Ex $T(n) = 2T(n/2) + n^2$

$a=2, b=2, f(n)=n^2$

$n^2 \Leftrightarrow n^{\log_b a}$

$n^2 > n$

$af(n/b) \leq c.f(n)$

$2.\frac{n^2}{4} \leq c.n^2$

$\frac{1}{2} \leq c \quad c=\frac{1}{2}$

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

Case 3

Ex $T(n) = 2T(n/2) + n$

$a=2, b=2, f(n)=n$

$f(n) = n^{\log_2 2}$

$n \Leftrightarrow n^{\log_2 2}$

$n = n$

$= \Theta(n^{\log_2 2} \log_2 n)$

$= \Theta(n \log_2 n)$

Case 2

Ex $T(n) = 4T(n/2) + n$

$a=4, b=2, f(n)=n$

$f(n) \Leftrightarrow n^{\log_b a}$

$n \leq n^{\log_2 4}$

$n \leq n^2$

$\Theta(n^{\log_2 4})$

$\Theta(n^2)$

Case 1

Ex $T(n) = c + T(n/2)$

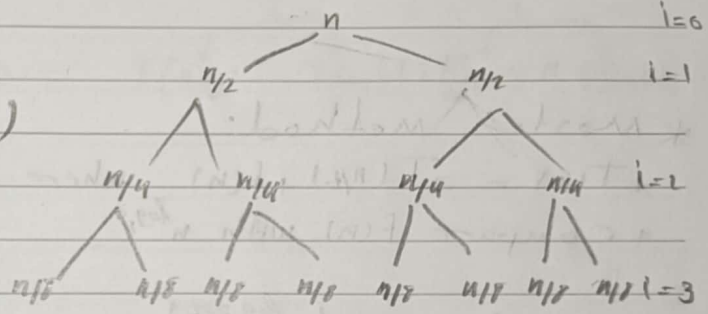
$c \Leftrightarrow n^{\log_2 1}$

$c = 1$

$T(n) = \Theta(n^{\log_2 1} \log_2 n)$

$= \Theta(\log_2 n)$

Ex $T(n) = 2T(n/2) + n$



nodes count $i = 2^i = 2^i$

Height $i = \log_b n$

size input $= n + n/2 + n/4 + \dots + n/2^i$

$T(1)$

$2^i = n \quad i = \log n$

$cost = \frac{n}{2^i}$

last leaves: $2^{\log_b n}, n^{\log_b a}$

Total complexity: $\sum_{i=0}^{\log_2 n - 1} n(\frac{1}{2})^i + n^{\log_b a}.T(1) = n + \log n$

$(\log_2 n + 1)n + n = n \log n = O(\log n)$

* master method

$T(n) = f(n) + aT(n/b)$

$a=2, b=2 \quad f(n)=n$

$n \Leftrightarrow n^{\log_b a}$

$n \Leftrightarrow n^{\log_2 2}$

$n = n \rightarrow \Theta(n \log n)$

* substition

$T(n) = 2T(n/2) + n \quad dn\log n - dn + n \leq dn\log n$

$T(n) = dn \log n$

$dn \geq n$

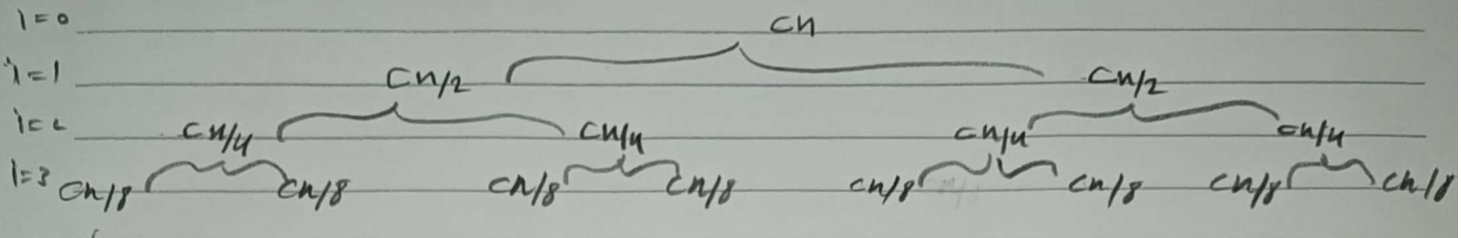
$T(n/2) = dn/2 \log n/2$

$d \geq 1$

$T(n/2) = dn/2 \log n - dn/2$

$2T(n/2) + n = dn \log n$

Ex $T(n) = 2T(n/2) + cn$



- Height $\Rightarrow i = n + n/2 + n/4 \dots + n/2^i \Rightarrow n = 2^i \Rightarrow \log_b n = i$
- nodes $2^i = 2^i$
- input size $= cn/2^i$
- last level leaves $= 2^{\log_b n}$ عدد النود
- cost $i = n$

Total complexity: $\sum_{i=0}^{\log n - 1} n + b^{\log_b^n} nc.T(1) = n\log n - n + nc.T(1)$

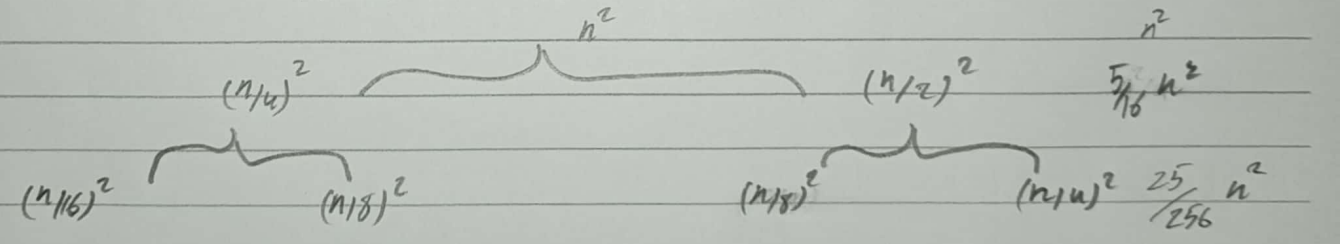
$\Rightarrow O(n\log n)$ Height × cost = total complexity

Ex $T(n) = T(\frac{2n}{3}) + 1$ $a=1, b=3/2, f(n)=1$

$1 \leq n^{\log_{3/2} 1}$

$1 = 1 \Rightarrow O(n^{\log_{3/2} 1} \cdot \log n) = O(\log n)$

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



$n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 \dots\right) = O(n^2)$

$n^2 \sum_{i=0}^{n-1} r^n = \frac{1-(5/16)^{\infty}}{1-5/16} = \frac{1}{1-5/16} = \frac{16}{11}$ مالها دلالة

Ex $T(n) = T(n/2) + c$

c $T(n)$

c $T(n/2)$

c $T(n/4)$

c $T(1)$

$$\sum_{i=0}^{\log_2 n - 1} c + c \cdot T(1)$$

c cost

$$c \cdot \log_2 n + c \cdot T(1) = O(\log n)$$

Ex $T(n) = T(n-1) + n$

n $T(n)$

$n-1$ $T(n-1)$

$n-2$ $T(n-2)$

1 $T(1)$

$$\sum_{i=0}^{n-1} i = \frac{n(n+1)}{2} - 1 = O(n^2)$$

* Master method:
  * $T(n) = aT(n/b) + f(n)$ where $a \geq 1$, $b > 1$ and $f(n) > 0$
  * Compare $f(n)$ with $n^{\log_b a}$

- Case1: $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then: $T(n) = \Theta(n^{\log_b a})$
- Case2: $f(n) = \Theta(n^{\log_b a})$, then: $T(n) = \Theta(n^{\log_b a} \log_2 n)$
- Case3: $f(n) = \Omega(n^{\log_b a})$, then: for $\varepsilon > 0$ and if $af(n/b) \leq cf(n)$ for some $c < 1$ and all sufficiently large n, then: $T(n) = \Theta(f(n))$

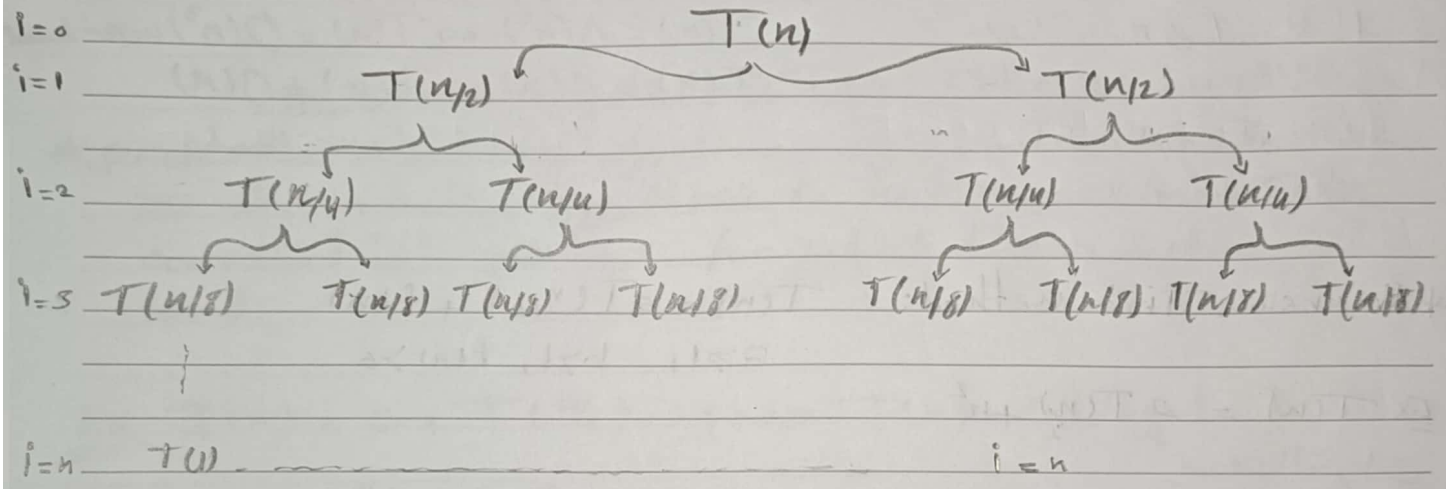
$$T(n) = n^2 \sum_{i=0}^{\log_2 n - 1} \left(\frac{1}{2}\right)^i + n^{\log_2 1} \cdot T(1)$$

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x} \; ; x < 1 \quad = 2n^2 + n \cdot T(1)$$

$$= O(n^2)$$

• تحديد تفرعات النود $T(n) = 2 \cdot T(n/2) + n^2$

$T(n) = a \cdot T(n/b) + F(n)$



- Number of nodes at level $i = 2^i \Rightarrow a^i$
- input size at level $i = \frac{n}{2^i} \Rightarrow \frac{n}{b^i}$
- Height of tree $= n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} \cdots + \frac{n}{2^i}$ // $\frac{n}{2^i} = \frac{n}{b^i}$, $2^i = b^i = n$

عدد المستويات Number of leaves $= a^{\log_b n}$, $n^{\log_b a}$ ⇐ Height ⇐ $i = \log_b n$

- cost at $i = \frac{n^2}{2} = \frac{n^2}{b^i}$
- Total complexity equal total complexities of all levels.

$$\sum_{i=0}^{\log_b n - 1} n^2 \left(\frac{1}{2}\right)^i + \underbrace{a^{\log_b n}}_{\downarrow} \cdot T(1)$$

* عدد الأوراق في المستوى الأخير $T(1)$

Ex $T(n) = n + T(n-1)$ is $O(n^2)$

$T(n) \le d \cdot g(n)$
$T(n) \le d \cdot n^2$
$T(n-1) \le d \cdot (n-1)^2$
$T(n-1) \le dn^2 - 2nd + d$
$T(n) \le d \cdot n^2$
$n + T(n-1) \le d \cdot n^2$
$n + dn^2 - 2nd + d \le dn^2$
$\frac{d}{2d-1} \le n$

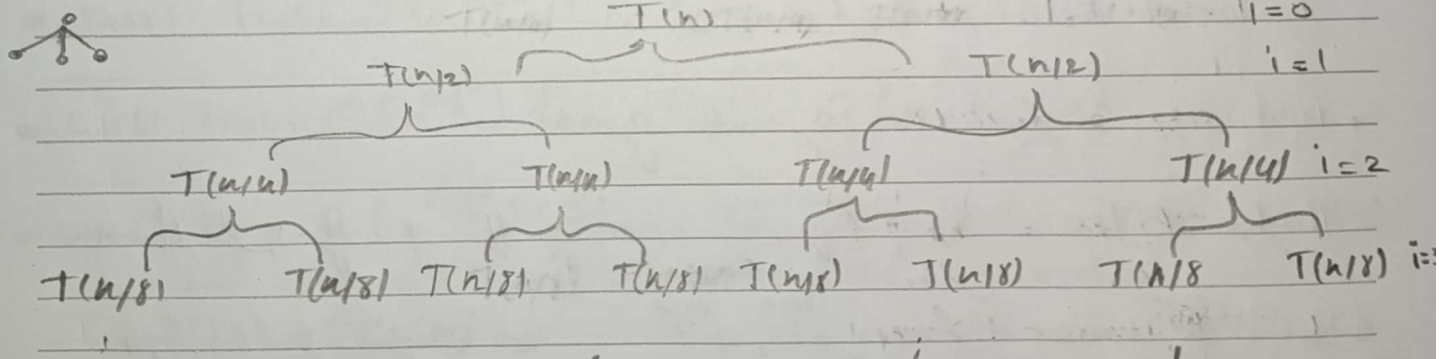
$d = 1 \quad 1 \le n$ صحيح

$T(n) \le d \cdot g(n)$
$T(n) \le d \cdot n$
$T(n-1) \le d \cdot (n-1)$
$T(n-1) \le dn - d$
$T(n) \le d \cdot n$
$n + T(n-1) \le dn$
$n + dn - d \le dn$
$n \le d$ غلط

$T(n) = O(n^2) \Rightarrow T(n) = O(n^3)$ upper bound
$T(n) = O(n^2) \Rightarrow T(n) \neq O(n)$
lower bound

* Recursion Tree method: $T(n) = aT(n/b) + f(n)$
$a \ge 1, \quad b > 1, \quad f(n) > 0$

Ex $T(n) = 2T(n/2) + n^2$



- $2^0 + 2^1 + 2^2 \dots 2^i$ = nodes || 2^i = nodes count
- input size: n/b^i
- Height of tree: $\frac{n}{2^i} = 1 \Rightarrow 2^i = n \quad i = \log_2 n$ (آخر نود (ورقة))
- cost at i: $T(n) = n^2, \quad T(n/2) = (n/2)^2 \dots$
$= \frac{n^2}{2^i}$
- total complexity: total complexity of of levels
Total complexity: $\sum_{i=0}^{\log_2 n - 1} \frac{n^2}{2^i} + n^{\log_b^a} * T(1)$

leaves = 2^i
$2^{\log_2 n}$
$n^{\log_b^a}$
عدد الأوراق

The substitution method.

1. Guess a solution
2. use induction to prove that the solution works

طريقة تعطيني إثبات رياضي لتحقيق الحل المرضي

* Guess a solution

$T(n) = O(g(n))$

induction goal: apply the definition of the asymptotic notation

$T(n) \le d \cdot g(n)$ for some $d > 0$ and $n \geq n_0$

ممكن اجبارها بالحل أو إعطائها قيمة ابتدائية

Induction hypothesis: $T(k) \le d \cdot g(k)$ for all $k < n$

* prove the induction goal:

use the induction hypothesis to find some values of the constant "d" and "n_0" for which the induction goal holds

Ex $T(n) = c + T(n/2)$ prove $T(n) = O(\log n)$

$T(n) = O(\log n)$

$T(n) \le d \cdot g(n)$

$T(n) \le d \cdot \log n$

$T(n/2) \le d \cdot \log n/2$

$T(n/2) \le d \cdot (\log n - \log_2 2)$

$T(n/2) \le d \log n - d$

$c + T(n/2) \le d \log n$

$c + d\log n - d \le d \log n$

$c \le d$

قيمة ابتدائية ثابتة

طلعت علاقة رياضية صحيحة

ولكن لو كانت $c > d$ فتصبح علاقة خاطئة

Ex $T(n) = n + T(n/2)$

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

$T(n) = n + n/2 + n/4 + T(n/8)$

$T(n) = n * (1/2)^{k-1} + T(n/2^k)$

$(\frac{1}{2})^{k-1} \Rightarrow \sum_{i=0}^{k-1} (\frac{1}{2})^i = \frac{1-(1/2)^k}{1-1/2} = 2 - 2^{1-k}$

$= n \times (2 - 2^{1-k})$; $2^k = n$ $\log n = k$

$= n \times (2 - 2 \cdot \frac{1}{n})$

$= 2n - 2$

$\Rightarrow T(n) = 2n - 2 + T(1) \rightarrow$ Base case

$T(n) = O(n)$ // * Bir önceki çözümde bir ya-da yanlışlık yaptım $(1/2)^{k-1}$ eksikti //

Ex $T(n) = n + 2T(n/2)$

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

$= n + n + 4T(n/4)$

$T(n) = n + n + 4(n/4 + 2T(n/8))$

$= n + n + n + 8T(n/8)$

$T(n) = n + n + n + 8[n/8 + 2T(n/16)]$

$= n + n + n + n + 16T(n/16)$

$T(n) = nk + 2^k T(n/2^k)$ Assume $2^k = n \rightarrow \log n = k$

$= n \log n + n \cdot T(1)$ $T(1) = 1$

$= n \log n + n \Rightarrow T(n) = O(n \log n)$

Ex $T(n) = n + T(n-1)$

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

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

$T(n) = n + n-1 + n-2 \cdots + 2 + T(n-k)$

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

$= \frac{n^2+n}{2} - 1 + T(0) \Rightarrow T(n) = O(n^2)$

* substitution method:

→ Show that the solution of $T(n) = T(n-1) + n$ is $O(n^2)$

We guess $T(n) \leq c.n^2$ $\quad c > 1/2$

$T(n) \leq c.(n-1)^2$

$= cn^2 - 2cn + c + n$

$= cn^2 - n(1-2c) + c \leq cn^2$

$4n^2 - 3n + 4 \leq 4n^2$

$T(n) = O(n^2)$

* iteration method:

$T(n) = T(n-1) + n \longrightarrow \Theta(n^2)$

$T(n) = T(n/2) + c \longrightarrow \Theta(\log n)$

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

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

نتاكد بالوصول الى

Ex $T(n) = c + T(n/2)$

$T(n) = c + c + T(n/4)$

$T(n) = c + c + c + T(n/8)$

$T(n) = k.c + T(n/2^k)$

Assume $2^k = n$ $\quad k = \log n$

$T(n) = c\log n + T(1) \rightarrow$ Base case

Ex $T(n) = n + T(n/2)$

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

$T(n) = n + n/2 + n/4 + T(n/8)$

$T(n) = n + n/2 + n/4 + n/8 \cdots n/2^k + T(n/2^k)$

$= n + (\frac{1}{2})^0 + (\frac{1}{2})^1 + (\frac{1}{2})^2 + \cdots (\frac{1}{2})^k)$

$= n \sum_{i=0}^{k} (\frac{1}{2})^i = \frac{1 - \frac{1}{2}^{k+1}}{1 - \frac{1}{2}} = 2 - \frac{1}{2}^k = n(2 - 2^{-k})$

$T(n) = 2n - 2^{-k} \cdot n + T(n/2^k) \quad k = \log n$

$T(n) = 2n - 2^{-\log n} + T(1) \Rightarrow T(n) = 2n - \frac{1}{n} + T(1)$

$O(n)$ Benim 2520m

Ex
```
Func Sum(int n) int {
    if n == 1 {
        return 1
    }
    return n + func(n-1);
}
```

$T(1) = 1$
$T(n) = 3 + T(n-1)$
$T(n) = 6 + T(n-2)$
$T(n) = 9 + T(n-3)$
$\vdots$
$T(n) = 3.k + T(n-k)$

$\Leftarrow \{ n-k = 1 \quad k = n-1$

$T(n) = 3n - 3 + T(1)$
$T(n) = 3n - 3 + 1 \quad = \quad T(n) = 3n - 2 \quad \therefore T(n) = O(n)$

Ex Complexity of Recursive Algorithm – The iteration Method.

$T(n) = C + T(n/2)$
$T(n) = C + C + T(n/4)$
$= C + C + C + T(n/8)$
Assume $n = 2^k$
$T(n) = C \log n + T(1)$
$= O(\log n)$

$T(n) = C + T(n/2)$
$T(n) = 2C + T(n/4)$
$T(n) = 3C + T(n/8)$
$\vdots$
$\therefore T(n) = k.C + T(n/2^k)$
Assum $n = 2^k$

$T(n) = \log n C + T(n/n)$
$T(n) = \log n C + T(1)$
$\Leftarrow T(n) = \log n . C + T(1)$

$O(n)$

Ex $T(n) = n + 2T(n/2)$
$T(n) = n + 2(n/2 + 2T(n/4))$
$= n + n + 4T(n/4)$
$= n + n + 4(n/4 + 2T(n/8))$
$= n + n + n + 8T(n/8)$
$= kn + 2^k T(n/2^k)$
$\Rightarrow$ Assume $n = 2^k$
$\rightarrow n \log n + nT(1) = O(n . \log n)$

Ex $T(n) = 1$ if $n = 1$
$T(n) = 2T(n-1)$ if $n > 1$
$T(1) = 1$
$T(n) = 2T(n-1)$
$T(n) = 2.2T(n-2)$

$T(n) = 2^3 . T(n-3)$
$T(n) = 2^k . T(n-k)$
$T(1) = 1 \Rightarrow n - k = 1 \Rightarrow k = n-1$
$T(n) = 2^{n-1} . T(1)$
$O(2^n)$

Recursion Time complexity

```
int factorial(int n)
{
    if (n==0)          -> 1 unit of time
    {
        return 1;
    }
    return n * factorial(n-1);
}                  1 unit of time   1 unit of time
```

$T(0) = 1$

$T(n) = 3 + T(n-1)$ if $n >= 1$

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

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

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

$T(n) = 6 + T(n-2)$

$T(n) = 9 + T(n-3)$

$T(n) = 3.k + T(n-k) \Rightarrow T(0) = 1 \quad n-k = 0$

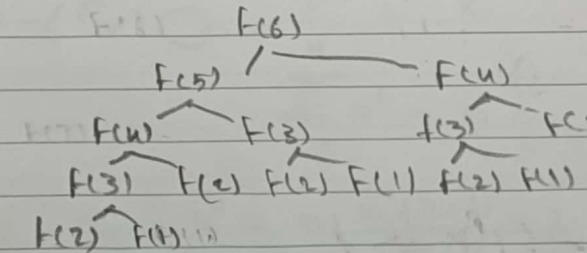
$T(n) = 3n + T(n-n) \Leftarrow n = k$

$T(n) = 3n + T(0)$

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

```
int fibonaci(int n)
{
  if (n<=1)
    return n;
  return Fibonaci(n-1)+fibonaci(n-2);
}
```

$O(2^n)$



f(3) called 3 time

f(1) called 3 time

f(2) called 5 time

```
int Fibonaci(int n)
{
  int n1,n2,n3;
  for(int i=2; i<=x; i++)
  {
    n3 = n1+n2;
    n1 = n2
    n2 = n3
  }
  if(n <= 1)
    return n;
  n1 = 0, n2=1;
```

$O(n)$

Ex
```
for (i=1; i<=n; i++)
{
    for (j=1; j<=i; j++)
    {
        for (k=1; k<=j; k++)
        {
            x = x+1;
        }
    }
}
```

$$\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{j} 1 = \sum_{i=1}^{n}\sum_{j=1}^{i} j = \sum_{i=1}^{n} \frac{i(i+1)}{2}$$

$$\sum_{i=1}^{n} \frac{i^2}{2}+\frac{i}{2} = \frac{1}{2}\sum_{i=1}^{n} i^2 + \frac{1}{2}\sum_{i=1}^{n} i = \frac{1}{2}\left[\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}\right]$$

$= O(n^3)$

Ex
```
for i=1 to n do
    for j=i to n do
        for k=i to j do
            m = i+j+k+m
```

$$\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=j}^{j} 1 = \sum_{i=1}^{n}\sum_{j=i}^{n} j$$

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

$$\frac{1}{2}n^3 + \frac{1}{2}n^2 + \frac{1}{2}\left(\frac{n(n+1)(2n+1)}{6}\right) - \frac{1}{2}\left(\frac{n(n+1)}{2}\right)$$

$$6n^3 + 6n^2 + (n^2+n)(2n+1) - 6n^2 - 6n$$

$$6n^3 + 6n^2 + 2n^3 + n^2 + 2n^2 + n - 6n^2 - 6n = 8n^3 + 3n^2 - 5n$$

$= O(n^3)$

Ex
```
for (int i=0; i<N; i++)
{
    for (int j=i+1; j<N; j++)
    {
        stmt;
    }
}
```

$$\sum_{i=0}^{n} \sum_{j=i+1}^{n} 1$$

$$\sum_{i=0}^{n} n-i-1+1 = \sum_{i=0}^{n} (n-i)$$

$$\sum_{i=0}^{n} n - \sum_{i=0}^{n} i = n^2 - n(n+1)/2$$

$$= \frac{2n^2 - n^2 - n}{2} = \frac{n^2 - n}{2}$$

$O(n^2)$

Ex
```
for (int i=0; i<N; i++) {
    for (int j=i+1; i<N; j++) {
        for (int k=j+1; k<N; k++) {
            stmt;
        }
    }
}
```

$$\sum_{i=0}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1}^{n} 1 = \sum_{i=0}^{n} \sum_{j=j+1}^{n} (n-j-1+1)$$

$$\sum_{i=0}^{n} \sum_{j=i+1}^{n} (n-j) = \sum_{i=0}^{n} \sum_{j=i+1}^{n} n - \sum_{i=0}^{n} \sum_{j=i+1}^{n} j$$

$$\sum_{i=0}^{n} n(n-i) - \sum_{i=0}^{n} \frac{n(n+1)}{2} - \frac{(j+1)(j+1-1)}{2}$$

$$\sum_{i=0}^{n} n^2 - ni - \sum_{i=0}^{n} \frac{n^2+n}{2} - \frac{j^2+j}{2}$$

$$\sum_{i=0}^{n} n^2 - n\sum_{i=0}^{n} i - \frac{1}{2}\sum_{i=0}^{n} n^2 - \frac{1}{2}\sum_{i=0}^{n} n + \frac{1}{2}\sum_{i=0}^{n} i^2 + \frac{1}{2}\sum_{i=0}^{n} i$$

$$n^3 - n\cdot\frac{n(n+1)}{2} - \frac{1}{2}n^3 - \frac{1}{2}n^2 + \frac{1}{2}\frac{n(n+1)}{2} + \frac{1}{2}\frac{n(n+1)(2n+1)}{6}$$

$$n^3 - \frac{n^3+n^2}{2} - \frac{n^3}{2} - \frac{n^2}{2} + \frac{n^2+n}{2} + \frac{n(n+1)(2n+1)}{6}$$

$$n^3 - \frac{n^3}{2} - \frac{n^2}{2} - \frac{n^3}{2} - \frac{n^2}{2} + \frac{n^2}{2} + \frac{n}{2} =$$

formula: $\sum_{i=1}^{n} 1 = n$

Ex
```
x=0
for i:1 to n do
    for j:1 to n do
    for k:1 to n do
        x i= x+1
```

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} 1 \qquad \sum_{i=1}^{n}\sum_{j=1}^{n} n = \sum_{i=1}^{n} n^2 = n^3$$

Ex
```
int i,x=0;
for (i=n*n; i<=n*n*n; i++){
    x++;
}
```

$$\sum_{i=n^2}^{n^3} 1 = n^3 - n^2 + 1 = f(n)$$

$f(n) \in O(n^3)$

Ex show that $5n$ belongs to $O(n)$

$5n \le cn$ $\quad k=1$

$\frac{1}{5} \le c$ $\quad n \ge 1$

$c = 6$

$5n \le 6n$

$5 \le 6$ $\quad \therefore f(n) \in O(n)$

* $\sum_{i=1}^{n} i \approx n^2$ * $\sum_{i=0}^{n-1} 2^i = 2^n - 1$ * $\sum_{i=1}^{n} i^2 \approx n^3$

Ex
```
for k=1 to n/2 do
{
---
}
for j=1 to n*n do
{
---
}
```

$$\sum_{k=1}^{n/2} 1 + \sum_{j=1}^{n^2} 1 = n/2 + n^2 = O(n^2)$$

Ex
```
for k=1 to n/2 do
{
    for j=1 to n*n do
    {
    ---
    }
}
```

$$\sum_{k=1}^{n/2}\sum_{j=1}^{n^2} 1 = \sum_{k=1}^{n/2} n^2$$

$$n^2 \sum_{k=1}^{n/2} 1 = n/2 \cdot n^2 = O(n^3)$$

Ex $f(n) = 4^n$, $g(n) = 8^n$ prove

$4^n \leq c \cdot 8^n$ for all $n \geq 2$ $\quad 4^n \leq \frac{1}{4} \cdot 8^n$ $\forall$ $n \geq 2$

$\frac{4^n}{8^n} \leq c$ for all $n \geq 2$ $\quad 1 \leq \frac{2^n}{4}$ $\forall$ $n \geq 2$

$\frac{1}{2^n} \leq c$ for all $n \geq 2$ $\quad 1 \leq 2^{n-2}$ $\{n-2 \geq 0\}$ $\{n \geq 2\}$

Choose $(c = \frac{1}{4}, k = 2)$ $\quad f(n) \in O(g(n))$

$4^n \in O(8^n)$

Ex $f(n) = 3n+1$, $g(n) = n$ Prove $f(n) = \Omega(g(n))$

$3n+1 \geq c \cdot n$

$3 + \frac{1}{n} \geq c$ $\quad c = 2$ choose

$k = 5$ choose

$3 + \frac{1}{n} \geq 2$

$3 + \frac{1}{5} \geq 2$ $\quad 3, 2 \geq 5 \rightarrow f(n) = \Omega(n)$

Ex prove 2^{n+1} is $O(2^n)$

$2^{n+1} \leq c \cdot 2^n$ $\quad 2^{n+1} \leq c \cdot 2^n$ $n \geq k$

$2 \leq c$ $\quad n \geq 1$

$2^{n+1} \leq 3 \cdot 2^n$ choose $k = 1$

$2^{n+1} \leq 2 \cdot 2^n$ for all $n \geq 1$ $\quad f(n) \in O(g(n))$ choose $c = 3$

Ex
```
int i, j, y, = 0;
for (i=1; i<= 2*n; i++) {
    for (j=1; j<=i; i++) {
        y = y+2
    }
}
```

$\sum_{i=1}^{2n} \sum_{j=1}^{i} 1 = \sum_{i=1}^{2n} i = \frac{2n(2n+1)}{2}$

$f(n) = \frac{4n^2 + 2n}{2} = 2n^2 + n$

Ex $f(n) = O(g(n))$

- if $f(n) \leq C \cdot g(n)$ for all $n \geq k$ when C are positive
- $\frac{f(n)}{g(n)} \leq C$ for all $n \geq k$

Show that $f(n) = n^2 + 2n + 1$ is $O(n^2)$

$$\frac{n^2+2n+1}{n^2} \leq C \quad \text{for all } n \geq 1$$

$$\frac{1+2+1}{1} \leq C \Rightarrow 4 \leq 4$$

$$\frac{n^2+2n+1}{n^2} \leq 4 \Rightarrow n^2+2n+1 \leq 4n^2 \quad \text{for all } n \geq 1$$

$$\Rightarrow f(n) = O(g(n)) = O(n^2)$$

Ex

```
for ( i=0 ; i<=n-1; i++ ) {
    for ( j=i+1; j<=n-1; j++ ) {
        loop body
    }
}
```

$$\sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} 1$$

$$\sum_{i=0}^{n-1} ((n-1) - (i+1) + 1) = n-1-i-1+1$$

$$\sum_{i=0}^{n-1} (n-1-i) = \sum_{i=0}^{n-1} n - \sum_{i=0}^{n-1} 1 - \sum_{i=0}^{n-1} i$$

$$\left(n^2 - n - \frac{n(n-1)}{2}\right) = \frac{2n^2 - 2n - n^2 + n}{2} = \frac{3n^2 - 3n}{2}$$

$f(n) = O(n^2)$

Ex $F(n) = 2n-2$ $g(n) = n$ Prove $F(n) = O(g(n))$

$2n-2 \leq C \cdot n$

Choose $k=1$, $C=3$

$2n-2 \leq 3n$

$2n \leq 3n+2$

$1 \leq \frac{3}{2} + \frac{1}{n}$ For all $n \geq 1$

$F(n) = O(n)$

summation formulas:

* $\sum_{i=1}^{n} ar^{i-1} = a\left(\frac{1-r^n}{1-r}\right)$ * $\sum_{i=m}^{n} c = c(n+1-m)$

* $\sum_{i=m}^{n} i = \frac{n(n+1)}{2} - \frac{m(m-1)}{2}$ * $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

* $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$ * $\sum_{i=0}^{n} x^i = \frac{(x^{n+1}-1)}{x-1}$

* $\sum_{0 \le i \le n} 2^i = \frac{2^{n+1}-1}{2-1} = 2^{n+1}-1$

Ex
```
int count = 0;
for(int i=3; i<n; i++)
    for(int j=0; j<i; j++)
        count++
```

$T(n) = \sum_{i=3}^{n-1} \sum_{j=0}^{i-1} c$

$= \sum_{i=3}^{n} (i-1-0+1).c$

$= \sum_{i=3}^{n-1} ci = c\sum_{j=3}^{n-1} i = c\left(\frac{(n-1)n}{2} - \frac{3.2}{2}\right) = c.\left(\frac{n^2-n-6}{2}\right)$

$T(n) = O(n^2)$

Ex
```
                                cost    times
i=1                             c1      1       c1+c2+c3 = c9
sum=0                           c2      1       c5+c8+c4+c8 = c10
while(i<=n)                     c3      n+1     c5+c6+c7 = c11
{
    j=1;                        c4      n       3c11n^2 + 4c10n + 3c9 = T(n)
    while(j<=n)                 c5      n(n+1)
    {                                           O(n^2)
        sum+=1;                 c6      n x n
        j=j+1;                  c7      n x n
    }
    i+=1;                       c8      n
}
```

insertion sort

```
* void InsertionSort(int[] arrNumbers)
  {
n+1   For(int i=1; i<arrNumbers.Length; i++)
      {
n         int j=i-1;
n         int key=arrNumbers[i];
nxn       while(j>=0 && key<arrNumbers[j])
          {
nxn           arrNumbers[j+1]=arrNumbers[j];
              j=j-1
          }
n         arrNumbers[j+1]=key;
2n²+4n+1 }
```

60, 34, 25, 12, 22 — Worst $O(n^2)$, Best $O(n)$

60, 60, 25, 12, 22
34, 60, 25, 12, 22
34, 60, 60, 12, 22
j=-1 34, 34, 60, 12, 22
i → 25, 34, 60, 12, 22
25, 34, 60, 60, 22
25, 34, 34, 60, 22
j=-1 25, 25, 34, 60, 22
→ 12, 25, 34, 60, 22
12, 25, 34, 60, 60
12, 25, 34, 60, 60
12, 25, 34, 34, 60
12, 25, 25, 34, 60
12, 22, 25, 34, 60

Selection Sort

مثل ترتيب ورق اللعبة

```
void SelectionSort(int[] arr)
{
    int minIndex=0;

    for(int i=0; i<arr.Length-1; i++)
    {   minIndex=i;
        for(int j=i+1; j<arr.Length; j++)
        {
            if(arr[minIndex] > arr[j])
            {
                minIndex = j;
            }
        }

        Swap(arr[j], arr[i]);
    }
```

Worst case $O(n^2)$
Best case $O(n^2)$

Bubble Sort

```
void BubbleSort(int[] arr)
{
    for(int i=0; i<arr.Length; i++){

        for(int j=0; j<arr.Length; j++){

            if(arr[j] > arr[j+1]){

                Swap(arr[j], arr[j+1])
            }
        }
    }
}
```

Best case and worst case $O(n^2)$

Alg1

```
int LinearSearch(int[] arrNumbers, int value)
{
    for(int i=0; i<arrNumbers.Length; i++)        n+1
    {
        if(arrNumbers[i] == value)                n
        {
            return i;                             1
        }
    }                                             f(n) = 2n+2
    return -1;                                    f(n) = O(n)
}
```

Best O(1), Avrage O(n), Worst O(n)

Alg2

```
int BinarySearch(List<int> lstNumbers, int value)
{
    int start = 0;
    int End = lstNumbers.Count;
    int mid;
    while(End >= start)                           2^k = n
    {                                             k = log n
        mid = (start + End)/2;                    O(log n) = f(n)
        if(lstNumbers[mid] == value)
        {                                         • Best O(1)
            return mid;                           • Worst O(log n)
        }                                         • Avrage O(log n)
        if(lstNumbers[mid] > value)
        {
            End = mid-1;
        }
        if(lstNumbers[mid] < value)
        {
            start = mid+1;
        }
    }
}
```

- algoritmaların doğrudan 10, 20 sayı için nasıl çalıştığı değil bizim için girdi sayısı ilerledikçe algoritmanın ne kadar hızlı değiştiği, ne kadar kötü duruma gittiği öğrenmek.

Ex	Time	cost
`i=1;`	1 unit of time	c_1
`sum=0;`	1	c_2
`while (i<=n) {`	$n+1$	c_3
`j=1;`	n	c_4
`while (j<=n) {`	$n(n+1) = n^2+n$	c_5
`sum = sum+i;`	n	c_6
`j = j+1;`	n	c_7
`}`		لتضرب معامل مشترك
`i = i+1;`	$n \times n$	بس مانو داخل
`}`		

$$2n^2+5n+3 = f(n) \rightarrow O(n^2) = f(n)$$

* Asimptotik Notasyonların karşılaştırması:

"Geçişlilik"

$f(n) = \Theta(g(n))$ ve $g(n) = \Theta(h(n))$ ise $f(n) = \Theta(h(n))$

(aynı kural O ve Ω için de)

"Dönüşlülük veya yansıma"

$f(n) = \Theta(f(n))$, $f(n) = O(f(n))$, $f(n) = \Omega(f(n))$

"simetri"

$g(n) = \Theta(f(n))$ olduğu durumda $f(n) = \Theta(g(n))$

- Transpose symmetry

$g(n) = \Omega(f(n))$ olduğu durumda $f(n) = O(g(n))$

$g(n) = \omega(f(n))$ olduğu durumda $f(n) = O(g(n))$

Ex
```
for (i=n/2; i<=n; i++)                    n/2
    for (j=1; j<=n; j=2*j)                logn
        for (k=1; k<=n; k=k*2)            logn
                stmt;
```
$f(n) = O[n.(\log n)^2]$

Ex
```
for (i=1; i<=n; i++)
    for (j=1; j<=n; j=j+i)
        stmt;
```

i	j	
1	1 → n	n/1 time
2	1 → n	n/2
3	1 → n	n/3
⋮		
n	n → n	n/n

$n(1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n})$

$\Downarrow$

$n(\log n) \Rightarrow f(n) = O(n \log n)$

Ex
```
int n=2^k
for (i=1; i<=n; i++) → n
{
    j=2;
    while (j<=n)
    {
        j=j^2
        stmt;
    }
}
```

k	n	j
1	4	2, 4
2	2^4	$2, 2^2, 2^4$
3	2^8	$2, 2^2, 2^4, 2^8$
⋮		
	2^k	$= n$

$\log n = 2^k$

$O(n.\log\log n) \Leftarrow \log\log n = k$

Ex
```
for (i=0; i<n; i++)                       n+1
{
    for (j=0; j<n; j++)                   n(n+1) = n^2+n
    {
        c[i,j] = 0;                       n×n = n^2
        for (k=0; k<n; k++)               n×n×(n+1) = n^3+n^2
        {
            c[i,j] = A[i,j]+B[i,j]        n×n×n
        }
    }
}
```

$2n^3+3n^2+2n+1 = f(n) \rightarrow O(n^3)$

Ex

```
int i,j,k,n;
for(i=1; i<=n; i++)
{
    for(j=1; j<=i; j++)
    {
        for(k=1; k<=100; k++)
        {
            stmt;
        }
    }
}
```

i	j	k
1	1	1x100
2	2	2x100
3	3	3x100
n	n	k = n x 100

$100\left(\frac{n(n+1)}{2}\right) \Rightarrow O(n^2)$

Ex

```
int i,j,k,n;
for(i=1; i<=n; i++)
{
    for(j=1; j<=i^2; j++)
    {
        for(k=1; k<=n/2; k++)
        {
            stmt;
        }
    }
}
```

i	j	k
1	1	$n/2 \cdot 1$
2	4	$n/2 \cdot 4$
3	9	$n/2 \cdot 9$
n	n^2	$n/2 \cdot n^2 = k$

$(1+4+9+\dots+n^2) = \frac{n}{2}\left(\frac{n(n+1)(2n+1)}{6}\right)$

$= O(n^4)$

Ex

```
for(i=1; i<n; i=i*2)
{
    stmt;
}
```

$i = 1, 2, 4, \dots 2^k$

$2^k = n \rightarrow O(\log n)$

Ex

```
for(i=n/2; i<=n; i++)          n/2
    for(j=1; j<=n/2; j++)      n/2
        for(k=1; k<=n; k=k*2)  log n
            stmt;
```

$O(n^2 \log n)$

Ex

```
i=0                 1
while (i<n)         n+1
{
    stmt;           n
    i++;            n
}
```

$f(n) = 3n+2$

$F(n) = O(n)$

Ex

```
a=1                 2
while (a<b)         1
{
    stmt;           2^2
    a=a*2;          2^3
}                   1
```

$2^k \geq b \Leftarrow 2^k$

$k = \log_2 b \Rightarrow f(n) = O(\log b)$

Ex

```
i=1;                i      k
k=1;                1      1+2
while (k<n)         2      1+2+3
{                   3      1+2+3+4
    stmt;           4      ----
    k=k+i;          i
    i++;            m      1+2---m+m+1
}
```

$\frac{m(m+1)}{2} + m + 1 \geq n$

$m^2 \geq n$

$m = \sqrt{n}$

$f(n) = O(\sqrt{n})$

```
for(i=1,k=1; k<n; i++)
{
    k=k+i;
}
```

Ex

```
while (m != n)
{
    if (m>n)
        m = m-n;
    else
        n = n-m;
}
```

	m=16	n=2
1	16	2
2	14	2
3	12	2
4	10	2
5	8	2
6	6	2
7	4	2
8	2	2

$\Rightarrow 16/2 = 8 \Rightarrow n/2$

$O(n)$

Ex

```
Algorithm test(n)
{
    if(n<5)
        print(n);          -> O(1)
    else
        for(i=0; i<n; i++)   1+1+1+...+1 = n+1   O(n)
            print(i);
}
```

Best: O(1) worst O(n)

Ex

```
p=0
for(i=1; i<n; i=i*2)   1, 2, 2^2, 2^3 ... 2^k >= n   O(logn)
{
    p++;        → O(logn)        p = logn
}

for(j=1; j<p; j=j*2)   1, 2, 2^2 ... 2^m >= p   m = log p
{
    stmt;  O(log log n)
}
```

Ex
```
for(i=0; i<n; i++)                (n+1)      ⇒ 2n.logn + n + 1
{
    for(j=1; j<n; j=j*2)          n.logn     ⇒ O(n.logn)
    {
        stmt;                     n.logn
    }
}
```

- for(i=0; i<n; i++) → O(n)
- for(i=0; i<n; i+=2) → n/2 → O(n)
- for(i=n; i>1; i--) → O(n)
- for(i=1; i<n; i=i*2) → O(log₂ n)
- for(i=1; i<n; i=i*3) → O(log₃ n)
- for(i=n; i>1; i=i/2) → O(log₂ n)
- for(i=n/2; i<n; i++) → n/2 → O(logn)

```
p=0
Ex for(i=1; p<=n; i++)
{
    p=p+i
}
```

i	p
1	1
2	1+2
3	1+2+3
:	
n	1+2+---+k

$p > n$

$\frac{k(k+1)}{2} > n$

$k^2 > n$

$k > \sqrt{n}$

$F(n) = O(\sqrt{n})$

```
Ex for(i=1; i<n; i=i*2)
{
    stmt;
}
```

i
1
1x2
1x2x2
:
2^k

$2^k \geq n$

$k \geq \log n$

$k = \log n$

$O(\log n) = f(n)$

```
Ex for(i=1; i<n; i++)
{
    stmt;
}
```

$i = 1+1+1+1 \text{---} = 1 \times n = k$

$\log(n)$

```
Ex for(i=n; i>=1; i=i/2)
{
    stmt;
}
```

i
n
$n/2$
$n/4$
:
$n/2^k$

$n < \frac{1}{2^n}$

$k = \log n \rightarrow O(\log n)$

```
Ex for(i=0; i*i<n; i++)
{
    stmt;
}
```

i	
1	1 → n
2	4 → n
3	9 → n
i	
	$k^2 \rightarrow n$

$i \times i = i^2 \geq n$

$i^2 = n$

$i = \sqrt{n}$

$O(\sqrt{n})$

```
Ex for(i=0; i<n; i++)
{
    stmt;          → n
}
for(j=0; j<n; j++)
{
    stmt;          → n
}
```

* Frequency time method

Arithmetic operations → 1 unit of time (+, −, ×, /)
Assignment operator → 1 unit of time (x=5)
Comprasion → 1 unit of time (x>y)
return statement → 1 unit of time

Ex return a+b; O(1)

Ex
```
i=1;                1          i
n=10;               1          1
while(i<n)          n          1x2
{                               1x2x2
    i*=2 → 2^k                  :
}                               2^k
```
$2^k \geq n$
$2^k = n$
$k = \log n$

⇒ f(n) = log n + 2
f(n) = O(log n)

Ex
```
sum=0
for(i=0 ; i<n ; i++) {      n+1
    sum = sum + i;          n
}
```
⇒ 2n+2 = f(n)
f(n) = O(n)

Ex
```
for(i=0; i<n; i++)
{
    For(j=0; j<i; j++)
    {
        stmt;
    }
}
```

i	j	time
0	0*	0
1	0<	1
	1*	
2	0<	2
	1<	
	2*	
	:	
n	0<	n
	n*	

$0+1+2+\dots+n = \frac{n(n+1)}{2}$

$f(n) = \frac{n^2}{2} + \frac{n}{2}$ $\quad f(n) = O(n^2)$

Ex prove that $2n^2+3n+6 \neq O(n^3)$

$c_1n^3 \leq 2n^2+3n+6 \leq c_2n^3$ $\quad 2n^2+3n+6 \leq c_2n^3$ ←

$c_1n^3 \leq 2n^2+3n+6$
Hence this is not possible
and $c_1n^3 \leq 2n^2+3n+6$ for $n \geq n_0$ is not true for any

if one side disproved so no need to prove other side

Ex prove that $2n^2+5n+6 = O(n^2)$ ($f(n)$, $g(n)$)

$2n^2+5n+6 \leq c\,g(n)$
$2n^2+5n+6 \leq 2n^2+5n^2+6n^2$
$2n^2+5n+6 \leq 13n^2$
$c=13 \quad n_0=1$

Ex prove that $2n+10 = O(n)$

$f(n) \leq c.g(n)$
$2n+10 \leq c.n$
$10 \leq cn-2n$
$10 \leq n(c-2)$
$\frac{10}{c-2} \leq n$

$c \geq 3, \quad n_0=10$

$2n+10 \leq 12n$
$c=12 \quad n=1$

Ex prove that $3n+2 = \Omega(n)$

$3n+2 \geq c\,g(n)$
$3n+2 \geq cn$
$2 \geq n(c-3)$
$\frac{2}{(c-3)} \geq n$

$n_0=2 \quad c=4$

Ex $5n^2+2n-3 = \Omega(n^2)$

$f(n) \geq g(n).c$
$c.n^2 \geq 5n^2+2n-3$
We need to prove $2n-3 \geq 0$
$2n-3 \geq 0 \quad n \geq \frac{3}{2} \Rightarrow n \geq 2$

$c=5, n_0=2$ proved

Ex prove that $5n^2 = \Omega(n)$

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

$5n^2 \geq nc$
$5n \geq c$
$n_0=1 \quad c=5$
proved

Ex $100n+5 \stackrel{?}{=} \Omega(n^2)$

$f(n) \geq c\,g(n)$
$100n+5 \geq c.n^2$ it is wrong
Hence this is not possible and $cn^2 \leq 100n+5$ for $n \geq n_0$ is not true for any combination of c and n_0.

Ex prove that $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

$c_1 n^2 \le \frac{1}{2}n^2 - 3n \le c_2 n^2$

$c_1 \le \frac{1}{2} - \frac{3}{n} \le c_2$

$\frac{1}{2} - \frac{3}{n} \ge c_1 \quad , n \ge 7 \quad , c_1 \le \frac{1}{14}$

$\frac{1}{2} - \frac{3}{n} \le c_2 \quad , n \ge 1 \quad , c_2 \ge \frac{1}{2}$

Ex $T(n) = n^2 + n \ge cn^2$?

$n^2 + n \ge cn^2$

$n^2 + n \ge c$ — $c=1$ olsun

$n=1$ olsun

$1^2 + 1 \ge 1$

$2 \ge 1$ $\quad n \ge n_0$

- $T(n)$ هو وقت تنفيذ الخوارزمية
- $g(n)$ هو دالة حد أعلى لتنفيذ خوارزمية
- c ثابت يعبر عن عامل التشبيه
- k, n_0 أعداد حقيقية يجب أن يكون n أكبر منها
- $n = g(n)$

Ex $T(n) = 2n^2 + 5n + 3$

"$O(n)$"

$T(n) \le c f(n) \quad n \ge n_0$

$n^2 = f(n), \quad c \ge 2, \quad n_0 \ge 1$

$T(n) \le cf(n)$ لتحقق $\Leftarrow O(n) = n^2, \ c \le 2,$

$T(n) = O(n^2)$ $\quad n_0 = 1$ iqin $n \ge 1$

"$\Omega(n)$"

$T(n) \ge cg(n)$

$2n^2 + 5n + 3 \ge cg(n)$

$T(n) = \Omega(n^2)$

"$\Theta(n)$"

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

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

Ex prove that $\frac{1}{2}n^2 - \frac{1}{2}n = \Theta(n^2)$

$c_1 n^2 \le \frac{1}{2}n^2 - \frac{1}{2}n \le c_2 n^2$

for lower bound;

$\frac{n^2}{2} - \frac{n}{2} \ge c_1 n^2$

if we assume $c_1 = \frac{1}{4}$

$\frac{n^2}{2} - \frac{n}{2} \ge \frac{1}{4}n^2$

$c_1 = \frac{1}{4} \quad n_0 = 2$

proved //

* for upper bound

$\frac{n^2}{2} - \frac{n}{2} \le c_2 n^2$

$c = \frac{1}{2} \Rightarrow \frac{1}{2}n^2 - \frac{1}{2}n \le \frac{n^2}{2}$

$n \ge 1 \Rightarrow c_2 = \frac{1}{2} \quad n_0 = 1$

proved //

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

02-04-2024

Ex $f(n) = 2n^2 + 3n + 4$

$2n^2 + 3n + 4 \geq 1 \times n^2$

$1 \times n^2 \leq 2n^2 + 3n + 4 \leq 9n^2$

$O(n^2), \Omega(n^2), \Theta(n^2)$

Ex $f(n) = n^2 \log n + n$

$1 \times n^2 \log n \leq n^2 \log n + n \leq 10 n^2 \log n$

$O(n^2 \log n), \Omega(n^2 \log n), \Theta(n^2 \log n)$

Ex $f(n) = n!$

$n! = n(n-1)(n-2) \cdots 3.2.1$

$1 \times 1 \times 1 \cdots \times 1 \leq 1 \times 2 \times 3 \cdots \times n \leq n \times n \times n \cdots \times n$

$\Omega(1), O(n^n)$

- we cannot say that $n!$ is always greater than n^{10} and less than n^{11} $\quad n^{10} < n! < n^{11}$

Ex $f(n) = \log n!$

$\log(1 \times 1 \times \cdots \times 1) \leq \log(1 \times 2 \times \cdots \times n) \leq \log(n \cdot n \cdots n)$

$1 \leq \log n! \leq \log n^n$

$1 \leq \log n! \leq n \log n$

$\Omega(1), O(n \log n)$

Ex $f(n) = 2n + 5 \in \Theta(n)$

$2n \leq 2n + 5 \leq 2n + 5n$

$2n \leq 2n + 5 \leq 8n$

for all $n \geq 1$ is given

Ex $f(n) = 5n^2 - 3n \in \Theta(n^2)$

$4n^2 \leq 5n^2 - 3n \leq 5n^2$

for all $n \geq 4$ is given

Ex $\frac{n^2}{2} - \frac{n}{2} \stackrel{?}{\in} \Theta(n^2)$

$\frac{n^2}{2} - \frac{n}{2} \leq \frac{n^2}{2} \quad (\forall n \geq 0) \quad c_2 = \frac{1}{2}$

$\frac{n^2}{2} - \frac{n}{2} \geq \frac{n^2}{2} - \frac{n}{2} \cdot \frac{n}{2} \quad (\forall n \geq 2) \quad c_1 = \frac{1}{4}$

Ex $n \neq \Theta(n^2)$: $c_1 n^2 \leq n \leq c_2 n^2$ only holds for : $n \leq 1/c_1$

$6n^3 \neq \Theta(n^2)$ $\quad c_1 n^2 \leq 6n^3 \leq c_2 n^2 \Rightarrow$ " " " : $n \leq c_2/6$

02-04-2024

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 \dots < 2^n < 3^n < n^n$

* Omega "Ω"

$f(n) \geq c * g(n) \quad \forall \; n \geq n_0$

Ex $f(n) = 2n+3$ $\quad f(n) \in \Omega(n)$

$2n+3 \geq 1n$ $\quad \| \quad$ $2n+3 \geq 1 \times \log n$ $\quad$ $2n+5 \geq n$

$n_0 \geq 0$ için

$P(n)$ c $g(n)$ $\quad$ $f(n)$ c $g(n)$

$c = 1$, $n_0 = 0$

$F(n) = \Omega(n)$ ✓

$f(n) = \Omega(\log n)$ ✓

$f(n) = \Omega(n^2)$ ✗

Ex $5n^2 - 3n = \Omega(n^2)$ olduğunu gösterin.

$n_0 \geq 0$ $\quad 5n^2 - 3n \geq n^2$

$c = 1$

$n_0 = 0$ değerleri için sağlar.

Ex $5n^2 = \Omega(n)$

$5n^2 \geq n$

$g(n) = n$

$c = 1$

$f(n) = 5n^2$

$n_0 = 0$

Ex $100n + 5 \neq \Omega(n^2)$

$0 \leq cn^2 \leq 100n + 5$

$100n + 5 \leq 100n + 5n \quad n \geq 1$

$cn^2 \leq 105n$

$n(cn - 105) \leq 0 \quad n = 0$

$n = 105/c$

n can't be smaller than a constant (n_0)

Theta Θ: $\quad c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

$f(n) = 2n + 3$

$1n < 2n+3 \leq 5n \quad f(n) = \Theta(n)$

$c_1 g(n) \quad f(n) \quad c_2 g(n)$

Ex $f(n) = 2n^2 + 3n + 4 \quad f(n) \stackrel{?}{=} \Theta(n)$

$2n^2 + 3n + 4 \leq 2n^2 + 3n^2 + 4n^2$

$2n^2 + 3n + 4 \leq 9n^2 \quad (n_0 \geq 1)$

$c \; g(n) \Rightarrow f(n) = O(n^2)$

02-04-2024

lower bound — upper bound

"$1 < \log n < \sqrt{n} < (n) < n\log n < n^2 < n^3 < \dots < 2^n < 3^n < n^n$"

avg bound

- $O(1)$ constant
- $O(n)$ linear
- $O(n^2)$ Quadratic
- $O(n^3)$ cubic
- $O(\log n)$ logrithemic

Ex $f(n) = 2n+3$ $\quad f(n) = O(n)$?

$2n+3 \leq 2n+3n$

$2n+3 = 2n+3n$

$3 \leq 3n$

$n \geq 1$

$\rightarrow f(n) = O(n)$

$f(n) = O(n^2)$

- $f(n) = 5$
- $f(n) = 300$ → $O(1)$
- $f(n) = 1$

- $f(n) = n+5000$
- $f(n) = n+3$ → $O(n)$
- $f(n) = 2n+3$
- $f(n) = \frac{n}{50} + 20$

Ex $2n^2 = O(n^3)$ için c, n_0 değerlerini bulunuz. ($n_0 \geq 1$)

$f(n) \leq c \cdot g(n) \rightarrow 2n^2 \leq cn^3$

$c=1$ için $n_0 = 2$ şartı sağlar.

Ex $\frac{1}{2}n^2 + 3n$ için üst sınırın $O(n^2)$ olduğunu gösteriniz. ($n_0 \geq 1$)

$c=1$ için: $\frac{1}{2}n^2 + 3n \leq n^2$

$3n \leq \frac{1}{2}n^2$

$6 \leq n$

$n_0 = 6$

Çözüm kümesini sağlayan kaç tane n_0 ve c çift olduğu önemli değildir, tek bir çift olması notasyonun doğruluğu için yeterlidir.

Ex $3n^2 + 2n + 5 = O(n^2)$ olduğunu gösteriniz. ($n_0 \geq 1$)

$3n^2 + 2n + 5 = 3n^2 + 2n^2 + 5n^2$

$8n^2 + 2n + 5 = 10n^2$

$8n^2 + 2n + 5 \leq 10n^2$

$c = 10 \quad n_0 = 1$

*Bir fonksiyonun Büyümesi (Growth)

- Fonksiyonların büyüme hızı matematikte Big O olarak ifade edilen fonksiyon gösterir.

Tanım: f ve g fonksiyonları reel sayılardan reel sayılara tanımlı iki fonksiyon olsun.

f(x) fonksiyonu için C ve k sabit olmak üzere O(g(x)) aşağıdaki şeklinde tanımlanır.

$f(x) \leq C * g(x) + k$
$x > k$ olmak üzere } $f(n) \leq C * g(n) \ \forall \ n \geq n_0$

Ex $f(x) = x^2 + 2x + 1$ fonksiyonun büyüme fonksiyonun x^2 olduğunu yani $(O(n^2))$ olduğunu gösteriniz.

$x > 1$ için (bu aynı zamanda $x > k$):

$x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2$

$x^2 + 2x + 1 \leq 4x^2$ $(C = 4)$, $(k = 1)$ → $x^2 + 2x + 1 = 4x^2$, $x_0 = k = 1$

$f(x) \leq cx^2 + k$, $x > k$ durumu sağlanır.

"C" إما نفرضه أو نعوض بأعلى أس ونجمع

$\Rightarrow f(x)$ is $O(x^2)$

* Definitions:

$f(n) = O(g(n))$ * $f(n) \leq n \cdot g(n) \ \forall \ n \geq n_0$

* $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$ * $f(x) \leq n \cdot g(x) + k \ \forall \ x > k$

Ex $f(n) = 2n + 6$

$f(n) = O(n)$?

$f(n) = O(g(n))$

- $g(n) = n$
- $c = 4$ نفرض

$2n + 6 \leq 4 \cdot n$

$6 = 2n$

$n = 3 \Rightarrow n \geq 1$

$c > 4$

Ex $f(n) = 2n + 3$

$f(n) \stackrel{?}{=} O(n)$

$2n + 3 \geq 10n$

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

$f(n) = 2n + 3$

$f(n) \stackrel{?}{=} O(n)$

$2n + 3 \leq 2n + 3n$

$3 \leq 3n$

$n \geq 1$

$f(n) = O(n)$

ممكن أيضاً $f(n) = O(n^2)$

$2n + 3 \leq 2n^2 + 3n^2$

$2n + 3 \leq 5n^2$

02-04-2024

Algoritma nedir? Bir problemin çözmek için bir prosedür veya formüldür.
- problemi çözmek için takip edilmesi gereken yönergeler kümesidir

Program? Bir programlı dilinde algoritmanın gerçekleştirilmesidir.

Algoritmaların özellikleri
Input: Girdi, bir kümedir.
Output: çıktı, bir kümedir.
Definiteness: kesinlik (algoritmanın adımlarının belirli olması)
Correctness: Doğruluk (bütün girdiler için algoritma tanımlı olması)
Finiteness: sonluluk (çalışma adımlarının sonlu olması)
Effectiveness: Verimlilik (Her adımın ve sonuca ulaşan yolun)
Generality: Genellenebilirlik (bir problem kümesi için)

What Big O is not?
* Big O is not going to give you an exact answer on how long a piece of code will take to run.
* Does not consider the performance of hardware.

* An algorithm's space complexity: spesifies the total amount of space or memory required to excute an algorithm as a function of the size of the input
* An algorithm's time complexity: spesifies how long it will take to excute an algorithm as a function of the input size.

* Things Affect program speed and Efficiency