MAT223 AYRIK MATEMATİK

Ağaçlar
8. Bölüm

Doç. Dr. Emrah Akyar

Anadolu Üniversitesi
Fen Fakültesi Matematik Bölümü, ESKİŞEHİR

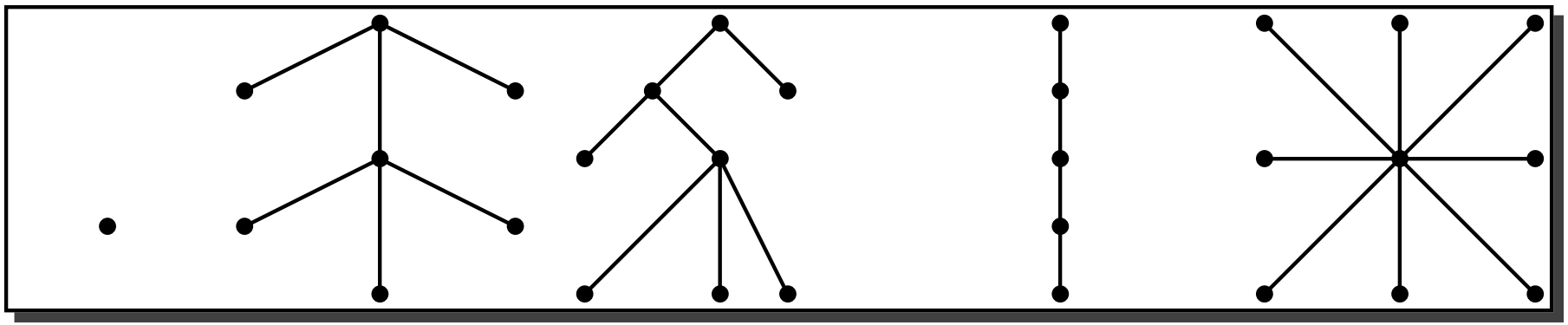
2011–2012 Güz Dönemi

Ağacın Tanımı

Tanım

Döngüsü olmayan tekparça $G = (V, E)$ çizgesine ağaç denir.

Bu tanıma göre aşağıdaki bazı çizgeleri ağaçlara örnek olarak verebiliriz.



Döngüsü olmayan çizgelere ise orman denir. Açıktır ki, ağaç tekparça bir ormandır.

Yukarıdaki ağaçların tümünü tek bir çizge gibi düşünürsek bu çizge bir orman olur.

Ağacın tanımındaki iki özellik yardımıyla

- Ağacın çok fazla sayıda kenar bulundurmadığını söyleyebiliriz (döngü içermiyor).
- Ağacın çok az sayıda kenar bulundurmadığını da söyleyebiliriz (tekparça).

Aşağıdaki teorem, ağaçların minimal tekparça ya da maksimal döngü içermeyen çizge olarak karakterize edilebileceğini ifade eder.

Teorem

- Bir G çizgesinin bir ağaç belirtmesi için gerek ve yeter koşul, G nin tekparça olması ancak, herhangi bir kenarı çıkarıldığında tekparça olmamasıdır.
- Bir G çizgesinin bir ağaç belirtmesi için gerek ve yeter koşul G nin döngü içermemesi ancak, yeni bir kenar eklendiğinde döngü içermesidir.

Kanıt.

Teoremin sadece ilk kısmını kanıtlayacağız. İkinci kısmı da benzer olarak kanıtlanır.

($\Rightarrow$) G çizgesi bir ağaç olsun. Bu durumda ağacın tanımından G tekparça olur. Geriye G den bir kenar çıkarıldığında tekparça olmadığını göstermek kalır.

Tersine, kabul edelim ki G den bir uv kenarı çıkarıldığında geriye kalan G' alt çizgesi tekparça olsun.

Bu durumda G' içinde u ve v noktalarını birleştiren bir P yolu var demektir.

Tekrar uv kenarını G' çizgesine eklersek, G de u ve v noktalarını birleştiren P ve uv gibi iki farklı yol elde edileceğinden, G bir döngü içerir.

Bu ise G nin ağaç olması ile çelişir.

O halde varsayımımız hatalıdır, G çizgesinden bir kenar çıkarıldığında tekparça olamaz. □

($\Leftarrow$) Koşullar sağlansın G ağaç mı?

G tekparça olarak verildiğinden geriye G nin bir döngü bulundurmadığını göstermek kalır.

Yine tersine, kabul edelim ki G bir döngü bulundursun.

O zaman bu döngüye ait kenarlardan birini silersek elde edilen çizge hala tekparça olur.

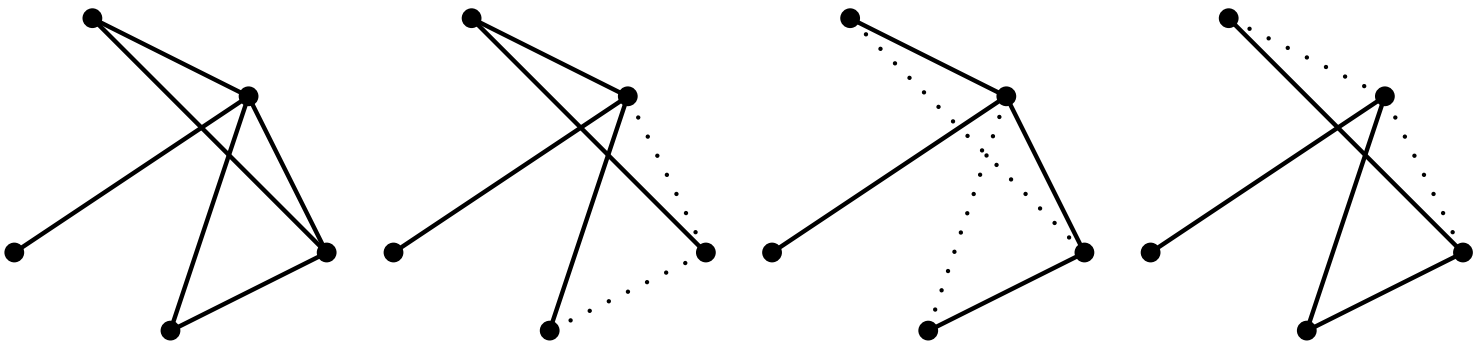
Bu ise verilen koşul ile çelişeceğinden G de bir döngü olamaz. □

Nokta sayısı n olan tekparça bir çizge verilsin.

Bu çizgenin bir e kenarı silindiğinde geriye kalan çizge tekparça olabilir de olmayabilir de. Eğer tekparça değilse, bu şekildeki e kenarına ayıran kenar (cut edge) denir.

Yukarıdaki teoremden bir ağacın her kenarının ayıran kenar olduğunu söyleyebiliriz.

Verilen tekparça bir G çizgesinin ayıran kenar olmayan kenarları silinerek bir ağaç elde edilebilir. Bu şekilde elde edilen ağaca G çizgesinin doğurduğu/ürettiği ağaç (spanning tree) denir.



G G nin doğurduğu ağaçlar

Çoğu zaman noktalarından biri özel olan ağaçlar çalışılır.

Bu özel olarak seçilen noktaya kök (root) denir.

Verilen bir ağacın herhangi bir noktası kök olarak seçilirse, bu ağaca köklü ağaç denir.

G köklü bir ağaç ve r noktası da G nin kökü olsun. G ağacının r noktasından farklı olan bir v noktasını ele alalım.v ile r noktasını birleştiren yolda v noktasının komşusu olan noktaya v noktasının babası denir.

Not: Açıktır ki, bir ağaçta herhangi iki düğümü birleştiren bir tek yol vardır (Alıştırma 8.1.3).

v noktasının diğer komşu noktalarına ise, v nin çocukları denir. G çizgesinin çocuğu olmayan noktalarına G nin yaprakları denir. Bir başka ifadeyle, ağacın derecesi 1 olan noktalarına ağacın yaprakları denir.

Açıktır ki, r noktasının babası yoktur. Ancak, r nin her komşusu r nin çocuğu olur.

Ağaçlar Nasıl Büyür?

Aşağıdaki teorem ağaçların en önemli özelliklerinin birinden söz etmektedir.

Teorem

En az iki noktası olan bir ağacın, derecesi 1 olan en az iki noktası vardır.

Bir başka ifadeyle en az iki noktası olan bir ağacın en az iki yaprağı vardır.

Kanıt.

Ağacı G ile gösterelim.

G ağacındaki en uzun yollardan birini biri alalım (en uzun yol tek olmayabilir).

Bu yol u noktasında başlayıp v noktasında bitsin. Bu durumda u ile v noktalarının derecesi 1 olmak zorundadır (aksi halde seçilen yol en uzun yol olmazdı).

Böylece kanıt biter. □

Alıştırma (8.5.4)

Noktalarından birinin derecesi d olan bir ağacın en az d tane yaprağı olduğunu kanıtlayınız.

Bir T ağacının derecesi d olan noktasını u ile gösterelim.

Eğer u noktasını silersek, d tane tekparça alt çizge elde ederiz (tekparça bileşeni). Bu bileşenler ya tek noktadır ya da en az iki noktası olan ağaçlardır.

Eğer bunlar en az iki noktası olan ağaçlar ise biliyoruz ki, nokta sayısı en az iki olan bir ağaçta en az iki yaprak vardır (Teorem 8.2.1).

Eğer bunlar tek nokta ise o zaman bu noktalar zaten T ağacının yaprakları olur.

O halde d tane bileşenin her biri en az bir yaprak içerdiğinden, T de en az d yaprak içerir.

Gerçek hayatta ağaçlar yeni sürgünler vererek büyürler. Çizge–ağaçlar da aşağıda anlatılan yöntemle gerçek ağaçlara benzer şekilde büyürler.

Tree–Growing Procedure

1. Adım G çizgesi tek nokta olsun.
2. Adım G çizgesine yeni bir nokta ekle. Bu yeni noktayı G nin herhangi bir noktası ile birleştir. Elde edilen çizgeyi yine G ile gösterelim.
   2. Adımı istenilen kadar tekrar edelim.

Teorem

Yukarıdaki yöntem ile elde edilen her çizge bir ağaç belirtir ve her ağaç bu şekilde elde edilebilir.

Bu şekilde elde edilen bir çizgenin ağaç olduğunu göstermek kolaydır. Başlangıçtaki tek noktadan oluşan çizge bir ağaçtır. Bundan sonra çizgeye her seferinde yeni bir nokta eklemekle elde edilen çizge de bir ağaç olur. Çünkü, yöntem gereği elde edilen çizgeler tekparçadır ve döngü içermez.

Şimdi her ağacın bu yöntemle elde edilebileceğini kanıtlayalım.
Kanıtı tümevarım yöntemi ile yapalım: Eğer G nin nokta sayısı 1 ise, 1. adımla bu ağaç elde edilebilir.
Şimdi G nin en az 2 noktasının olduğunu kabul edelim.

- Bu durumda G nin derecesi 1 olan en az iki noktası vardır.
- v noktası G nin derecesi 1 olan noktalarından biri olsun.
- v noktasını ve bu noktanın uç nokta olduğu kenarı silelim ve elde edilen çizgeyi G' ile gösterelim.
- G' nün herhangi iki noktası G deki bir yol ile birleştirilebildiğinden G' tekparçadır (v nin derecesi 1 olduğundan bu yol v den geçmez. Yani, v yi çıkarmak bir sorun oluşturmaz).
- Ayrıca, G döngü içermediğinden G' de döngü içermez.
- O halde G' çizgesi bir ağaçtır.

Tümevarım hipotezi: G den daha az sayıda noktası olan her çizge bu yöntemle elde edilebilsin. Dolayısıyla G' de bu yöntemle elde edilebilsin.

G' çizgesine 2. adımda v noktasını ekleyerek, G çizgesini elde edebileceğimizden kanıt tamamlanır.

Tree–Growing Procedure ile ağaçların birçok özelliğini elde etmek mümkündür. Aşağıdaki teorem ile verilen özellik bunlar içinde belki de en önemlisidir.

Teorem

Nokta sayısı n olan bir ağacın $n-1$ tane kenarı vardır.

Kanıt.

Yukarıdaki yönteme göre her ağaç tek noktadan (0 tane kenar) başlanarak, her seferinde 1 nokta ve 1 kenar eklenerek elde edilebildiğine göre n noktalı bir ağaç $n-1$ kenar eklenerek elde edilebilir.

Böylece kanıt biter. □

Ağaçların Sayısı

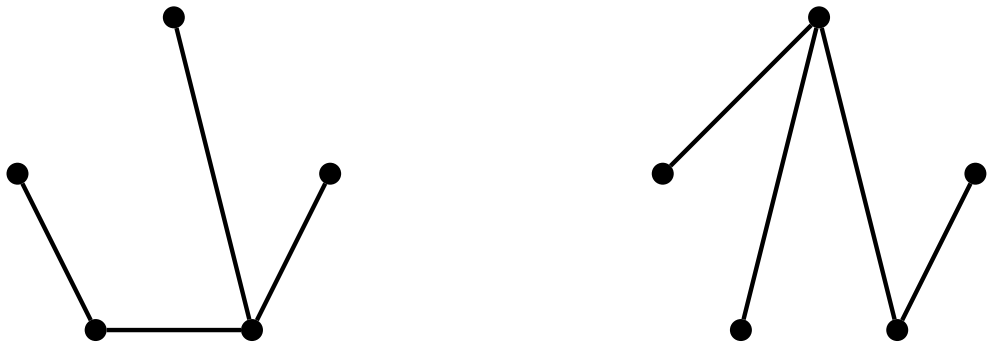
Bu dersin başından beri her türlü şeyin sayısını hesapladık. Bu durumda şu soruyu sormak gayet doğaldır:

Soru

Nokta sayısı n olan kaç tane ağaç vardır

Cevap

Duruma göre değişir!



Bu iki ağaç aynı mı?

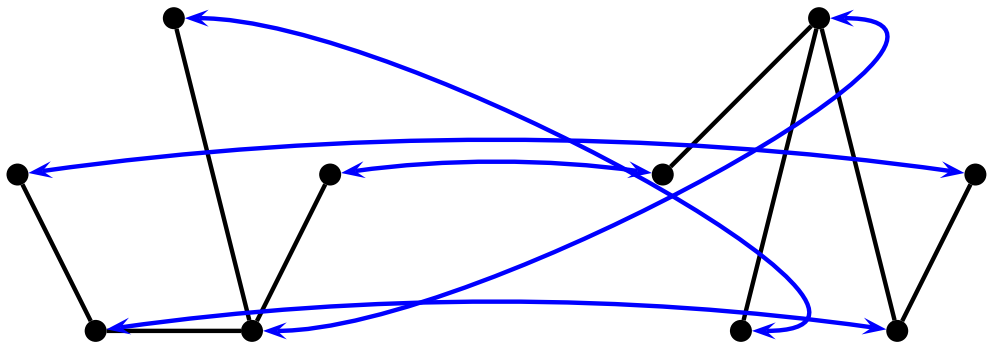
- Eğer noktaları şehirler, kenarları da bu şehirler arasındaki yollar şeklinde düşünürsek yukarıda verilen iki ağaç birbirinden farklı olur (Birinde iki şehir arasında yol varken diğerinde aynı şehirler arasında yol yok).

  Noktaları adlandırılmış ağaç!

- Eğer noktaları adlandırmadan tüm noktaları aynı kabul edersek iki ağaç aynı olur (Noktaların yerlerini değiştirebiliriz).

  Noktaları adlandırılmamış ağaç!

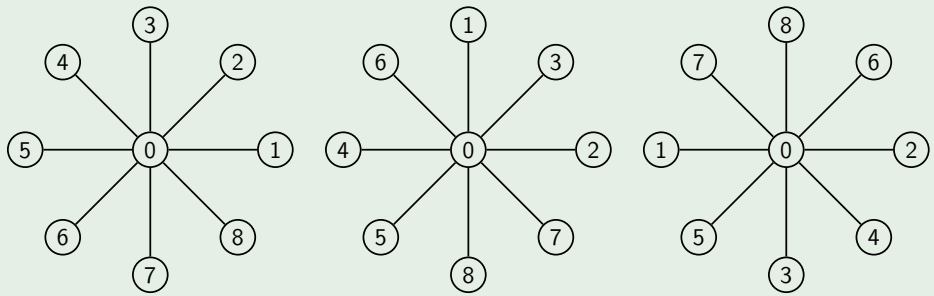
Eğer iki çizgenin (ağacın) noktaları arasında birebir bir eşleme varsa öyle ki, iki noktayı birleştiren bir kenar varsa eşlenen noktalar arasında da bir kenar olacak ve tersi de doğru olacak. Bu durumda bu iki çizgeye izomorf çizgeler denir.



Alıştırma (8.3.2)

Noktaları adlandırılmış, n noktalı kaç farklı yıldız ağaç vardır? Yol vardır?

Yıldız ağaçta yaprak olan noktaların kendi arasında yer değiştirmesi ağacı değiştirmez. Sadece merkezdeki nokta değiştiğinde farklı ağaçlar elde edilir. Merkezdeki nokta için n tane seçenek olduğundan cevap n olur.



Aynı ağaç!

Nokta sayısı n olan yol üzerindeki noktalar $n!$ farklı şekilde sıralanabilir. Ancak, $u_1 u_2 u_3 \ldots u_{n-1} u_n$ yolu ile $u_n u_{n-1} \ldots u_3 u_2 u_1$ yolu aynı ağacı verir.

O halde cevap $\dfrac{n!}{2}$ olur.

Aşağıdaki teorem noktaları adlandırılmış ağaçların sayısını vermektedir.

Teorem (Cayley Teoremi, 1889)

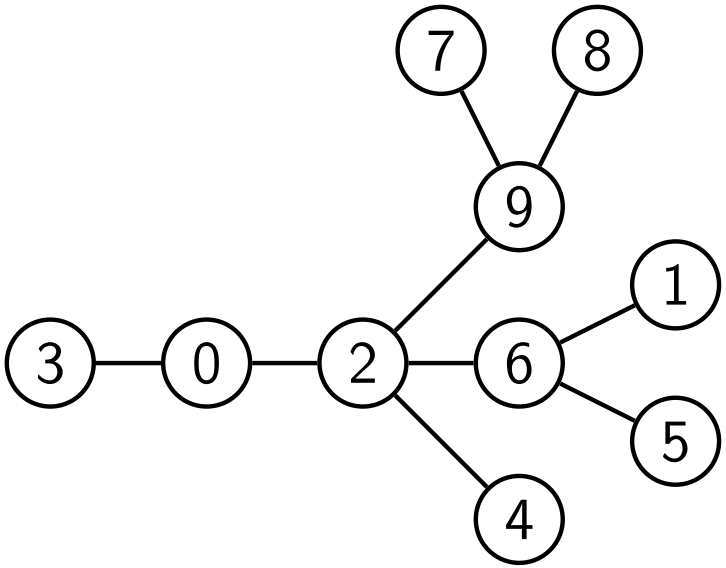
Nokta sayısı n olan n^{n-2} tane noktaları adlandırılmış ağaç vardır.

Formül gayet basit gözükmesine karşın, teoremin kanıtı çok da kolay değildir. Teoremin anlaşılır bir kanıtını verebilmek için önce ağaçların bilgisayarın belleğinde en az yer kaplayacak şekilde nasıl saklanabileceğini inceleyeceğiz. Daha sonra buradan çıkaracağımız sonuçlar ile teoremin kanıtını vereceğiz.

Ağaçları Bilgisayarın Belleğinde Nasıl Saklarız?

Soru

Aşağıdaki ağacı bilgisayarın belleğinde nasıl saklayabiliriz?

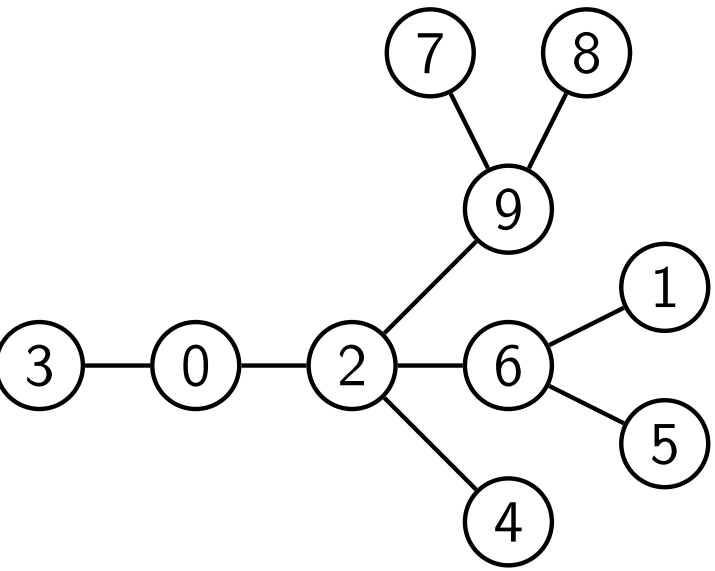


Elbette bu sorunun cevabı ağaç ile ilgili hangi bilgileri saklayacağımıza ve bu bilgileri hangi sıklıkta kullanacağımıza göre değişir.

Ağaçları bilgisayarın belleğinde en az yer kaplayacak şekilde saklamak istiyoruz.

1. Yöntem: Nokta sayısı n olan G çizgesini ele alalım. İlk akla gelen yöntem n tane satırı ve n tane sütunu olan bir tablo yapmak ve eğer çizgenin i. ve j. noktaları arasında bir kenar varsa tablonun (i, j). bileşenine 1, aksi taktirde 0 yazmaktır.

Örneğin, az önceki ağacı aşağıdaki tablo/matris ile gösterebiliriz.



	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	1
4.	0	1	0	0	0	0	0	0	0	0
5.	0	0	0	0	0	1	0	0	0	0
6.	1	1	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	0	0	1	0
9.	0	1	0	0	0	0	1	1	0	0
0.	0	1	1	0	0	0	0	0	0	0

Bu yöntemle n noktalı bir ağacı bellekte saklamak için n^2 bit gerekir.

	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	1
4.	0	1	0	0	0	0	0	0	0	0
5.	0	0	0	0	0	1	0	0	0	0
6.	1	1	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	0	0	1	0
9.	0	1	0	0	0	0	1	1	0	0
0.	0	1	1	0	0	0	0	0	0	0

(Adjacency Matrix)

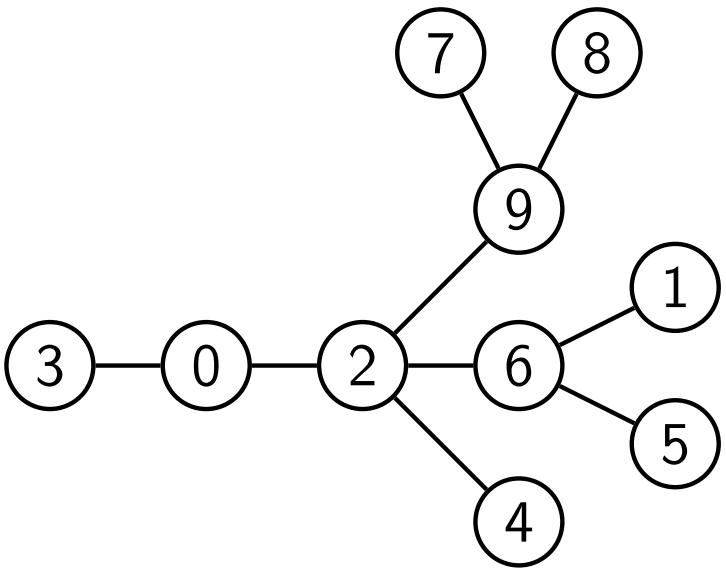
- Köşegen üzerindekiler daima 0 olacaktır. Bu nedenle bunları saklamaya gerek yok.
- i. nokta ile j. nokta arasında bir kenar varsa, j. nokta ile i. nokta arasında da kenar vardır. Yani matris simetrik!

O halde sadece köşegenin altında ya da üstünde kalan parça yeterlidir. Böylece bu yönteme göre n noktalı bir ağacı saklamak için,

$$1 + 2 + \cdots + (n-1) = \frac{(n-1)n}{2} = \frac{n^2 - n}{2}$$

bit yeterlidir.

2. Yöntem: Noktaları saklamayıp, sadece kenarları saklayabiliriz. Örneğin,



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

Böylece 1. yöntemdeki gibi n satır kullanmak yerine sadece 2 satır kullandık.

Nokta sayısı n olan bir ağacın kenar sayısı $n-1$ olduğundan $n-1$ tane sütun gereklidir.

Ancak, bu kez "0" ve "1" ler (bit) yerine 0 ile $n-1$ arasındaki tamsayıları kullandık.

Hatırlatma

Bir n pozitif tamsayısını ikilik sistemde yazmak için $[\![\log_2 n]\!] + 1$ bit gerekir.

O halde bu yönteme göre n noktalı bir ağacı bellekte saklamak için

$$2 \cdot (n-1) \cdot ([\![\log_2 n]\!] + 1)$$

bit gerekir.

Yeterince büyük n sayıları için

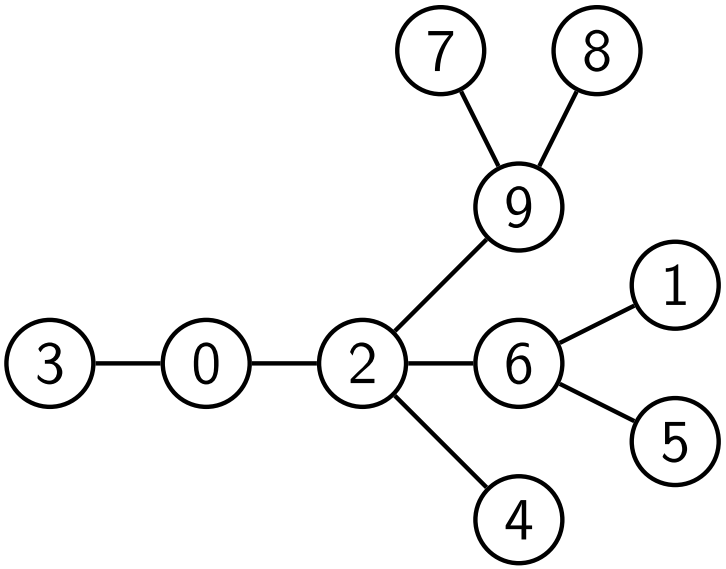
$$2 \cdot (n-1) \cdot ([\![\log_2 n]\!] + 1) < \frac{n^2 - n}{2}$$

olur.
Bu yazım şekli tek değildir (Önce hangi kenarın yazılacağı ya da bir kenar yazılırken kenarın hangi uç noktasının üstte hangisinin altta yazılacağı belli değil). Ancak, belli kurallar ile bu yazım şeklinde tek türlü hale getirebiliriz (Örneğin, küçükten büyüğe sıralama, vb).

3. Yöntem: (Father Code) Bu yöntemde 0. noktayı kök olarak kabul edeceğiz.

2. Yönteme benzer olarak yine sadece kenarları listeleyeceğiz. Ancak, bu sefer bir kenarı yazarken üste o kenarın kökten uzakta olan uç noktasını, alta ise kenarın köke yakın olan uç noktasını yazacağız. Yani, alt satırdakiler üst satırdakilerin babası olacak.

Kenarları da kendi içerisinde üst satırdaki sayılara göre sıralayacağız. Örneğin,



Çocuk

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

Baba

Bu yönteme göre,

- 0 üst satırda bulunmaz (Kökün babası yok).
- 0 hariç tüm noktalar üst satırda bulunmak zorundadır (Kök hariç hepsinin babası var).
- Üst satırda hiç bir rakam iki kez bulunmaz (Tüm noktaların bir tek babası var).
- Böylece, n noktalı bir ağaçta üst satır $1, 2, 3, \ldots, n-1$ sayılarından oluşmak zorunda (yani sabit).
- O zaman ağacı saklamak için sadece alttaki satır yeterli olur.
- Yani 0 ile $(n-1)$ arasındaki $n-1$ tane tamsayıyı saklamak yeterlidir.
- Bunun için ise $(n-1) \cdot (\llbracket \log_2 n \rrbracket + 1)$ bit gereklidir.

Ne yazık ki her ağaca karşılık böyle bir kod yazabilmemize karşın, böyle her kod bir ağaç belirtmeyebilir.

Alıştırma (8.4.1)

Aşağıda verilenlerden hangileri bir ağaç için geçerli bir "father code" olur?

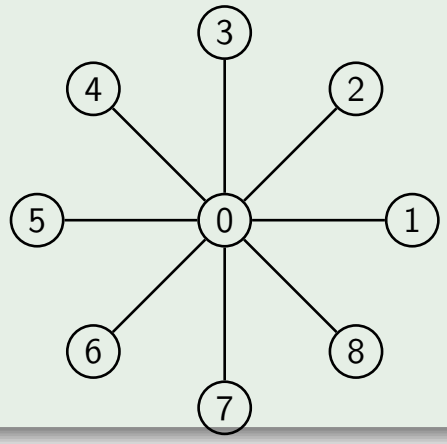
- $(0, 1, 2, 3, 4, 5, 6, 7)$

	1	2	3	4	5	6	7	8	$\rightarrow$ (çocuklar)
	0	1	2	3	4	5	6	7	$\rightarrow$ (babaları)

(Yol!)

- $(0, 0, 0, 0, 0, 0, 0, 0)$

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



Alıştırma (8.4.1)

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

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

Bir ağaç belirtmez!

- $(2, 3, 1, 2, 3, 1, 2, 3)$

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

Kök nerede? Ağaç belirtmez!

4. Yöntem: (Prüfer Code) Bu yöntem 3. Yöntemin iyileştirilmesidir. 0 yine kök olsun.

Bu yöntemle noktaları adlandırılmış, n noktalı ağacı $n-1$ yerine $n-2$ tane $0, 1, \ldots, n-1$ tamsayılarından oluşan dizilerle ilişkilendireceğiz.

Bu yöntemin en önemli yanı bu tür diziler ile n noktalı, noktaları adlandırılmış ağaçlar arasında birebir bir eşleme sunmasıdır.

Bu durumda n noktalı noktaları adlandırılmış ağaçların sayısı

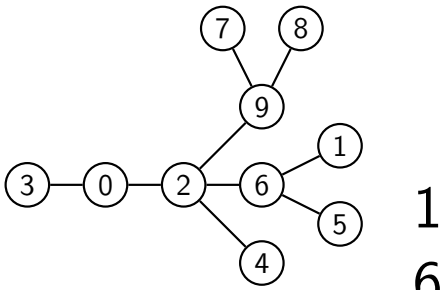
$$\underbrace{\square\square\square\ldots\square}_{n-2 \text{ tane}} = n^{n-2}$$

olur (Cayley Teoremi).

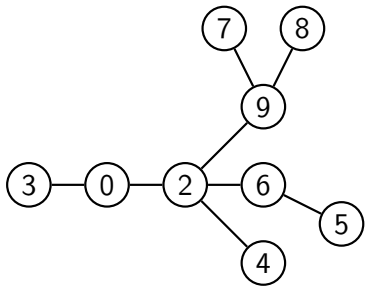
Yine ağacın kenarlarını önceki yöntemde olduğu gibi üst satırda çocuklar, alt satırda ise babaları olacak şekilde yazalım. Ancak, bu sefer kenarları sıralarken büyüklüklerine göre sıralamayacağız.

Sıralamayı şu kurala göre yapacağız:

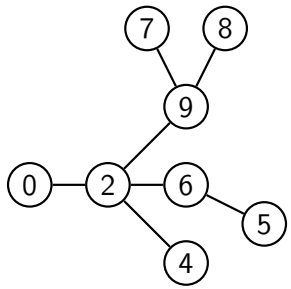
- Derecesi 1 olan ve kökten farklı olan noktalardan en küçük olanını seç ve babasıyla birlikte yaz.
- Daha sonra bu noktayı ve kenarı silerek birinci adıma geri dön.



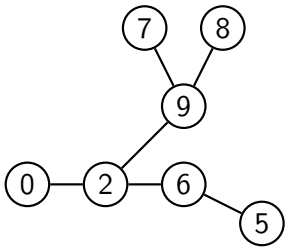
1
6



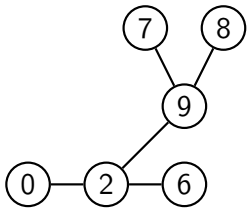
1 3
6 0



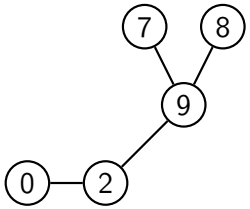
1 3 4
6 0 2



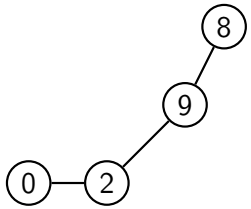
1 3 4 5
6 0 2 6



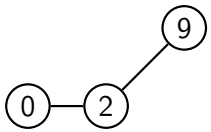
1 3 4 5 6
6 0 2 6 2



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

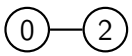


1	3	4	5	6	7	8
6	0	2	6	2	9	9



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

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



(Extended Prüfer Code)

En sona kök olan nokta kalacağı için alt satırın son terimi daima 0 olur.

Lemma

Prüfer dizisinin ikinci satırı birinci satırını belirler.

Lemmanın kanıtını yapmadan önce birkaç örnek ile sadece ikinci satır verildiğinde extended Prüfer code un nasıl bulunacağını görelim.

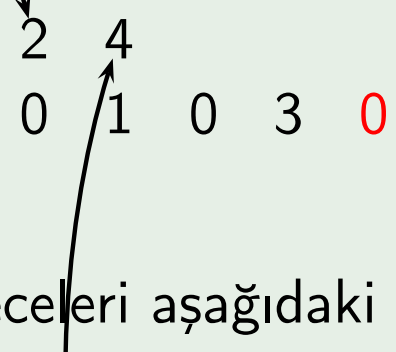
Alıştırma

$(0, 1, 0, 3)$ Prüfer dizisine karşılık gelen ağacı çiziniz.

- Prüfer dizisi $n - 2$ tane tamsayıdan oluştuğuna göre, bu ağaç 6 noktalıdır.
- Verilen diziye göre ağacın noktalarının dereceleri aşağıdaki gibidir:

Noktalar	0	1	2	3	4	5
Dereceleri	3	2	1	2	1	1

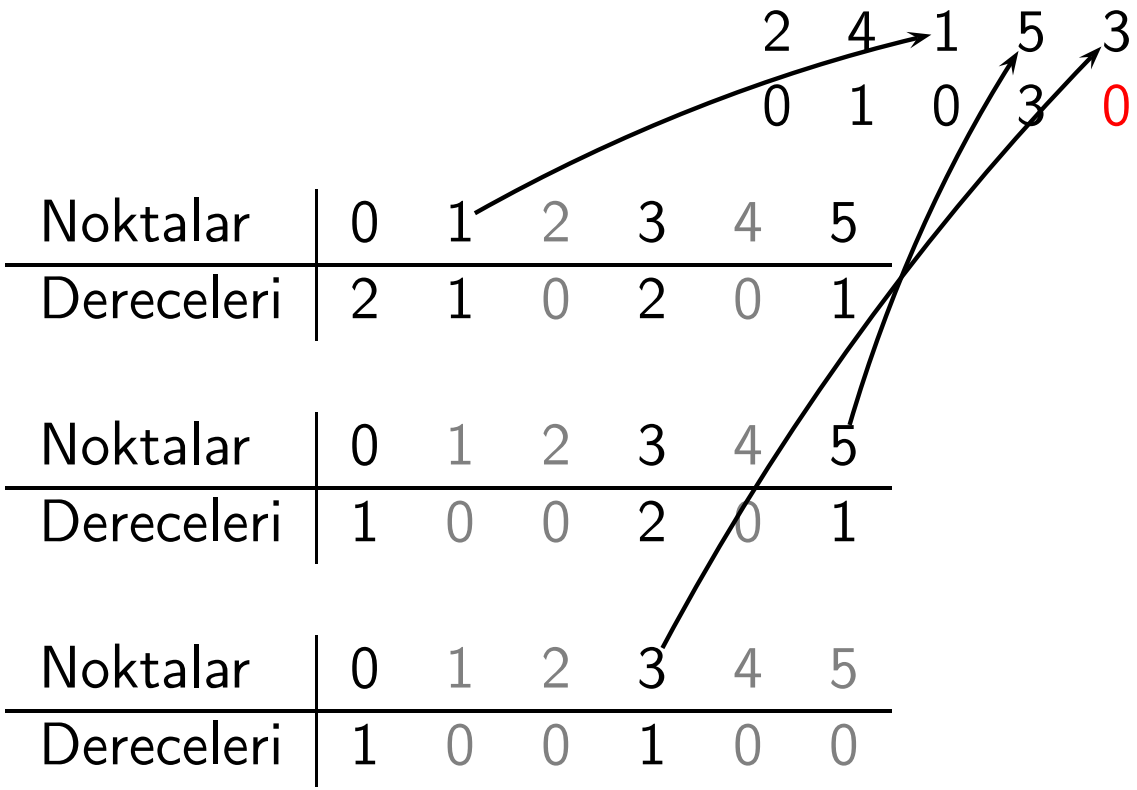
- Yukarıdaki tabloya göre derecesi 1 olan en küçük nokta 2 dir. Tersten gidersek, ilk silinen nokta 2 noktasıdır.



- Şimdi kalan noktaların dereceleri aşağıdaki gibi olur:

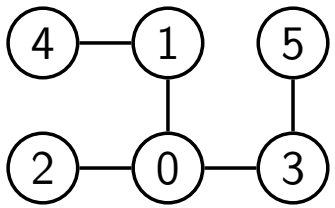
Noktalar	0	1	2	3	4	5
Dereceleri	2	2	0	2	1	1

- Bu şekilde devam edilirse...



2 4 1 5 3
0 1 0 3 0

$\Leftrightarrow$



Kanıt (Lemma).

Yukarıda izlenen yöntem genelleştirilebilir. Yani, Extended Prüfer kodun ilk satırındaki her bir terim ikinci satır yardımıyla yukarıda izlenen yönteme göre elde edilebilir.

(İlk satırdaki her bir terim kendinden önce yazılmamış olan altındaki ve altının sağında yer almayan en küçük tamsayıdır) □

O halde sadece ikinci satırdaki $n-2$ terimi saklamak yeterlidir (son terimin her zaman 0 olduğunu biliyoruz).

İkinci satırdaki bu $n-2$ terimin oluşturduğu diziye ise Prüfer kodu denir.

Prüfer kodunun diğer yöntemlere göre önemi aşağıda verilmiştir:

İddia

$n-2$ tane 0 ile $n-1$ arasındaki sayıdan oluşan her dizi n noktalı bir ağacın Prüfer kodu olur.

Bu iddianın doğruluğu açıktır. Önce verilen dizinin sonuna 0 ekler ve yukarıdaki yönteme göre extended Prüfer kodu elde ederiz. Daha sonra bu koda göre ağacı elde ederiz.

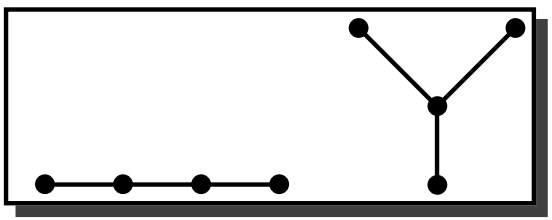
Noktaları Adlandırılmamış Ağaçların Sayısı

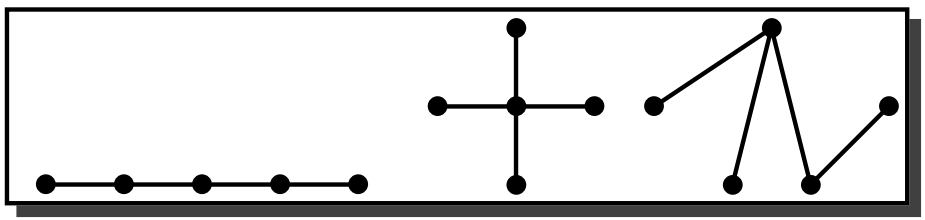
Noktaları adlandırılmamış, n noktalı ağaçların sayısı genellikle T_n ile gösterilir.

Ne yazık ki T_n için Cayley Teoremi gibi basit bir formül vermek mümkün değildir.

Bu bölümde sadece T_n sayısının hangi büyüklükte olduğunu kestirmeye çalışacağız.

Nokta Sayısı (n)	1	2	3	4	5
Noktaları Adlandırılmış Ağaçların Sayısı	1	1	3	16	125
Noktaları Adlandırılmamış Ağaçların Sayısı (T_n)	1	1	1	2	3



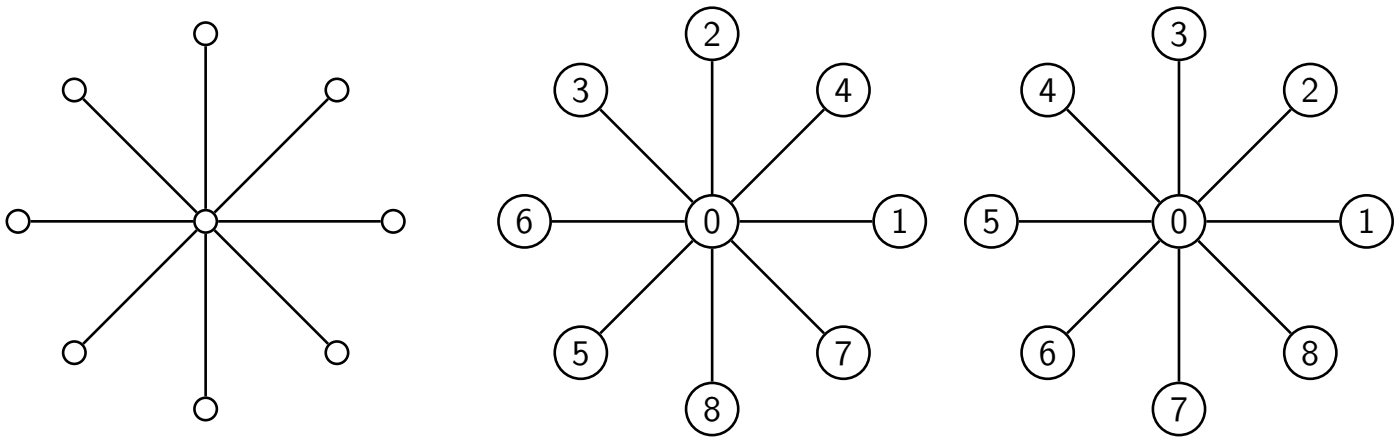


Yukarıdaki tablodan T_n sayılarının çok daha küçük olduğu görülüyor.

Bu sayıların küçük olması normaldir çünkü noktaları adlandırılmamış bir ağaç verildiğinde bu ağacın noktalarını çok farklı sayıda adlandırabiliriz.

Örneğin, n noktalı ve noktaları adlandırılmamış bir ağacın noktalarını $n!$ farklı biçimde adlandırabiliriz.

Ancak, bu durumda elde edilen ağaçların bazıları aynı olabilir. Mesela ağaç bir yıldız çizge ise



merkezdeki nokta değişmedikçe adlandırılan çizgeler hep aynı olur.

Yine de en kötü değerlendirme olarak $n!$ değerini kullanabiliriz.

Noktaları adlandırılmış, n noktalı ağaçların sayısı n^{n-2} olduğundan noktaları adlandırılmamış olanların sayısı en azından $\frac{n^{n-2}}{n!}$ olur.

Böylece,

$$T_n \geq \frac{n^{n-2}}{n!}$$

değerlendirmesini elde ederiz.

Stirling Formülünü kullanacak olursak,

$$T_n \geq \frac{n^{n-2}}{\left(\frac{n}{e}\right)^n \sqrt{2\pi n}} = e^n n^{-n} n^{n-2} n^{-1/2} \sqrt{2\pi} = e^n n^{-5/2} \sqrt{2\pi}$$

olur.

Sonuç olarak noktaları adlandırılmamış, n noktalı ağaçların sayısı için bir alt sınır bulmuş olduk.

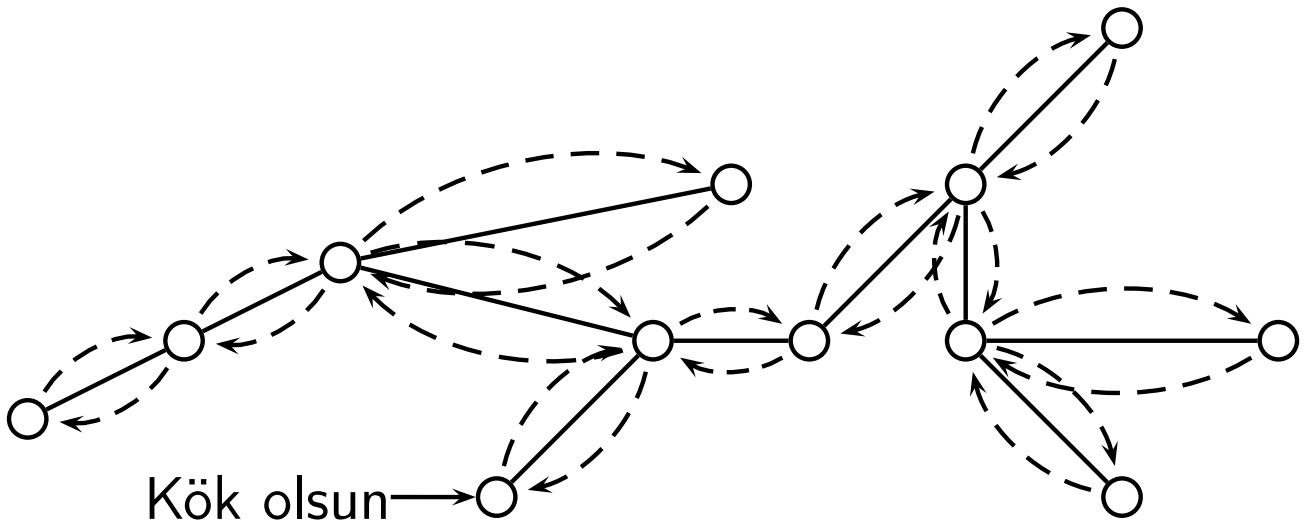
Şimdi de noktaları adlandırılmamış, n noktalı ağaçların sayısı için bir üst sınır belirlemeye çalışalım.

Acaba bir ağacı bilgisayarın belleğinde noktaları nasıl adlandırıldığından ziyade şeklini nasıl saklayabiliriz?

Aşağıda bunun için bir yöntem sunulmuştur:

- Nokta sayısı n olan bir G ağacını ele alalım. G ağacının *yapraklarından birini* kök olarak seçelim.
- G ağacını düzlemde kenarları birbiriyle kesişmeyecek şekilde çizelim.
- Şimdi G nin herbir kenarını düzleme dik olan duvarlar olarak düşünüp, kökten başlayarak sağ elimiz duvarda olacak şekilde bir turu düşünelim.

  Belli bir adım sonra tekrar başladığımız kök olan noktaya ulaşırız. Ağacın $n-1$ tane kenarı olduğundan ve herbir kenarın her iki yanından da geçtiğimizden bu turun uzunluğu $2(n-1)$ olur.
- Bu tur esnasında ulaştığımız nokta bir önceki noktanın çocuğu ise bunu "1" ile, babası ise "0" ile gösterelim.



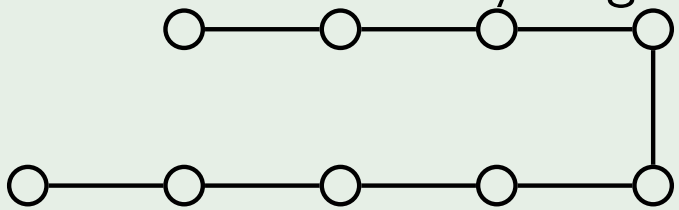
11110010011110110100000

Bu diziye düzlemsel kod (planar code) denir.

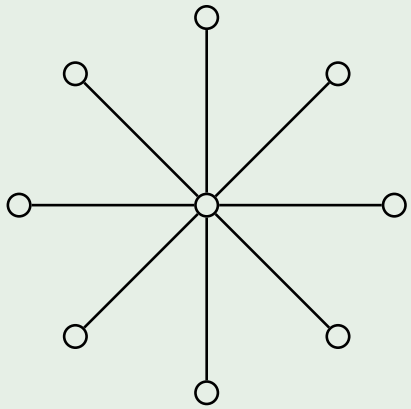
Acaba verilen koda karşılık gelen ağacı nasıl çizebiliriz?

Alıştırma (8.5.1)

- 111111110000000 düzlemsel koduna karşılık gelen ağacı çiziniz.



- 1010101010101010 düzlemsel koduna karşılık gelen ağacı çiziniz.



- 1100011100 düzlemsel koduna karşılık gelen ağacı çiziniz.

  Bir ağaç belirtmez!

  Başlangıçta 2 tane "1", 3 tane "0" var. Bu durum mümkün değildir!

Böylece yukarıdaki alıştırmadan "0" ve "1" lerden oluşan her kodun bir ağaç belirtmeyebileceğini görmüş olduk.

Diğer taraftan, "0" ve "1" lerden oluşan $2(n-1)$ uzunluğundaki farklı kodların sayısı $2^{2(n-1)}$ olur.

Öyleyse, noktaları adlandırılmamış, n noktalı ağaçların sayısı en fazla $2^{2(n-1)} = 4^{n-1}$ olur.

O halde aşağıdaki teorem geçerlidir.

Teorem

Noktaları adlandırılmamış, n noktalı ağaçların sayısı T_n,

$$\frac{n^{n-2}}{n!} \leq T_n \leq 4^{n-1}$$

eşitsizliğini sağlar.

Düzlemsel kodun birçok dezaavantajı da vardır. Örneğin, ağacın hangi noktasının kök olarak seçildiğine ya da ağacın nasıl çizildiğine bağlı olarak bir ağacın birden fazla düzlemsel kodu olabilir.

George Polya $a = 0.5349\ldots$ ve $b = 2.9557\ldots$ için

$$T_n \to an^{-5/2} b^n$$

olduğunu göstermiştir.