

CENG 114 BİLGİSAYAR BİLİMLERİ İÇİN AYRIK YAPILAR

Prof. Dr. Tufan TURACI

tturaci@pau.edu.tr

- Pamukkale Üniversitesi
- Mühendislik Fakültesi
- Bilgisayar Mühendisliği Bölümü
- Hafta 15

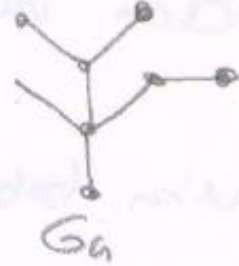
Ders İçeriği

- **Ağaçlar**
- **Ağaçların Bilgisayarlarda Saklanması**
- **En Küçük kapsayan Ağaçların Bulunması**
 - Prim Algoritması
 - Kruskal Algoritması

Ağaç Graflar ve Ağaç Grafların Bilgisayarlarda Saklanması

Tanım: Gevre içermeyen bağlantılı bir grafa ağaç denir.

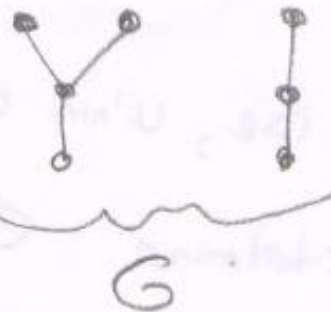
(11) m



\Rightarrow Hepsi ağaçtır.

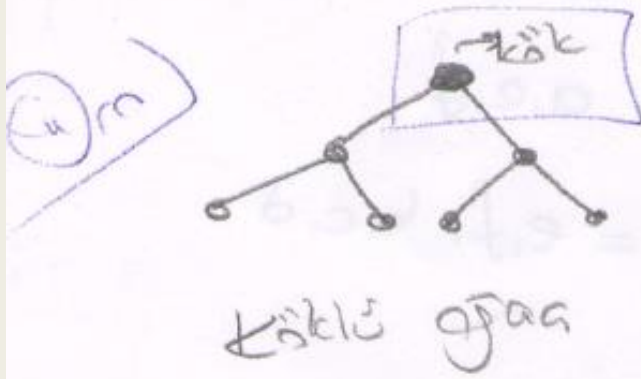
Tanım: Birden fazla ağaç grafin birleşmesiyle oluşan grafa Orman denir.

(12) m



\Rightarrow G , 2 parçalı bir ormandır.

Tanımı Çoğu zaman tepelerinden biri özel olan ağaçlar ile çalışılır. Özel olarak seçilen bu tepeye kök denir. Bu grafik de köklü ağaç dır.



1) Tanım: G köklü bir ağaç ve r noktesi de G 'nin kökü olsun. G ağacının r noktesinden farklı olan bir v noktesini alalım. v ile r noktesini birleştiren yolda v noktesinin komşusu olan noktaya v noktesinin babası denir.

(nm)



v 'nin babası a 'dır.



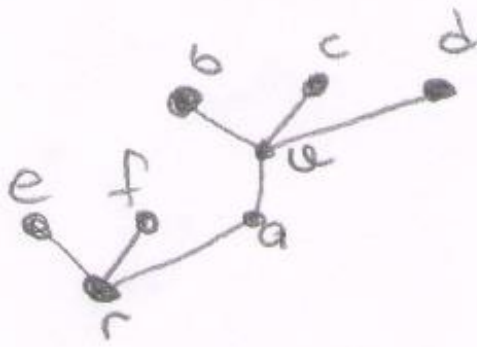
u 'nin babası r 'dir.

Tanım: u noktasının diğer noktalarına ise, u 'nin çocukları denir. G ağzının çocuğu olmayan noktalarına G 'nin yaprakları denir.

1 dereceli tefelere G 'nin yaprakları denir.

Öğr.: r bir kök olmak üzere r 'nin babası yoktur.

(Ör.)



G grafı

u 'nin babası = a

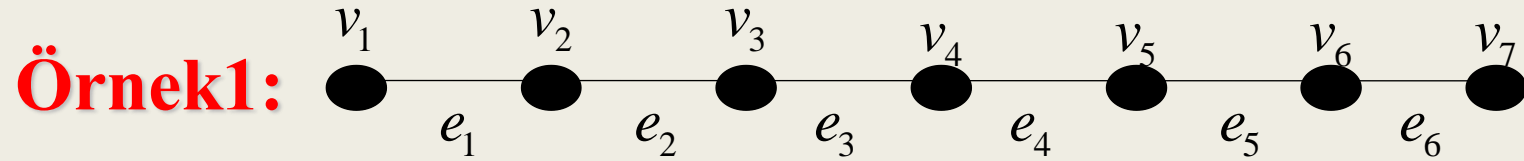
u 'nin çocukları = b, c, d

r 'nin çocukları = e, f

G 'nin yaprakları = e, f, b, c, d

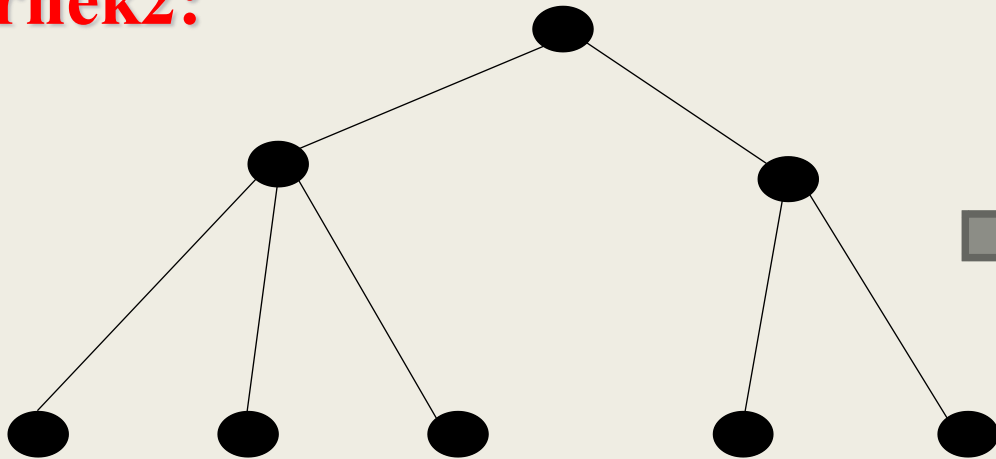
Teorem: G , p tepeli q ayrıtlı bir ağaç graf olsun. Bu durumda $p=q+1$ dir.

Kanıt: Tümevarım yardımıyla kanıt yapılır.




P_7 Grafı  7 tepeli, 6 ayrıtlıdır. Böylece, $7=6+1$ elde edilir.

Örnek2:



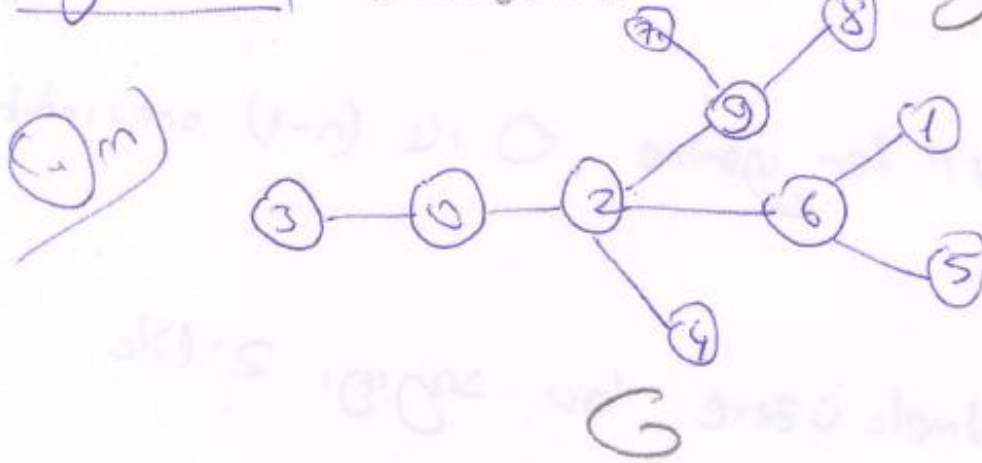
T Ağacı

 T ağacı; 8 tepeli, 7 ayrıtlıdır. Böylece, $8=7+1$ elde edilir.

④ = Ağaçları Bilgisayar Belleğinde Nasıl saklarız? =

4 farklı yöntem ile ağaçları bilgisayar belleğinde saklayabiliriz.

1. Yöntem: / Konsoluk matrisi yardımıyla.



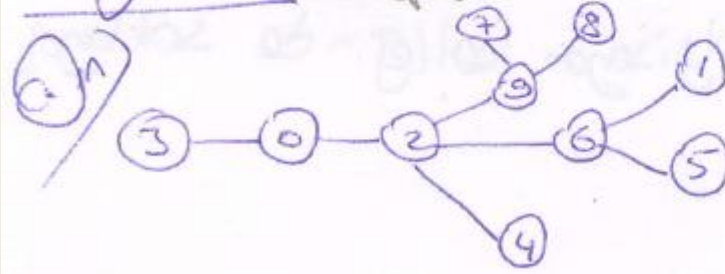
G ağacını ağızındaki matris yardımıyla bellekte saklayabiliriz.

G şeması aşağıdaki matris yardımıyla bellekte saklanabilir.

	1	2	3	4	5	6	7	8	9	0
1	0	0	0	0	0	1	0	0	0	0
2	0	0	0	1	0	1	0	0	1	1
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	1	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	1	0	0	0
9	0	1	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0

* Bu yöntemle; n taneli bir şema saklamak için n^2 bit gereklidir. Fakat matris simetrik olduğundan ve köşegen elemanları sıfır olduğundan $\frac{n^2-n}{2}$ bit yeterlidir.

2. yöntem / Tepeleri soldan doğruya, sadece kollarını soldan doğruya.



7 8 9 6 3 0 2 6 6
9 9 2 2 0 2 4 1 5

Böylece, bu yöntemde sadece 2 soluk kullandık ve $(n-1)$ sütun kullandık.
Ancak bu kez, "0" ile "1" ler yerine 0 ile $(n-1)$ arasındaki temsilleri kullandık.

★ n pozitif bir temsili olmak üzere bu sayıyı 2' ile sistemde yapmak için $\log_2 n + 1$ bit gerekir.

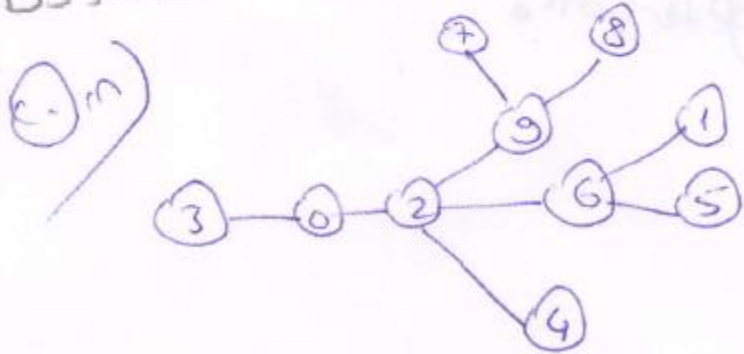
Bu durumda; n tane bir işlem bellekte soldan doğruya için;

$$2 \cdot (n-1) \cdot (\log_2 n + 1) < \frac{n^2 - n}{2}$$

Böylece, 2. yöntem için daha az bit gereklidir.

3. yöntem / (Father Code)

Bu yöntemde 0 nolu taşı kök kabul ederiz. 2. yöntemle benzer olarak yine kenarlar satırlar. Ancak kural üst satıra çocuklar, alt satıra babalar yazılır.



çocuklar	1	2	3	4	5	6	7	8	9
babalar	6	0	0	2	6	2	9	9	2

Bu yöntemde,

- 0 (kötü) üst sırada bulunmaz.
- 0 heria tüm noktalar üst sırada bulunmak zorundadır.
- Üst sırada hiç bir nokta 2 kez kullanılmaz.
- Üst sıradaki sayıları seçilmeye göre gelir. Çünkü

Sınırlılar.

- Sadece alt sınır seçilir.
- Sıra olarak seçilmeye izin $(n-1) \cdot (\log_2 n + 1)$ kadar ihtiyac vardır.

⊗ Her şeye için bu kod yazılabilir. Fakat her kod
bir şeye belirtilebilir!!

(m)

(0, 1, 2, 3, 4, 5, 6, 7)

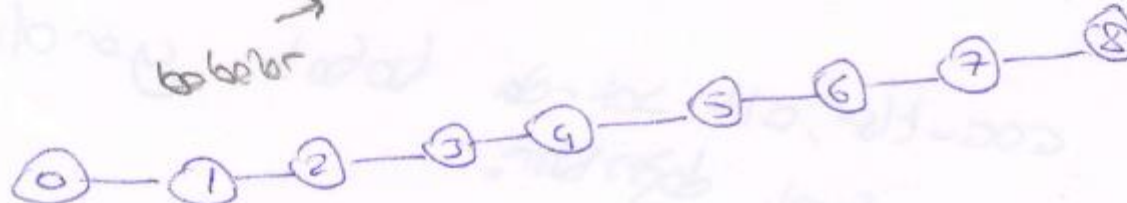
⇒ 8 sayı var 5 tane var "father code"
dizisi bir ağ için

Durum?

→ çocuklar

1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7

→ babalar



⇒ P_g grafi

(n)

(0, 0, 0, 0, 0)

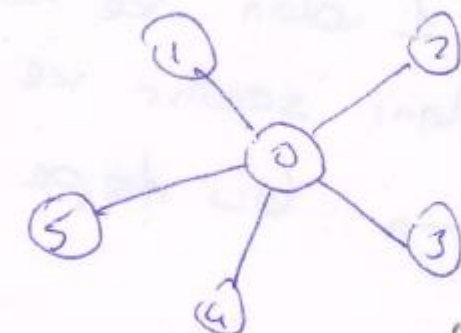
⇒ 5 sayı var 6 tane var "father code"
dizisi bir ağ için

Durum?

→ çocuklar

1	2	3	4	5
0	0	0	0	0

→ babalar



K_{1,5} yıldız grafi.

ör

(7, 6, 5, 4, 3, 2, 1, 0) bir gecen "let her code" olurmu?

1	2	3	4	5	6	7	8
7	6	5	4	3	2	1	0

let her code olmaz.

ör

(2, 3, 1, 2, 3, 1, 2, 3) "bir gecen "let her code" olurmu?

1	2	3	4	5	6	7	8
2	3	1	2	3	1	2	3

\Rightarrow 0 (kötü) yollar.
let her code olmaz!!

4. yöntem (Prüfer Kod)

Bu yöntem 3. yöntemin iyileştirilmiş versiyonudur. O yine kök olarak kabul edilir.

Bu yöntemde n tane bir sıra olan $(n-1)$ çift yerine $(n-2)$ çift silinir.

yöntemde;

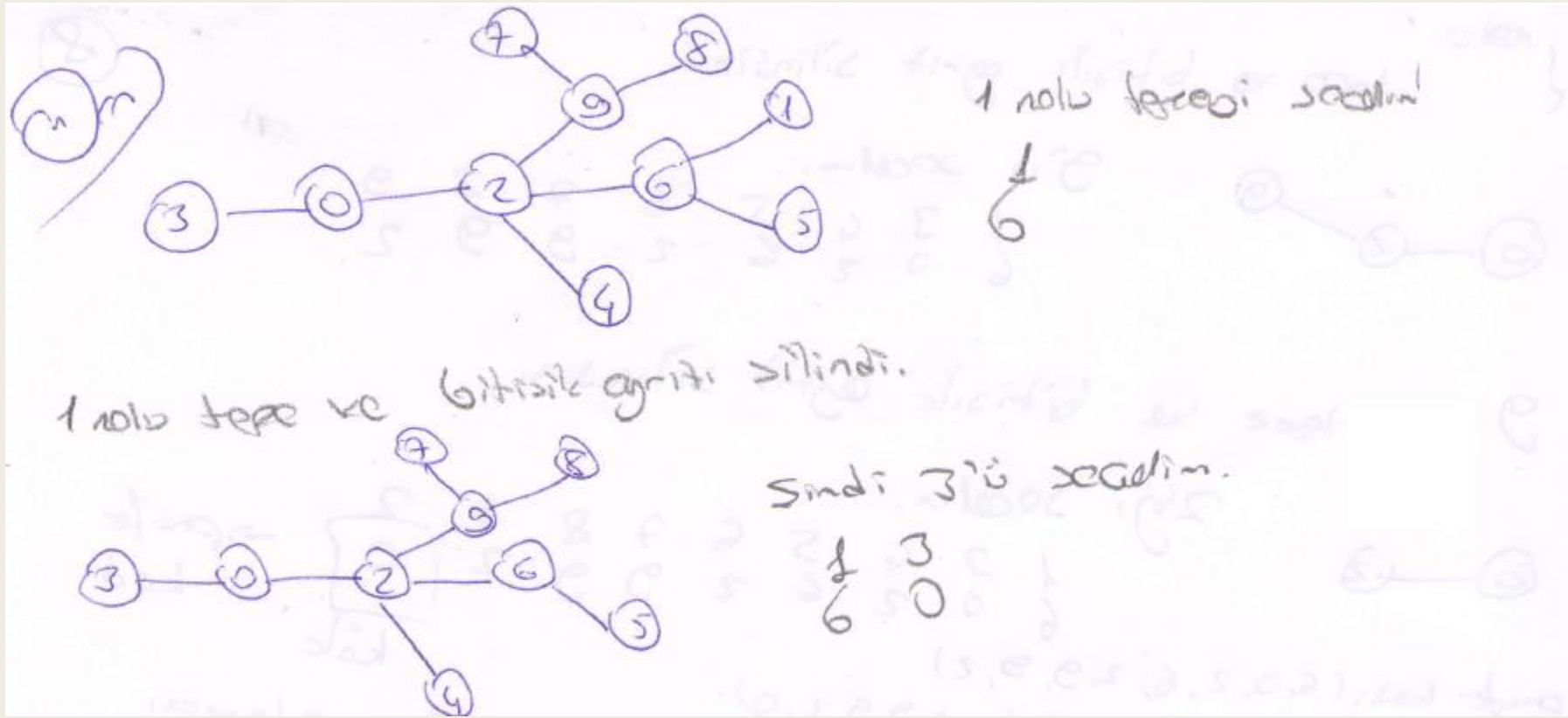
üst sırada çiftler, alt sırada tekler yer alır.

Ancak bu şekilde üst sıra sıralı değildir.

Sıralama şu şekilde yapılır:

A1: Derecesi 1 olan ve kökten farklı olan noktelerden en küçük olanı seçilir ve boşluğu birliğe yerilir.

A2: Daha sonra bu tepe ve birliğe çifti silinerek A1'e git.



3 nolu tere ve bitirile egrisi silindi.

Simdi 4'is degerlem.

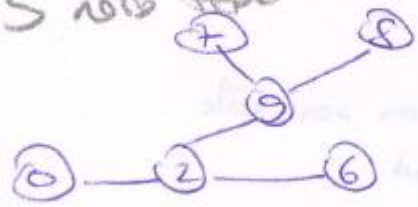
1	3	4
6	0	2

4 nolu tere ve bitirile egrisi silindi.

Simdi 5'is degerlem.

1	3	4	5
6	0	2	6

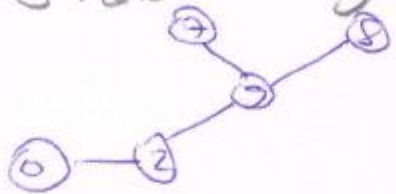
5 no lu tepe ve bitirilemiyor silindi.



Sindi 6'yi soçelim.

1	3	4	5	6
6	0	2	6	2

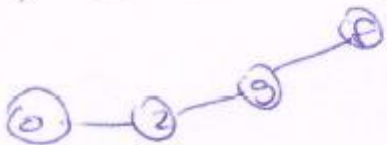
6 no lu tepeyi ve bitirilemiyor silelim.



Sindi 7'yi soçelim.

1	3	4	5	6	7
6	0	2	6	2	9

7 ve bitirilemiyor silindi



8'i soçelim

1	3	6	5	6	7	8
6	0	2	6	2	9	9

8'inci tere ve bitizile aynit silinsin.



9'u silelim.

1	3	4	5	6	7	8	9
6	0	2	6	2	9	9	2

9'ncü tere ve bitizile aynit silinsin.



2'yi silelim.

1	3	4	5	6	7	8	9	2
6	0	2	6	2	9	9	2	0

→ prefer kod

kök

prefer kod: (6, 0, 2, 6, 2, 9, 9, 2)

Extended prefer kod = (6, 0, 2, 6, 2, 9, 9, 2, 0)

En son kök köleceğinden alt solun son elemanı

0 (sıfır) olur.

Teorem: Prüf dizisinin 2. satırı 1. satırın çolukları.

(n, m) Derece dizisi $(3, 2, 1, 2, 1, 1)$ ve Prüf kodu'su

$(0, 1, 0, 3)$ olan geçici giriniz.

Tere	0	1	2	3	4	5
Derece	3	2	1	2	1	1

Şimdi 2'ye 0'in derecesini 1 indirelim.
Schublen den 2 ile bitmiş oluyor.

Tere	0	1	2	3	4	5
Derece	2	2	0	2	1	1

\otimes Derece en küçük olan ilk tere 2'ye tepedir. 0 zaman

2 ? ? ? ?

0 1 0 3 0

en son külc
extended Prüf kod.

\otimes Derece en küçük ilk tere 4

0 zaman

2 4 ? ? ?

0 1 0 3 0

4'ün ve 1'in derecesini 10 zollalım.

En küçük derece: ilk tere = 1

Tere	0	1	2	3	4	5
Derece	2	1	0	2	0	1

2	4	1	?	?
0	1	0	3	0

Simdi 0'ın ve 1'in derecesini 10 zollalım.

0 kenar (köşe) en küçük derece: tere = 5

Tere	0	1	2	3	4	5
Derece	1	0	0	2	0	1

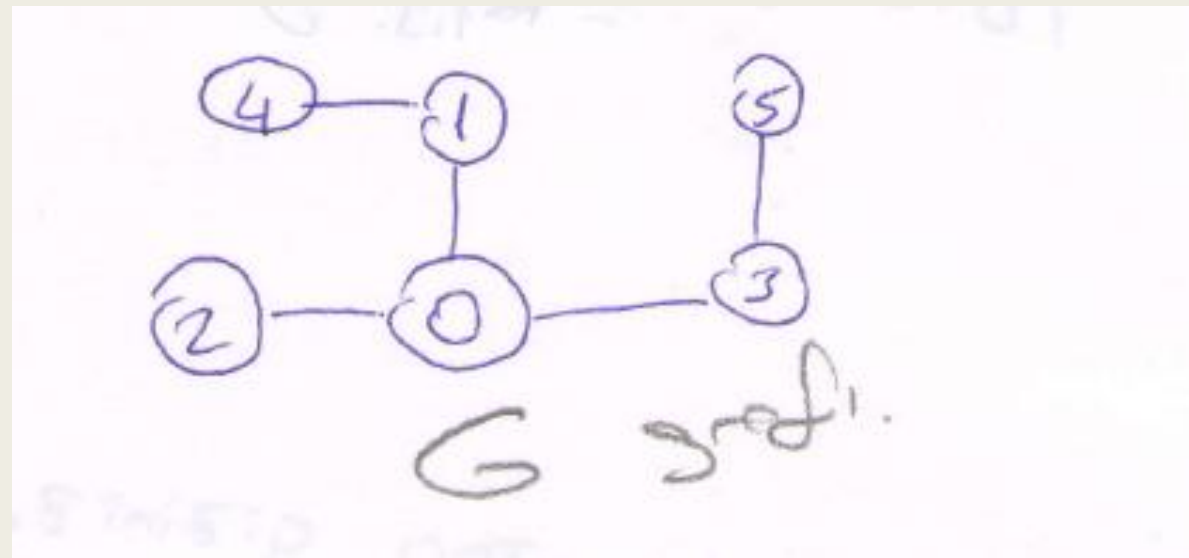
2	4	1	5	?
0	1	0	3	0

Simdi 5 ve 3'ün derecesini 10 zollalım.

0 kenar derece: en küçük tere = 3

Tere	0	1	2	3	4	5
Derece	1	0	0	1	0	0

2	4	1	5	3
0	1	0	3	0



Tanım: (Düzensel Kod):

Tepe sayısı n olan bir geçi alalım. Bu geçi kaleden olsun ve n tane başlayıp geçen tüm ayrıtlardan 2 kez geçerek bir tur yoyalım. Bu turda ulaştığımız tepe bir önceki tepenin karşısına ise buna "1" ile, başka ise "0" ile gösterilmek üzere elde edilen koda düzensel kod denir.

Örnek

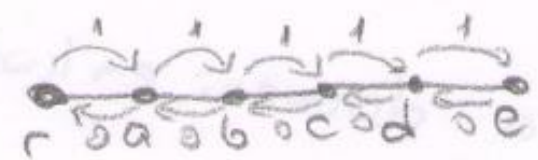


Turumu?

refgfeccbaabceder
 √ √ √ √ √ √ √ √ √ √ √ √ √ √ √
 1 1 1 0 0 1 1 1 0 0 1 0 0 0

Öm)

1111100000 düzensel kodum karşılık gelen ağac.

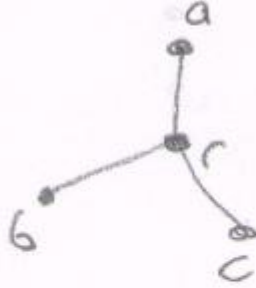


= P6 yol grafi

Qm)

101010

düzlemsel koduna karşılık gelen ağac.



rarbrcr
vvvvvv
101010

= K_{1,3} yıldız graf.



Qm)

1100011100 düzensiz kodlu ağacı çiziniz.

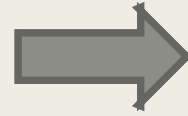
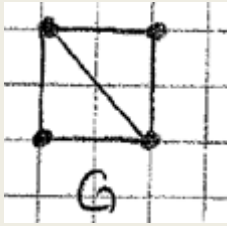
En başta 2 tane 1, felaket sonra enle enlaya 3 sıfır

Obduktan böyle bir ağac çizilemez.

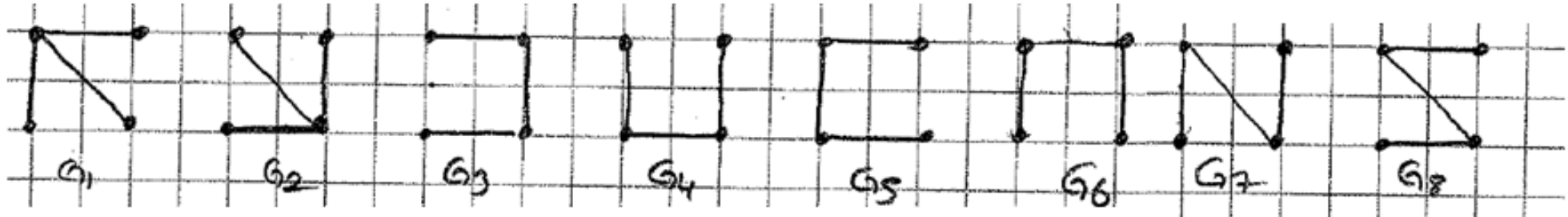
En Küçük Kapsayan Ağaçların Bulunması (Dallanmış Alt ağaçlar)

- Bir G grafının tüm tepelerini içeren birleştirilmiş bir alt grafa dallanmış **alt graf** denir.
- Eğer dallanmış alt graf çevre içermiyorsa **dallanmış alt ağaç** (kapsayan ağaç- **spanning tree**) denir.

Örnek:



G grafının dallanmış alt ağaçları nelerdir?

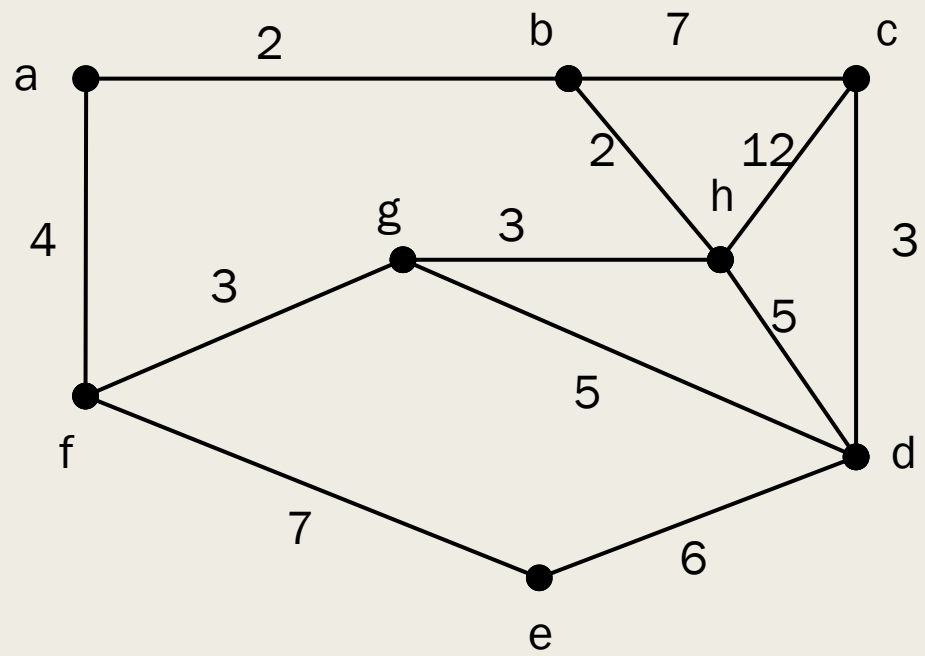


Kruskal ve **Prim** algoritmaları yardımıyla ayrıtları ağırlıklandırılmış bir G grafindaki en küçük maliyetli kapsayan ağaçlar bulunur.

Kruskal'ın Algoritması:

Bu algoritmayı aşağıdaki problemi, ele alarak inceleyeceğiz.

Problem: Bir eğlence parkının oluşturulmak istendiğini kabul edelim. Bu parkta yapılması olası olan tüm yollar daha önce belirlenmiş olup, parkın sahibi bu yollardan en az maliyetlisini seçerek, yaptırmak istiyor. Aşağıdaki grafta, parkın yapısı, olası tüm yollar ve herhangi iki nokta arasında yapılacak yolun maliyeti belirtildiğine göre en az maliyetli yolun hangisi olduğunu bulunuz.



Bu problemin çözümü için en küçük ağırlıklı dallanmış ağacı bulmalıyız. Bunun için Kruskalın algoritmasını kullanırız.

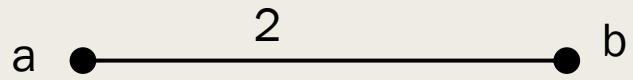
KRUSKAL ALGORİTMASI

Adım1: Graftaki en küçük ağırlıklı ayrıtı (birden fazla ise herhangi birisini) seç, ve bu ayrıt ile ağacı oluşturmaya başla.

Adım2: Henüz ağaçta olmayan ve ağaca eklendiğinde çevre içermeyen en küçük ağırlıklı bir ayrıtı seç ve ağaca ekle

Adım3: Dallanmış ağaca sahip olup olmadığını kontrol et, Eğer sahip ise dur, aksi halde Adım 2 ye git.

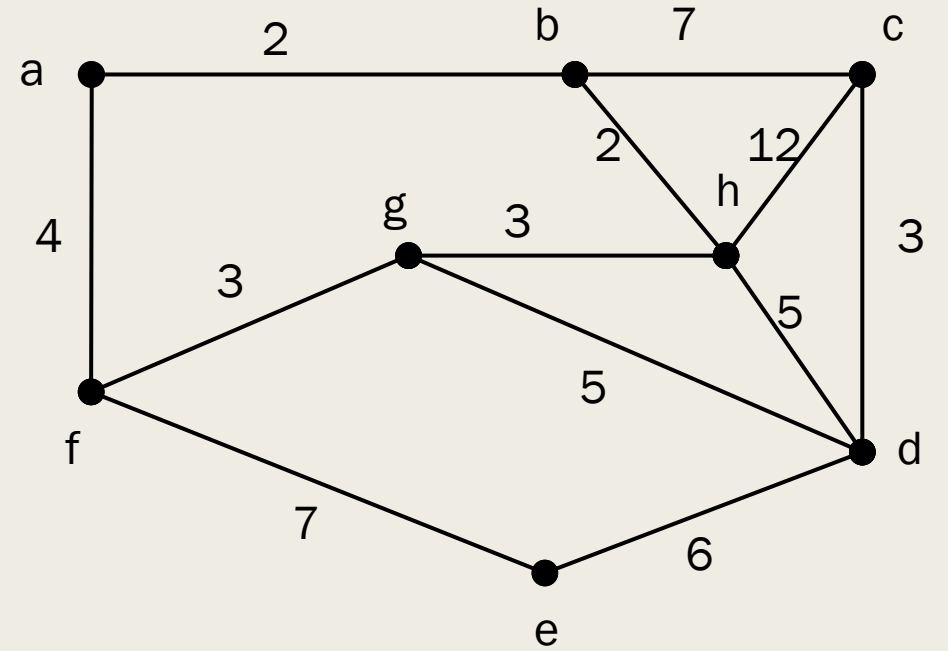
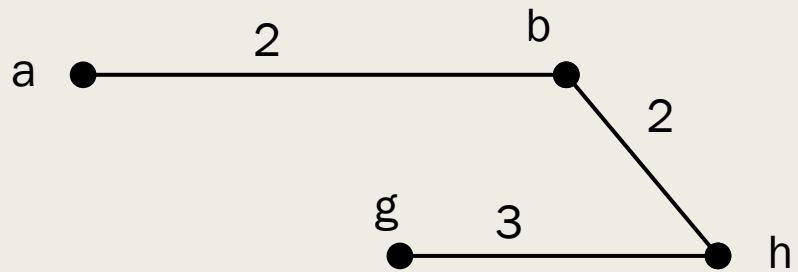
1)



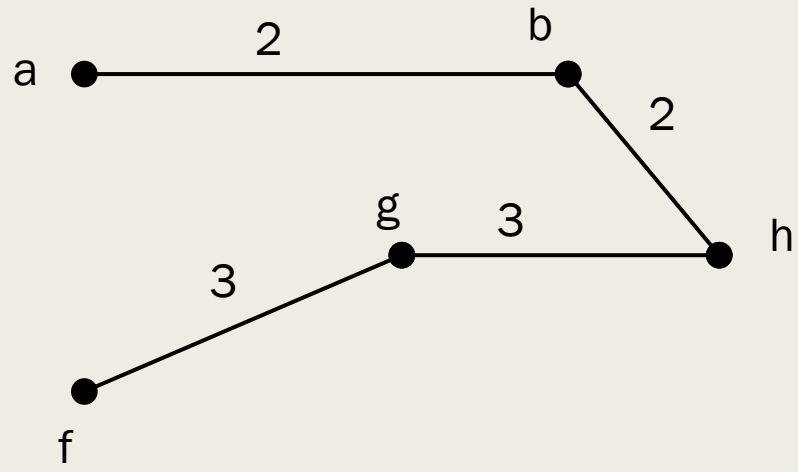
2)



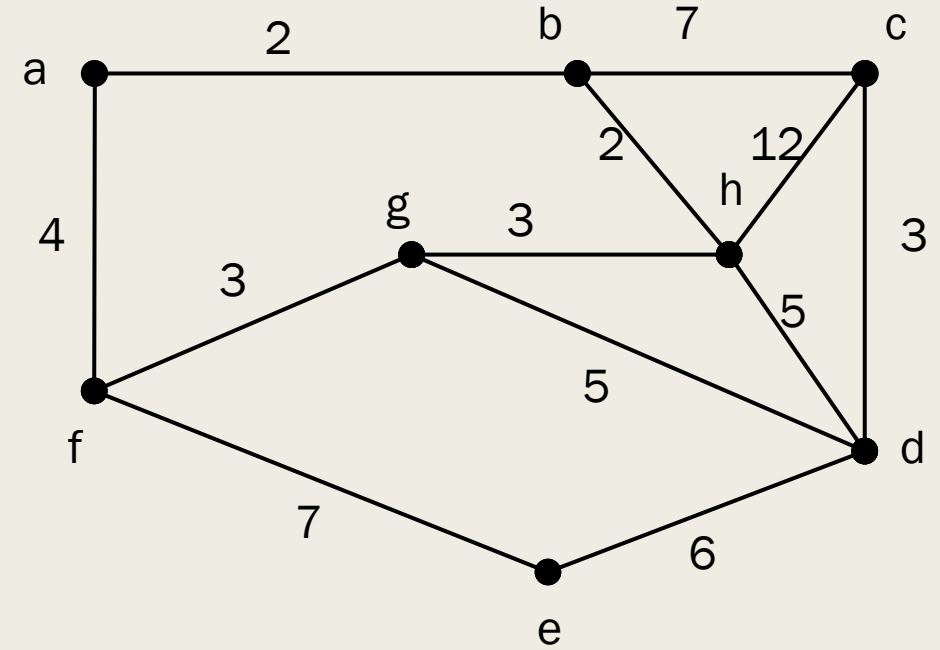
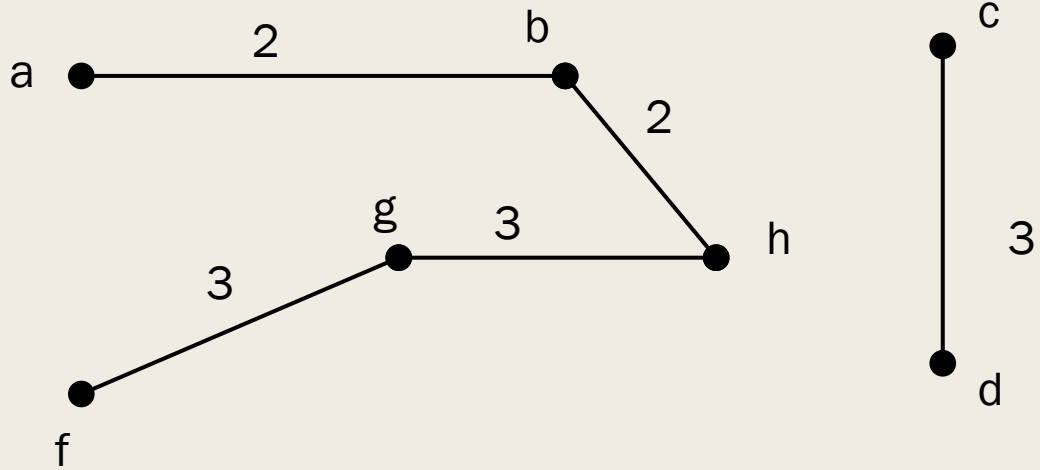
3)



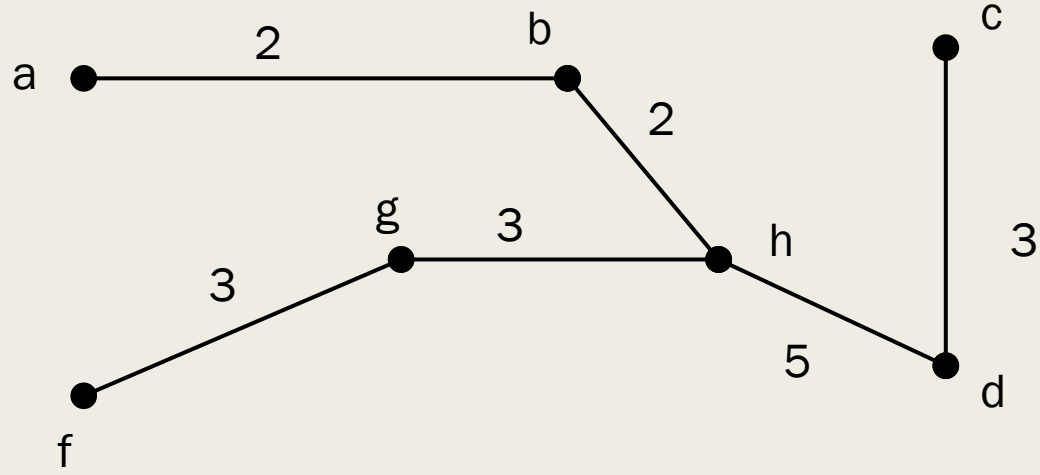
4)



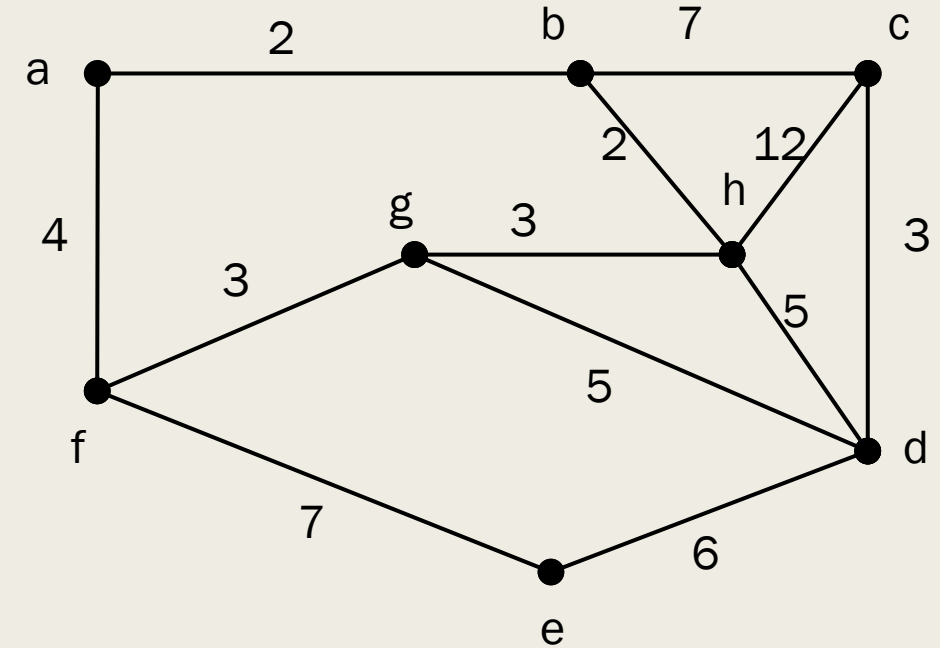
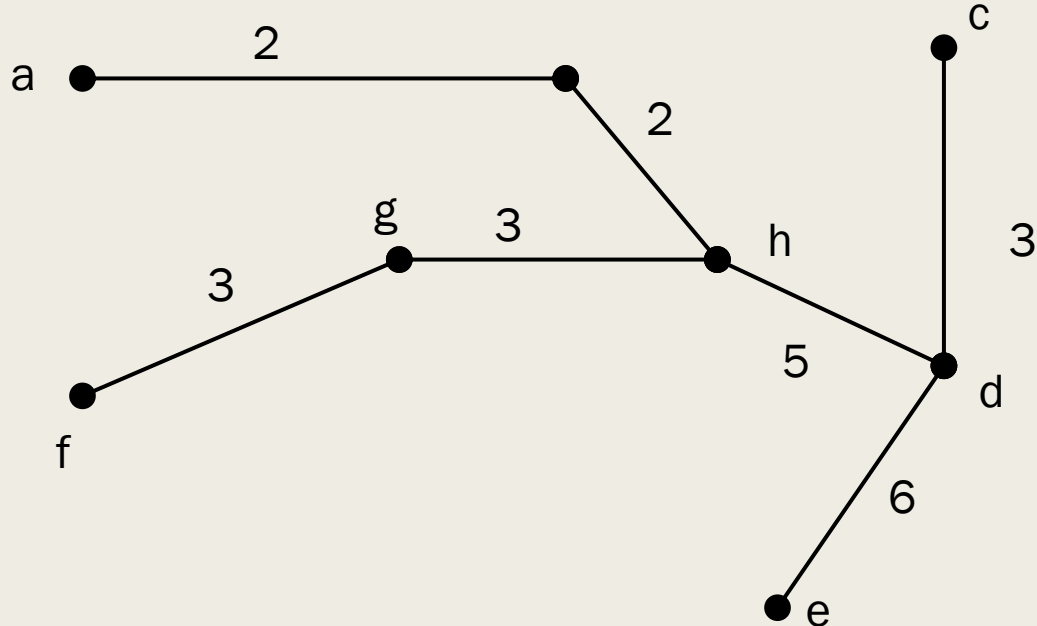
5)



6)



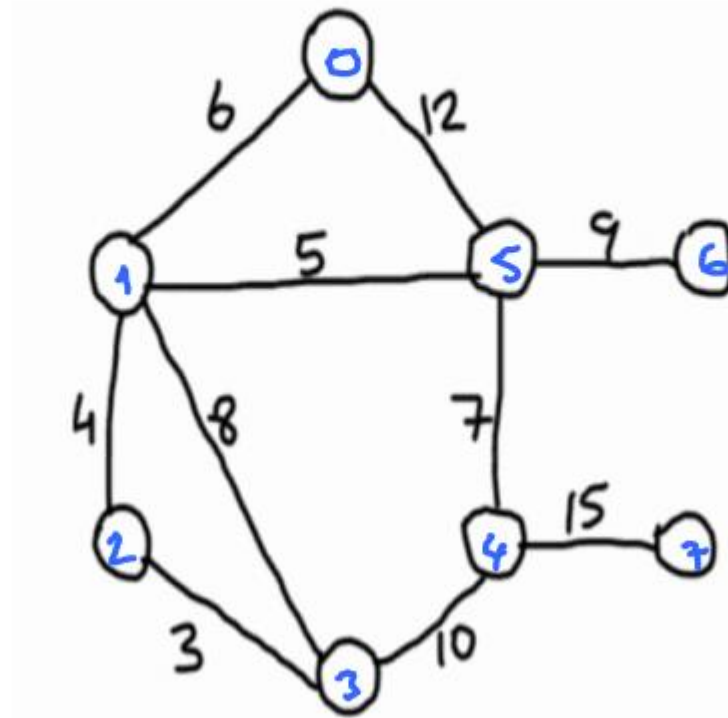
7)



Sonuç olarak; minimum ağırlıklı kapsayan ağaç elde edildi. (Ağırlık=24)

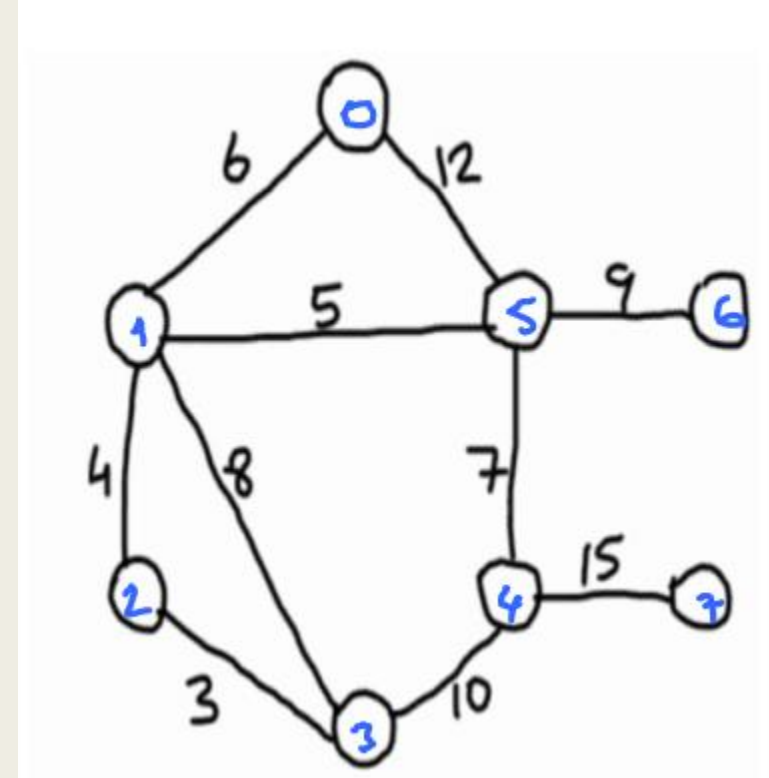
Kruskal Algoritması Nasıl Programlanabilir?

Örnek: Aşağıda verilen G grafinin en küçük kapsayan ağacını **Kruskal Algoritması** ile bulunuz.



G grafi

<u>1. type</u>	<u>2. type</u>	<u>Ağırlık</u>
0	1	6
1	2	4
2	3	3
3	4	10
4	5	7
5	0	12
1	5	5
1	3	8
5	6	9
4	7	15



Ağırlığa göre küçükten büyüğe sıralanır:

<u>1. tane</u>	<u>2. tane</u>	<u>Ağırlık</u>
2	3	3
1	2	4
1	5	5
0	1	6
4	5	7
1	3	8
5	6	9
3	4	10
5	0	12
4	7	15

0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0

Tepeleğin
ziyaret edilip
edilmediği
kontrolü

1.02m

0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0

<u>1. tepe</u>	<u>2. tepe</u>	<u>Ağırlık</u>	
2	3	3	→ Ağırlık Eklendi.
1	2	4	
1	5	5	
0	1	6	
5	5	7	
1	3	8	
5	6	9	
5	4	10	
5	0	12	
4	7	15	

2. adım

0	1	2	3	4	5	6	7
0	1	1	1	0	0	0	0

1. tane

2. tane

Ağırlık

2

3

3

→ Ağırlık Eklendi.

1

2

4

→ Ağırlık Eklendi.

1

5

5

0

1

6

4

5

7

1

3

8

5

6

9

3

4

10

5

0

12

4

7

15

3. adım



1. tane

2. tane

Ağırlık

2

3

3

→ Ağrıt Eklendi.

1

2

4

→ Ağrıt Eklendi.

1

5

5

→ Ağrıt Eklendi.

0

1

6

5

5

7

1

3

8

5

6

9

5

4

10

5

0

12

4

7

15

4. adım



<u>1. tane</u>	<u>2. tane</u>	<u>Ağırlık</u>		
2	3	3	→	Ağırlık Eklendi.
1	2	4	→	Ağırlık Eklendi.
1	5	5	→	Ağırlık Eklendi.
0	1	6	→	Ağırlık Eklendi.
4	5	7		
1	3	8		
5	6	9		
3	4	10		
5	0	12		
4	7	15		

5. adım

0	1	2	3	4	5	6	7
1	1	1	1	1	1	0	0

<u>1. tane</u>	<u>2. tane</u>	<u>Ağırlık</u>		
2	3	3	→	Ağırlık Eklendi.
1	2	4	→	Ağırlık Eklendi.
1	5	5	→	Ağırlık Eklendi.
0	1	6	→	Ağırlık Eklendi.
4	5	7	→	Ağırlık Eklendi.
1	3	8		
5	6	9		
3	4	10		
5	0	12		
4	7	15		

6.01.2020

1. tane

2. tane

Ağırlık

2

3

3

→ Ağırlık Eklendi.

1

2

4

→ Ağırlık Eklendi.

1

5

5

→ Ağırlık Eklendi.

0

1

6

→ Ağırlık Eklendi.

4

5

7

→ Ağırlık Eklendi.

1

3

8

→ Eklendi.

5

6

9

3

4

10

5

0

12

4

7

15

7. adım

0	1	2	3	4	5	6	7
1	1	1	1	1	1	1	0

<u>1. tepe</u>	<u>2. tepe</u>	<u>Ağırlık</u>
2	3	3 → Ağırlık Eklendi.
1	2	4 → Ağırlık Eklendi.
1	5	5 → Ağırlık Eklendi.
0	1	6 → Ağırlık Eklendi.
4	5	7 → Ağırlık Eklendi.
1	3	8 → Eklendi.
5	6	9 → Ağırlık Eklendi.
3	4	10
5	0	12
4	7	15

8. ve 9. adımlarda Ağrılar Eklennme

1. tane

2. tane

Ağırlık

2	3
1	2
1	5
0	1
4	5
1	3
5	6
3	4
5	0
4	7

3	→ Ağrı Eklenir.
4	→ Ağrı Eklenir.
5	→ Ağrı Eklenir.
6	→ Ağrı Eklenir.
7	→ Ağrı Eklenir.
8	→ Eklennedi.
9	→ Ağrı Eklenir.
10	→ Eklennedi.
12	→ Eklennedi.
15	

to.ozim

0	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1

<u>1. tane</u>	<u>2. tane</u>	<u>Ağırlık</u>
2	3	3 → Ağırlık Eklendi.
1	2	4 → Ağırlık Eklendi.
1	5	5 → Ağırlık Eklendi.
0	1	6 → Ağırlık Eklendi.
4	5	7 → Ağırlık Eklendi.
1	3	8 → Eklendi.
5	6	9 → Ağırlık Eklendi.
3	4	10 → Eklendi.
5	0	12 → Eklendi.
4	7	15 → Ağırlık Eklendi.

$$w(T) = 3 + 4 + 5 + 6 + 7 + 9 + 15 = 49.$$

PRİM ALGORİTMASI

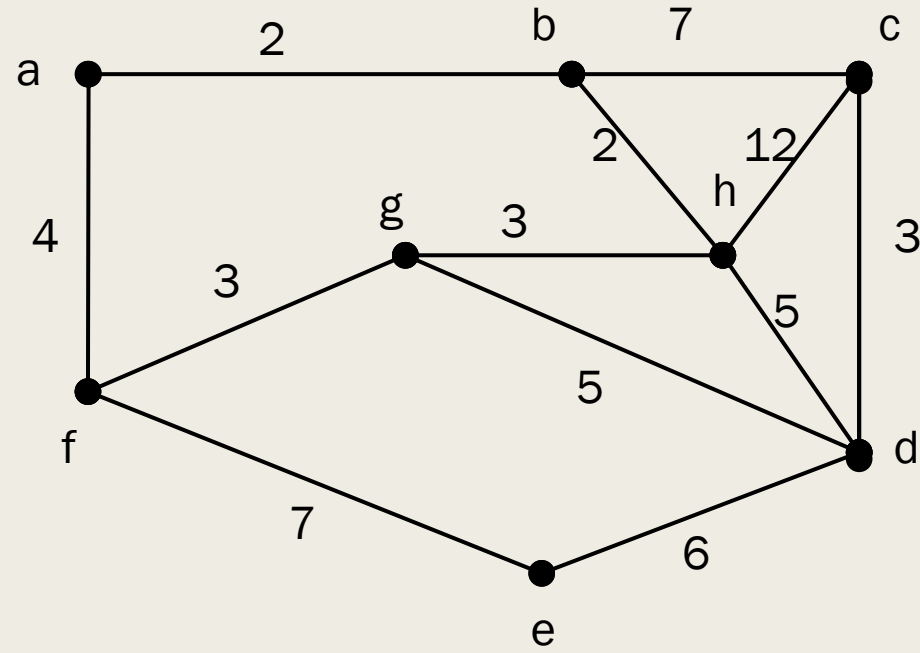
Ağırlıklandırılmış ve yönsüz graflarda dallanmış alt ağacı bulur.

Adım1: Graftaki herhangi v tepesi seç, ve bu tepe ile birlikte en düşük maliyetli ayrıt ile ağacı oluşturmaya başla.

Adım2: Henüz ağaçta olmayan, ziyaret edilmiş tepelere bitişik olan ve ağaca eklendiğinde çevre içermeyen en küçük ağırlıklı bir ayrıtı seç ve ağaca ekle.

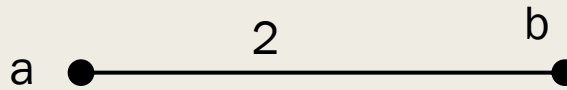
Adım3: Dallanmış ağaca sahip olup olmadığını kontrol et, Eğer sahip ise dur, aksi halde Adım 2 ye git.

Örnek:



1. adım

a tepesinden başlayalım.

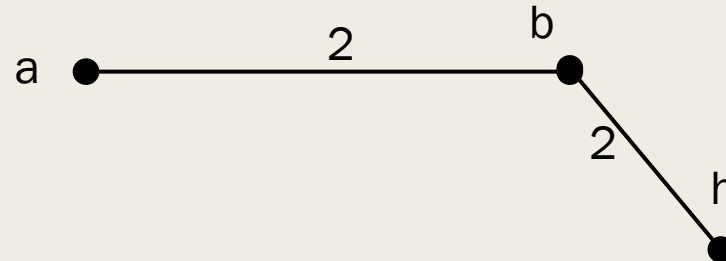


2. adım

a – f arası :4

b – c arası :7

b – h arası : 2 (2. adımda eklenir.)



3. adım

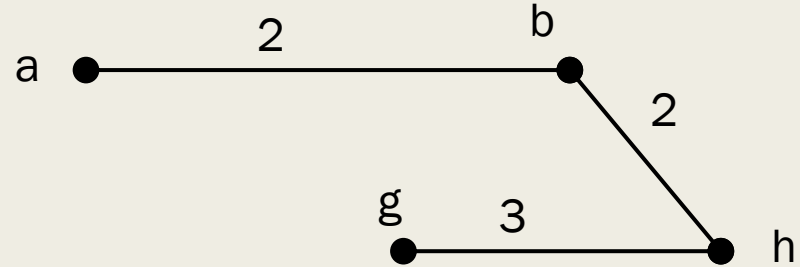
a - f arası :4

b - c arası :7

h - g arası : 3 (3. adımda eklenir.)

h - c arası : 12

h - d arası : 5



4. adım

a - f arası :4

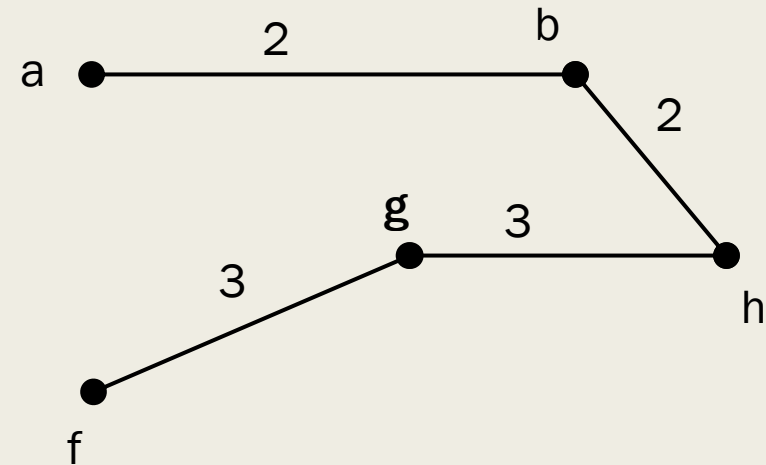
b - c arası :7

h - c arası : 12

h - d arası : 5

g - d arası : 5

g - f arası : 3 (4. adımda eklenir.)



5. adım

a – f arası :4 eklenemez, çevre olur.

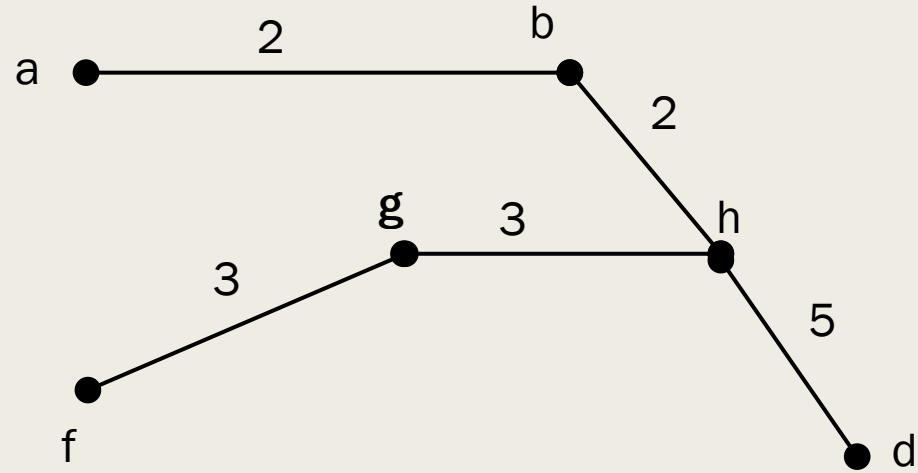
b – c arası :7

h – c arası : 12

h – d arası : 5 (5. adımda eklenir.)

g – d arası : 5

f – e arası: 7



6. adım

b – c arası :7

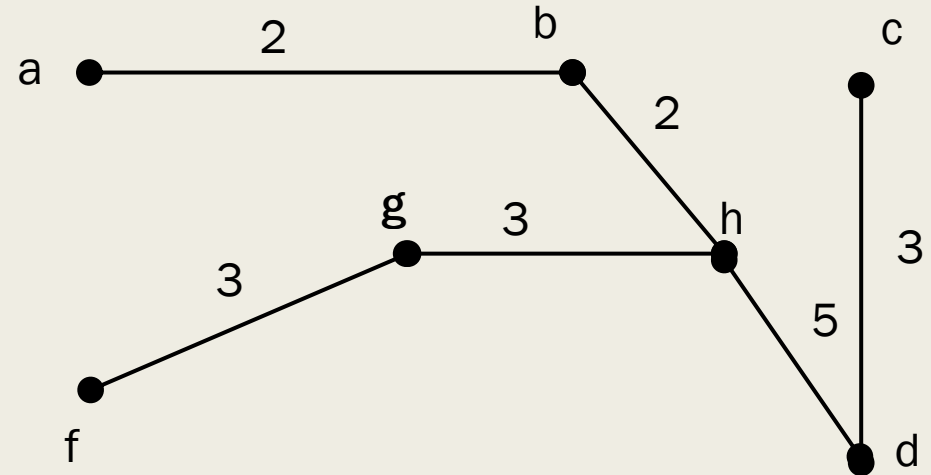
h – c arası : 12

g – d arası : 5

f – e arası: 7

d – c arası: 3 (6. adımda eklenir.)

d – e arası: 6



7. adım

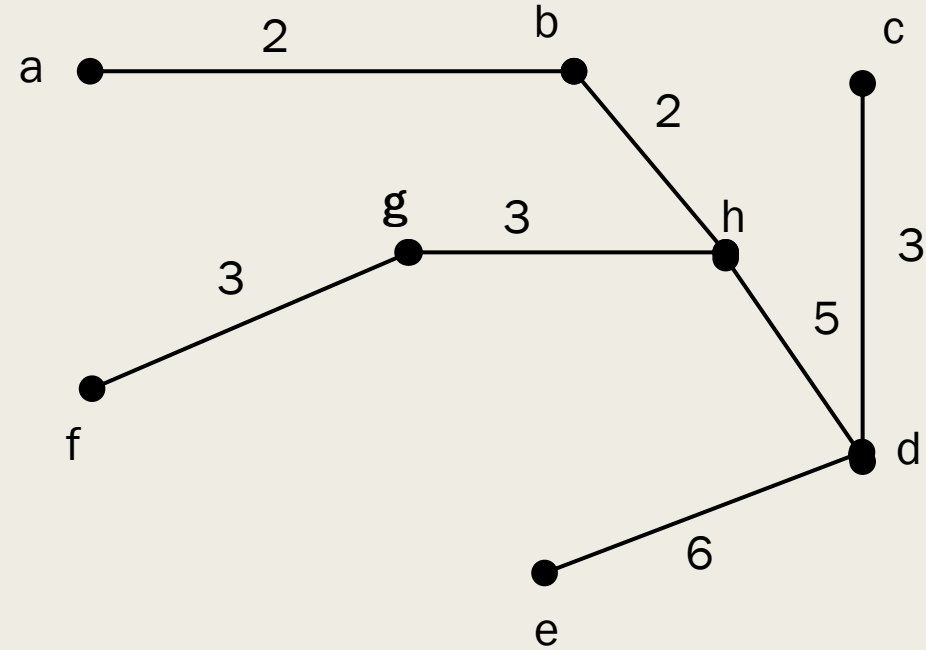
b – c arası :7

h – c arası : 12

g – d arası : 5 **eklenmez, çevre oluşturur.**

f – e arası: 7

d – e arası: 6 (7. adımda eklenir.)



8. adım

Dallanmış alt ağaç oluştu, dur.

Oluşturulan ağacın ağırlığı=24

Sonuç: Kruskal algoritması ile aynı minimum ağırlıklı dallanmış ağacı bulur.

Kaynaklar

- *Discrete Mathematics and Its Applications*, Kennet H. Rosen
(Ayrık Matematik ve Uygulamaları, Kennet H. Rosen (Türkçe çeviri),
Palme yayıncılık)
- *Discrete Mathematics: Elementary and Beyond*, L. Lovász, J. Pelikán,
K. Vesztergombi, 2003.
- *Introduction to Algorithms*, T.H. Cormen, C.E. Leiserson, R.L. Rivest,
C. Stein, 2009.
- *Introduction To Design And Analysis Of Algorithms*, A. Levitin, 2008.