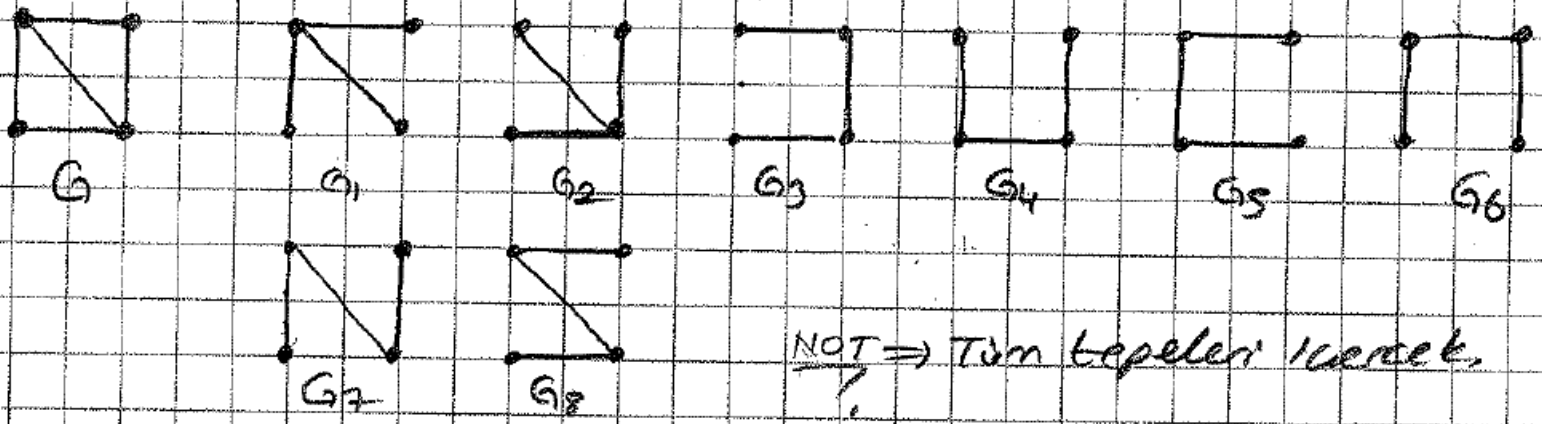
BİLGİSAYAR BİLİMLERİNDE GÜNCEL KONULAR II

Hafta 9

. Dallanmış Ağaç Algoritmaları

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 denir.



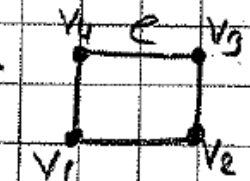
NOT ⇒ Tüm tepeleri içerecek.

Teorem: (Cayley)

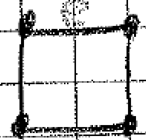
$Z(G)$, dallanmış ağaçların sayısını göstermek üzere,

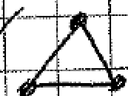
$$Z(G) = Z(G-e) + Z(G\cdot e)$$ büzülme

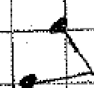
⇒ **büzülme işlemi:**



$G.e =$

 = 4

Z(G) = Z() + Z()

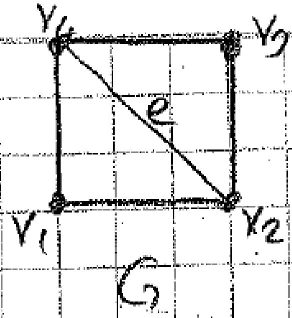
+ Z() + Z() (bukle)

+ Z() + 8 = 4

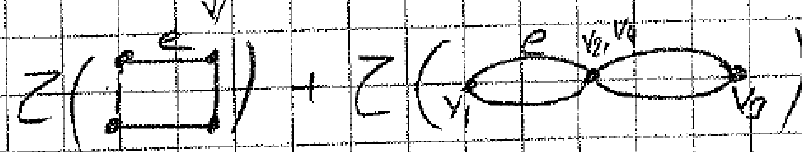
(1) (2) (3) (4)

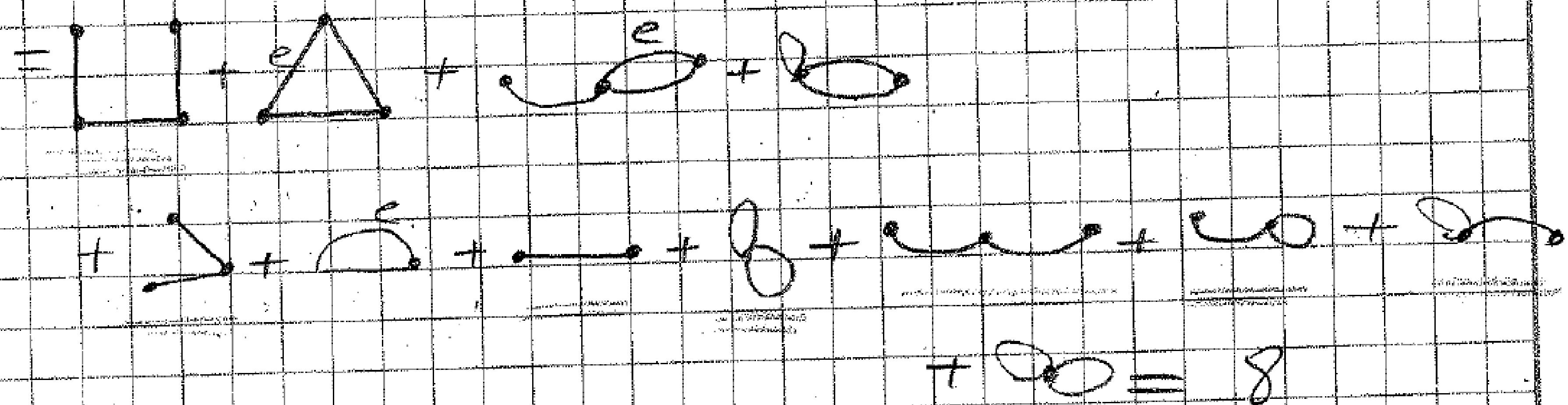
Yön 1

 ⇒ 4 tane



$$Z(G) = Z(G-e) + Z(G.e)$$





DALLANMIŞ AĞAÇ ALGORİTMALARI

Verilen bir G grafının dallanmış ağaçlarının herhangi birini bulmak için çeşitli algoritmalar kullanılır. Aşağıdaki algoritma ağaç oluşturma algoritması olarak bilinir.

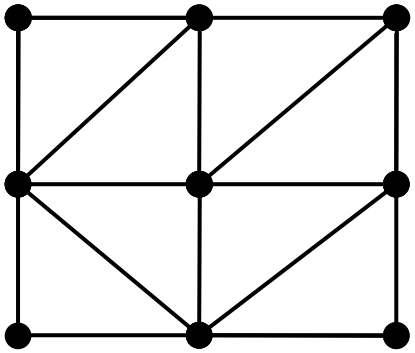
Adım1: Grafın herhangi bir e ayrıtını seç ve bu ayrıtın uç noktalarını v_1,v_2 ile isimlendir.(i=1, j=2 olsun.)

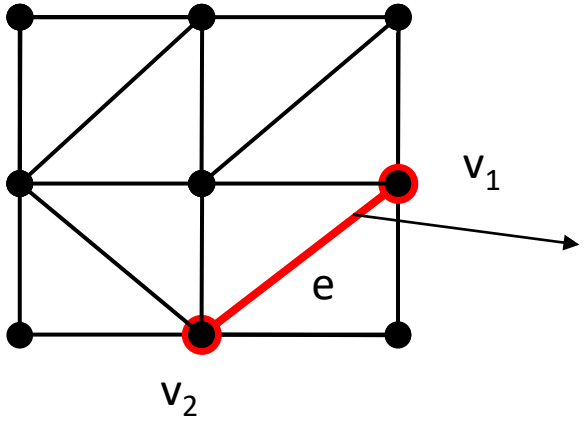
Adım2: v_i tepesinin , verilen graftaki tüm komşularının oluşturulan ağaçta olup olmadığını kontrol et.

Adım3: vi tepesinin grafta olup oluşturulan ağaçta olmayan bir komşu tepesi varsa bunu vj ile isimlendir ve (v_i v_j) ayrıtını ağaca ekle. Eğer j, grafın tepe sayısına eşitse dur. Şu an dallanmış bir ağaca sahipsin. Aksi halde j yi 1 arttır ve adım2 ye dön.

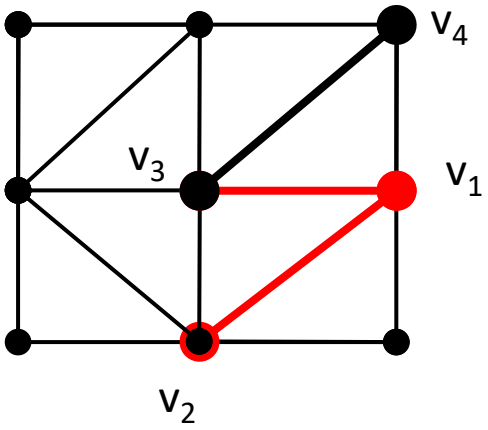
Bunu programlamak için matris (veri yapısını) kullanmak gereklidir.

Örnek:

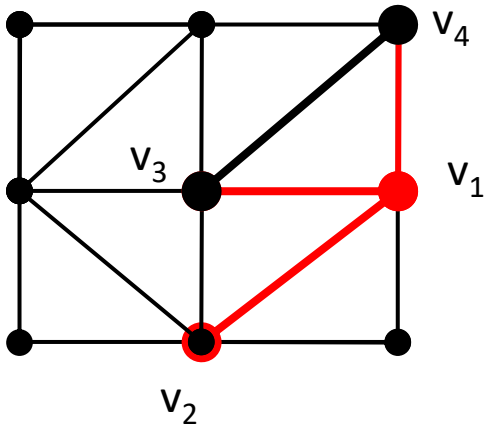




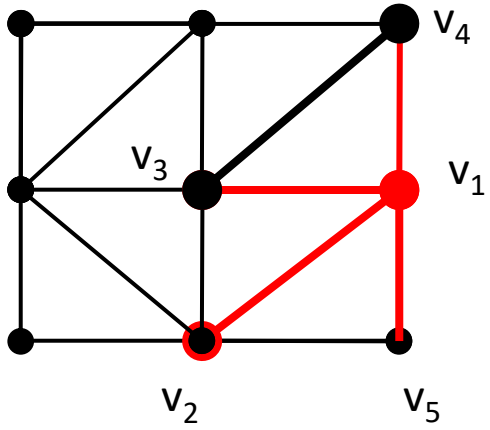
ayrıtını
seçelim i=1
ve j=2



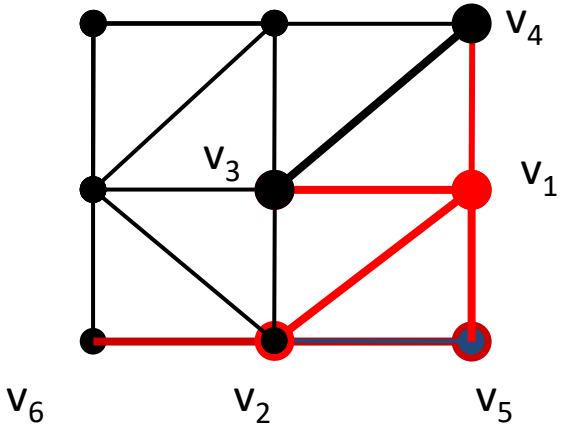
j=3



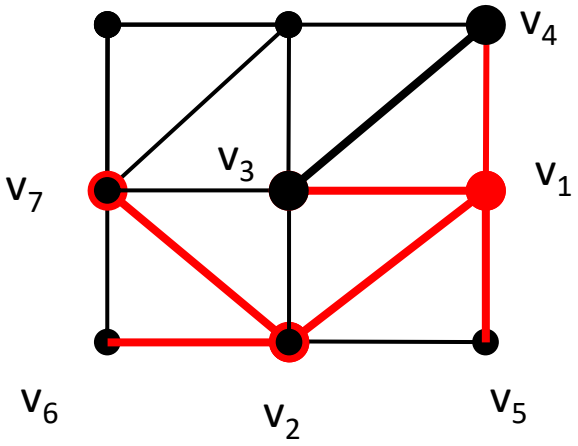
j=4



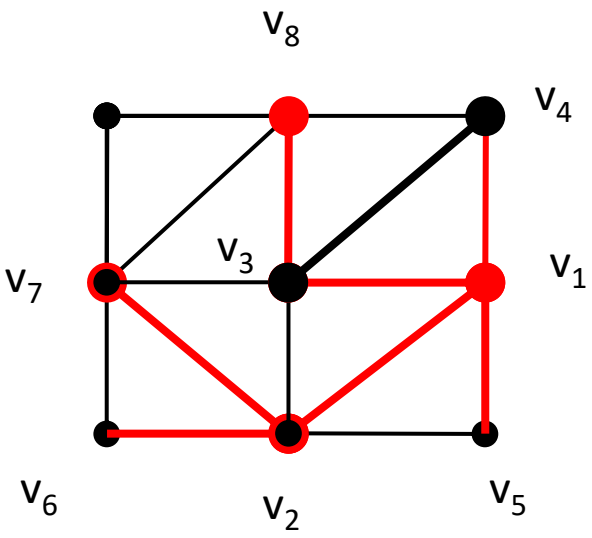
j=5



j=6 , i=2



j=7,



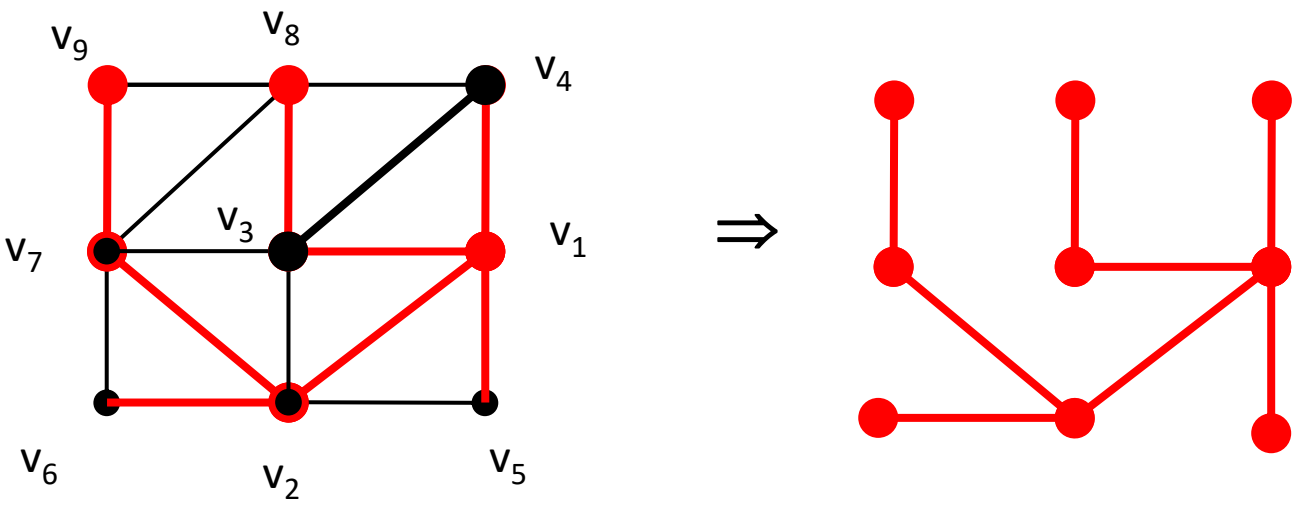
j=8, i=3

J=9, i=4

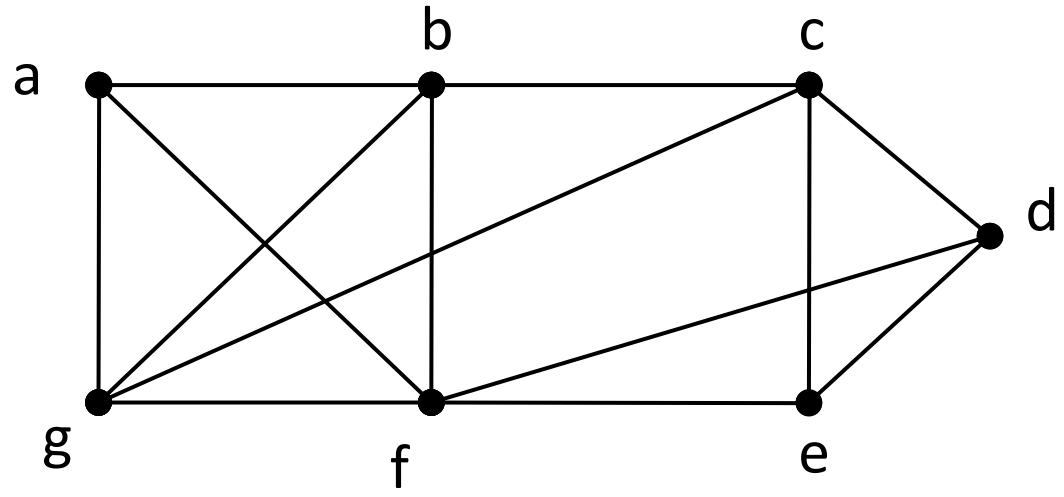
j=9, i=5

j=9, i=6

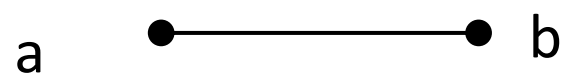
j=9, i=7



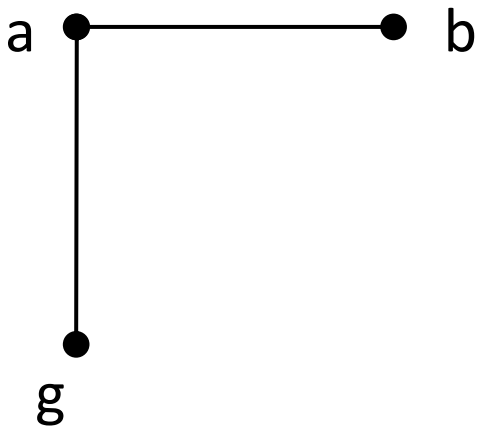
Örnek : Aşağıdaki grafın spanning (dallanmış) ağacını bulunuz.



1)

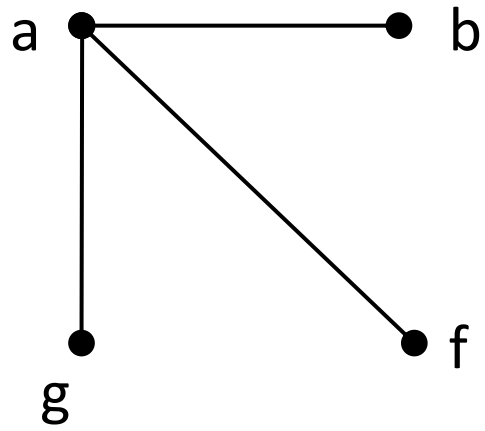


2)

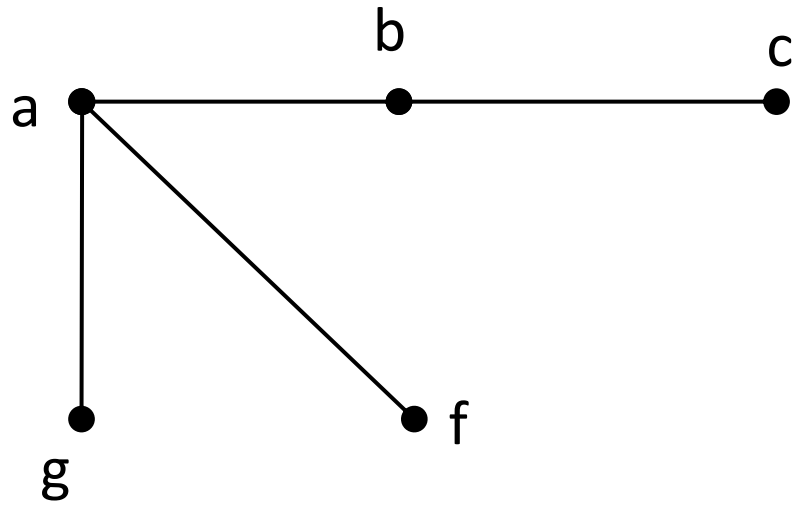


Çevre içermeyecek

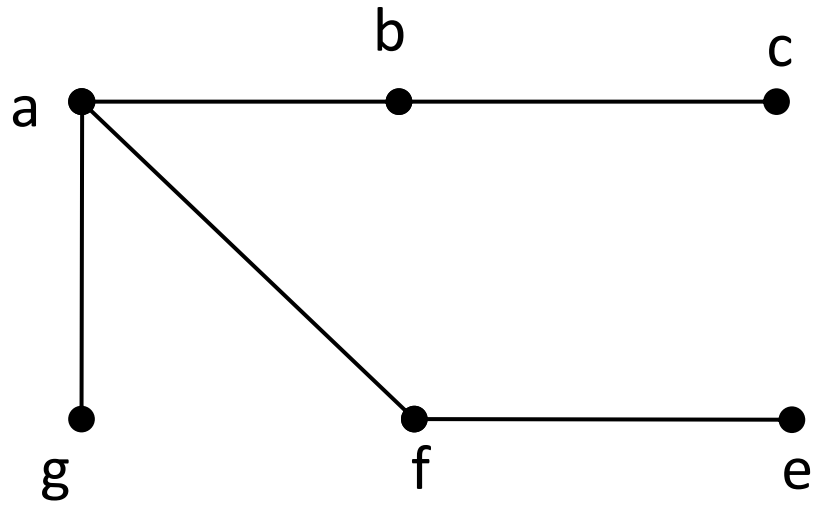
3) A tepesine bitişik ve ağaçta olmayan başka tepe seçilecek.



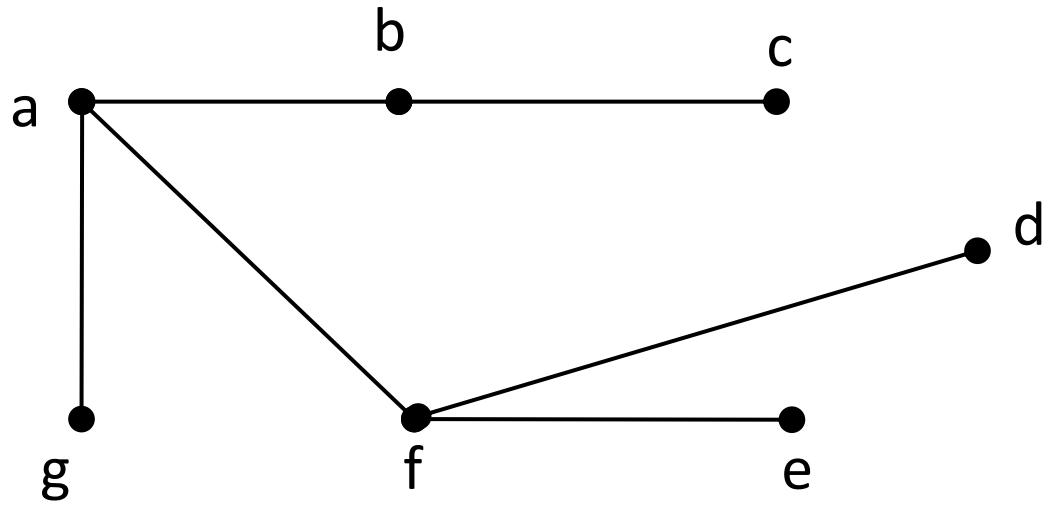
4) bc ağaçta olmadığı için seçebiliriz.



6) fe ağaçta olmadığı için seçebiliriz.



7) fd ağaçta olmadığı için seçebiliriz.

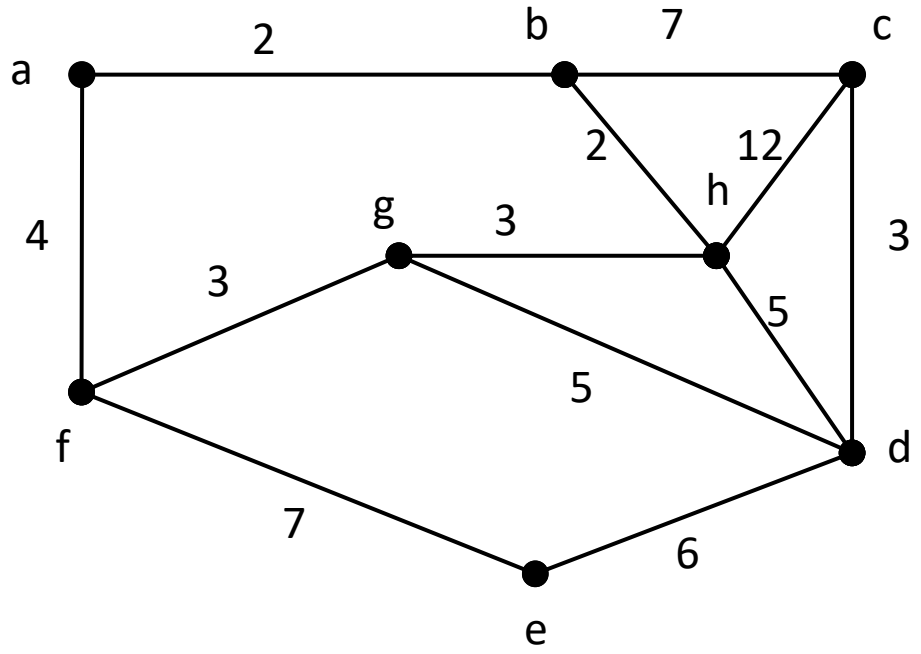


Başka ayrıt ekleyemeyiz. Hangi ayrıt eklenirse eklensin ağaç oluşur. Bu da istenmeyen durumdur.

Kruskal'ın Algoritması:

Bu algoritmayı aşağıdaki problemi, ele alarak inceleyeceğiz. Aslında bu algoritma, önceki algoritma ile aynı mantığa sahip olup, sadece o algoritmanın ağırlıklı graflara uygulanmasıdır.

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.


a
b
c
d
e
f
g
h
2
7
2
12
4
3
3
3
5
5
7
6

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

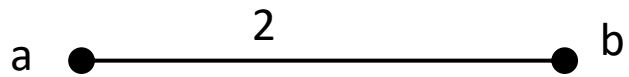
KRUSKAL ALGORİTMASI

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

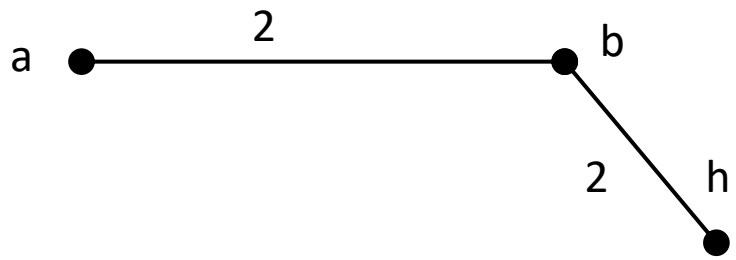
Adım2: Henüz ağaçta olmayan ve ağaca eklendiğinde çevre içermeyen en küçük ağırlıklı bir ayrıtı seç ve ağaca ekle

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

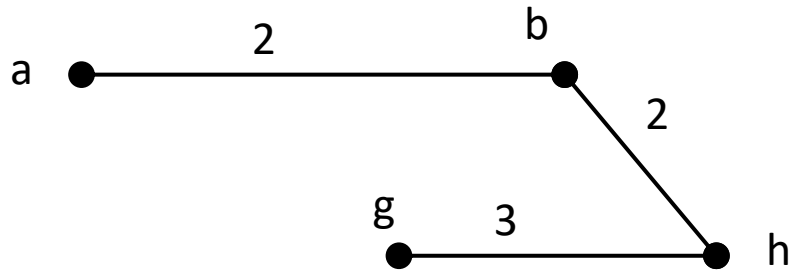
1)



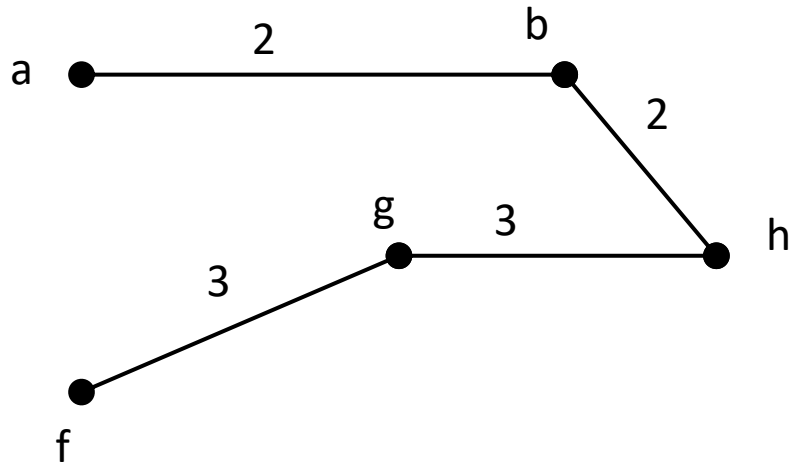
2)



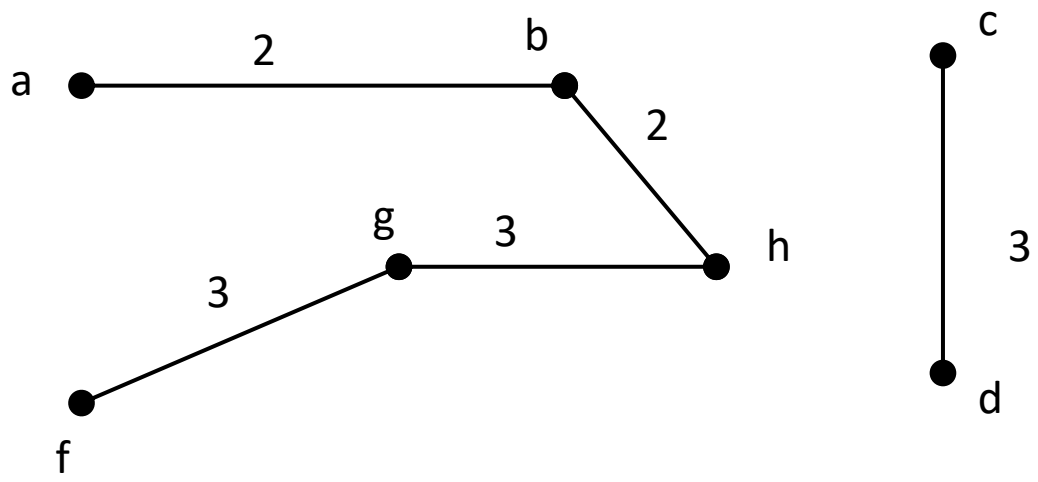
3)



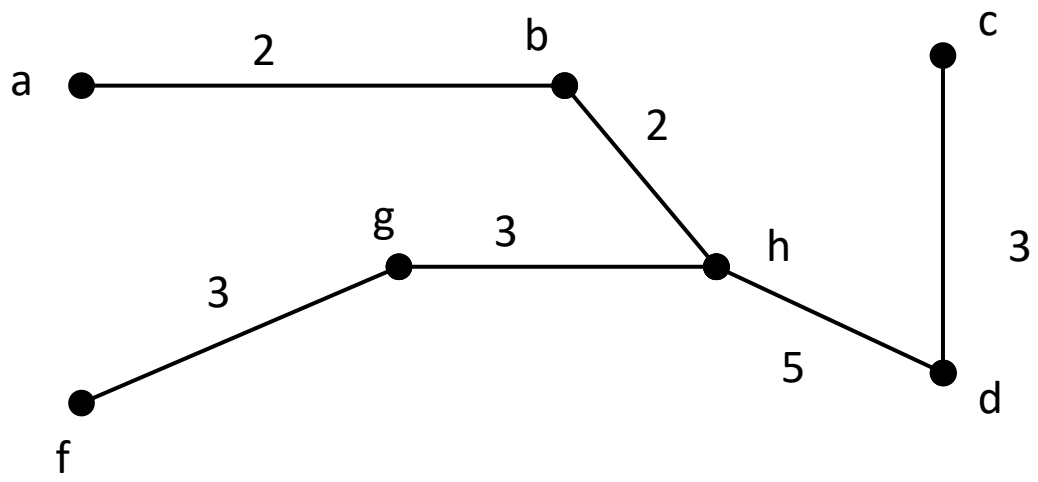
4)



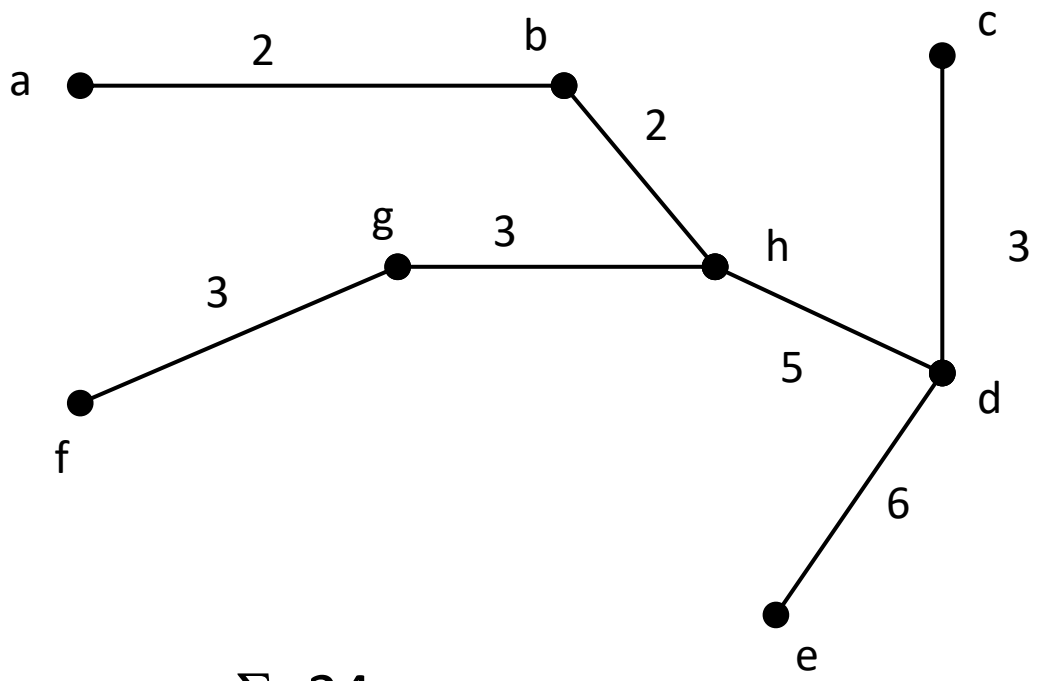
5)



6)



7)



Σ=24

Sonuç: Bir önceki algoritma ile arasındaki fark biri rasgele dallanmış ağacı, diğeri minimum ağırlıklı dallanmış ağacı bulur.

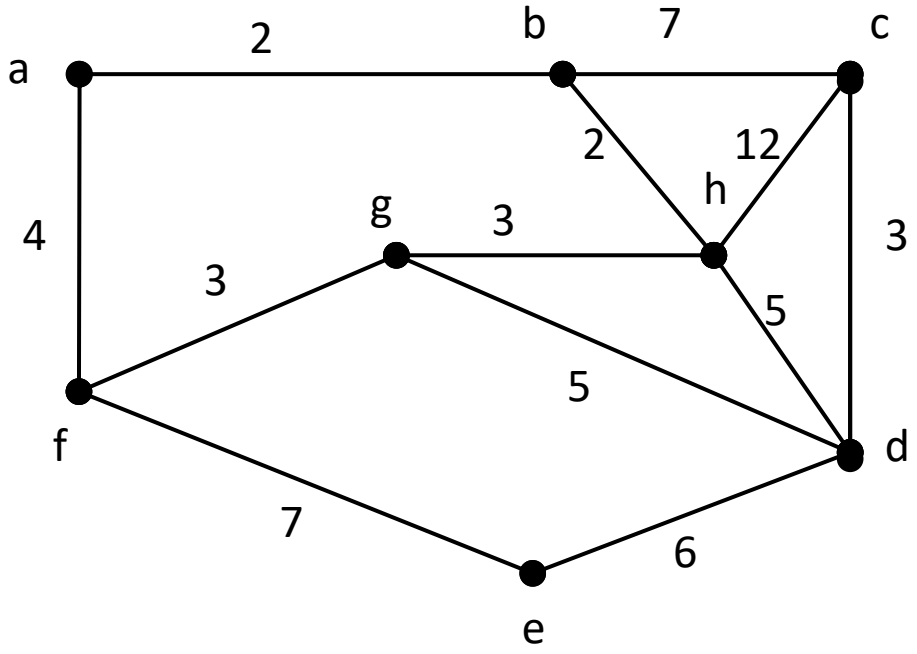
PRİM ALGORİTMASI
Ağırlıklandırılmış ve yönsüz graflarda dallanmış alt ağacı bulur.

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

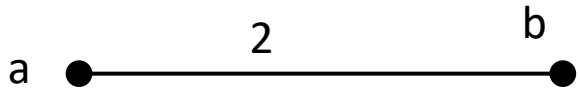
Adım2: Henüz ağaçta olmayan, ziyaret edilmiş tepelere bitişik olan ve ağaca eklendiğinde çevre içermeyen en küçük ağırlıklı bir ayrıtı seç ve ağaca ekle.

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

Örnek:



a tepesinden başlayalım.

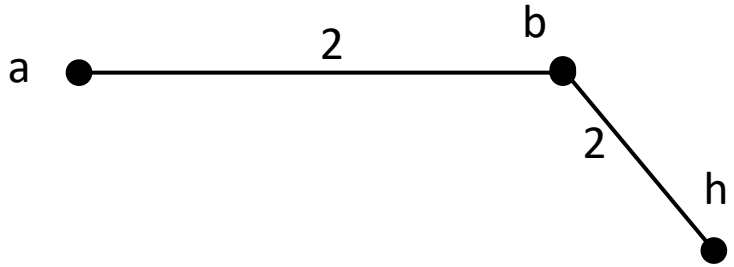


2. adım

a – f arası :4

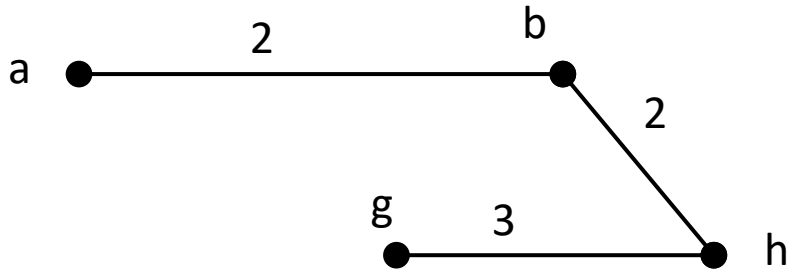
b – c arası :7

b – h arası : 2 2. adımda eklenir.



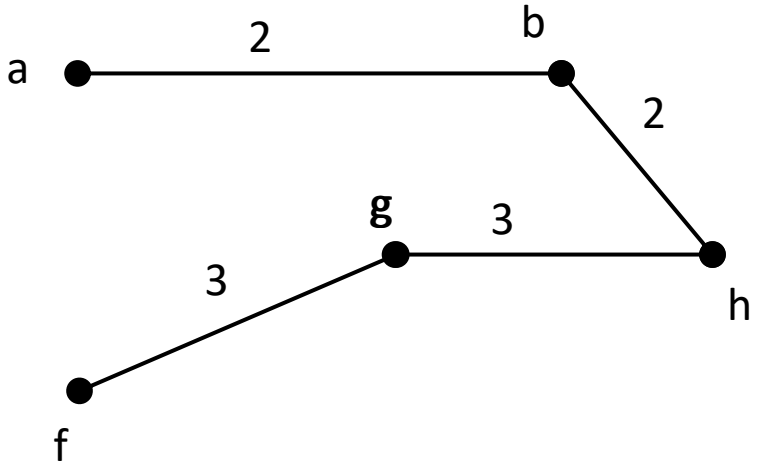
3. adım

a – f arası :4
b – c arası :7
h – g arası : 3 3. adımda eklenir.
h – c arası : 12
h – d arası : 5



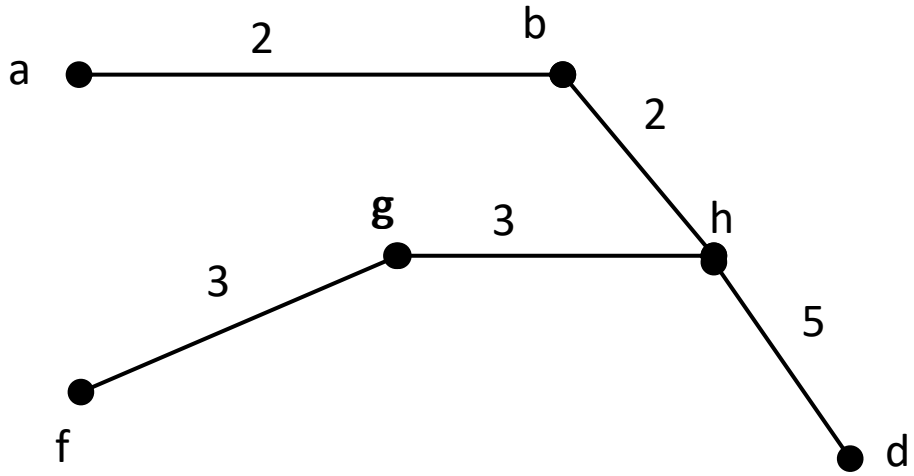
4. adım

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.



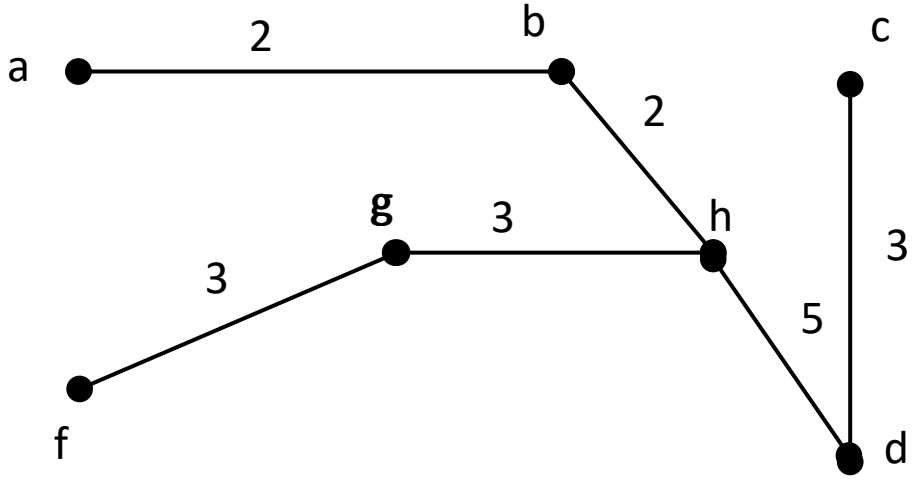
5. adım

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

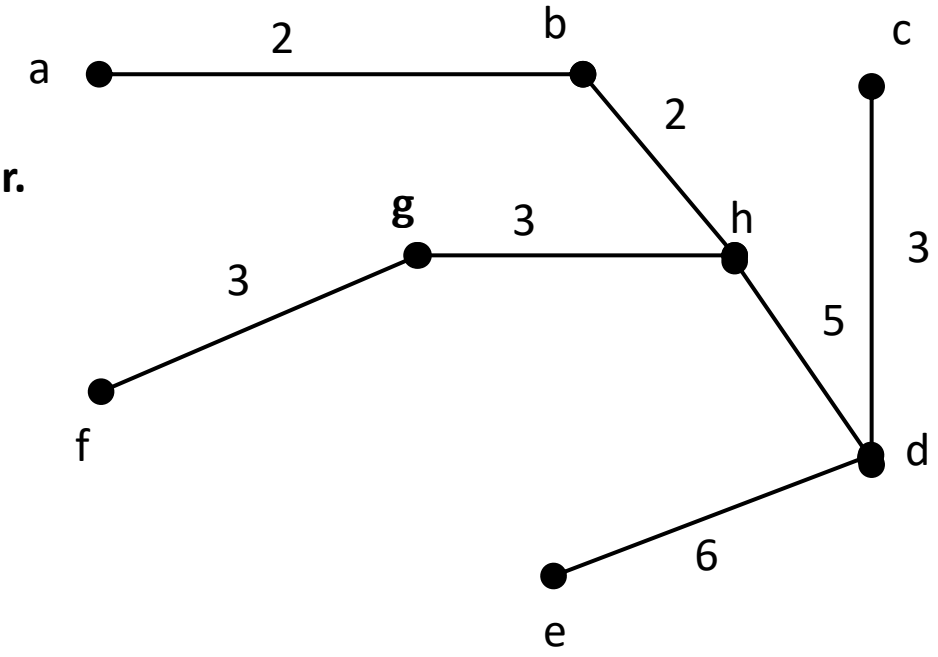
b – c arası :7

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.

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

KAYNAKLAR

[1] Chartrand, G.-Lesniak, L., (1986) : *Graphs and Digraphs*, Wadsworth & Brooks, California

[2] West D.B. (2001) : *Introduction to Graph Theory, Prentice Hall*, USA.

[3] Graf Teoriye Giriş, Şerife Büyükköse ve Gülistan Kaya Gök, Nobel Yayıncılık

[4] Discrete Mathematical Structures for Computer Science, Ronald E. Prather, Houghton Mifflin Company, (1976).

[5] Christofides, N., 1986. Graph Theory an Algorithmic Approach, Academic Press, London