Algoritma Analizi (2)

ÖZYİNELEMELER (RECURRENCES)

Bir ***dizi****,* ***seri****,* ***fonksiyon****,* ***algoritma*** **kendi cinsinden tanımlanmasına özyineleme denir.**

Tanım bölgesi negatif tamsayılar olmayan fonksiyon tanımlanırken:

Temel Adım: Fonksiyonun sıfırdaki değeri belirtilir.

Özyinelemeli adım: Fonksiyonun *bir tamsayıdaki değeri hesaplanırken, fonksiyonun daha küçük tamsayılardaki değer(ler)ini kullanarak* bu değeri veren kural belirtilir.

ÖZYİNELEMEMELER (RECURRENCES)

- *Örneğin ikinin kuvvetlerinden oluşan dizi aşağıdaki gibi ifade edilebilir*

$\boldsymbol{a_n = 2^n}$

Fakat bu dizi özyinelemeli olarak aşağıdaki gibi ifade edilebilir.

$\boldsymbol{a_0}$ =1, $\boldsymbol{a_{n+1}}$ =2*$\boldsymbol{a_n}$

ÖZYİNELEMELER (RECURRENCES)

- **Örnek:** ***f*** fonksiyonu öz yinelemeli olarak aşağıdaki tanımlanmış olsun;

$\boldsymbol{f(n+1)=2f(n)+3}$ *ve* $f(0)=3$, *olsun* $\boldsymbol{f(1)}$, $\boldsymbol{f(2)}$, $\boldsymbol{f(3)}$ değerleri nedir?

$\boldsymbol{f(1)}= 2f(0)+ 3 = 2 * 3 + 3 = \mathbf{9}$

$\boldsymbol{f(2)} = 2f(1) + 3 = 2 * \mathbf{9} + 3 = \mathbf{21}$

$\boldsymbol{f(3)} = 2f(2) + 3 = 2 * \mathbf{21} + 3 = \mathbf{45}$

$\boldsymbol{f(4)} = 2f(3)+ 3 = 2 * \mathbf{45} + 3 = 93$

ÖZYİNELEMELER (RECURRENCES)

Yinelemeleri çözmek için genel <u>bir prosedür yada yöntem yoktur</u>.

<u>Sadece birtakım teknikler vardır</u>. Bunlardan bir kısmı oluşturulan yineleme için bunlardan çalışır.

Yinelemeler integral, türev, vs. denklemlerinin çözümlerine benzer.

ÖZYİNELEMELER (RECURRENCES)

❖ **Yinelemeleri çözmek konusunda 3 ana yöntem vardır.**

1. **Yerine koyma metodu (substitution)**
2. **Özyineleme ağacı (recursion tree)**
3. **Ana Metot (Master metod)**

❖ Ana yöntemlerin dışında tekrarlı bağıntılar *karakteristik denklemler* kullanarak çözülebilir.

1.Yerine koyma metodu (yöntemi)

- *Bazı tekrarlı bağıntıların çözümü yapılmadan* çözümünün nasıl olabileceği hakkında **tahmin yapılabilir.**
- **Daha sonra yerine konulur.**
- Yerine koyma (tahminler) metodu **bir sınırdır** ve **tahmini ispatlamak** için **<u>tümevarım yöntemi kullanılır.</u>**

1.Yerine koyma metodu (yöntemi)

- **En genel yöntem:**

1.Çözümün şeklini tahmin edin.

2. Tümevarım ile doğrulayın.

3. Sabitleri çözün.

1.Yerine koyma metodu (yöntemi)

Çözümün biçimin tahmin edilmesi:

Ne yazık ki özyinelemelerin çözümü için **doğru bir tahmin yapmanın genel bir yolu yoktur.**

Eğer özyineleme daha önceden gördüğünüz **özyineleme ile benzer ise çözüm tahmini yapmak anlamlıdır.**

$$T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$$

Eğer **n** değeri **oldukça büyük olduğunda** $\lfloor n/2 \rfloor$ ile $\lfloor n/2 \rfloor + 17$ arasındaki **fark çok büyük olmaz. Bir başka ifadeyle 17 sabiti göz ardı edilebilir.** Sonuç olarak yerine koyma yöntemi ile $T(n) = O(n \lg n)$ tahmini yapabiliriz.

1.Yerine koyma metodu (yöntemi)

Çözümün biçimin tahmin edilmesi:

İyi bir tahmin yapmanın başka bir yolu da özyineleme üzerinde *kabaca* ***üst*** ve ***alt*** ***sınır*** *tahmini* yapıp **belirsizlikleri indirgemektir.**

Örneğin $T(n) = 2T(\lfloor n/2 \rfloor) + n$ özyinelemesi için **alt sınır** $T(n) = \Omega(n)$ ve **üst sınır** $T(n) = O(n^2)$ olarak tahmin edebiliriz.

Daha sonra doğru çözüme ulaşıncaya kadar, **azar azar üst sınırı düşürerek ve alt sınırı yükselterek** $T(n) = \Theta(n \lg n)$ çözüme ulaşabiliriz.

1.Yerine koyma metodu (yöntemi)

Örnek: T(n)=T(n/2)+c, n≥2, ve T(1)=1.

- T(2)=1+**1**c, T(4)=1+**2**c, T(8)=1+**3**c, T(16)=1+**4**c, …
- $T(2^k)$=1+**k**c olur.

Burada **n= 2^k**, **T(n)=1+clogn** olur.

1.Yerine koyma metodu (alt sınır)

Örnek: **T(n)=3T(n/2)+cn**, $n \geq 2$, ve T(1)=1.

- **T(2)=3+2c**
- $T(4)=3(\mathbf{T(2)})+4c=3(\mathbf{3+2c})+4c=\mathbf{9+10c}=\mathbf{3^2}+[\underline{3^1 2^1 c}]+3^0 2^2 \mathbf{c}$
- $T(8)=\mathbf{27+38c}=3^3+[3^2 2^1 c+3^1 2^2 c]+3^0 2^3 c$
- T(16)=81+130c
- …
- $T(2^k)=3^k+[3^{k-1}2^1c+3^{k-2}2^2c+\ldots+3^1 2^{k-1}c]+3^0 2^k c$, **$2^k c$ parantezine alalım**
- $T(2^k)=3^k+\mathbf{2^k c}[(3/2)^{k-1}+(3/2)^{k-2}+\ldots+(3/2)]$, *serisinin genel denklemi*

$$f(n) = \sum_{0 \leq i \leq n} x^i = (x^{n+1}-1) / (x-1)$$

1.Yerine koyma metodu (yöntemi)

- $T(2^k)=3^k+2^k c[((3/2)^k-1)/((3/2)-1)]$, burada **$n=2^k$** ve **$k=\log n$**
- $T(n)=3^{\log n}+cn[((3^{\log n}/n-1)/(1/2)]$, burada $a^{\log_b x} = x^{\log_b a}$
- $T(n)=n^{\log 3}+2cn(n^{\log 3-1}-1) = n^{1,59}+2cn(n^{0,59}-1)$
- $T(n)=n^{1,59}(1+2c)-2cn$
- $T(n)\in O(n^{1,59})$ dır.

Yerine koyma metodu (üst sınırı tahmin ederek)

Örnek : $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$

Burada sezgisel tahminimiz : $T(n) = O(n)$

$T(n) \le cn$

Tahminimizi doğrulamaya çalışırsak :

$$\begin{aligned} T(n) &\le c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 \\ &= cn + 1\,, \end{aligned}$$

Burada 1 sabiti olduğu için tahminimiz $T(n) \le cn$ geçerli olmayacaktır.

Yeni tahmin : $T(n) = O(n^2)$

Yalnız burada **c** oldukça büyük değer seçilirse bu sabit göz ardı edilebilir. Bu sefer yeni tahmin: $T(n) \le cn - d,\ \ d \ge 0$

$$\begin{aligned} T(n) &\le (c\lfloor n/2 \rfloor - d) + (c\lceil n/2 \rceil - d) + 1 \\ &= cn - 2d + 1 \\ &\le cn - d\,, \end{aligned}$$

d sabiti çıkarılır. c değeri oldukça büyük ve d>0 olduğu sürece tahminimiz doğru olacaktır.

Örnek: **$T(n) = 4T(n/2) + n$**, ise

- [**$T(1) = \Theta(1)$** olduğunu varsayın.]
- **$O(n^3)$**'ü tahmin edin. (O ve Ω ayrı ayrı kanıtlayın.)
- **$k< n$** için **$T(k) \leq ck^3$** olduğunu varsayın.
- **$T(n) \leq cn^3$**'ü tümevarımla kanıtlayın.

Yerine koyma örneği

T(n) = <u>4T(n/2)</u> + n

$T(n) \le cn^3$ *Üst sınır*

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &\le 4c(n/2)^3 + n \\ &= (c/2)n^3 + n \\ &= cn^3 - \underline{((c/2)n^3 - n)} \leftarrow \textit{istenen –kalan} \\ &\le cn^3 \leftarrow \textit{istenen} \end{aligned}$$

ne zaman ki $(c/2)n^3 - n \ge 0$, örneğin,
eğer $\underline{c \ge 2}$ ve $\underline{n \ge 1}$.

kalan

- ***n* sabiti çıkarılsa acaba durum ne olurdur? Yine büyük mü olur yoksa küçük mü?**
- Örneğin ***c*** yerine 2 ve ***n*** yerine 1 seçilirse n^3-1 olacağından üst sınır sağlayacaktı.

Sıkı olmayan bir üst sınır

- Başlangıç koşullarını da ele almalı, yani, tümevarımı taban şıklarına (base cases) dayandırmalıyız.
- **Taban:** T(n) = Θ(1) *tüm* **n <** $\boldsymbol{n_0}$ için, ki $\boldsymbol{n_0}$ uygun bir sabittir.
- **1 ≤ n <** $\boldsymbol{n_0}$ için, elimizde ***"Θ(1)" ≤ cn3,*** olur; yeterince büyük bir **c** değeri seçersek.

- **Bu, sıkı bir sınır değildir !**

Yerine koyma örneği-Daha sıkı bir üst sınır

$T(n) = O(n^2)$ olduğunu kanıtlayacağız.

Varsayın ki $T(k) \le ck^2$, $k < n$ için olsun :

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\le 4c(n/2)^2 + n \\
&= cn^2 + n \\
&= O(n^2)
\end{aligned}
$$

Yerine koyma örneği-Daha sıkı bir üst sınır

$T(n) = O(n^2)$ olduğunu kanıtlayacağız.

Varsayın ki $T(k) \le ck^2$; $k < n$: için

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\le 4c(n/2)^2 + n \\
&= cn^2 + n \\
&= \cancel{O(n^2)} \quad \textit{Yanlış!}\ \text{I.H.(tümevarım hipotezini) kanıtlamalıyız.} \\
&= cn^2 - (-n) \quad [\ \text{istenen} - \text{kalan}\ ] \\
&\le cn^2 \quad \text{seçeneksiz durum } c > 0.\ \text{Kaybettik!}
\end{aligned}
$$

-n >= 0 sağlanmaz (n değeri negatif olamaz)

Yerine koyma örneği-Daha sıkı bir üst sınır

Fikir: Varsayım hipotezini güçlendirin.

- Düşük-düzeyli bir terimi ***çıkartın***.

Varsayım hipotezi: $T(k) \le c_1 k^2 - c_2 k$; $k < n$ için.

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &= 4(c_1(n/2)^2 - c_2(n/2)) + n \\ &= c_1 n^2 - 2c_2 n + n \\ &= c_1 n^2 - c_2 n - (c_2 n - n) \\ &\le c_1 n^2 - c_2 n \text{ eğer } c_2 \ge 1. \end{aligned}$$

c_1'i başlangıç koşullarını karşılayacak kadar büyük seçin.

2. Özyineleme ağacı (Recursion tree) metodu

- **Bu metot genelde hep çalışır.**
- **Çok sıkı kuralları olan bir yöntem değildir.**
- *Uygularken çok dikkatli olunması gerekir.* Yanlış cevaplar üretilebilir.
- Özyineleme **cevabın ne olduğunu bulmak** ve **sonrada bu cevabın doğru olup olmadığını kanıtlamak için yerine koymak metodunu bulmaktır.**
- Öte yandan özyineleme-ağacı metodu "öngörü" olgusunu geliştirir.

2.Özyineleme ağacı (Recursion tree) metodu - örnek

- $T(n) = T(n/4) + T(n/2) + n^2$: çözün

$$n^2$$

$$T(n/4) \qquad T(n/2)$$

Alt dallardaki maliyet **genişletilerek** bulunur.

$\mathbf{n}\ \boldsymbol{yerine}\ \frac{\boldsymbol{n}}{\mathbf{4}}\ \boldsymbol{konulur}.$

$$n^2$$

$$(n/4)^2 \qquad (n/2)^2$$

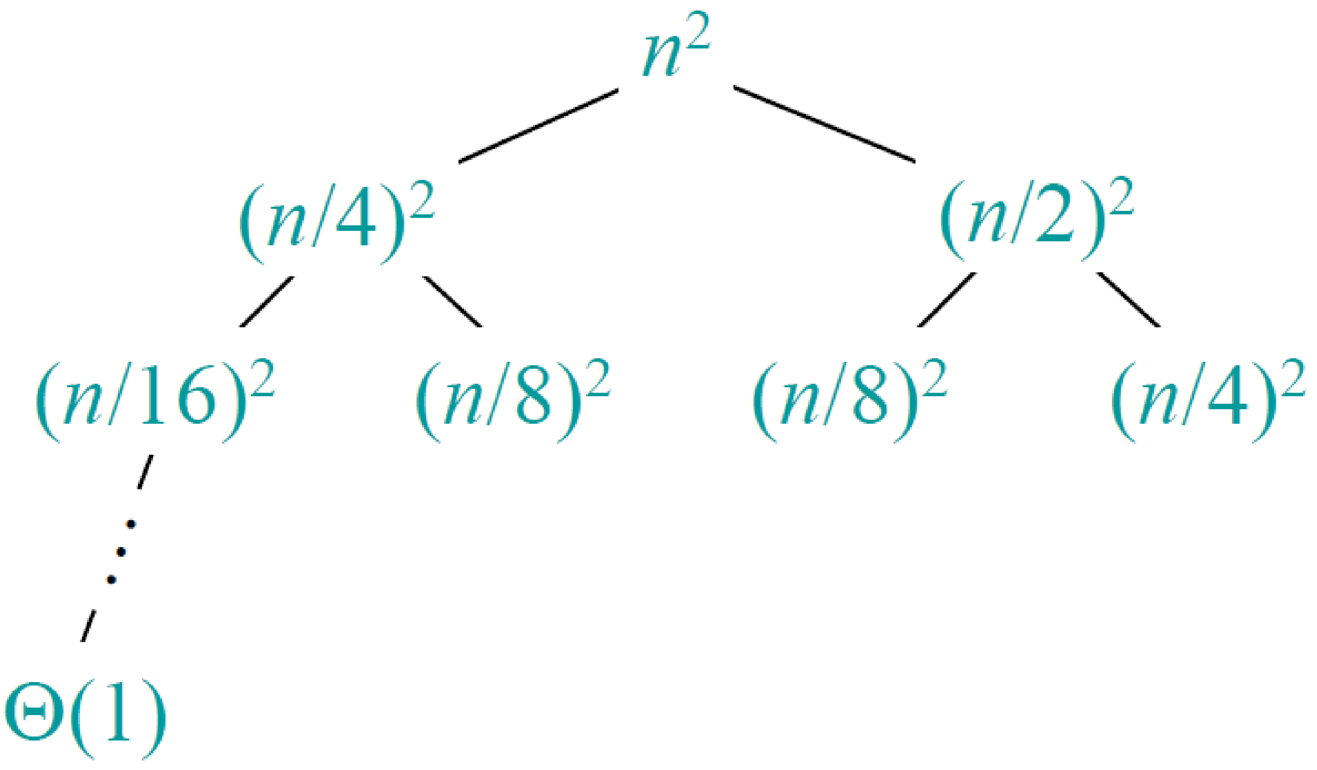
$$T(n/16) \quad T(n/8) \quad T(n/8) \quad T(n/4)$$

* Çözüm için **<u>üst sınır bulunmaya odaklanılır. Genellikle küçük bir miktar "yanlışlığa" tolerans gösterebilir.</u>**

- Burada n sayısı 2'nin tam kuvveti olup tüm alt problemler tamsayı boyutludur.
- En üst seviyedeki *maliyet* ile *alt dallardaki maliyet* genişletilerek bulunulur.

2. Özyineleme ağacı (Recursion tree) metodu-örnek

$$T(n) = T(n/4) + T(n/2) + n^2:$$



2. Özyineleme ağacı (Recursion tree) metodu -örnek

$T(n) = T(n/4) + T(n/2) + n^2$:

$$n^2 \quad\cdots\cdots\quad n^2$$

$$(n/4)^2 \qquad (n/2)^2 \quad\cdots\cdots\quad \frac{5}{16}n^2$$

$$(n/16)^2 \qquad (n/8)^2 \qquad (n/8)^2 \qquad (n/4)^2 \quad\cdots\cdots\quad \frac{25}{256}n^2$$

$$\vdots \qquad\qquad \vdots$$

$$\Theta(1)$$

$$\text{Total} = n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \cdots\right)$$

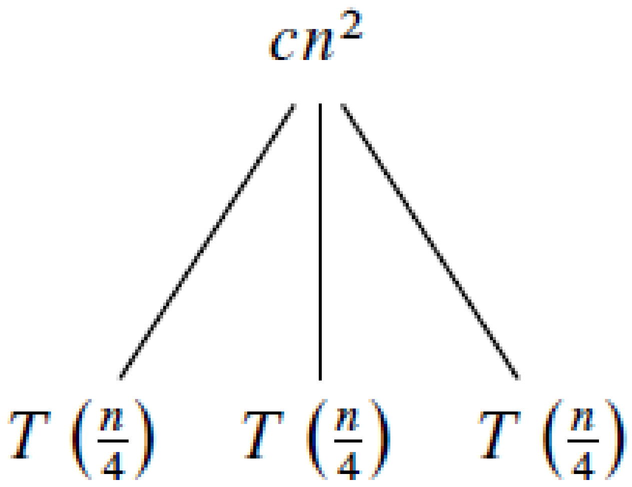
$$= \Theta(n^2)$$

Geometrik seri

2. Özyineleme ağacı (Recursion tree) metodu- örnek

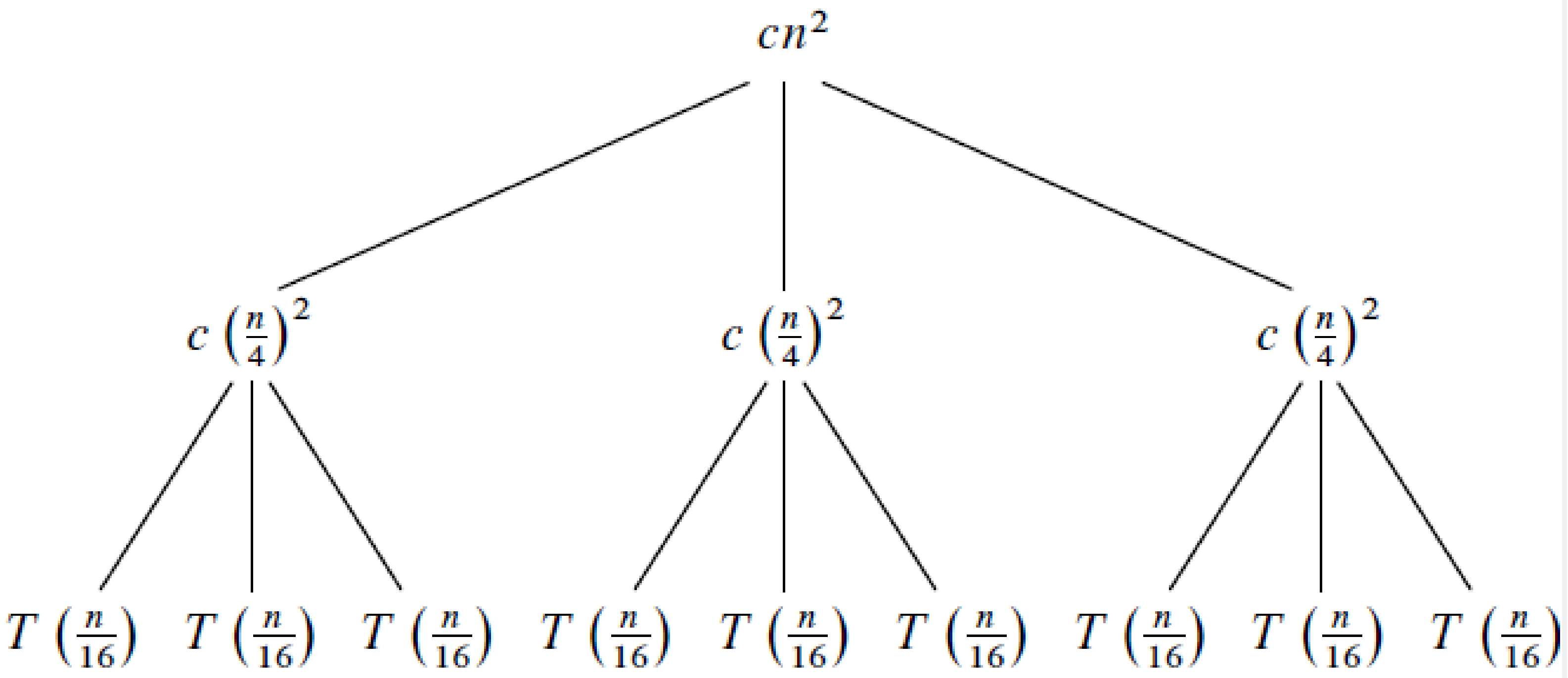
$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$$

$$T(n) = 3T(n/4) + cn^2$$



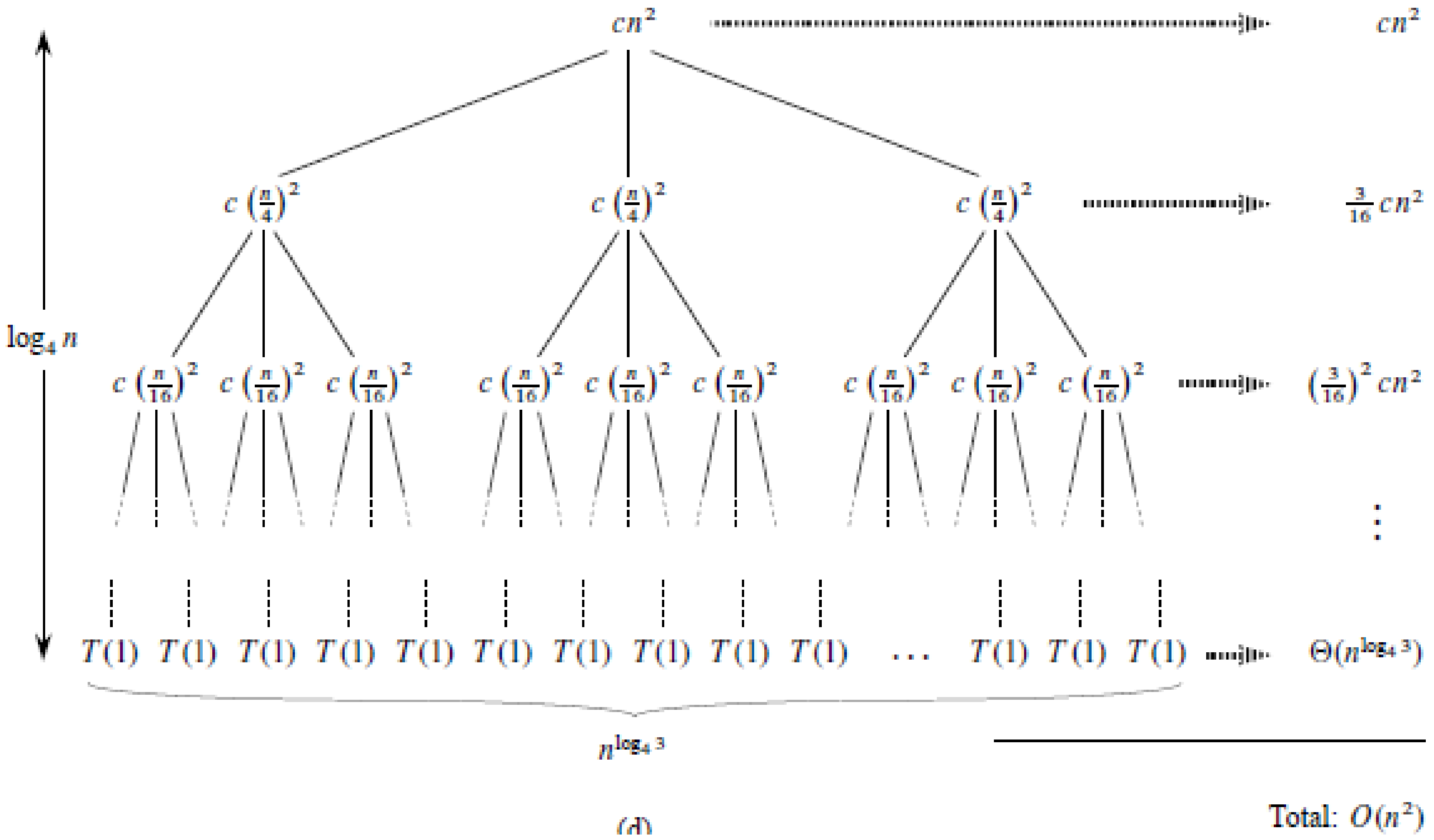
2. Özyineleme ağacı (Recursion tree) metodu- örnek

$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$$



2. Özyineleme ağacı (Recursion tree) metodu- örnek

$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$$



Burada n, dördün katı olduğu farz edilmiştir. Dolayısıyla 1'e ulaşılıncaya kadar 4'e bölünür.
Bu nedenle de ağacın yüksekliği $\log_4 n$ olur.

3. Ana Metod (The Master Method)

- **Ana method** aşağıda belirtilen yapıdaki yinelemelere uygulanır:

  - $T(n) = a\,T(n/b) + f(n)$,

burada $\boldsymbol{a \geq 1}$, $\boldsymbol{b > 1}$, ve f **asimptotik olarak pozitiftir.**

Not: Asimptotik pozitif, *n değeri yeterince büyük* $(n \geq n_0)$ *olduğunda f değeri pozitiftir.*

- *Özyineleme ağacı ile çözümünden farklı olarak* burada **her problem aynı boyutta olması gerekir.**

3. Ana Metod (The Master Method)

$T(n) = a\,T(n/b) + f(n)$

- ❖ ***T(n)*** *bir algoritmanın çalışma süresidir.*
- ❖ ***n/b*** boyutunda ***a*** *tane alt problem* **özyineleme olarak çözülür** ve her biri ***T(n/b)***süresindedir.
- ❖ ***f(n)*** **problemin bölünmesi** ve **sonuçların birleştirilmesi için** geçen süredir.

- **Örnek:** Merge-sort için $T(n)=2T(n/2)+\Theta(n)$ yazılabilir.

3. Ana Metod (The Master Method)

- $T(n) = a\,T(n/b) + f(n)$

Burada **n/b** ya da $\lfloor \boldsymbol{n/b} \rfloor$ ya da $\lceil \boldsymbol{n/b} \rceil$ anlamında yorumlarız.

O zaman ***T(n)*** **aşağıdaki asimptotik sınırlara sahiptir**.

1. Eğer bazı $\boldsymbol{\varepsilon > 0}$ sabiti için $f(n)=O(n^{\log_b^{a-\varepsilon}})$ ise $T(n)= \theta(n^{\log_b^a})$ olur.
2. Eğer $f(n)= O(n^{\log_b^a})$ *ise o zaman* $T(n)= \theta(n^{\log_b^a} \lg n)$ *olur.*
3. *Eğer bazı* $\boldsymbol{\varepsilon > 0}$ sabiti için $\mathbf{f(n)}=\Omega(n^{\log_b^{a+\varepsilon}})$ ve ***bazı c<1 sabiti*** ve de ***yeterince büyük n*** için $a*f(n/b) \leq c*f(n)$ ise o zaman $T(n)= \theta(f(n))$ olur.

- *Özyineleme ağacında alt dallara inildikçe f küçülmelidir.*
- *Yani toplamdaki artış miktarı azalmalıdır.*

3. Ana Metod (The Master Method) -Örnek

- $T(n) = 9T(n/3) + n$
- $a = 9,\ b = 3$ ve $\underline{f(n) = \boldsymbol{n}}$ $\Rightarrow n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$

$\varepsilon = 1$ için ***f(n)*** $O(n^{\log_3 9-\varepsilon})$ olduğundan ana teoremin **1. durumu** uygulayabiliriz ve $\boldsymbol{T(n)=\Theta(n^2)}$ sonucunu elde ederiz.

3. Ana Metod (The Master Method) -Örnek

- $T(n) = T(2n/3) + 1$

❑ a=1 , b=3/2 ve **f(n)=1** ve $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$ olur.

❑ $\Theta(n^{\log_b a}) = \Theta(1)$ ve de $f(n)$ 'de 1'e eşit olduğundan

2. durum uygulanır ve böylece özyinelemenin çözümü $f(n)= \Theta(\lg n)$ olur.

3. Ana Metod (The Master Method) -Örnek

- T(n)=3 T(n/4)+n lgn

- a=3, b=4 ve f(n)=n lgn ve $n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$
- $\varepsilon \approx 0.2$ olduğunda f(n)=$\Omega(n^{\log_4 3+\varepsilon})$ *için düzenlilik koşulunu sağladığını gösterirsek* ***3. durum uygulanır.***

a*f(n/b)≤ c*f(n) durumu sağlayan c değeri var mı?

- ***Yeterince büyük n ve c=3/4 için***
- *af(n/b) = 3(n/4) lg(n/4) ≤ (3/4) n lg n=cf(n) olur.*
- ***Sonuç olarak, durum 3 uygulanabilir. Bu durumda özyineleme sonucu*** $T(n) = \theta(n\ lgn)$ *olur*.

3. Ana Metod (The Master Method) -Örnek

- **$T(n)=2\,T(n/2)+n\lg n$**

a=2, b=2 ve f(n)=n lgn ve $n^{\log_b a} = n^{\log_2 2} = O(n)$

f(n)=n lgn ***asimptotik olarak*** $n^{\log_b a + \varepsilon} = n$ **değerinden** **büyüktür.**

Bu sonuca bakarak durum 3 'ün uygulanması düşünülebilir.

Fakat her zaman bu ifade polinomiyal olarak büyük değildir.

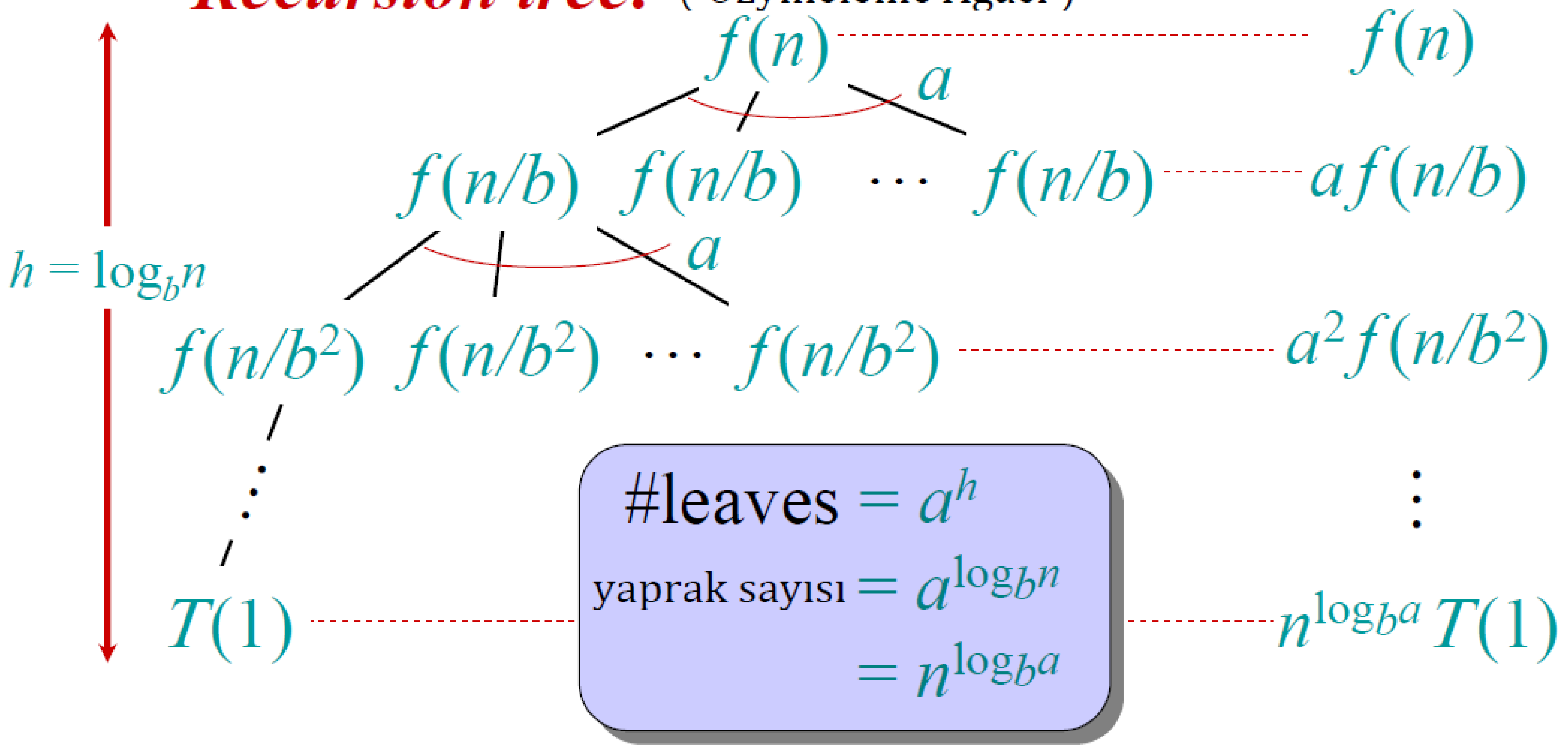
Herhangi bir pozitif ε ***değeri için*** $f(n)/n^{\log_b a} = n\lg n/n = \lg n$ oranı asimptotik olarak n^{ε} *'den azdır.*

Sonuç olarak bu problem durum 2 ve durum 3 arasında boşluğa düşer.

Master teoremdeki düşünce

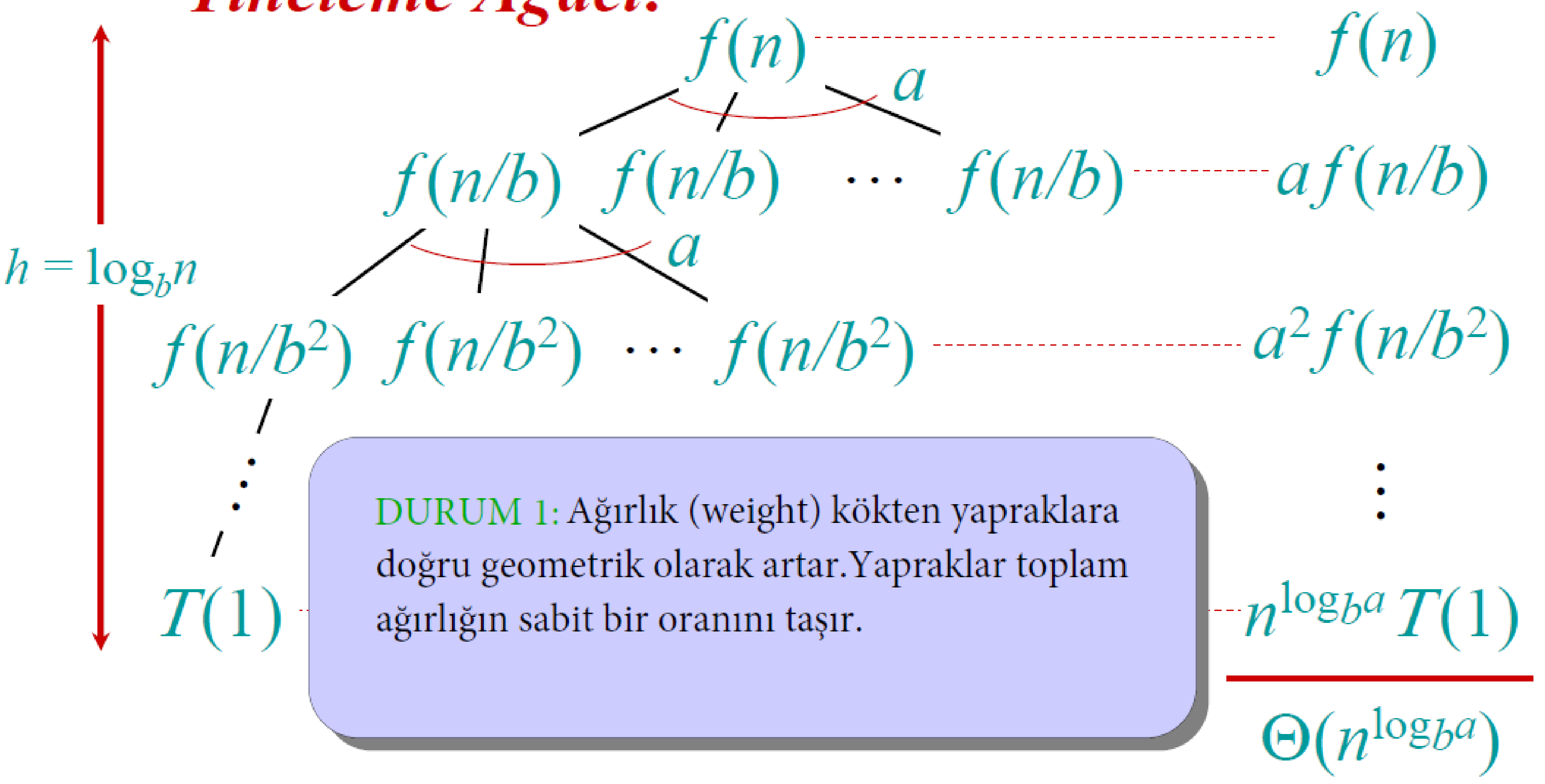
a T(n/b) + f (n)

Recursion tree: (Özyineleme Ağacı)



Master teoremdeki düşünce

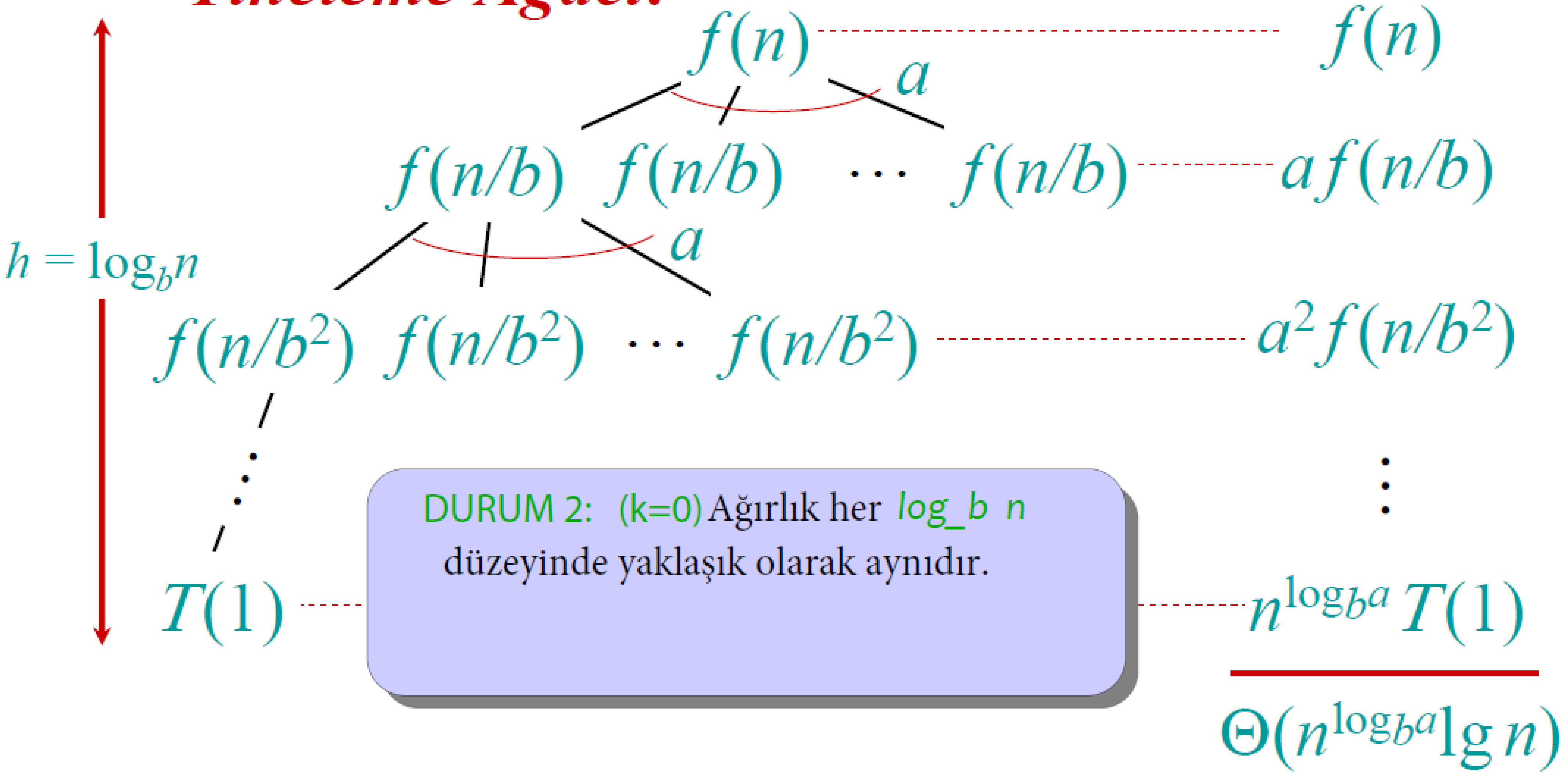
Yineleme Ağacı:



DURUM 1: Ağırlık (weight) kökten yapraklara doğru geometrik olarak artar. Yapraklar toplam ağırlığın sabit bir oranını taşır.

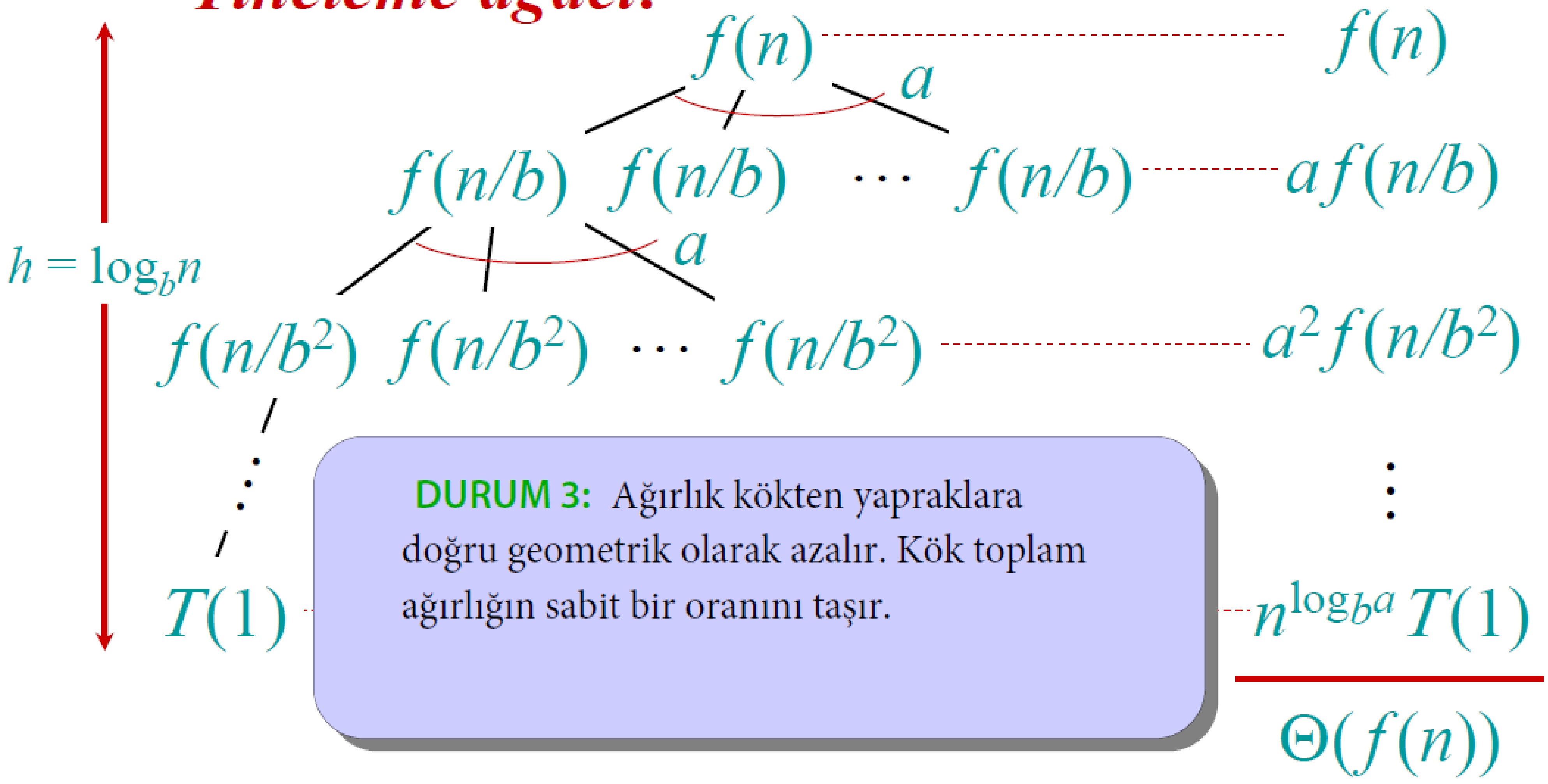
Master teoremdeki düşünce

Yineleme Ağacı:



Master teoremdeki düşünce

Yineleme ağacı:



DURUM 3: Ağırlık kökten yapraklara doğru geometrik olarak azalır. Kök toplam ağırlığın sabit bir oranını taşır.

Böl ve hükmet tasarım paradigması

1. Problemi (anlık durumu) alt problemlere **böl**.
2. Alt problemleri özyinelemeli olarak çözüp, onları **fethet**.
3. Alt problem çözümlerini **birleştir**.

Örnek: Birleştirme sıralaması (Merge Sort)

1. ***Bölmek:*** Kolay.
2. ***Hükmetmek:*** 2 altdizilimi özyinelemeli sıralama.
3. ***Birleştirmek:*** Doğrusal-zamanda birleştirme.

$$T(n) = 2\,T(n/2) + \Theta(n)$$

altproblem sayısı

altproblem boyutu

bölme ve birleştirme işi

İkili arama (Binary Search)

- Sıralı dizilimin bir elemanını bulma:

1. ***Böl:*** Orta elemanı belirle.
2. ***Hükmet:*** 1 alt dizilimde özyinelemeli arama yap.
3. ***Birleştir:*** Kolay.

İkili arama

Sıralı dizilimin bir elemanını bulma:

1. Böl: Orta elemanı belirle.

2. Hükmet: 1altdizilimde özyinelemeli arama yap.

3.Birleştir: Kolay.

Örnek: 9' u bul.

3	5	7	8	9	12	15

İkili arama

Sıralı dizilimin bir elemanını bulma:

1. ***Böl:*** Orta elemanı belirle.
2. ***Hükmet:*** 1altdizilimde özyinelemeli arama yap.
3. ***Birleştir:*** Kolay.

Örnek: 9'u bul.

3	5	7	8	9	12	15

İkili arama

3	5	7	8	9	12	15
3	5	7	8	9	12	15
3	5	7	8	9	12	15
3	5	7	8	9	12	15

İkili arama için yineleme

$$T(n) = 1\,T(n/2) + \Theta(1)$$

altproblem sayısı

altproblem boyutu

bölme ve birleştime işi

İkili arama için yineleme (Master Metot)

$$T(n) = 1\,T(n/2) + \Theta(1)$$

altproblem sayısı

altproblem boyutu

bölme ve birleştime işi

f(n)=1

$$n^{\log_b a} = n^{\log_2 1} = n^0 = 1 \Rightarrow \text{DURUM 2 } (k = 0)$$

$$\Rightarrow T(n) = \Theta(\lg n)\,.$$

Bir sayının üstellenmesi

Problem: a^n 'yi $n \in \mathbb{N}$ iken hesaplama.

Saf (Naive) algorithm: $\Theta(n)$.

Böl-ve-fethet algoritması:

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2} & n \text{ çift sayıysa;} \\ a^{(n-1)/2} \cdot a^{(n-1)/2} \cdot a & n \text{ tek sayıysa.} \end{cases}$$

* Birleştirme işlemi, 1 veya 2 çarpma yapmaktır.
Bu da sabit bir zamandır.

$$T(n) = T(n/2) + \Theta(1) \Rightarrow T(n) = \Theta(\lg n)\ .$$

Matrislerde çarpma
Standart algoritma

for $i \leftarrow 1$ **to** n *(i 1'den n'ye kadar)*

do for $j \leftarrow 1$ **to** n *(j 1'den n'ye kadar)*

do $c_{ij} \leftarrow 0$

for $k \leftarrow 1$ **to** n

do $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$

Koşma süresi $= \Theta(n^3)$

Böl-ve-fethet algoritması

FİKİR:

$n{\times}n$ matris = $(n/2){\times}(n/2)$ altmatrisin $2{\times}2$ matrisi:

$$\left[\begin{array}{c|c} r & s \\ \hline t & u \end{array}\right] = \left[\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right] \cdot \left[\begin{array}{c|c} e & f \\ \hline g & h \end{array}\right]$$

$$C \quad = \quad A \quad \cdot \quad B$$

$$\left.\begin{aligned} r &= ae + bg \\ s &= af + bh \\ t &= ce + dh \\ u &= cf + dg \end{aligned}\right\}$$

recursive (özyinelemeli)
8 çarpma $(n/2){\times}(n/2)$ altmatriste,
4 toplama $(n/2){\times}(n/2)$ altmatriste.

- Elimizde 8 tane (n/2) × (n/2) boyutlu küçük matris oldu. **Sonucu bulmak için özyinelemeli çarpım yapıyoruz.**
- (n/2) × (n/2) boyutlu küçük matrisleri topluyoruz.
- **Herhangi iki matrisi toplamak için gerekli süre n^2 dir.**
- **Not: Matrisler 2'nin katı değilse 0 ile tamamlanır.**

Böl ve fethet algoritması

$$T(n) = 8\,T(n/2) + \Theta(n^2)$$

altmatris sayısı

altmatris boyutu

altmatrisleri toplama işi

$$n^{\log_b a} = n^{\log_2 8} = n^3 \;\Rightarrow\; \text{DURUM 1} \;\Rightarrow\; T(n) = \Theta(n^3)$$

Master teoremine göre

Sıradan algoritmadan daha iyi değil.

Strassen Yöntemi

$$T(n) = 8\,T(n/2) + \Theta(n^2)$$

- Toplama işleminde maliyet değişmeyecek *fakat çarpma işleminde özyineleme sayısını azaltabilme* fikrine dayanmaktadır.
- A ve B 'yi **(n/2) × (n/2) alt matrislere** bölünmüş + ve - kullanarak **çarpılabilecek terimler** oluşturulmuştur.

```
                        [ 2  3  4  0 ]
[ 2  3  4 ]     ===>    [ 1  5  7  0 ]
[ 1  5  7 ]     ===>    [ 0  0  0  0 ]
                        [ 0  0  0  0 ]
```

Strassen Yöntemi

- **2×2 matrisleri yalnız 7 özyinelemeli çarpmayla çözülmesi:**

$$\begin{bmatrix} r & s \\ t & u \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

- $P_1 = a \cdot (f - h)$
- $P_2 = (a + b) \cdot h$
- $P_3 = (c + d) \cdot e$
- $P_4 = d \cdot (g - e)$
- $P_5 = (a + d) \cdot (e + h)$
- $P_6 = (b - d) \cdot (g + h)$
- $P_7 = (a - c) \cdot (e + f)$

$$r = P_5 + P_4 - P_2 + P_6$$
$$s = P_1 + P_2$$
$$t = P_3 + P_4$$
$$u = P_5 + P_1 - P_3 - P_7$$

Toplama işlemi sıra bağımsızdır. Çarpma işleminde sıra bağımsızlık yoktur. **7 çarpma**, **18** toplama /çıkarma işlemi var.

Strassen Yöntemi

2×2 matrisleri yalnız 7 özyinelemeli çarpmayla çözülmesi:

$$P_1 = a \cdot (f - h)$$
$$P_2 = (a + b) \cdot h$$
$$P_3 = (c + d) \cdot e$$
$$P_4 = d \cdot (g - e)$$
$$P_5 = (a + d) \cdot (e + h)$$
$$P_6 = (b - d) \cdot (g + h)$$
$$P_7 = (a - c) \cdot (e + f)$$

$$\begin{aligned} r &= P_5 + P_4 - P_2 + P_6 \\ &= (a + d)(e + h) \\ &\quad + d(g - e) - (a + b)h \\ &\quad + (b - d)(g + h) \\ &= ae + ah + de + dh \\ &\quad + dg - de - ah - bh \\ &\quad + bg + bh - dg - dh \\ &= ae + bg \end{aligned}$$

$$\left[\begin{array}{c|c} r & s \\ \hline t & u \end{array}\right] = \left[\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right] \cdot \left[\begin{array}{c|c} e & f \\ \hline g & h \end{array}\right]$$

Strassen Algoritması

1. ***Böl:*** A ve B'yi (n/2)×(n/2) altmatrislere böl. + ve – kullanarak çarpılabilecek terimler oluştur.
2. ***Fethet:*** $(n/2)\times(n/2)$ altmatrislerde özyinelemeli 7 çarpma yap.
3. ***Birleştir:*** +ve – kullanarak $(n/2)\times(n/2)$ altmatrislerde C 'yi oluştur.

$$T(n) = 7\,T(n/2) + \Theta(n^2)$$

Strassen Algoritması

$$T(n) = 7\,T(n/2) + \Theta(n^2)$$

$$n^{\log_b a} = n^{\log_2 7} \approx n^{2.81} \;\Rightarrow\; \text{CASE 1} \;\Rightarrow\; T(n) = \Theta(n^{\lg 7})$$

2.81 değeri 3' den çok küçük görünmeyebilir ama, fark üstelde olduğu için, yürütüm süresine etkisi kayda değerdir. Aslında, $n \geq 32$ değerlerinde Strassen'in algoritması günün makinelerinde normal algoritmadan daha hızlı çalışır.

Bugünün en iyi değeri (teorik merak açısından): $\Theta(n^{2.376\cdots})$.

UYGULAMALAR

- **İkili arama algoritmasını özyineleme şekilde çözünüz.**
- **Strassen Algoritması uygulayarak 2 matrisin çarpımını yapınız.**

Kaynakça

- Algoritmalar : Prof. Dr. Vasif NABİYEV, Seçkin Yayıncılık
- Algoritmalara Giriş : Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, Palme YAYINCILIK
- Algoritmalar : Robert Sedgewick , Kevin Wayne, Nobel Akademik Yayıncılık
- M.Ali Akcayol, Gazi Üniversitesi, Algoritma Analizi Ders Notları
- Doç. Dr. Erkan TANYILDIZI, Fırat Üniversitesi, Algoritma Analizi Ders Notları