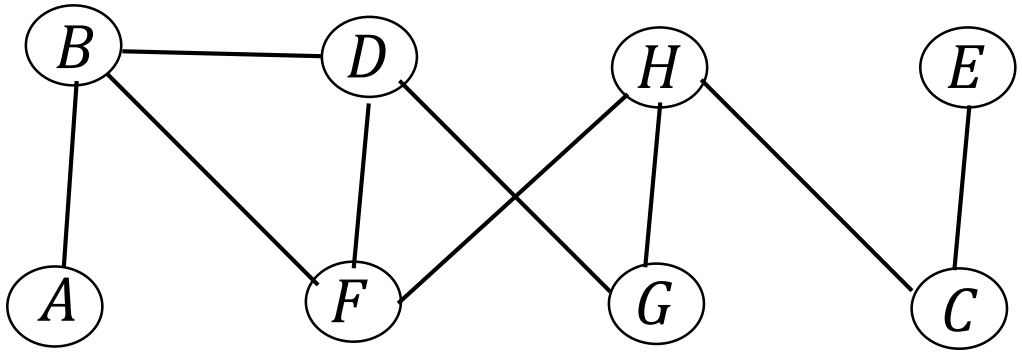
Bu tanım bağlı graf tanımını getirir.

## Bağlı Graf (Connected Graph)

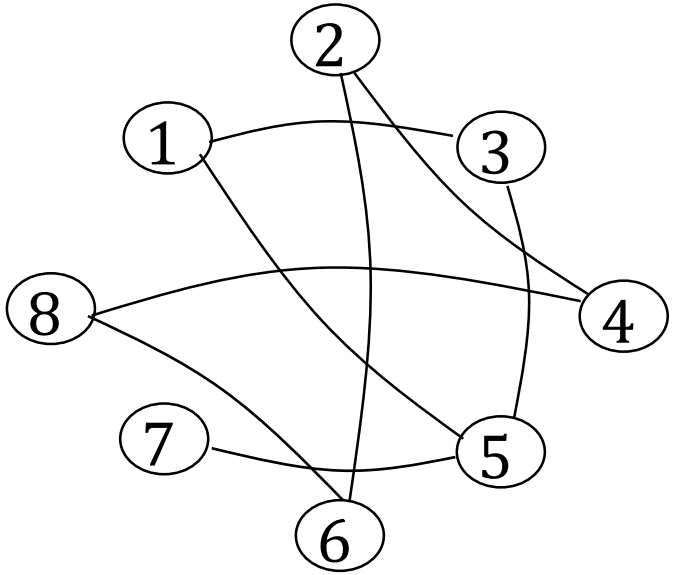
Bir $G$ grafında eğer tüm düğümler birbine bağlı ise (yani tüm düğümlerden düğer düğümlere bir yol varsa) bu grafa bağlı graf denir.

# Bağlı Graf (Connected Graph)

ör.



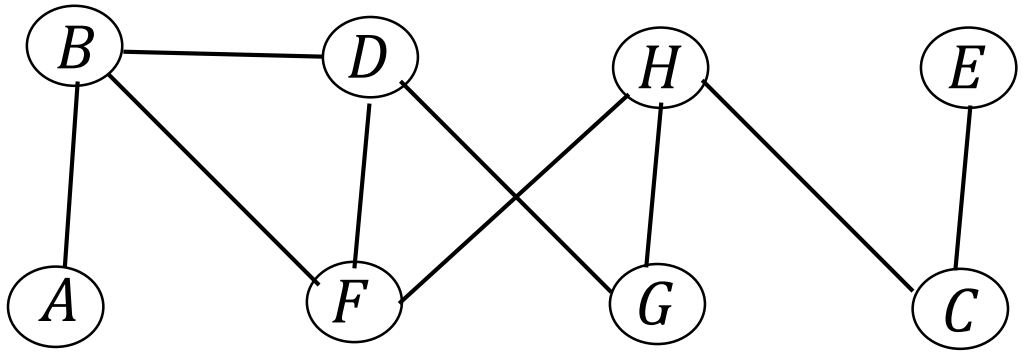

I.



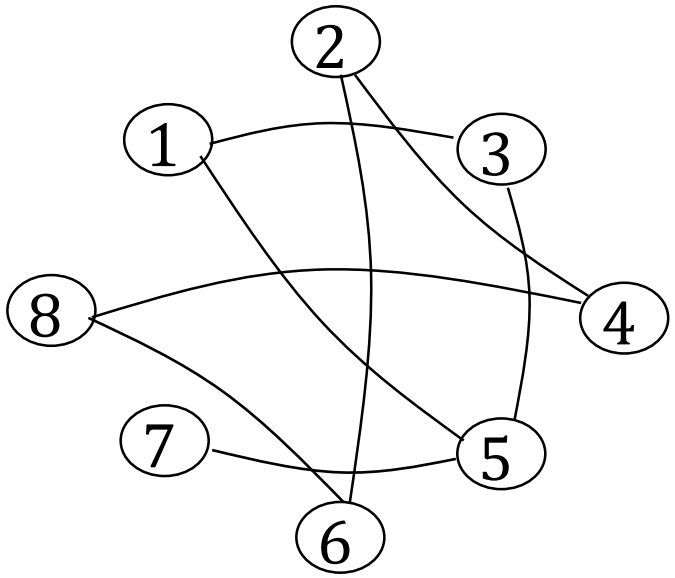

II.

# Bağlı Graf (Connected Graph)

ör.

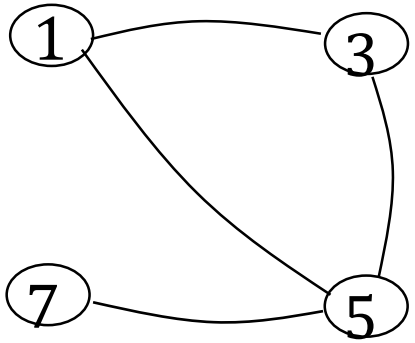


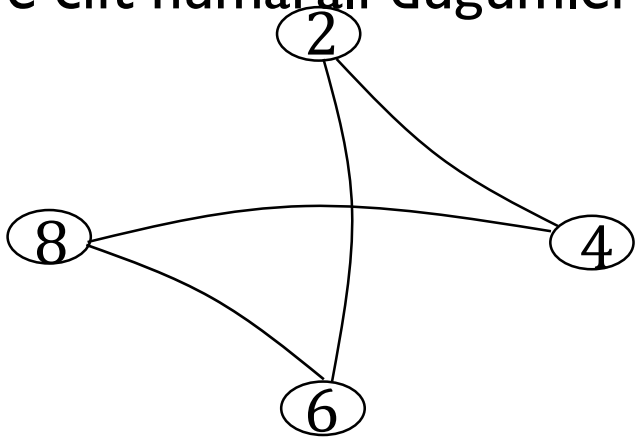

I.




II.

Yukardaki graflardan ilki bagli graftir; ikincisi ise bagli graf degildir. Fakat ikinci grafta tek numaralı düğümler kendi aralarinda baglidir, ve cift numarali düğümler kendi aralarinda baglıdır.



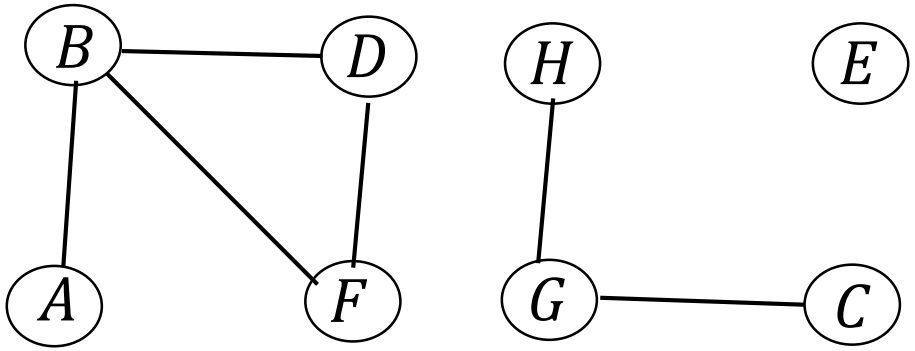

Bağlı graflar

# Bağlı Bileşen (Connected Component)

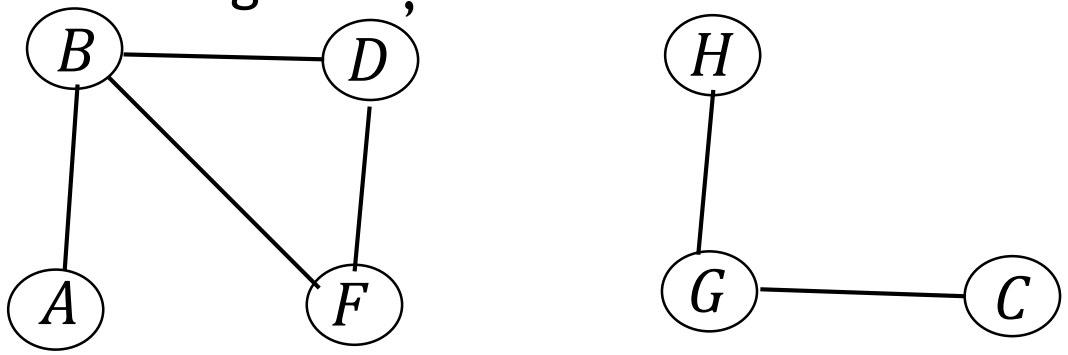
Bir onceki ornekte verilen graf bagli olmamasina ragmen bazi bileşenleri (parçaları) bağli idi; yani bu bileşenlerdeki düğümlerin tamamı birbirine bağli idi.

Bu şekilde, bir grafin birbirine bagli düğümlerinin oluşturduğu kümeye bağlı bileşen diyeceğiz.

ör.




Grafının bazı bağlı bileşenleri:




Bileşen 1 Bileşen 2 Bileşen 3

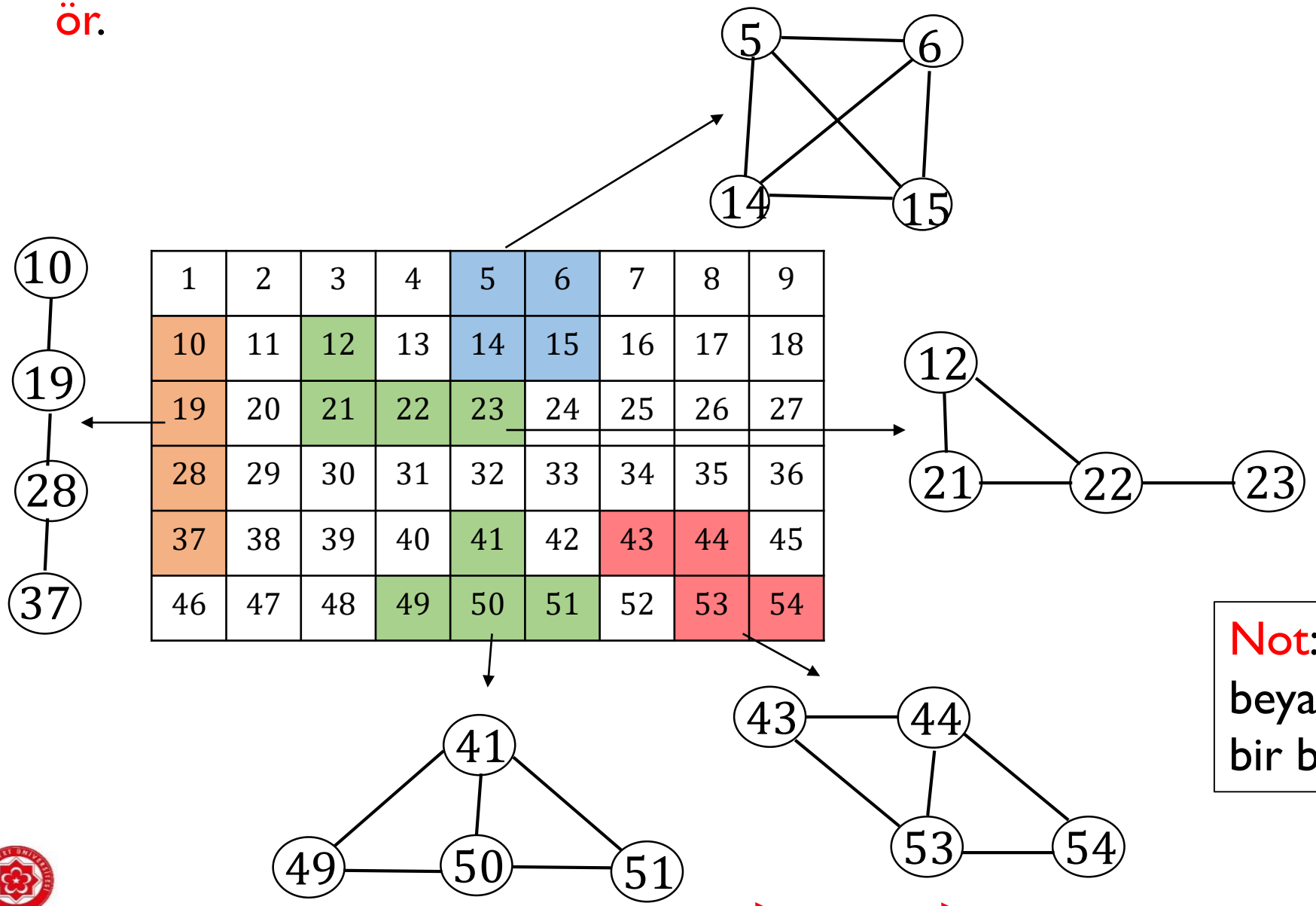
# Bağlı Bileşen (Connected Component)

ör. Bilgisayarla Görme (Computer Vision) alanındaki önemli görevlerden biri resimlerdeki objeleri anlayabilme (ayırt edebilme) dir. Örneğin otonom (sürücüsüz) bir araba etrafindaki cisimleri ayirt edebilmelidir.


Cisimleri aytirt etmek icin baglı bileşen kavramindan faydalanabiliriz. Bunun icin resimdeki pikselleri birer düğüm olarak ele alırız. Eğer iki piksel komşuysa (bitişikse) ve renkleri birbirine çok yakınsa bu piksellere karşılık gelen düğümler arası kenar çizilir. Böylece ortaya çıkan graftaki bağlı bileşenler birer cisme karşılık gelir.

# Bağlı Bileşen (Connected Component)

ör.



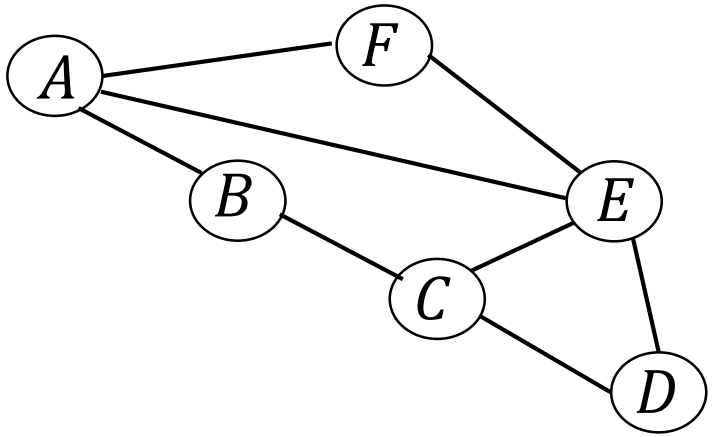

**Not**: Bu bağlı bileşenlere ilaveten beyaz pikseller birleşmesiyle de bir bağlı bileşen oluşur.

# En Kısa Yol (Shortest Path) – Uzaklık (Distance)

Şimdiye kadar yalnizca verilen iki düğüm arasinda bir yol olup olmadığı ile ilgilendik (yani verilen iki düğümun birbiriyle bağlı olup olmadığıyla). Şimdi ise bağlı iki düğüm arasındaki olabilecek en kısa yol ile ilgilenecegiz.

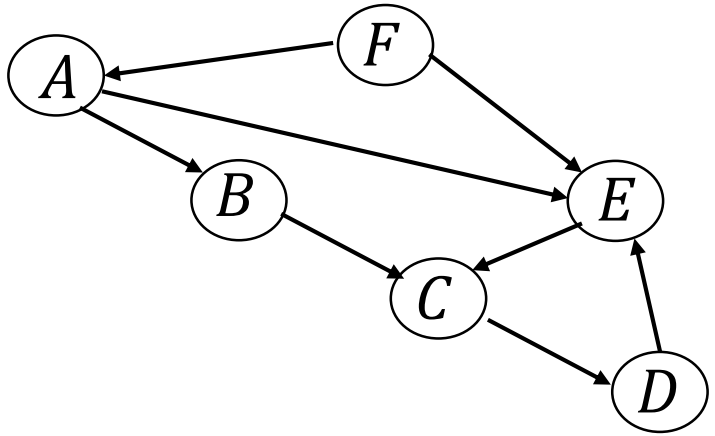
Kısaca iki düğüm arasındaki en kısa yol, bu iki düğümü birbirine bağlayan yollardan uzunluğu en az olan yoldur. En kısa yolun uzunluğu uzakliği verir.

ör.




$A$ düğümünün diğer düğümlere olan uzaklıkları :

| $A$ | $B$ | $C$ | $D$ | $E$ | $F$ |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 2 | 1 | 1 |




$A$ düğümünün diğer düğümlere olan uzaklıkları :

| $A$ | $B$ | $C$ | $D$ | $E$ | $F$ |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 1 | $\infty$ |

# Genişlik Öncelikli Arama / Breadth – First Search (BFS)

Genişlik öncelikli arama, grafta bir düğümden başlanarak ziyaret edilebilecek tüm düğümlerin listesini verir. Böylece herhangi iki düğümün birbiriyle bagli olup olmadığının kontrolu de yapılmiş olur.

## BFS Algoritmasının Çalışma Prensibi

BFS'de iki liste bulunur: ziyaret edilen düğümler listesi: $Z$ ve sıradaki düğümler listesi: $S$

Algoritmanin baslangicinda $Z$ listesi boş; $S$ listesi yalnızca başlangıç düğümünden oluşur.

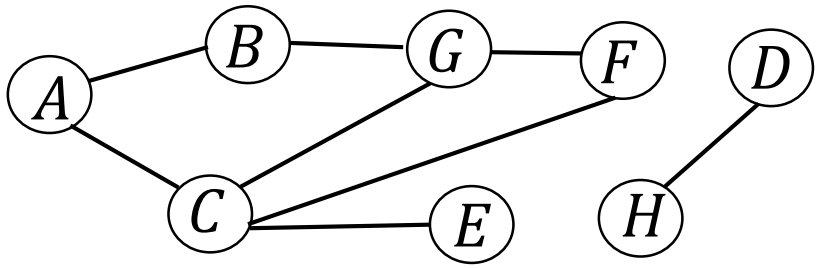
$S$ listesinde hiçbir düğüm kalmayana kadar her adımda:

$S$'nin ilk elemanı $Z$'ye konur ve bu elemanin eğer $Z$'de veya $S$'de olmayan komşuları varsa bu komşular $S$ listesinin sonuna konulur. (Boylece ilk gelen ilk gider)

$S$'de hic eleman kalmayinca BFS sonlanir. $Z$'deki elemanlar başlangıç düğümünden erişilebilen düğümler olur.

# Genişlik Öncelikli Arama / Breadth – First Search (BFS)

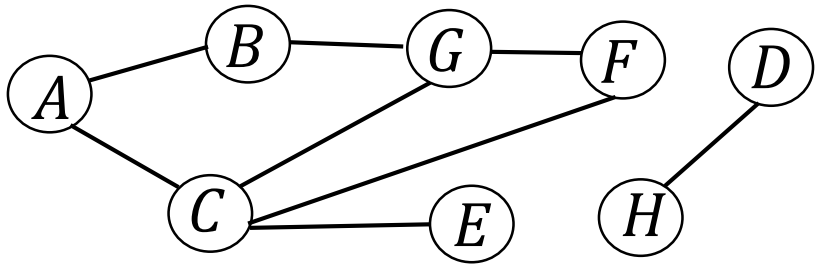
ör. Aşağıdaki grafta $A$ düğümünden başlayarak erişilebilen düğümleri BFS algoritması kullanarak bulunuz.




Çözüm:

Başlangıç:

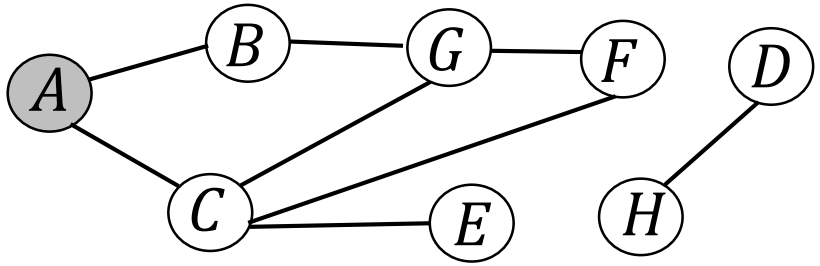
| $Z$ | $S$ |
|---|---|
| | $A$ |




---

Adım 1:

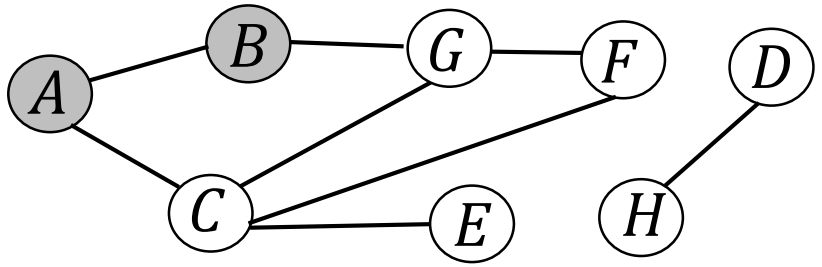
| $Z$ | $S$ |
|---|---|
| $A$ | $B$ |
| | $C$ |



# Genişlik Öncelikli Arama / Breadth – First Search (BFS)

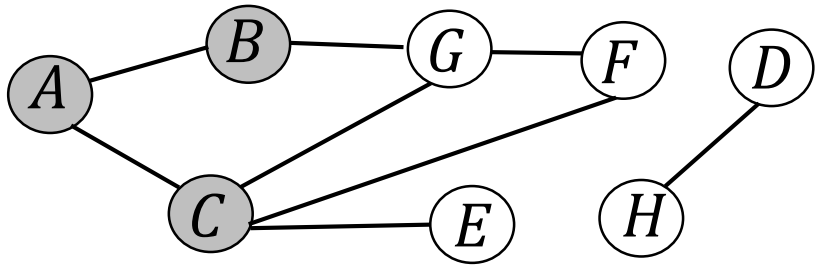
Adım 2:

| $Z$ | $S$ |
|---|---|
| $A$ | $C$ |
| $B$ | $G$ |



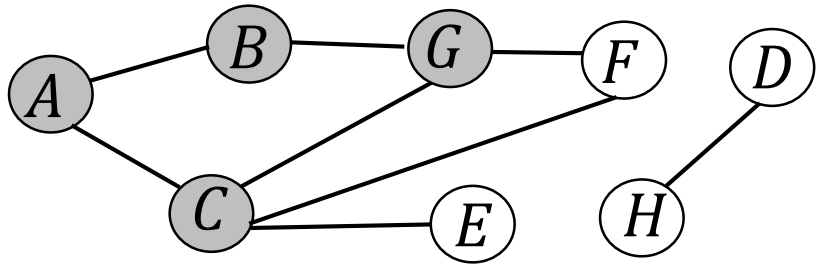

Adım 3:

| $Z$ | $S$ |
|---|---|
| $A$ | $G$ |
| $B$ | $E$ |
| $C$ | $F$ |




Adım 4:

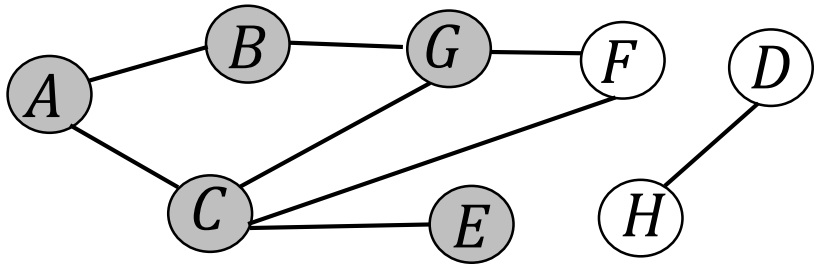
| $Z$ | $S$ |
|---|---|
| $A$ | $E$ |
| $B$ | $F$ |
| $C$ | |
| $G$ | |



# Genişlik Öncelikli Arama / Breadth – First Search (BFS)

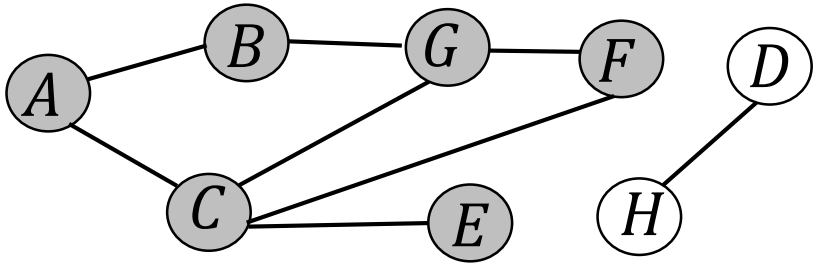
Adım 5:

| $Z$ | $S$ |
|---|---|
| $A$ | $F$ |
| $B$ | |
| $C$ | |
| $G$ | |
| $E$ | |




Adım 6:

| $Z$ | $S$ |
|---|---|
| $A$ | |
| $B$ | |
| $C$ | |
| $G$ | |
| $E$ | |
| $F$ | |



# Derinlik Öncelikli Arama / Depth – First Search (DFS)

Derinlik öncelikli arama, grafta aynı BFS'de olduğu gibi bir düğümden başlanarak ziyaret edilebilecek tüm düğümlerin listesini verir. BFS'den farkli olarak arama grafta dikey dogrultuda yapilir.

## DFS Algoritmasının Çalışma Prensibi

DFS'de iki liste bulunur: ziyaret edilen düğümler listesi: $Z$ ve sıradaki düğümler listesi: $S$

Algoritmanin baslangicinda $Z$ listesi boş; $S$ listesi yalnızca başlangıç düğümünden oluşur.

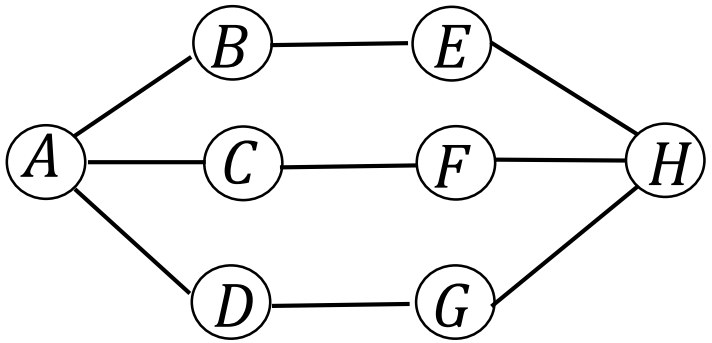
$S$ listesinde hiçbir düğüm kalmayana kadar her adımda:

$S$'nin ilk elemanının $Z$'de veya $S$'de olamayan bir komsu varsa bu komsu $S$'listesinin başına konur aynı zamanda $Z$ listesinin sonuna konur. Eğer böyle bir komşusu yoksa bu eleman $S$'den çıkartılır.

$S$'de hic eleman kalmayinca BFS sonlanir. $Z$'deki elemanlar başlangıç düğümünden erişilebilen düğümler olur.

# Derinlik Öncelikli Arama / Depth – First Search (DFS)

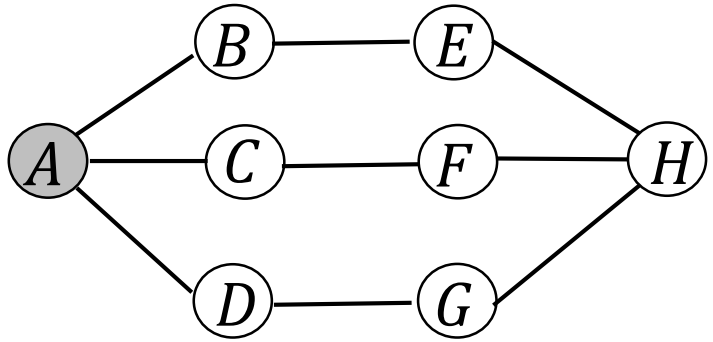
ör. Aşağıdaki grafta $A$ düğümünden başlayarak erişilebilen düğümleri DFS algoritması kullanarak bulunuz.




Çözüm:

Başlangıç:

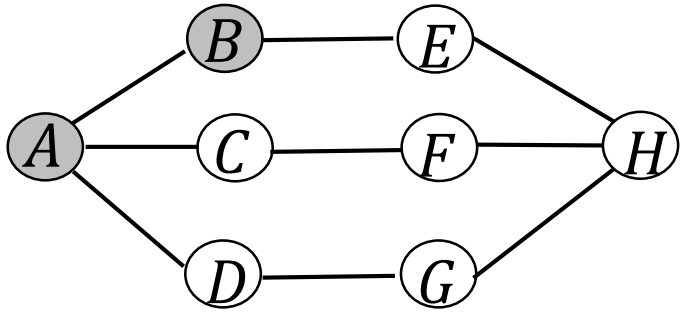
| $Z$ | $S$ |
|---|---|
| $A$ | $A$ |




---

Adım 1:

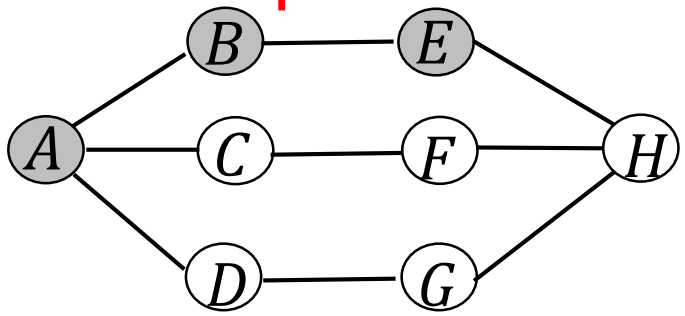
| $Z$ | $S$ |
|---|---|
| $A$ | $B$ |
| $B$ | $A$ |



# Derinlik Öncelikli Arama / Depth – First Search (DFS)

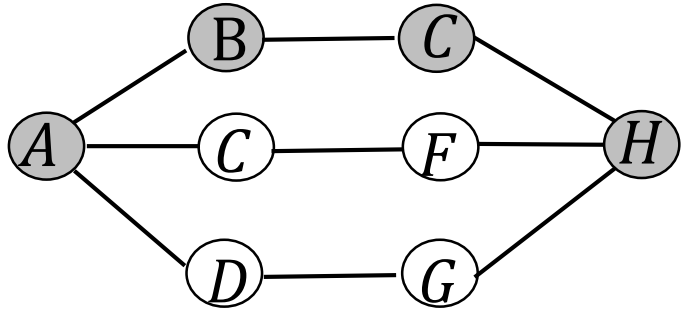
Adım 2:

| $Z$ | $S$ |
|---|---|
| $A$ | $E$ |
| $B$ | $B$ |
| $E$ | $A$ |



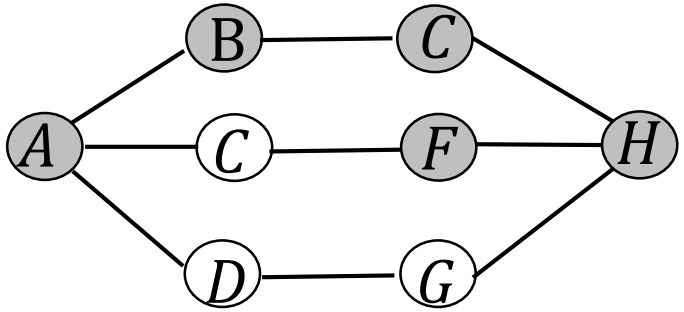

Adım 3:

| $Z$ | $S$ |
|---|---|
| $A$ | $H$ |
| $B$ | $E$ |
| $E$ | $B$ |
| $H$ | $A$ |




Adım 4:

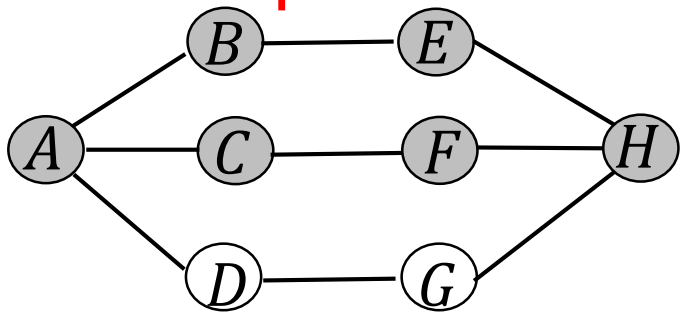
| $Z$ | $S$ |
|---|---|
| $A$ | $F$ |
| $B$ | $H$ |
| $E$ | $E$ |
| $H$ | $B$ |
| $F$ | $A$ |



# Derinlik Öncelikli Arama / Depth – First Search (DFS)

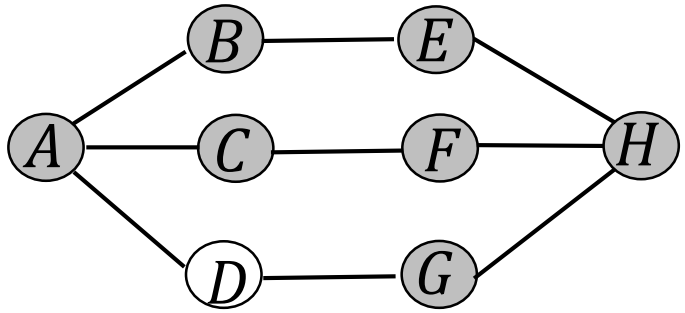
Adım 5:

| $Z$ | $S$ |
|---|---|
| $A$ | $C$ |
| $B$ | $F$ |
| $E$ | $H$ |
| $H$ | $E$ |
| $F$ | $B$ |
| $C$ | $A$ |




Adım 6:

| $Z$ | $S$ |
|---|---|
| $A$ | $G$ |
| $B$ | $H$ |
| $E$ | $E$ |
| $H$ | $B$ |
| $F$ | $A$ |
| $C$ | |
| $G$ | |

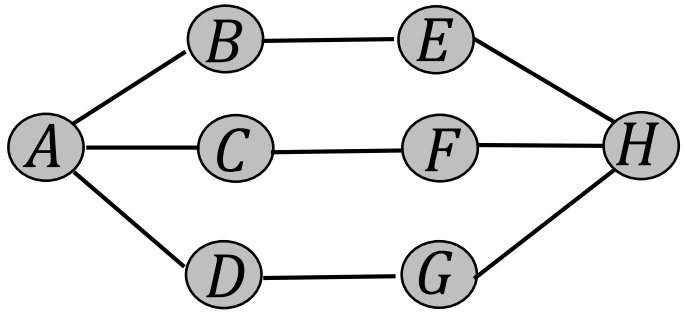



Not: Bu adımda sıranın en üstundeki $C$'nin $Z$'de yada $S$'de komsusu olmadigindan $C$'yi S listesinden sildik. Boylece $F, S$ listesinin en ustundeki eleman oldu. Ayni sekilde $F$'nin de uygun bir komsusu olmadigindan $F$'yi de sildik. Bu sekilde $S$ listesinin en ustune
$H$ cikti. $H$'nin $Z$'de ve $S$'de olmayan tek komsusu $G$ oldugundan $G$'yi $Z$'ye ve $S$'ye ekledik.

# Derinlik Öncelikli Arama / Depth – First Search (DFS)

Adım 7:

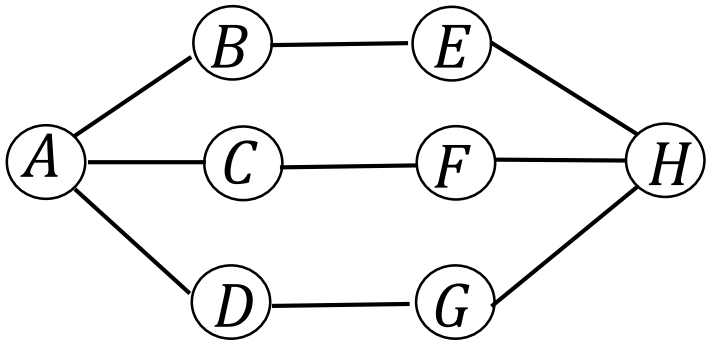
| $Z$ | $S$ |
|---|---|
| $A$ | $D$ |
| $B$ | $C$ |
| $E$ | $F$ |
| $H$ | $H$ |
| $F$ | $E$ |
| $C$ | $B$ |
| $D$ | $A$ |




Daha sonraki adımlarda $Z$'ye ve $S$'ye yeni bir dugum (eleman) eklenmez. Her bir adimda $S$ listesinin en ustundeki eleman cikarilir. $S$'de hic eleman kalmayinca DFS sonlanir.

# Genişlilk Öncelikli Arama / Depth – First Search (BFS)

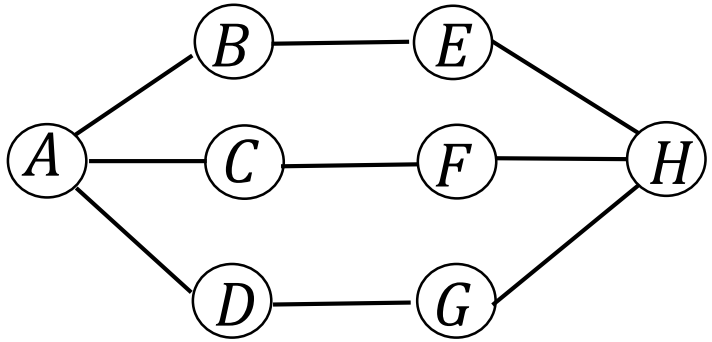
ör. Aşağıdaki grafta $A$ düğümünden başlayarak erişilebilen düğümleri BFS algoritması kullanarak bulunuz.




Çözüm:

Başlangıç:

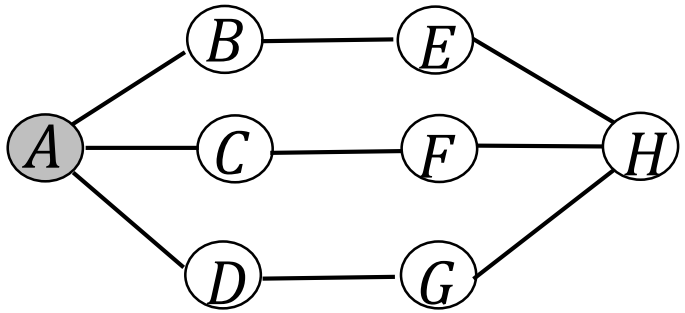
| $Z$ | $S$ |
|---|---|
| | $A$ |




---

Adım 1:

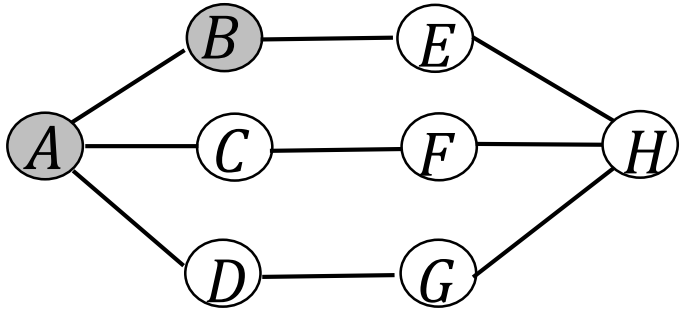
| $Z$ | $S$ |
|---|---|
| $A$ | $B$ |
| | $C$ |
| | $D$ |



# Genişlilk Öncelikli Arama / Depth – First Search (BFS)

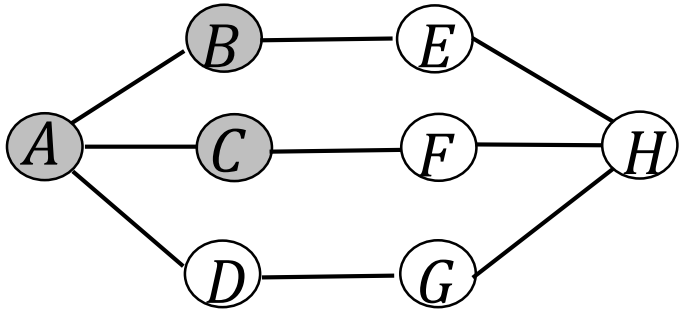
Adım 2:

| $Z$ | $S$ |
|---|---|
| $A$ | $C$ |
| $B$ | $D$ |
| | $E$ |



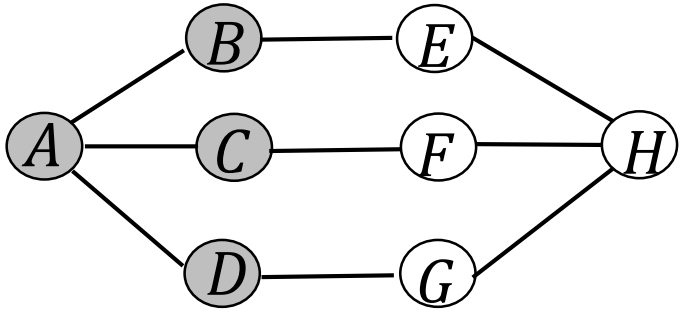

Adım 3:

| $Z$ | $S$ |
|---|---|
| $A$ | $D$ |
| $B$ | $E$ |
| $C$ | $F$ |




Adım 4:

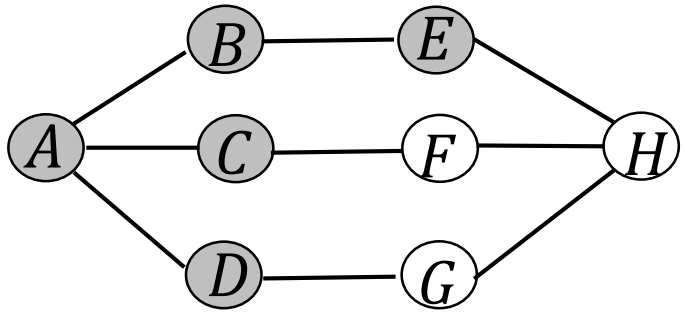
| $Z$ | $S$ |
|---|---|
| $A$ | $E$ |
| $B$ | $F$ |
| $C$ | $G$ |
| $D$ | |



# Genişlilk Öncelikli Arama / Depth – First Search (BFS)

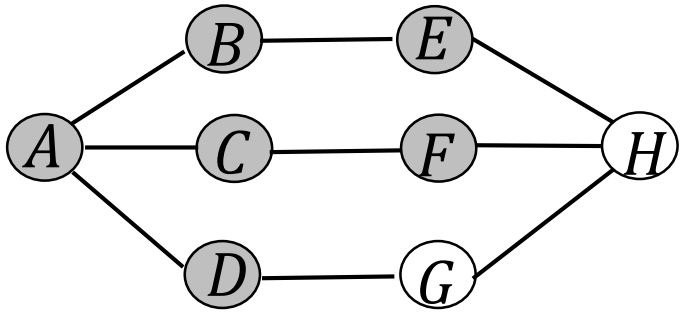
Adım 5:

| $Z$ | $S$ |
|---|---|
| $A$ | $F$ |
| $B$ | $G$ |
| $C$ | $H$ |
| $D$ | |
| $E$ | |




Adım 6:

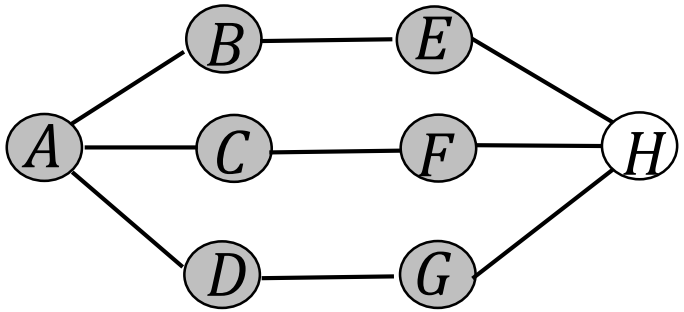
| $Z$ | $S$ |
|---|---|
| $A$ | $G$ |
| $B$ | $H$ |
| $C$ | |
| $D$ | |
| $E$ | |
| $F$ | |



# Genişlilk Öncelikli Arama / Depth – First Search (BFS)

Adım 7:

| $Z$ | $S$ |
|---|---|
| $A$ | $H$ |
| $B$ | |
| $C$ | |
| $D$ | |
| $E$ | |
| $F$ | |
| $G$ | |




Adım 8:

| $Z$ | $S$ |
|---|---|
| $A$ | $H$ |
| $B$ | |
| $C$ | |
| $D$ | |
| $E$ | |
| $F$ | |
| $G$ | |
| $H$ | |