Ağaçlar

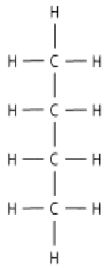
Ders 10

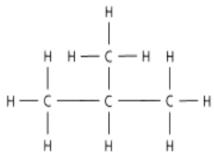
Ağaçlar

- İçerisinde çevrim (cycle) bulundurmayan bağlantılı grafların özel bir türüne ağaç adı verilir.
- Ağaçlar, fizikçi ***Gustava Kirşof*** tarafından 1847'de kablo ağlarındaki elektrik akışını formülize etmek için kullanılmıştır.
- Kirşof yasaları olarak adlandırılan denklemlerin tamamı bağımsız olmadığından Kirşof ağaçları kullanarak bu denklemlerin hangilerinin bağımsız olduğunu belirlemiştir.
- Ağaç terimi, bu çalışmalardan on yıl sonra İngiliz matematikçi ***Arthur Cayley*** tarafından verilmiştir.
- Cayley matematik içerisindeki bir problemi incelemek için ağaçlar üzerine odaklanarak çalışmalar gerçekleştirmiştir.
- Ağaçlar bir çok nedenden dolayı graf teorisi içerisinde önemli bir yere sahiptir. Ayrıca, ağaçlar graf teorisinin birçok uygulamasında ön plana çıkmaktadır.

Ağaçlar

- Cayley kimyadaki izomerleri ağaçları kullanarak incelemiş ve teoriyi uygulamaya taşımına becerisini göstermiştir.
- Son zamanlarda, bilgisayar bilimcileri, hiyerarşik veritabanı kullanarak belirli tipteki verilerin depolanma ve düzenlenme uygulamasını, ağaçları kullanarak gerçekleştirmişlerdir.
- Bunlara ek olarak, sıralama, kodlama teorisi ve bazı optimizasyon problemlerinin incelenmesinde de ağaçlar kullanılmaktadır.



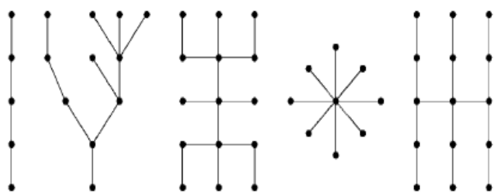


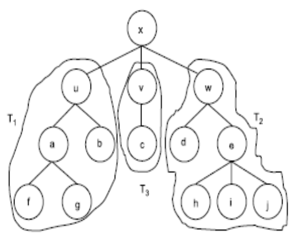
Arthur Cayley (1821–1895)

Ağaçlar

- Tanım: Ağaçlar içerisinde çevrim bulundurmayan bağlantılı graflardır.

- Buradan ağaçların döngü ve çoklu ayrıt bulunduramayacağı açıktır.
- Herhangi bir döngünün kendisinin bir çevrim olduğu unutulmamalıdır.





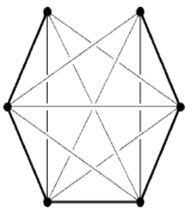
Ağaçlar

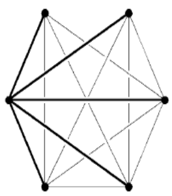
- Graf teorisinde ağaçların öneminin bir nedeni, her bağlantılı grafın özel bir ağaç içermesidir. Bu tip ağaçlar geren ağaç (spanning tree) olarak adlandırılır.
- Diğer özelliklerin yanı sıra; geren ağaçlar grafın düğümlerinin herhangi bir çiftini bağlayan yolların bir uygun kümesini sunar.
- Tanım: Γ, V düğüm kümesi olan bir bağlantılı graf olsun. Geren ağaç, düğüm kümesi V olan bir ağaçtır ve Γ'nın bir alt grafıdır.
- Dikkat: Geren ağaç, grafın tüm düğümlerinden oluşmaktadır.
- Teorem: Her bağlantılı graf bir geren ağaç içerir.

10-5

Ağaçlar

- Uyarı: Verilen bir Γ bağlantılı grafı genellikle birden fazla geren ağaca sahiptir.
- Örneğin K_6 tam grafı için iki farklı geren ağaç aşağıdaki şekilde gösterilmektedir.





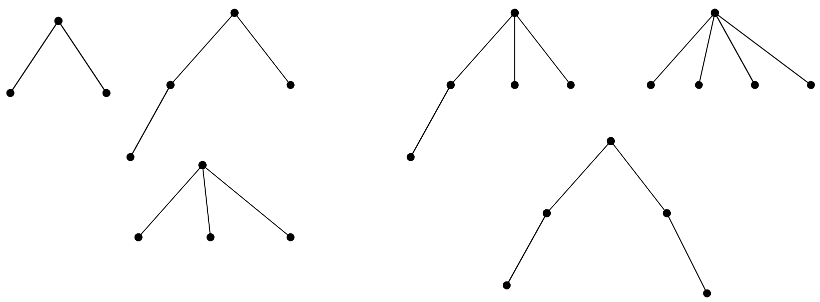
10-6

Ağaçlar

- Teorem: T, düğüm kümesi V ve ayrıt kümesi E olan bir ağaç olsun. Bu durumda
  i. Her u ve w gibi iki ayrık düğüm çifti için T içerisinde bir birlerini bağlayan tek bir yol vardır.
  ii. T'deki herhangi bir ayrıtın silinmesi her birisi ağaç olan iki bileşenli bir graf oluşturur.
  iii. $|E| = |V| - 1$

- Sonuç: Yukarıdaki özelliklerden herhangi birini sağlayan bağlantılı graf bir ağaçtır.

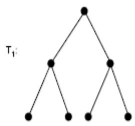
- Örnek: Aşağıdaki düğüm sayılarına sahip kaç tane izomorf olmayan ağaç vardır.
  i. Üç düğüm 1 tane
  ii. Dört düğüm 2 tane
  iii. Beş düğüm 3 tane
  iv. Altı düğüm 6 tane



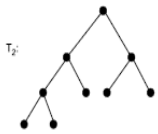
Tanımlar

i. Bir ağaçta tam bir tane düğümün derecesi 2 ve diğerlerinin dereceleri 1 veya 3 ise tam ikili ağaç (full binary tree) olarak adlandırılır.
ii. Derecesi 2 olan düğüm kök (root) olarak,
iii. Derecesi 3 olan düğümleri karar düğümleri (decesion vertices),
iv. Derecesi 1 olan düğümlerde yaprak düğümler (leaf vertices) olarak adlandırılır.

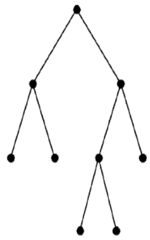
- İkili ağaçlar, genellikle bilgisayar bilimlerinde sıralama ve arama algoritmalarının tasarlanmasında kullanılmaktadır.



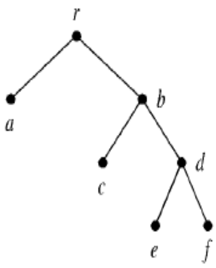
(Full Binary Tree)



(Complete Binary Tree)

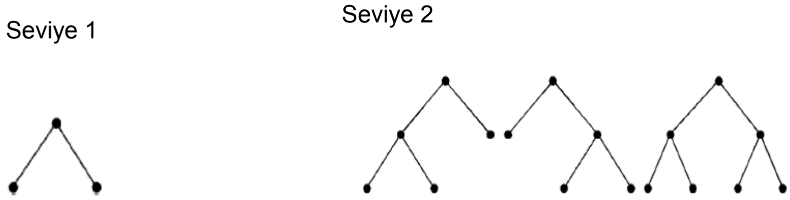


- Tanım: T, r köküne sahip tam ikili ağaç olsun. v düğümünün seviyesi (level), T içerisinde r ile v yi birleştiren tek bir yolun uzunluğudur.



- Bu şekil 3 seviyeli tam ikili ağaçtır.
- 0 seviyeli düğüm kök olan r dir.
- 1 seviyeli düğümleri a ve b dir.
- 2 seviyeli düğümleri c ve d dir.
- 3 seviyeli düğümleri e ve f dir.

- Örnek: Seviye 1 ve seviye 2 tam ikili ağaçları çiziniz.

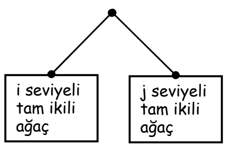


- Örnek: a_n , n seviyeli tam ikili ağaçların sayısını göstersin. n ≥ 2 için

$$a_n=2\, a_{n-1}(a_0+a_1+\cdots+a_{n-2})+(a_{n-1})^2$$

olduğunu gösteriniz.

n seviyeli bir ağaç; $0 \le i,j \le n-1$ ve i ve j den en az birisi n-1' e eşit olmak üzere iki parçadan oluştuğunu düşünelim:



Eğer j=0 ise i = n-1: $a_0\, a_{n-1}$ tane böyle ağaç vardır.
Eğer j=1 ise i = n-1: $a_1\, a_{n-1}$ tane böyle ağaç vardır.
...
Eğer j = n-2 ise i = n-1: $a_{n-2}\, a_{n-1}$ tane böyle ağaç vardır.
Eğer j = n-1 ise i sayısı 0 ile n-1 arasındaki herhangi bir tamsayı değeri alabilir :

$$a_{n-1}(a_0+a_1+\cdots+a_{n-2}+a_{n-1})$$

tane böyle ağaç vardır.

- Böylece toplam n seviyeli tam ikili ağaçların sayısı

$$\begin{aligned} a_n &= a_0\, a_{n-1} + a_1\, a_{n-1} + \cdots + a_{n-2}\, a_{n-1} + a_{n-1}(a_0+a_1+\cdots+a_{n-2}+a_{n-1}) \\ &= a_{n-1}(a_0+a_1+\cdots+a_{n-2}) + a_{n-1}(a_0+a_1+\cdots+a_{n-2}) + (a_{n-1})^2 \\ &= 2\, a_{n-1}(a_0+a_1+\cdots+a_{n-2}) + (a_{n-1})^2 \end{aligned}$$

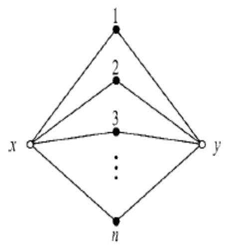
Örnek:

- Bir önceki örnekteki formülü kullanarak a_3 ve a_4 değerlerini hesaplayınız.
  - $a_0=1$, $a_1=1$ ve $a_2=3$ olduklarını bildiğimize göre
  - $a_3 = 2a_2(a_0+a_1)+a_2^2 = 2 \times 3 \times (1+1)+3^2=21$
  - $a_4 = 2a_3(a_0+a_1+a_2)+a_3^2 = 2 \times 21 \times (1+1+3)+21^2=651$

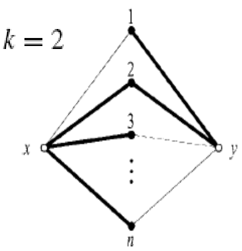
- Teorem:
  i. T bir ağaç ise ayrıtlarının sayısı $|E|=|V|-1$
  ii. T herhangi bir bağlantılı graf ise $|E| \geq |V|-1$
  iii. T ağaç olmayan bir bağlantılı graf ise $|E| > |V|-1$

- Örnek: $K_{n,2}$ içerisinde kaç tane geren ağaç vardır.

- V 'nin { {x,y} , {1,2,…,n} } şeklinde parçalanışa sahip olduğunu kabul edelim. Bu durumda $K_{n,2}$ yandaki şeklinde şekil yardımıyla ifade edilebilir.



$k = 2$



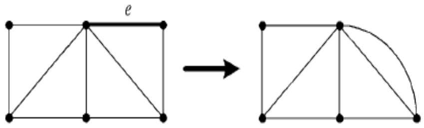
$K_{n,2}$ nin geren ağaçlarını şu şekilde tanımlayabiliriz:

- İlk olarak {1,2,…,n} kümesinden bir k düğümünü seçelim ve bu düğümü x ve y'nin her ikisi ile birleştirelim.
- Daha sonra $e \in \{1,2,\ldots,n\} - \{k\}$ şeklindeki her bir düğüm için rx veya ry den birisini geren ağacın bir ayrıtı olarak seçelim. K düğümünü seçmek için n tane durum vardır.
- Geriye kalan ayrıtların seçimi için ise n-1 eleman 2 farklı şekilde seçileceğinden 2^{n-1} durum söz konusudur.
- Sonuç olarak geren ağaçların sayısı

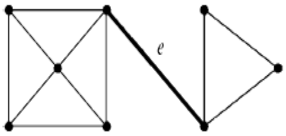
  $$n\, 2^{n-1}$$

  tane olur.

- Tanım: Γ bağlantılı bir graf olsun. t(Γ), Γ'nın içerisindeki geren ağaçların sayısını göstersin. e'de Γ'nın bir ayrıtı olsun. Bu durumda
  - **Γ- e** : Γ daki e ayrıtının silinmesiyle elde edilen grafı gösterir.
  - **$\Gamma \setminus$ e** : Γ dan e ayrıtının kısaltılması veya daraltılmasıyla (contracting) elde edilen grafı gösterir.



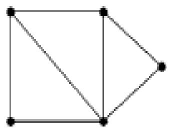
  - Eğer Γ-e bağlantılı graf değilse e ayrıtı köprü (bridge) olarak adlandırılır.



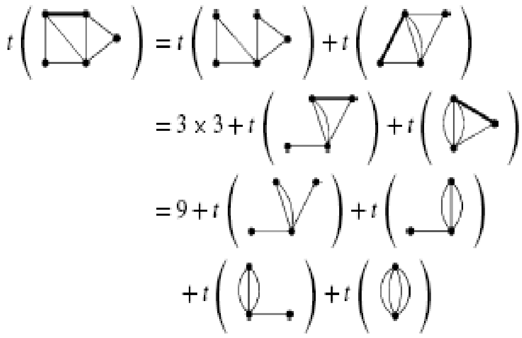
- Ödev: e, Γ'nın köprü olmayan bir ayrıtı olsun.

  t(Γ)=t(Γ-e)+t($\Gamma\setminus$e)

  olduğunu gösteriniz.



- Örnek: Yandaki şekilde verilen grafın geren ağaçlarının sayısını yukarıdaki formülü kullanarak hesaplayınız.
- Yukarıdaki formülü farklı seçimler yaparak bu örneğe uygulamak mümkündür.



$$=9+2+3+3+4=21.$$

- Tanım: u ve v, G grafının iki düğümü olsun. u ve v arasındaki uzaklık d(u,v) ile gösterilir ve u-v arasındaki en kısa uzaklıktır. Eğer u ile v arasında bir yol yok ise d(u,v)=∞ olur.

- Tanım: V, G grafının düğüm kümesi olsun. v düğümünün eccentricity e(v) ile gösterilir ve

$$e(v)=Max\{d(u,v): u\in V \text{ ve } u\neq v\}$$

şeklinde tanımlanır.

- Tanım: G bir graf olsun G nin yarıçapı rad(G) ile gösterilir ve

$$rad(G)=Min\{e(v): v\in V\}$$

şeklinde tanımlanır.

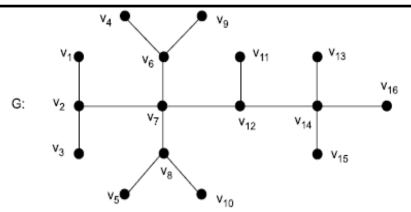
- Tanım: G bir graf olsun G nin çapı diam(G) ile gösterilir ve

$$rad(G)=Max\{e(v): v\in V\}$$

şeklinde tanımlanır.

- Tanım: V, G grafının düğüm kümesi olsun. Eğer e(v)=rad(V) ise v∈V düğümü merkez noktasıdır denir.G'nin tüm merkez noktalarının kümesi G nin merkezi olarak adlandırılır.

10-17



$V = \{v_1, v_2, v_3, \ldots, v_{16}\}$

- Yukarıdaki her bir kenar uzunluğu 1 olan G grafını göz önüne ala

$d(v_1,v_2) = 1$; $d(v_1,v_3) = 2$; $d(v_1,v_4) = 4$; $d(v_1,v_5) = 4$; $d(v_1,v_6) = 3$;
$d(v_1,v_7) = 2$; $d(v_1,v_8) = 3$; $d(v_1,v_9) = 4$; $d(v_1,v_{10}) = 4$; $d(v_1,v_{11}) = 4$;
$d(v_1,v_{12}) = 3$; $d(v_1,v_{13}) = 5$; $d(v_1,v_{14}) = 4$; $d(v_1,v_{15}) = 5$; $d(v_1,v_{16}) = 3$.

$e(v_1) = \text{Max}\{1, 2, 3, 4, 5\} = 5.$

$d(v_2,v_1) = 1$; $d(v_2,v_3) = 1$; $d(v_2,v_4) = 3$; $d(v_2,v_5) = 3$; $d(v_2,v_6) = 2$; $d(v_2,v_7) = 1$;
$d(v_2,v_8) = 2$; $d(v_2,v_9) = 3$; $d(v_2,v_{10}) = 3$; $d(v_2,v_{11}) = 3$; $d(v_2,v_{12}) = 2$; $d(v_2,v_{13}) = 4$;
$d(v_2,v_{14}) = 3$; $d(v_2,v_{15}) = 4$; $d(v_2,v_{16}) = 4$.

$e(v_2) = \text{Max}\{1, 2, 3, 4\} = 4.$

$e(v_3) = 5$; $e(v_4) = 5$; $e(v_5) = 5$; $e(v_6) = 4$; $e(v_7) = 3$;
$e(v_8) = 4$; $e(v_9) = 5$; $e(v_{10}) = 5$; $e(v_{11}) = 4$; $e(v_{12}) = 3$;
$e(v_{13}) = 5$; $e(v_{14}) = 4$; $e(v_{15}) = 5$; $e(v_{16}) = 5$.

radius = rad (G) = Min $\{e(v), v \in V\}$ = Min $\{5, 4, 3\}$ = 3 and diameter = diam (G) = Max $\{e(v), v \in V\}$ = Max $\{5, 4, 3\}$ = 5.

the central points are v_7 and v_{12} and center = $\{v_7, v_{12}\}$.

10-18

Prim Algoritması

- Prim algoritması, bilgisayar bilimcisi Robert Prim tarafından 1957 yılında üretilmiştir.
- Gelişi güzel seçilen bir düğümden başlanarak en küçük geren ağacın inşası gerçekleştirilir.
- Her bir adımda o andaki en küçük geren ağaç içerisinde olmayan en yakın noktayı ağaca bağlayan ayrıtın ağaca eklenmesiyle oluşturulur.
- Bu ağaç verilen grafın tüm düğümlerinden oluşuncaya kadar genişletilir.
- Bu strateji; her bir adımda kısmi geren ağaç tüm mümkün bitişik ayrıtların arasındaki en küçük olan ayrıt ile artırılarak gerçekleştirildiğinden aç gözlülük prensibi üzerine çalışıyor da denir.

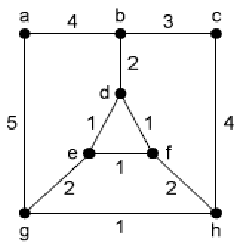
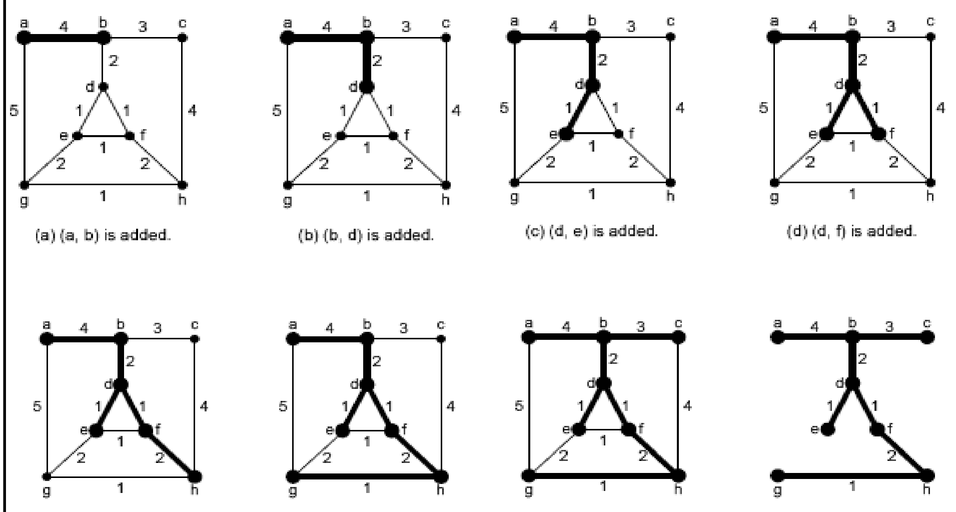
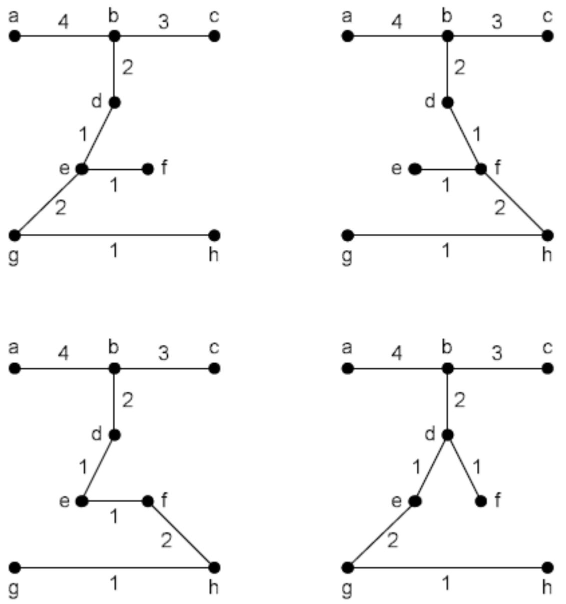


FIGURE 2.1: A weighted graph.



(a) (a, b) is added.
(b) (b, d) is added.
(c) (d, e) is added.
(d) (d, f) is added.
(e) (f, h) is added.
(f) (g, h) is added.
(g) (b, c) is added.
(h) A minimum spanning tree.

Bazı Farklı Geren Ağaçlar



10-21

Prim Algoritması

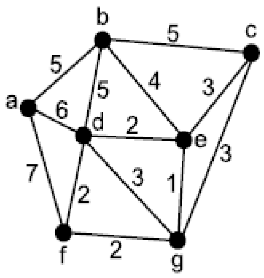
Prim algoritması

1. $i \leftarrow 1$, $v_1 \in V$, $P = \{v_1\}$, $N = V - \{v_1\}$, $T = \emptyset$
2. $1 \leq i \leq n-1$ için:
   $P = \{v_1, v_2, \ldots, v_i\}$, $T = \{e_1, e_2, \ldots, e_{i-1}\}$, $N = V - P$.
   öyle bir $v_{i+1} \in N$ düğümü seç ki, bir $x \in P$ düğümü için
   $e = (x, v_{i+1}) \notin T$, $wt(e)$ minimum
   $P \leftarrow P + \{v_{i+1}\}$, $N \leftarrow N - \{v_{i+1}\}$, $T \leftarrow T + \{e\}$
3. $i \leftarrow i + 1$
   - $i = n \Rightarrow e_1, e_2, \ldots, e_{n-1}$ ayrıtlarından oluşan G altçizgesi bir en hafif kapsayan ağaçtır
   - $i < n \Rightarrow$ 2. adıma git

10-22

Prim Algoritması Örneği

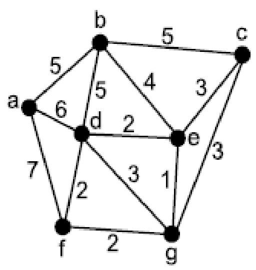
Örnek (başlangıç)



- $i \leftarrow 1$
- $P = \{a\}$
- $N = \{b, c, d, e, f, g\}$
- $T = \emptyset$

Prim Algoritması Örneği

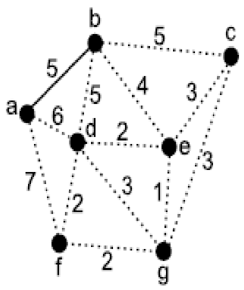
Örnek ($1 < 7$)



- $T = \{(a, b)\}$
- $P = \{a, b\}$
- $N = \{c, d, e, f, g\}$
- $i \leftarrow 2$

Prim Algoritması Örneği

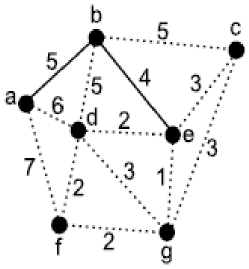
Örnek $(2 < 7)$



- $T = \{(a,b),(b,e)\}$
- $P = \{a,b,e\}$
- $N = \{c,d,f,g\}$
- $i \leftarrow 3$

Prim Algoritması Örneği

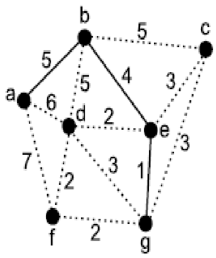
Örnek $(3 < 7)$



- $T = \{(a,b),(b,e),(e,g)\}$
- $P = \{a,b,e,g\}$
- $N = \{c,d,f\}$
- $i \leftarrow 4$

Prim Algoritması Örneği

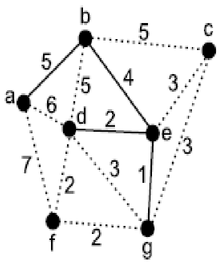
Örnek (4 < 7)



- $T = \{(a,b),(b,e),(e,g),(d,e)\}$
- $P = \{a,b,e,g,d\}$
- $N = \{c,f\}$
- $i \leftarrow 5$

Prim Algoritması Örneği

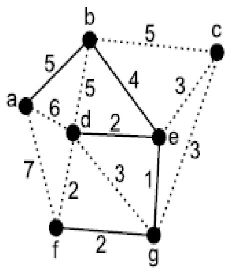
Örnek (5 < 7)



- $T = \{$
  $(a,b),(b,e),(e,g),$
  $(d,e),(f,g)$
  $\}$
- $P = \{a,b,e,g,d,f\}$
- $N = \{c\}$
- $i \leftarrow 6$

Prim Algoritması Örneği

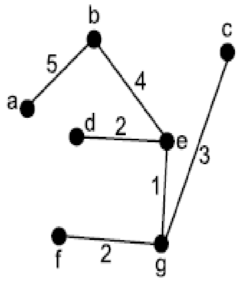
Örnek (6 < 7)



- $T = \{$
  $(a, b), (b, e), (e, g),$
  $(d, e), (f, g), (c, g)$
  $\}$
- $P = \{a, b, e, g, d, f, c\}$
- $N = \emptyset$
- $i \leftarrow 7$

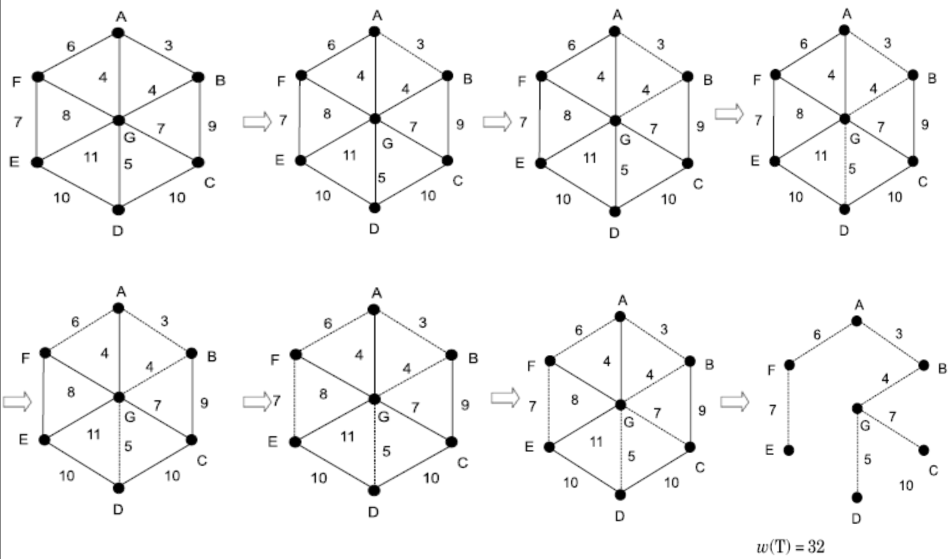
Prim Algoritması Örneği

Örnek ($7 \not< 7$)



- toplam ağırlık: 17

Prim Algoritması - Örnek



10-31

Kruskal Algoritması

- Kruskal algoritması, Joseph Kruskal tarafından 1956 yılında verilmiştir.
- Grafın her bir düğümünün başlangıçta ayrık ağaç olduğu bir orman oluşturulur.
- Daha sonra ayrıtlar ağırlıklarına göre sıralanılır.
- Sıralanmış her bir (u,v) kenarları için; eğer u ve v iki farklı ağacın düğümleri ise iki ağaç tek bir ağaca birleştirilerek (u,v) ormana dahil edilir.
- Bu işlem tüm ayrıtlar işleninceye kadar devam eder.

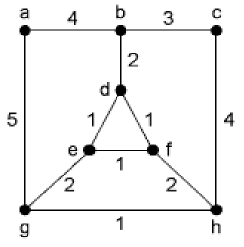
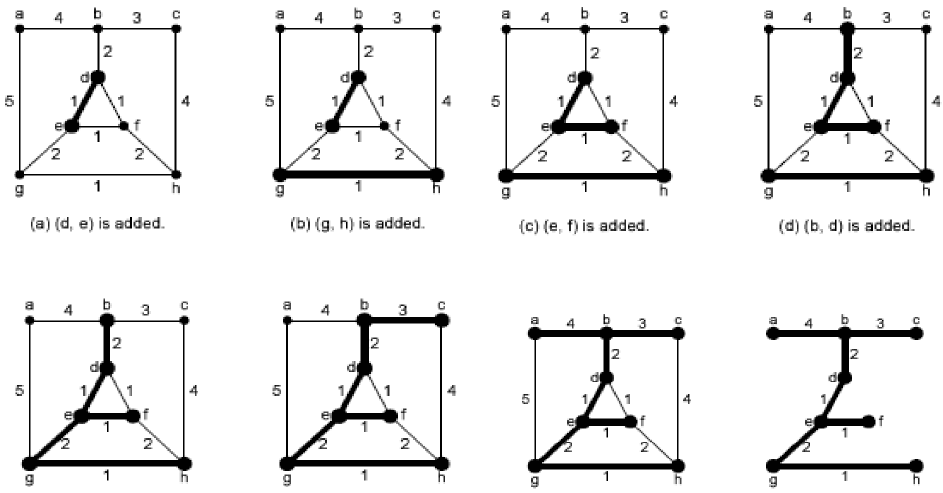


FIGURE 2.1: A weighted graph.

10-32



(a) (d, e) is added. (b) (g, h) is added. (c) (e, f) is added. (d) (b, d) is added.

(e) (e, g) is added. (f) (b, c) is added. (g) (a, b) is added. (h) A minimum spanning tree.

10-33

Kruskal Algoritması

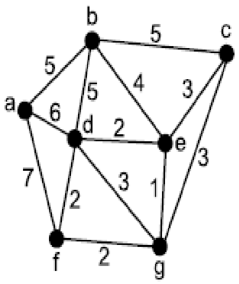
Kruskal algoritması

1. $i \leftarrow 1$, $e_1 \in E$, $wt(e_1)$ minimum
2. $1 \le i \le n-2$ için:
   şu ana kadar seçilen ayrıtlar $e_1, e_2, \ldots, e_i$ ise, kalan ayrıtlardan öyle bir e_{i+1} seç ki:
   - $wt(e_{i+1})$ minimum
   - $e_1, e_2, \ldots, e_i, e_{i+1}$ altçizgesi çevre içermiyor
3. $i \leftarrow i+1$
   - $i = n-1 \Rightarrow e_1, e_2, \ldots, e_{n-1}$ ayrıtlarından oluşan G altçizgesi bir en hafif kapsayan ağaçtır
   - $i < n-1 \Rightarrow$ 2. adıma git

10-34

Kruskal Algoritması Örneği

Örnek (başlangıç)

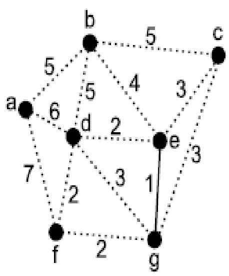


- $i \leftarrow 1$
- en düşük ağırlık: 1 (e, g)
- $T = \{(e, g)\}$

10-35

Kruskal Algoritması Örneği

Örnek $(1 < 6)$

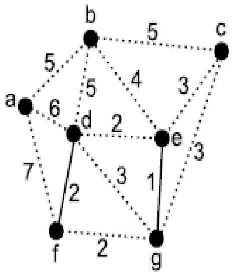


- en düşük ağırlık: 2 $(d, e), (d, f), (f, g)$
- $T = \{(e, g), (d, f)\}$
- $i \leftarrow 2$

10-36

Kruskal Algoritması Örneği

Örnek $(2 < 6)$

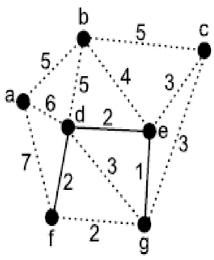


- en düşük ağırlık: 2
  $(d, e), (f, g)$
- $T = \{(e, g), (d, f), (d, e)\}$
- $i \leftarrow 3$

10-37

Kruskal Algoritması Örneği

Örnek $(3 < 6)$

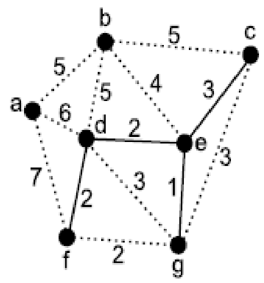


- en düşük ağırlık: 2
  (f, g) çevre oluşturuyor
- en düşük ağırlık: 3
  $(c, e), (c, g), (d, g)$
  (d, g) çevre oluşturuyor
- $T = \{(e, g), (d, f), (d, e), (c, e)\}$
- $i \leftarrow 4$

10-38

Kruskal Algoritması Örneği

Örnek $(4 < 6)$

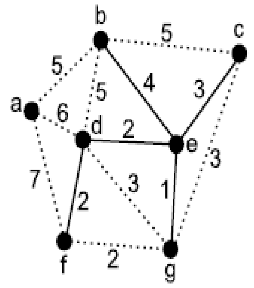


- $T = \{$
  $(e, g), (d, f), (d, e),$
  $(c, e), (b, e)$
  $\}$
- $i \leftarrow 5$

10-39

Kruskal Algoritması Örneği

Örnek $(5 < 6)$

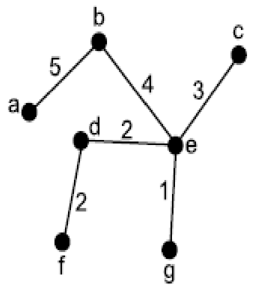


- $T = \{$
  $(e, g), (d, f), (d, e),$
  $(c, e), (b, e), (a, b)$
  $\}$
- $i \leftarrow 6$

10-40

Kruskal Algoritması Örneği

Örnek ($6 \not< 6$)

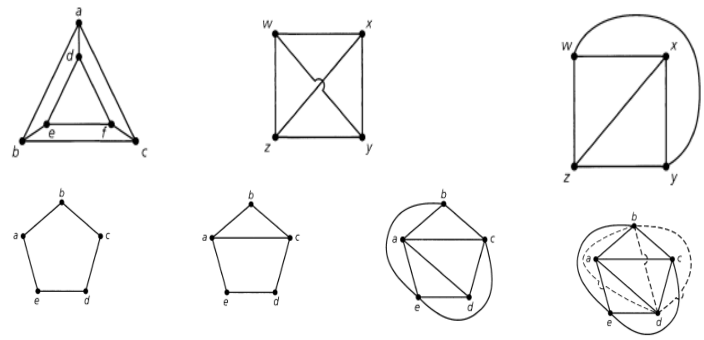


- toplam ağırlık: 17

10-41

Düzlemsel Graflar

- **Tanım:** Düğümleri düzlemde noktalar ve ayrıtları sadece grafın düğümlerinde kesişen doğrular veya yaylar olan grafa düzlem grafı denir.
- Bir graf, eğer bir düzlem grafıyla izomorfik ise örneğin düzlemde hiçbir ayrıtı kesişmeden bir diyagram ile temsil edilebiliyorsa düzlemsel (planar) graftır.

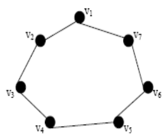


10-42

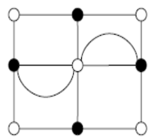
Euler Formülü

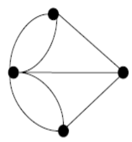
- *G* bağlı düzlemsel bir graf olsun. Düzlemde çizilen *G* 'nın diyagramı 'yüz' (face) adını verdiğimiz bölgelere ayırır.
- Bağlı düzlemsel bir grafın düğümlerinin, ayrıtlarının ve yüzlerinin sayısı arasında bir ilişki kurmak için basit bir formül vardır.
- Aşağıdaki tablo bu formülü görmek için faydalı olabilir.

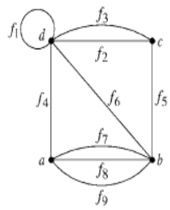
Graf	*Düğüm Sayısı*	*Ayrıt Sayısı*	*Yüz Sayısı*
Şekil 7.2 (a)	7	7	2
Şekil 7.3 (a)	9	14	7
Şekil 7.4 (b)	4	7	5
Şekil 7.6 (b)	4	9	7
Herhangi ağaç	n	n-1	1



Şekil 7.2 (a)


Şekil 7.3 (a)


Şekil 7.4 (b)



Şekil 7.6 (b)

- Bütün bu graflar bağlıdır ve düzlemseldir ve |F|, |E|, |V| sırasıyla yüzlerin, ayrıtların ve düğümlerin sayısı olmak üzere

$$|F| = |E| - |V| + 2$$

ilişkisini sağlarlar. Bu ilişki tüm bağlı düzlemsel graflar için sağlanır ve Euler' in formülü olarak bilinir.

Teorem: *G*, |V| düğümlü, |E| ayrıtlı ve düzlemi |F| yüze veya bölgeye ayıran herhangi bir bağlı düzlemsel graf olsun. Bu durumda,

$$|F|=|E| -|V| + 2$$

olur.

İspat: *G'* nın ayrıt sayısına tümevarım yöntemi uygulayarak ispat yapılabilir. |E|=0 ise |V|=1 (*G* bağlıdır o halde iki veya daha fazla düğüm olamaz) ve tek bir yüz vardır yani |F|=1. Bu nedenle bu durum için teorem doğrudur.

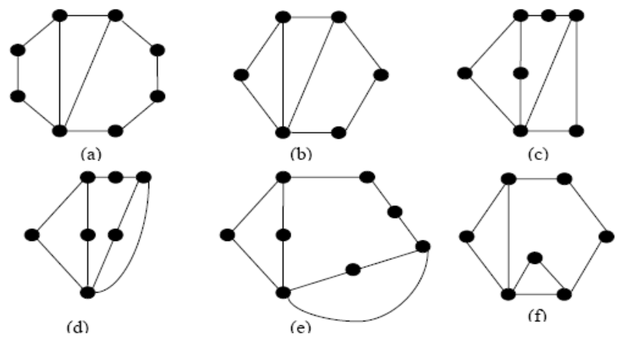
Şimdi, teoremin n ayrıttan az graflar için de sağlandığını düşünelim. *G*, n ayrıtlı bağlı düzlemsel graf olsun; yani |E|=n. *G* bir ağaç ise |V|=n+1 (teorem 7.5) ve |F|=1 o halde, teorem bu durumda da sağlanır. Eğer *G* bir ağaç değilse *G*'deki herhangi bir devreyi seç ve bir ayrıtını sil. Sonuçtaki graf *G'* bağlıdır, düzlemseldir, n-1 ayrıtı, |V| düğümü, ve |F|-1 yüzü vardır. Tümevarımsal hipoteze dayanarak Euler' in formülü *G'* için sağlanır.

$$|F|-1 = (|E|-1) - |V| + 2 \quad \text{o halde,}$$
$$|F| = |E| - |V| + 2.$$

Tanım: Eğer bir graf, diğer bir grafın ayrıtlarına derecesi 2 olan düğümler ekleyerek veya çıkararak elde edilebiliyorsa bu iki graf homeomorfiktir (izomorfik kopyasıdır).

Örnek: Şekil 7.8 de gösterilen grafların hepsi homeomorfiktir.

- (a) daki graftan (b) dekini elde etmek için 2 düğüm sileriz.
- (b) dekinden (c)' deki grafı elde etmek için bir düğüm sileriz ve iki düğüm ekleriz.
- (d) den (e) yi elde etmek için bir düğüm ekleriz.
- (e) ve (f) deki graflar izomorfiktir- herhangi bir düğümün eklenmesine veya çıkarılmasına gerek yoktur.

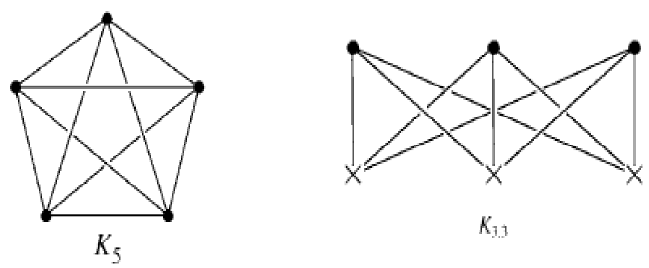


Şekil 7.8

10-45

Kuratowski 'nin Teoremi

- Bir Grafın düzlemsel olması için ancak ve ancak K_5 ve $K_{3,3}$'e homeorfik hiçbir alt grafının olmamasıdır.

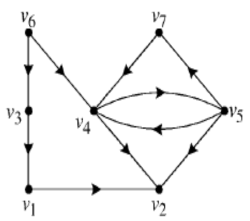


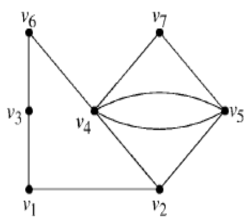
10-46

Yönlü Graflar

Tanım: Yönlü graf veya digraf D, sonlu $V=V_D$ düğümlerinin, $E=E_D$ yönlü ayrıtlarının boş olmayan ve $\delta:E\rightarrow V\times V$ tasvirinden oluşur. Eğer $\delta(e)=(u,v)$ ise u, e ayrıtının başlangıç düğümü ve v ise son düğümü olarak adlandırılır.

- Verilen D digrafında ayrıtların yönlendirilmesi ihmal edilerek yönsüz Γ grafı elde edilir.
- Eğer döngü ve çoklu kenarlar yoksa digraf basit olarak adlandırılır.
- Bitişiklilik, bağlı olmak ve derece kavramları graflardakinin aynısıdır.





$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

10-47

Tanım: Eulerian digraf, her ayrıtı içeren kapalı yönlü yol içeren digraftır.

Tanım: Hamiltonian digraf her düğümden geçen yönlü çevrim içeren digraftır.

Tanım: Son düğümü v olarak düşünülen ayrıtların sayısı v düğümünün iç-derecesidir.

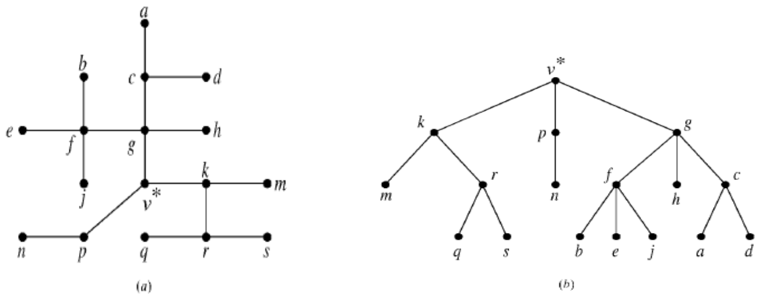
Tanım: Başlangıç düğümü v olarak düşünülen ayrıtların sayısı dış-derecesi olarak adlandırılır.

Teorem: Bağlantılı digraf ancak ve ancak her düğümünün iç-derecesi dış-derecesine eşit ise Euleriandir.

10-48

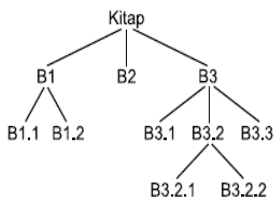
Köklü Ağaçlar

- Tanım: Köklü ağaç, T bir ağaç ve $v^* \in V_T$ olan (T,v^*) çiftidir. Buradaki ayrıcalıklı v düğümü ağacın köküdür.
- Köklü ağaçtaki yaparak derecesi bir olan ve köke eşit olmayan düğümdür.
- Karar düğümü (iç düğüm) kök veya yaprak olmayan düğümdür.
- Tanım: (T,v^*) köklü ağaç olsun. T'nin w düğümünün seviyesi, T içerisindeki v^* ile w tekbir yolun uzunluğudur.
- Tanım: T'nin yüksekliği, düğüm seviyelerinin maksimumudur.



Köklü Ağaç Örneği

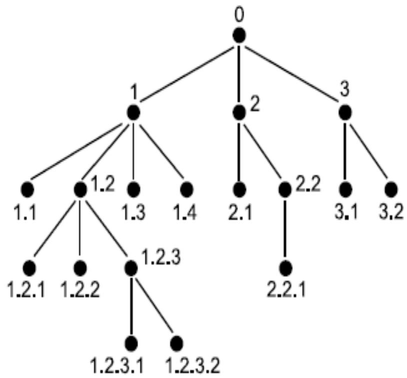
Örnek (kitap düzeni)



Kitap
- B1
  - B1.1
  - B1.2
- B2
- B3
  - B3.1
  - B3.2
    - B3.2.1
    - B3.2.2
  - B3.3

Sıralı Köklü Ağaç

Örnek (evrensel adresleme sistemi)



- 0 - 1 - 1.1 - 1.2
  - 1.2.1 - 1.2.2 - 1.2.3
  - 1.2.3.1 - 1.2.3.2
  - 1.3 - 1.4 - 2
  - 2.1 - 2.2 - 2.2.1
  - 3 - 3.1 - 3.2
- sözlük sırası

- Tanım: (T,v*) köklü ağaç olsun ve p ise k > 0 seviyeli düğüm olsun. P'ye bitişik olan k-1 seviyeli q düğümü p'nin ebeveyni, p ise q'nun çocuğu olur. q ya bitişik olan k seviyeli düğüm p'nin kardeşi olarak adlandırılır.

- m-li ağaç, ebeveyn düğümü en fazla m çocuğa sahip köklü ağaçtır.
- İkili ve üçlü ağaçlar sırasıyla m=2 ve m=3 durumları için kullanılır.
- Eğer m-li ağacın her ebeveyn düğümü m çocuğa sahip ise ağaç dolu olarak adlandırılır.
- Eğer tüm yaprak düğümler köklü ağacın yüksekliği olan h seviyesinde ise tam olarak adlandırılır.

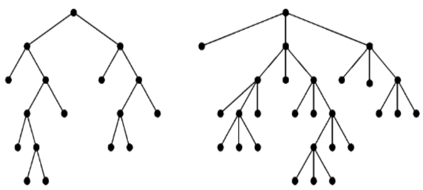

(a) (b)

Figure 11.5

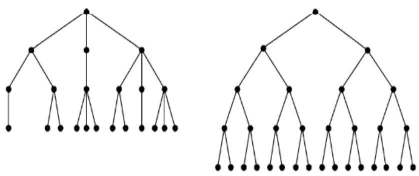
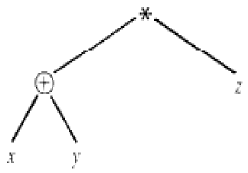

(a) (b)

Figure 11.6

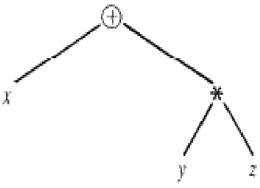
Cebirsel ifadelerin Köklü Ağaçlar Yardımıyla İfadesi

- Bir ifade rekürsif olarak XαY şeklinde tanımlanabilir. Burada α ikili işlemin operatörü X ve Y ise işlemin operantları olarak adlandırılır.
- Böyle ifadeler, kökü α, sol çocuğu X ve sağ çocuğu Y olan ikili ağaç olarak ifade edilebilir.
- Aşağıdaki olağan notasyon infix form olarak da adlandırılır.

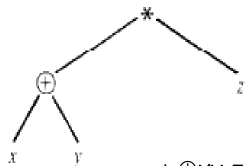
$(x \oplus y) * z$



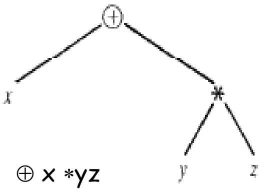
$x \oplus (y * z)$



- Bir ifadenin Prefix (Polish) form operatörün iki operandın önüne yazılmasıdır. XαY ifadesinin prefix formu αXY olur.



$* \oplus xy\ z$



$\oplus\ x\ *yz$

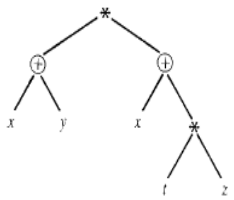
- Bir ifadenin Postfix (reverse Polish) form operatörün iki operandın ardına yazılmasıdır. XαY ifadesinin prefix formu XYα olur.

$xy\oplus\ z\ *$ $\qquad$ $x\ yz\ *\ \oplus$

1. içek geçişi: sol altağacı tara, köke uğra, sağ altağacı tara
2. önek geçişi: köke uğra, sol altağacı tara, sağ altağacı tara
3. sonek geçişi: sol altağacı tara, sağ altağacı tara, köke uğra
   - *ters Polonyalı gösterilimi*

Örnek:

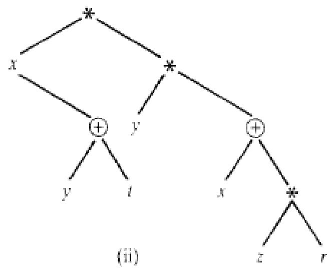
Aşağıdaki ikili ağaçları infix, prefix ve postfix formunda yazınız.



Infix: $(x \oplus y) * (x \oplus (t * z))$

Prefix: $* \oplus x y \oplus x * t z$

Postfix: $x y \oplus x t z * \oplus *.$



(ii)

10-55