CENG 114 BİLGİSAYAR BİLİMLERİ İÇİN AYRIK YAPILAR

Prof. Dr. Tufan TURACI
tturaci@pau.edu.tr

- Pamukkale Üniversitesi
- Mühendislik Fakültesi
- Bilgisayar Mühendisliği Bölümü

- Hafta 6

Ders İçereği

- **Boolean Fonksiyonlar (Soru Çözümü)**
- **Algoritma Analizi**

  --- Yürütme Zamanı

  --- Karmaşıklık

Algoritma ve Algoritma Analizi

Bir algoritma, bir problemi çözmek ya da bir işlevi hesaplamak için izlenecek sonlu, açıkça belirtilen talimat dizisidir.

Bir algoritma genel olarak

- *Bir (birkaç) girdi alır.*
- *Sınırlı bir süre içerisinde komutlar bir çıktı üretmektedir.*

Etkili bir talimat, temelde kalem ve kağıt kullanarak gerçekleştirmenin mümkün olduğu kadar basit bir işlemdir.

Algoritmaları İfade Etmek

Algoritmalar şu şekilde gösterilebilir:

- **doğal diller**
  - *ayrıntılı ve belirsizdir.*
  - *nadiren karmaşık veya teknik algoritmalar için kullanılır.*
- **Pseudocode(sözde kod), akış diyagramları:**
  - --- algoritmaları ifade etmek için yapısal yöntemlerdir.
  - --- doğal dilde ifadelerde belirsizliklerden kaçınır.
  - --- belirli bir uygulama dilinden bağımsızdır.
- **Programlama dilleri:**
  - --- algoritmaları bir bilgisayar tarafından yürütülebilecek biçimde ifade etmeyi amaçlar.
  - --- algoritmaları belgelemek için kullanılabilir.

Örnek:

Problem: Sıralanmamış bir listede en büyük elemanı bulmak

Fikir: Her elemana bir kere bakmak.

Doğal Dil:

- Listedeki ilk elemanın en büyük olduğunu varsay.
- Listenin sonuna kadar daha büyük bir sayı var mı diye arat.
- Liste tarama işlemi bittiğinde en son not edilen en büyük elemandır.

Örnek:

Sözde Kod:

Algorithm LargestNumber

Input: A non-empty list of numbers L.

Output: The *largest* number in the list L.

$largest \leftarrow L_0$

for each *item* **in** the list $L_{i \geq 1}$, **do**

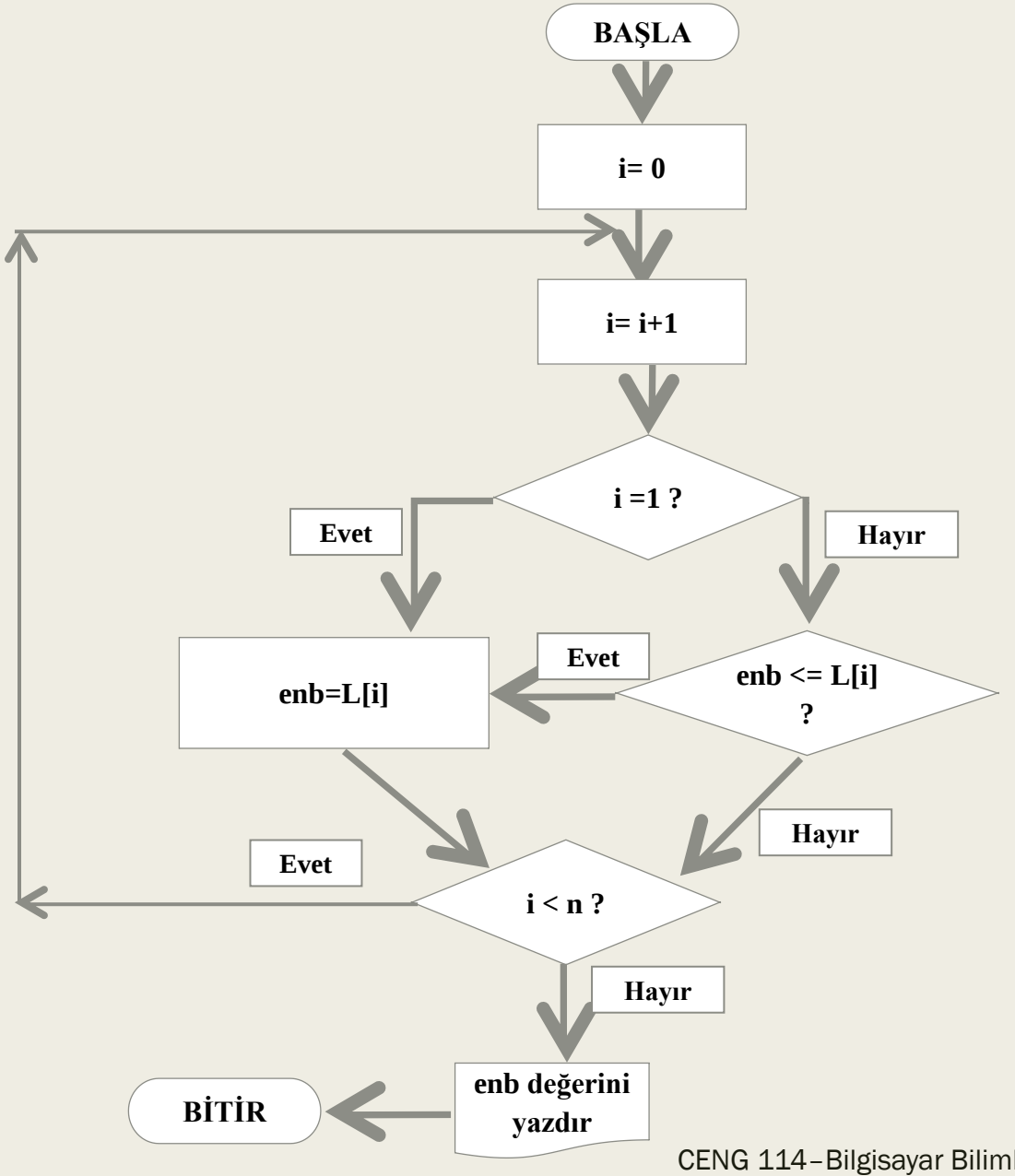
if the *item* > *largest*, **then**

$largest \leftarrow$ the *item*

return *largest*

Örnek:

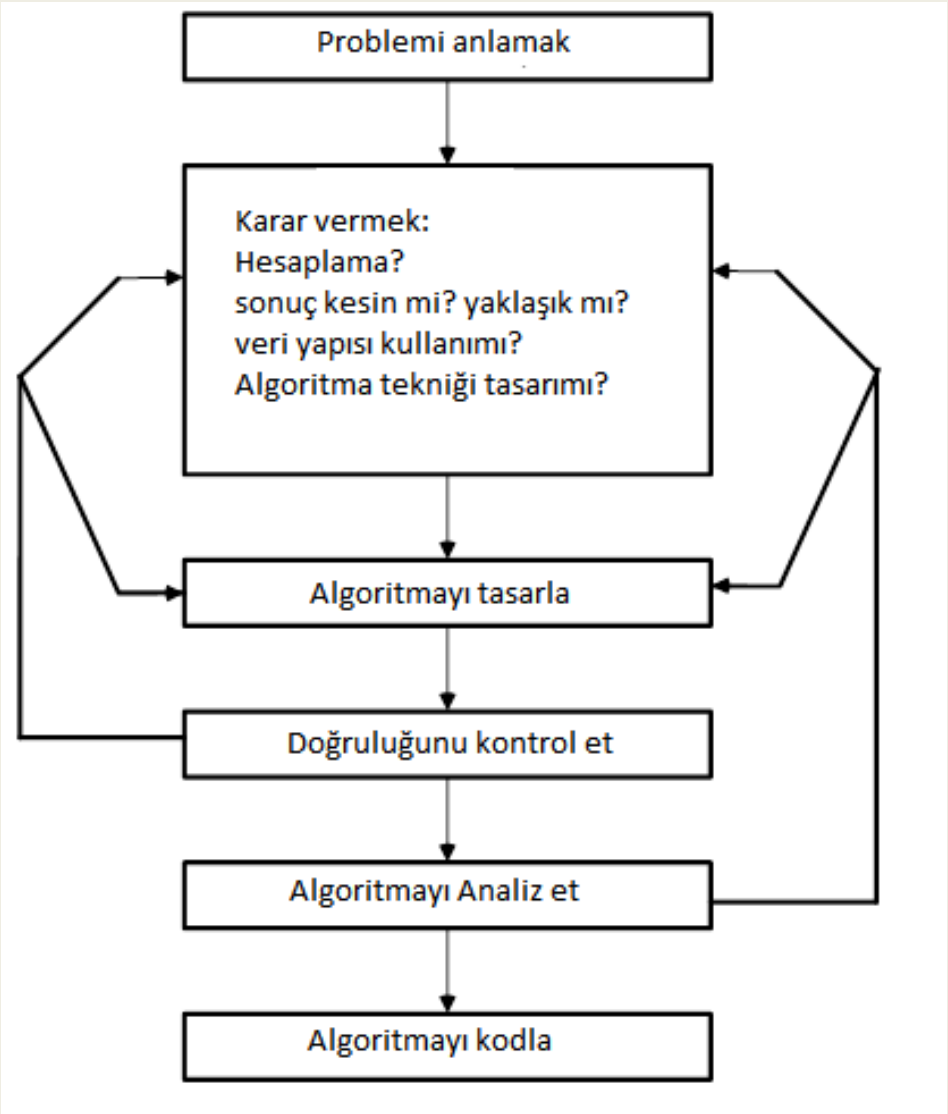
Akış Diyagramı



Algoritmanın Özellikleri

- **Effectiveness (Etkinlik)**
  - *Talimatlar basit olmalı.*
  - *kalem ve kağıtla yapılabilir.*
- **Definiteness (Kesinlik)**
  - *Talimatlar net*
  - *Anlamı tek olmalıdır.*
- **Correctness (Doğruluk)**
  - *Algoritma doğru cevabı verir (Olası tüm durumlar için)*
- **Finiteness (Sonluluk)**
  - *Algoritma makul sürede durmalı ve bir çıktı üretmelidir.*

Algoritma Tasarım Süreci



Algoritmaların Analizi

- Bir algoritmanın complexity'sini (karmaşıklığı) çalışma
  - *Algoritma ne kadar iyi?*
  - *Diğer algoritmalarla karşılaştırma işlemi nasıl yapılacak?*
  - *En iyi yazılabilecek algoritma bu mudur?*

- Karmaşıklık
  - *Alan Karmaşıklığı*
    - Bit sayısı
    - Eleman sayısı
  - *Zaman Karmaşıklığı*
    - Toplamda çalıştırılacak işlem sayısı
      - *Modele göre değişir*
      - *RAM*

Algoritmaların Run-Time (Çalışma Zamanı) Analizi

- Algoritma karmaşıklığı, problemin boyutunu gösteren parametre n' nin bir fonksiyonu olarak hesaplanabilmektedir.
- Zaman karmaşıklığı, T (n), algoritmanın en önemli işlemi olan - temel işlem olarak adlandırılan – işlemin çalıştırılma sayısı olarak hesaplanabilir.
- Space (Alan) karmaşıklığı, S (n), genellikle algoritmanın yürütülmesi sırasında kullanılan bellek alanının büyüklüğü olarak hesaplanır.

Tablo Metodu

- *Tablo Metodu, bir algoritmanın karmaşıklığını hesaplamak için kullanılır*

Örnek: Bir dizinin elemanlarını toplama

	İşlem sayısı
`top = 0`	1
`for (i=0; i<n; i++){`	1+n+1+n
`top = top + a[i]}`	n
`print top`	1
	3n+4

Kaç işlem yapılır? T(n)=3n+4 (Yürütme zamanı)

Tablo Metodu

- Örnek:

Matris Toplama

a, b, c 'nin mxn boyutunda matrisler olduğunu varsayalım.

	işlem	toplam
`for (i=0; i<m; i++){`	1+m+1+m	2m+2
`  for (j=0; j<n; j++){`	m(1+n+1+n)	2mn+2m
c[i,j] = a[i,j] + b[i,j] } }	mn	mn
		3mn+4m+2

Eğer m=n ise $T(n) = 3n^2 + 4n + 2$ (Yürütme zamanı)

Asimptotik Notasyon Ve Temel Verimlilik Sınıfları

(Büyüme Sırası) Order of growth

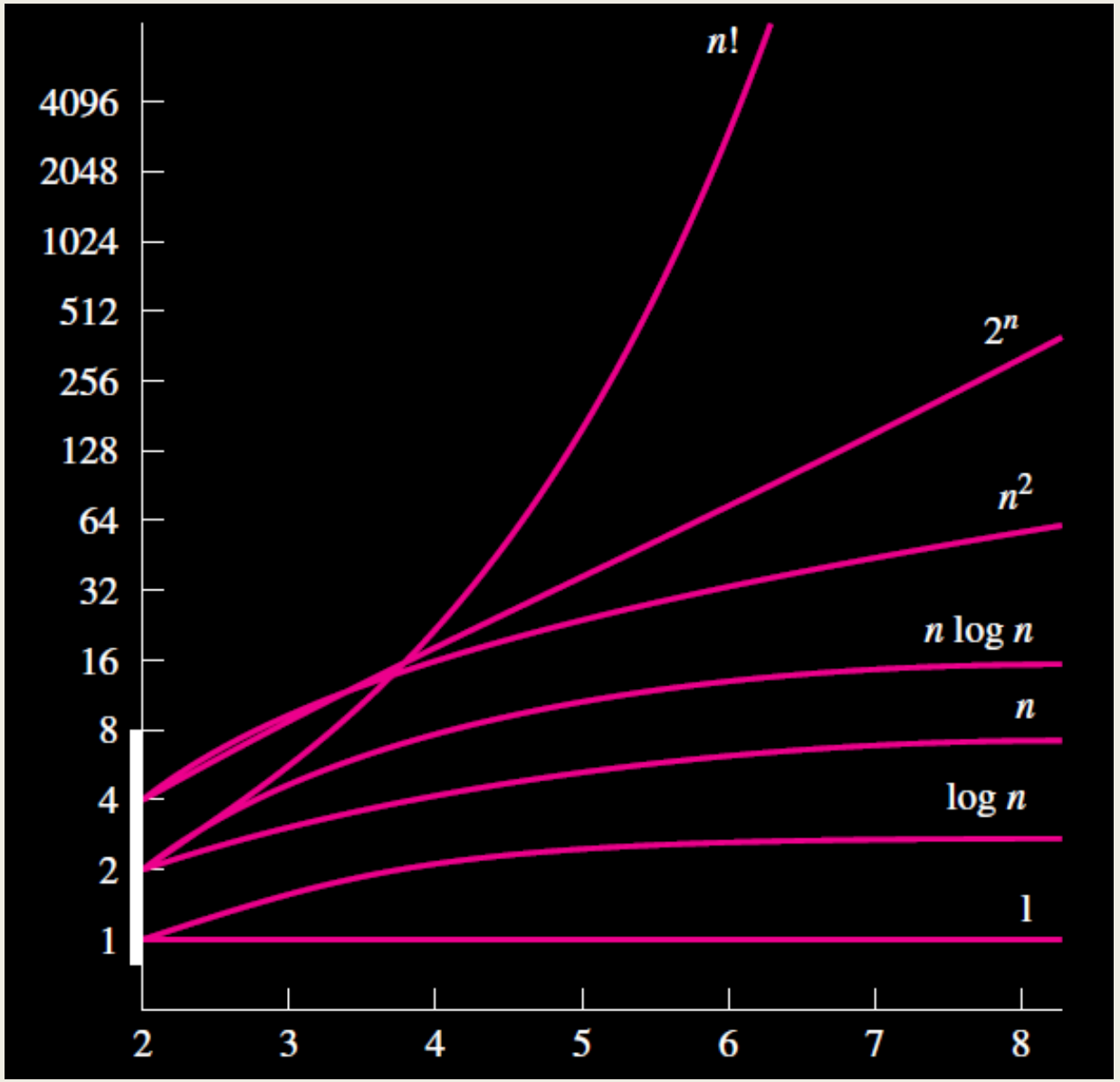
- En önemlisi : $n \to \infty$'a giderken algoritmanın performansı hangi sınırlarda bunu anlayabilmektir.

- Örnek:
  - *İki katı kadar hızlı bir bilgisayarda algoritma ne kadar hızlanıyor?*
  - *Girdi boyutu iki katına çıktığında algoritma ne kadar yavaşlıyor?*

$n \to \infty$ giderken bazı önemli fonksiyonların değerleri

Tablo: Algoritma analizi için bazı önemli fonksiyonların değerleri

n	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n	$n!$
10	3.3	10^1	$3.3 \cdot 10^1$	10^2	10^3	10^3	$3.6 \cdot 10^6$
10^2	6.6	10^2	$6.6 \cdot 10^2$	10^4	10^6	$1.3 \cdot 10^{30}$	$9.3 \cdot 10^{157}$
10^3	10	10^3	$1.0 \cdot 10^4$	10^6	10^9		
10^4	13	10^4	$1.3 \cdot 10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7 \cdot 10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0 \cdot 10^7$	10^{12}	10^{18}		


n!
4096
2048
1024
512
256
128
64
32
16
8
4
2
1
2^n
n^2
n log n
n
log n
1
2
3
4
5
6
7
8

Örnek:

- 2^100'ü hesaplamak saniyede 10^{12} işlem yapan bir bilgisayar için $4x10^{10}$ yıl alacaktır.
- Bu, 100! Değerini hesaplamak için gereken süreden kısadır. Fakat, 100!i hesaplamak ise dünya gezegeninin tahmini yaşından 4,5 milyar ($4.5 .10^{9}$) yıl daha uzundur.
- 2^{n} ve n! fonksiyonlarının büyüme sıraları arasında muazzam bir fark vardır.

Asimptotik Büyüme Dereceleri

Asimptotik Analiz

$n \to \infty$ iken $T(n)$'nin büyümesi nedir?

Asimptotik Notasyon

O — Büyük O

Ω — Büyük omega

Θ — Teta

(Big O) (Big Omega) (Theta)

O Notasyonu (Üst Sınır)

Ω Notasyonu (Alt Sınır)

Θ Notasyonu (Sıkı Sınır, Ortalama durum)

Kısaltmalar ve Anlamları

O — $\leq$

Ω — $\geq$

Θ — $=$

$\Theta(n^2)$ algoritması aynı problemi çözen bir $\Theta(n^3)$ algoritmasından daha hızlıdır.

Asimptotik Büyüme Dereceleri

Fonksiyonların büyüme hızlarını karşılaştırmak için kullanılan, sabit çarpanları ve küçük girdi boyutlarını yok sayan bir yöntem.

- O($g(n)$): $g(n)$ fonksiyonundan daha hızlı büyümeyen $f(n)$ fonksiyonlarını kapsar.
- Θ($g(n)$): $g(n)$ fonksiyonları ile aynı derecede büyüyen $f(n)$ fonksiyonlarını gösterir.
- Ω($g(n)$): en az $g(n)$ fonksiyonları kadar hızda büyüyen $f(n)$ fonksiyonlarını belirtmek için kullanılır.

Big-oh

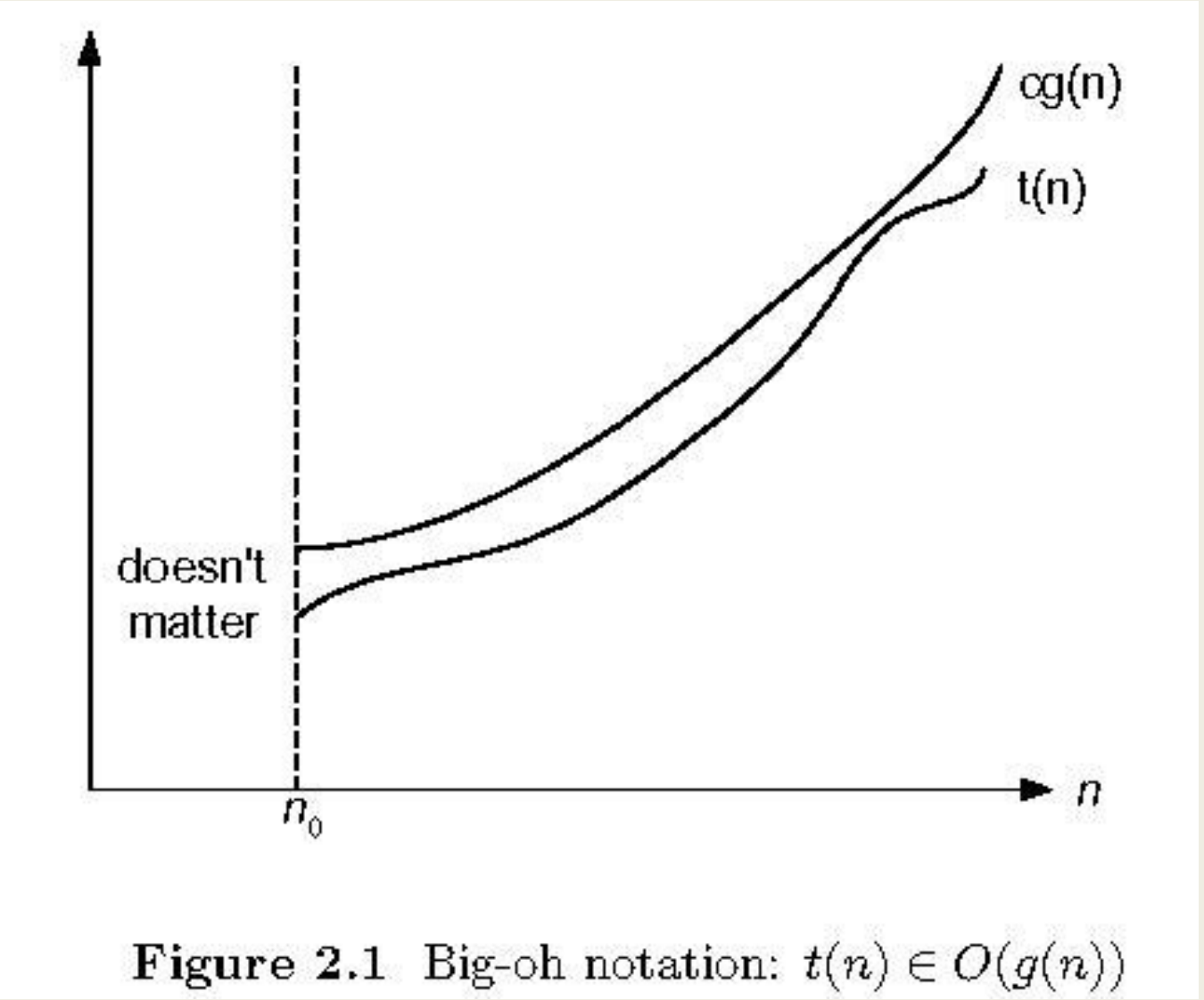


Figure 2.1 Big-oh notation: $t(n) \in O(g(n))$

Big-omega

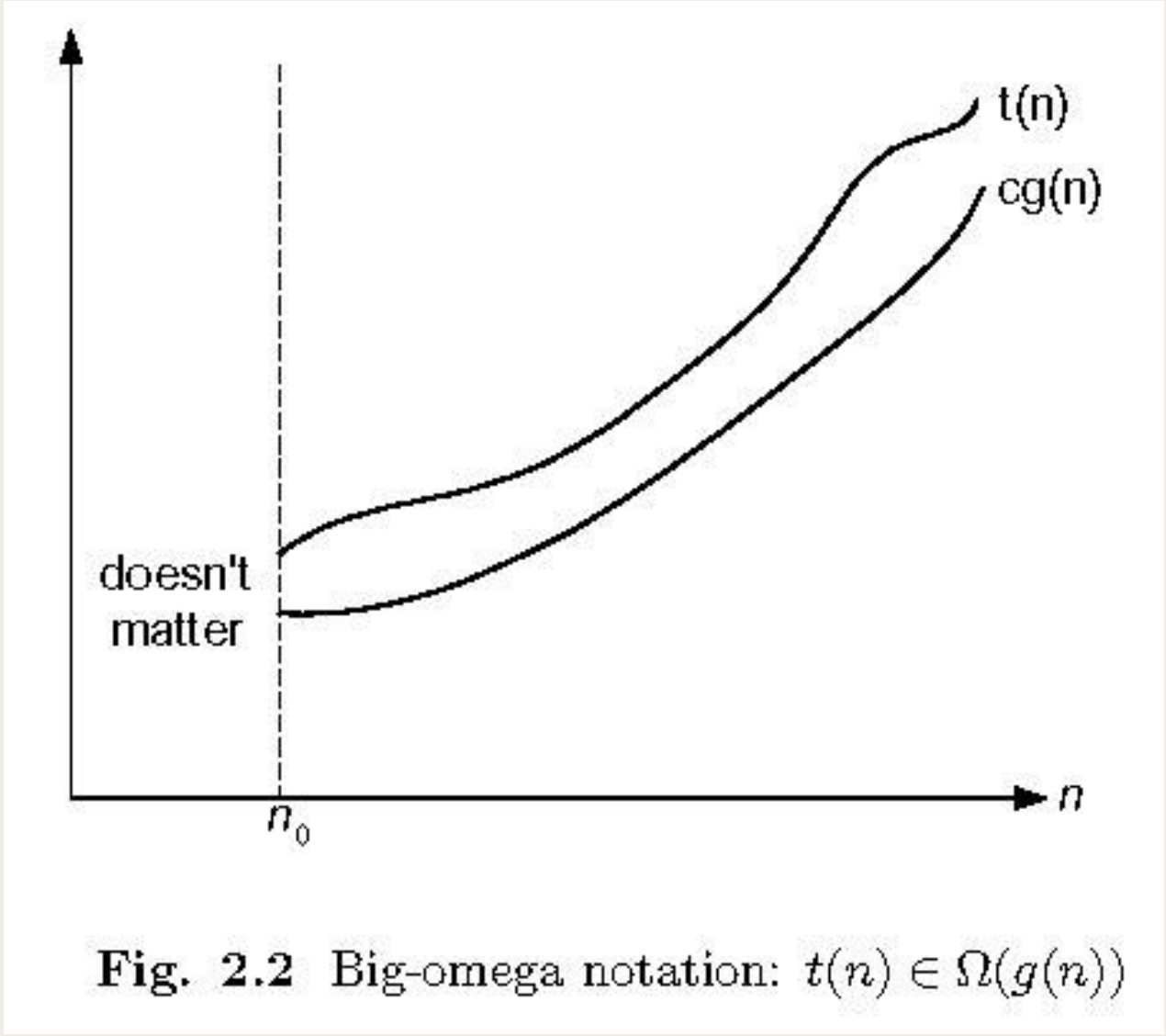


Fig. 2.2 Big-omega notation: $t(n) \in \Omega(g(n))$

Big-theta

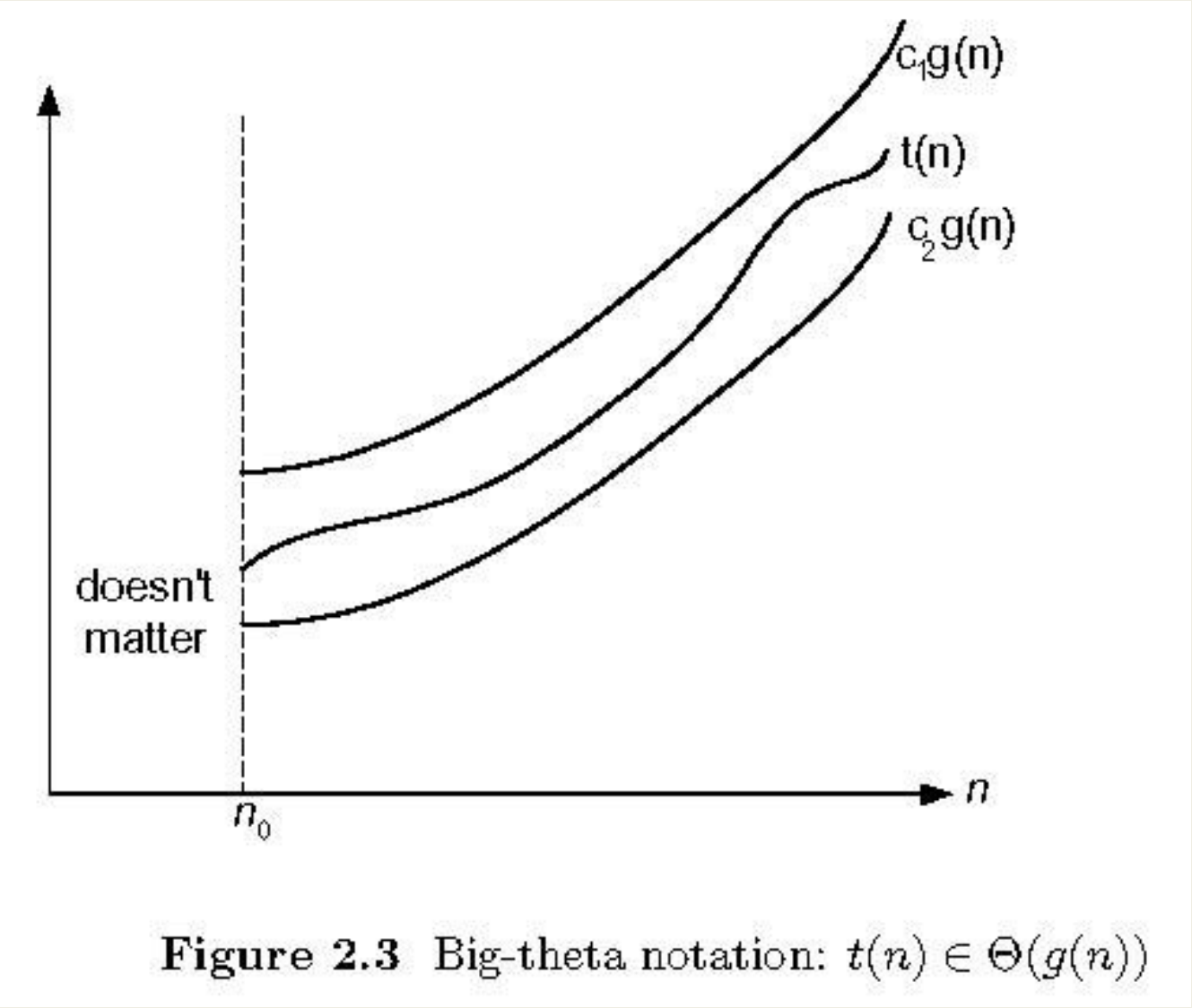
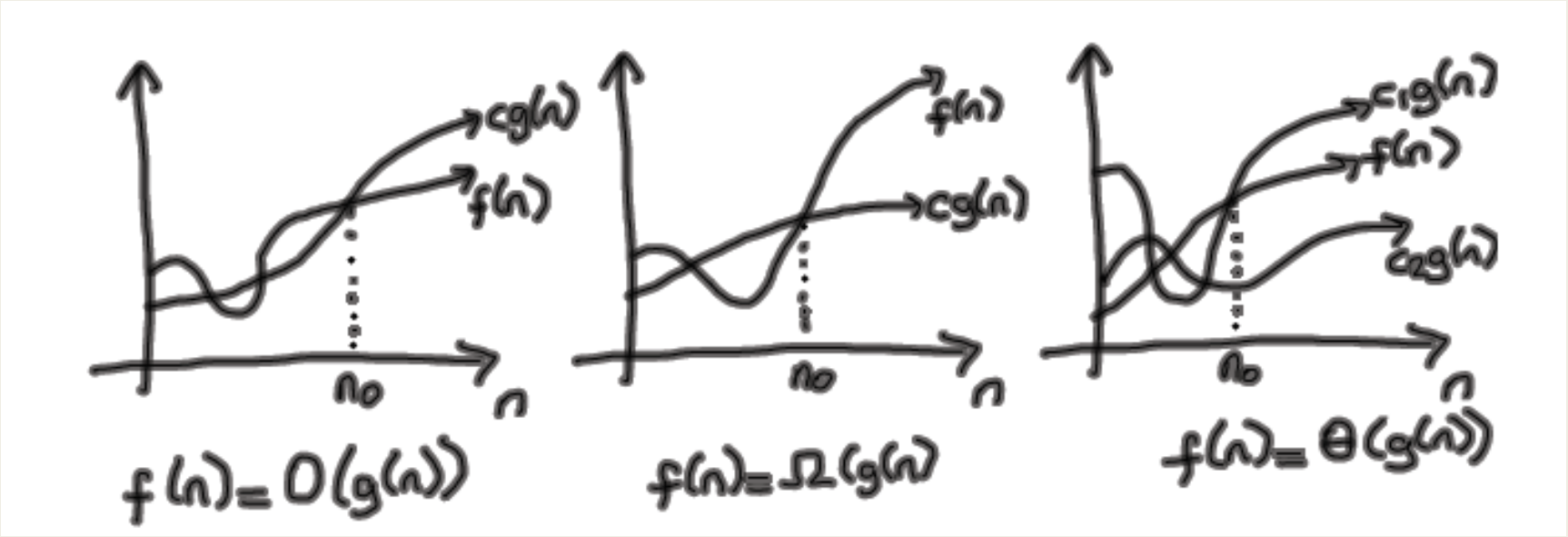


Figure 2.3 Big-theta notation: $t(n) \in \Theta(g(n))$

Asimtotik Notasyonların Grafik Üzerinde Gösterimi



~~#~~) Asimtotik analiz $n \to \infty$ iken fonksiyonun büyümesi ile ilgilidir.

Big *O* Formal Tanımı

Tanım: $f(n) \in O(g(n))$ ise, f(n) fonksiyonunun büyüme derecesi, $g(n)$'in büyüme sabit bir sayı ile çarpımının büyüme derecesinden küçüktür.

$$f(n) \leq c\, g(n)\ ,\ \forall\ \ n \geq n_0$$

Eşitsizliğini sağlayan pozitif bir sabit c ve pozitif bir tamsayı $n_0$vardır.

--- O notasyonu ilk olarak alman Matematikçi Bochmann tarafından 1894 yılında tanımlanmıştır.

--- Algoritmanın en kötü durum analizini yapmak için kullanılan notasyondur.

Örnek:

T(n)=3n+8 = O(n) dir. (3n+8 ∈ O(n))

3n+8<= c*n

n>=1 için 3n+8<=3n+8n<=11n (c=11, n_0 =1)

n>=4 için 3n+8<= 5n (c=5, n_0 =4)

Diğer c ve n_0 değerleri bulunabilir.

Örnek:

T(n)=3n+8= O(n^2) dir. (3n+8 ∈ O(n^2))

3n+8<= c*n^2

n>=5 için 3n+8<= c * n^2 (c=1, n_0 =5)

n>=3 için 3n+8<= c * n^2 (c=2, n_0 =3)

Diğer c ve n_0 değerleri bulunabilir.

Örnek:

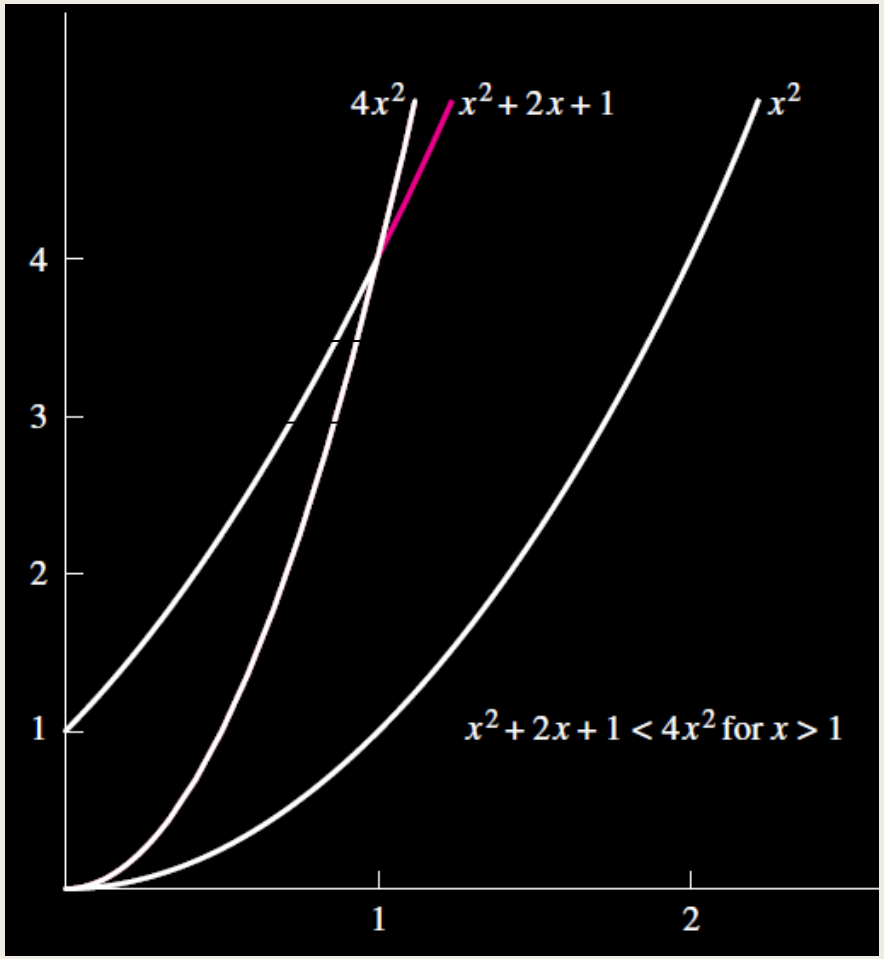
- $f(x) = x^2 + 2x + 1$ ise $0(x^2)$ dir.
- Eğer $x>1$ ise $x<x^2$ dir ve, eğer $x>1$ ise $1<x^2$ dir.
- Çünkü:

  $0 \leq x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2 = 4x^2$

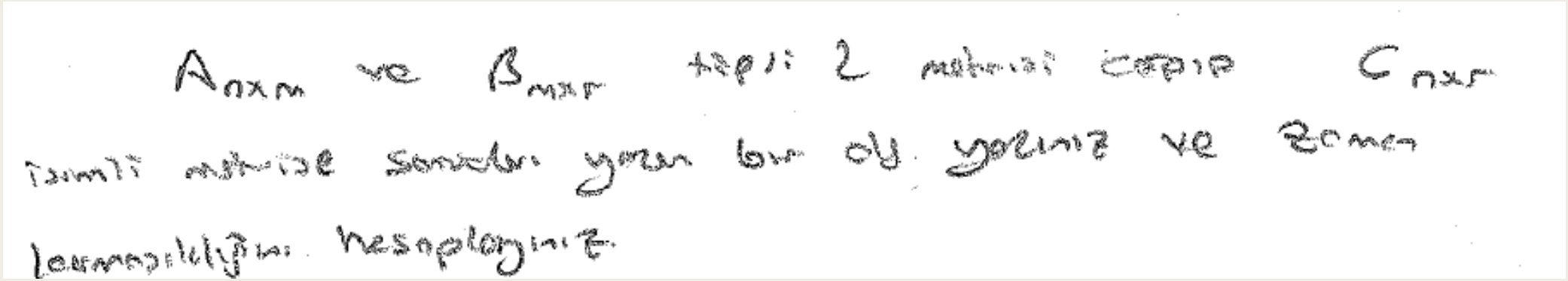
$x>1$ olduğu her değer için

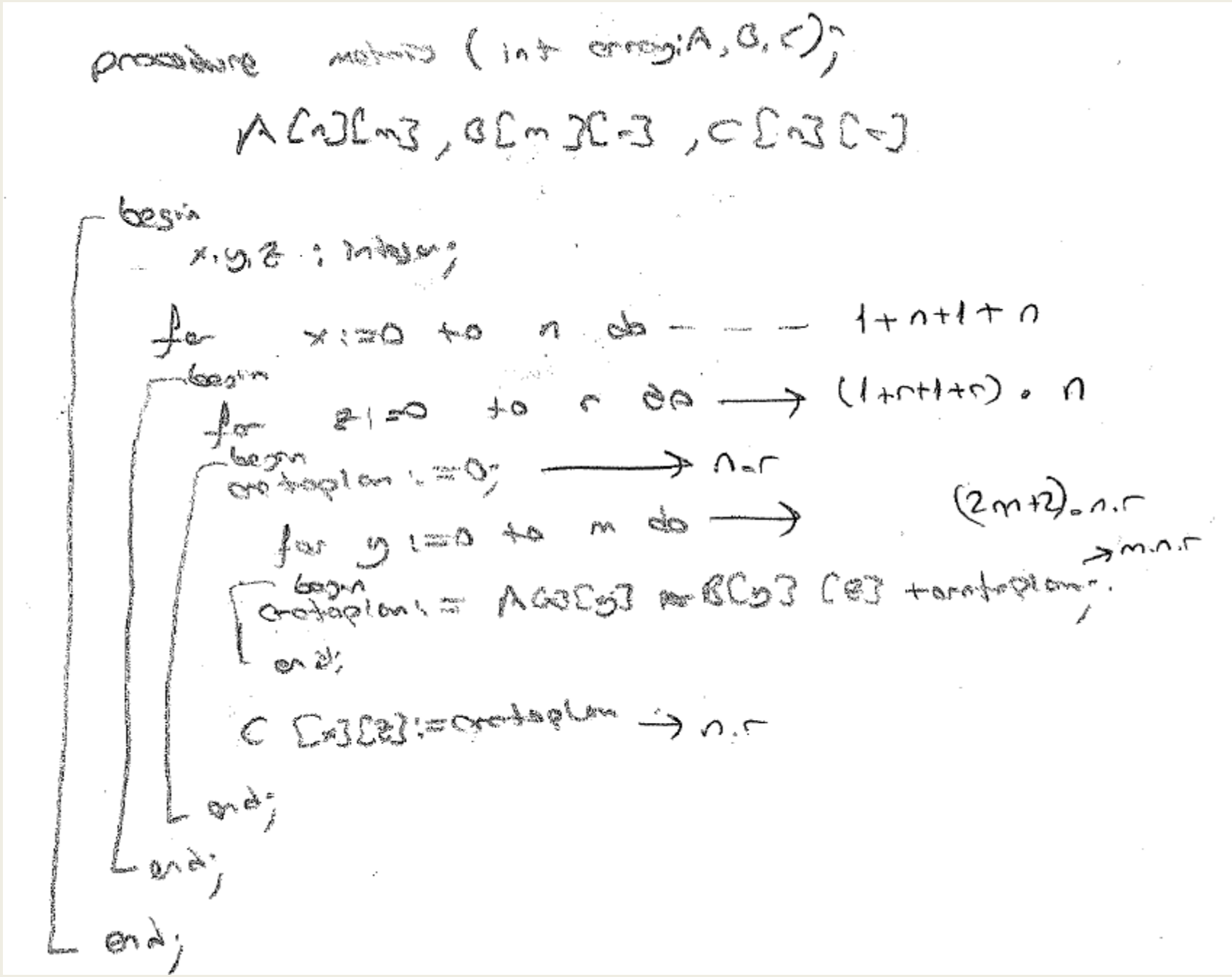
- Böylece, $c = 4$ ve $n_0 = 1$ **alınırsa** $f(x) \in 0(x^2)$ elde edilir.

Grafiği



Örnek:


Anxm ve Bmxr tipi 2 matrisi
Cmxr
tamsayı matrisle sonuçları yazan bir alt yordamı ve
hesaplayınız.


begin
for x:=0 to n do ——— 1+n+1+n
for z:=0 to n do ⟶ (1+n+1+n) · n
for y:=0 to m do ⟶ (2m+2)·n·n
end;
end;
end;

$T(n,m,r) = 2n+2+(2r+2)\cdot n + n\cdot r + (2m+2)\cdot r\cdot n + n\cdot m\cdot r + r n$

$= 3mnr + 6nr + 4n + 2$

$m = n = r$ olursa

$T(n) = 3n^3 + 6n^2 + 4n + 2$

$O(n^3)$ → karmaşıklık!!!

Önemli Teoremler

Teo: Eğer $f(n) = O(g(n)) \Rightarrow c \cdot f(n) = O(g(n))$

İspat)

$f(n) = O(g(n)) \Rightarrow \exists k, m \quad \forall n > m, \; f(n) \leq |k| \cdot g(n)$

$\Rightarrow |c \cdot f(n)| \leq k \cdot |c| \cdot |f(n)| \Rightarrow c f(n) = O(g(n))$

Teo: Eğer $f(n) = O(g(n))$ ve $h(n) = O(g(n)) \Rightarrow$

$(f+h)(n) = O(g(n))$ 'dir.

İspat:

$f(n) = O(g(n)) \Rightarrow \exists k, n_1 : \forall n \geq n_1 \quad |f(n)| \leq k \cdot |g(n)| \quad ---(1)$

$h(n) = O(g(n)) \Rightarrow \exists l, n_2 : \forall n \geq n_2 \quad h(n) \leq l \cdot |g(n)| ---(2)$

$n = \max\{n_1, n_2\} \overset{(1),(2)}{\Longrightarrow} \forall n \geq n_1$

$|f(n) + h(n)| \leq |f(n)| + |g(n)| \leq k \cdot |g(n)| + l|g(n)|$

$= (k+l) \quad |g(n)|$

$\Rightarrow (f+h)(n) = O(g(n))$

Teorem:

Eğer $f(n) = O(g(n))$, $h(n) = O(e(n)) \Rightarrow (f \cdot h)(n) = O((g \cdot e)(n))$

İspat:

$f(n) = O(g(n)) \Rightarrow \exists k, m_1 : \forall n \geq m_1, |f(n)| \leq k \cdot |g(n)|$

$h(n) = O(e(n)) \Rightarrow \exists \ell, m_2 : \forall n \geq m_2, |h(n)| \leq \ell \cdot |e(n)|$

$m = \max\{m_1, m_2\} \Rightarrow \forall n \geq m$ için:

$$|f(n) \cdot h(n)| \leq |f(n)| \cdot |h(n)| \leq k \cdot |g(n)| \cdot \ell \cdot |e(n)| =$$

$$= k \cdot \ell \cdot |g(n) \cdot e(n)|$$

$\Rightarrow (f \cdot h)(n) = O((g \cdot e)(n))$

Çalışma Sorusu:

Teorem:

Eğer $p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_2 n^2 + a_1 n + a_0$

$\Rightarrow p(n) = O(n^k)$ 'dır.

Kanıt??

Bazı Önemli Fonksiyonların Büyüme Dereceleri

- Tüm logaritmik fonksiyonlar $\log_a n$ aynı asimptotik sınıfa sahiptir.

  O(log *n)* logaritmanın tabanı $a > 1$ önemli değil.

- Aynı derece *k'ye sahip olan tüm polinomlar* aynı asimptotik sınıfa sahiptir. :
- $a_k n^k + a_{k-1} n^{k-1} + \ldots + a_0 \in O(n^k)$

- Üstel fonksiyonlar a^n, a değerine göre farklı büyüme sınıfına aittir.

- order log n < order n^α (α>0) < order a^n < order $n!$ < order n^n

Temel Asimptotik Verimlilik Sınıfları

1	sabit
$\log n$	logaritmik
n	lineer
$n \log n$	n-log-n or linerritmetik
n^2	quadratic
n^3	kübik
2^n	üstel
$n!$	faktoriyel

Ω- Formal Tanımı

- **Tanım:** $f(n) \in \Omega(g(n))$ ise, f(n) fonksiyonunun büyüme derecesi, $g(n)$'in sabit bir sayı ile çarpımının büyüme derecesinden büyük veya eşittir.

$$f(n) \geq c\, g(n)\ ,\ \forall\ n \geq n_0$$

Eşitsizliğini sağlayan pozitif bir sabit c ve pozitif bir tamsayı n_0 vardır.

Örn: $T(n)=3n+5 \in \Omega(n)$

$3n+5 >= 3n$, tüm $n>=1$ için sağlanır ($c=3$, $n_0=1$)

Örnek: $n = \Omega(\lg n)$ 'dir.

$f(n)$ $g(n)$

$\log_2 n = \lg n$

$c \cdot g(n) \leq f(n)$

$1 \cdot \lg n \leq n$

$n = 2$ için $1 \leq 2$ ✓

$c = 1$ ve $\forall n \geq 10$ için, $c \cdot g_n \leq f(n)$

Olduğundan $f(n) = \Omega(g(n))$ 'dir.

$n = \Omega(\lg n)$ 'dir.

Çalışma Sorusu: $n^3 = \Omega(n^2)$?

Θ Formal Tanımı

- **Tanım:** $f(n) \in \Theta(g(n))$ ise, f(n) fonksiyonunun büyüme derecesi, $g(n)$ fonksiyonun bir sabit katından yüksek aynı zamanda $g(n)$ fonksiyonun bir sabit katından da düşük olmaktadır.

$$c_1 g(n) \leq f(n) \leq c_2 g(n) , \ \forall \ n \geq n_0$$

Eşitsizliğini sağlayan pozitif sabit c_1, c_2 *sayıları* ve pozitif bir tamsayı n_0 vardır.

$\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$ 'dir.

Hem alttan, hem de üstten sınırlıdır.

$5n^2 + n - 7 = \Theta(n^2)$ 'dir

*)

Θ notasyonunda küçük terimler ihmal edilir

Örnek: $\frac{1}{2}n^2 - 2n \in \Theta(n^2)$ olduğunu gösteriniz.

$f(n) = \frac{1}{2}n^2 - 2n$ $g(n) = n^2$ dir.

$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ $\forall n \geq n_0$

$c_1 c_2 > 0$

$c_1 < c_2$

$\frac{1}{4} \cdot n^2 \leq \frac{1}{2}n^2 - 2n \leq \frac{1}{2} \cdot n^2$ $\Rightarrow \forall n$ için doğru dur.

$\frac{1}{4} n_0^2 = \frac{1}{2} n_0^2 - 2n_0$

$2n_0 = \frac{1}{4} n_0^2$

$\boxed{n_0 = 8}$ $\Rightarrow$ eşik değeri.

Böylece: $c_2 = \frac{1}{2} > 0$

$c_1 = \frac{1}{4} > 0$

$n_0 = 9$ için

$\frac{1}{2} n^2 - 2n \in \Theta(n^2)$ dir.

(Ornek) $n_0 = 9$ için

$$\frac{1}{4} \cdot 9^2 \leq \frac{1}{2} \cdot 9^2 - 2 \cdot 9 \leq \frac{1}{2} \cdot 9^2$$

$$\frac{81}{4} \leq \frac{45}{2} \leq \frac{81}{2}$$

Kaynaklar

- *Discrete Mathematics and Its Applications*, Kennet H. Rosen

(Ayrık Matematik ve Uygulamaları, Kennet H. Rosen (Türkçe çeviri), Palme yayıncılık)

- *Discrete Mathematics: Elementary and Beyond*, L. Lovász, J. Pelikán, K. Vesztergombi, 2003.

- *Introduction to Algorithms*, T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, 2009.

- *Introduction To Design And Analysis Of Algorithms*, A. Levitin, 2008.