4.2.1 Big-O Notasyonu

- → algoritmanın etkinliğini yürütme zamanı cinsinden ifade edeceğiz
- → yürütme zamanı = (algoritmanın gereksinim duyduğu) işlem sayısı
- → veya adım sayısı
- → temel işlem birimi nedir?

sumOfN - v1

→ toplama örneği

→ temel işlem birimi = atama

```
1        def sumOfN(n):
2            sum = 0
3            for i in range(1,n+1):
4                sum = sum + i
5
6            return sum
```

→ as2: sum = 0 $\Longrightarrow$ 1

→ s4: sum = sum + i $\Longrightarrow$ n

→ sonuçta 1 + n atama işlemi

Amacımız

- → problemin boyutu: $T(n) = 1 + n$
- → n boyutlu bir problem $1 + n$ adımda sonuç üretir.
- → problem boyutu (n) ile yürütme zamanı arasındaki ilişkiyi çıkartmak
- → bilg.bil.'de bunu bir adım öte götürmüşler
- → işlem sayısı, T(n)'deki baskın parça
- → T(n) ifadesindeki **baskın** bileşen nedir?
- → karşılaştırmalarda bu baskın parça kullanılır

genliğin derecesi (order of magnitude)

→ n arttıkça T(n)'nin en hızlı artan parçasını tanımlar

→ genliğin derecesi kısaca

genliğin derecesi (order of magnitude)

→ n arttıkça T(n)'nin en hızlı artan parçasını tanımlar

→ genliğin derecesi kısaca **Big-O** notasyonu olarak adlandırılır

genliğin derecesi (order of magnitude)

- → n arttıkça T(n)'nin en hızlı artan parçasını tanımlar
- → genliğin derecesi kısaca **Big-O** notasyonu olarak adlandırılır
- → O(f(n)) biçiminde gösterilir
- → burada f(n), T(n)'nin baskın parçası
- → Ör. T(n) = 1 + n ise

genliğin derecesi (order of magnitude)

- → n arttıkça T(n)'nin en hızlı artan parçasını tanımlar
- → genliğin derecesi kısaca **Big-O** notasyonu olarak adlandırılır
- → O(f(n)) biçiminde gösterilir
- → burada f(n), T(n)'nin baskın parçası
- → Ör. $T(n) = 1 + n$ ise $f(n) = n$

genliğin derecesi (order of magnitude)

- → n arttıkça T(n)'nin en hızlı artan parçasını tanımlar
- → genliğin derecesi kısaca **Big-O** notasyonu olarak adlandırılır
- → O(f(n)) biçiminde gösterilir
- → burada f(n), T(n)'nin baskın parçası
- → Ör. T(n) = 1 + n ise f(n) = n
- → Ör. T(n) = 5n^2 + 27n + 1005 ise

genliğin derecesi (order of magnitude)

- → n arttıkça T(n)'nin en hızlı artan parçasını tanımlar
- → genliğin derecesi kısaca **Big-O** notasyonu olarak adlandırılır
- → O(f(n)) biçiminde gösterilir
- → burada f(n), T(n)'nin baskın parçası
- → Ör. T(n) = 1 + n ise f(n) = n
- → Ör. T(n) = 5n^2 + 27n + 1005 ise f(n) = n^2

genliğin derecesi (order of magnitude)

- → n arttıkça T(n)'nin en hızlı artan parçasını tanımlar
- → genliğin derecesi kısaca **Big-O** notasyonu olarak adlandırılır
- → O(f(n)) biçiminde gösterilir
- → burada f(n), T(n)'nin baskın parçası
- → Ör. T(n) = 1 + n ise f(n) = n
- → Ör. T(n) = 5n^2 + 27n + 1005 ise f(n) = n^2

Big-O

- T(n) işlevi için, T(n) <= c0*f(n), n>=n0 (c0 ve n0 sabit)
- varsa O(f(n))'dir

Big-O

- → T(n) işlevi için, T(n) <= c0*f(n), n>=n0 (c0 ve n0 sabit)
- → varsa O(f(n))'dir
- → ör. T(n)= 3n + 2, 3n+2 <= 4*n, n>=2, burada f(n)= n (c0=4, n0=2)
- → dolayısıyla O(f(n)) $\Longrightarrow$ O(n)'*dir*

→ Ör. T(n)= 2n^2 + 6n + 3 için O(f(n))?

Big-O

- → T(n) işlevi için, T(n) <= c0*f(n), n>=n0 (c0 ve n0 sabit)
- → varsa O(f(n))'dir
- → ör. T(n)= 3n + 2, 3n+2 <= 4*n, n>=2, burada f(n)= n (c0=4, n0=2)
- → dolayısıyla O(f(n)) $\Longrightarrow$ O(n)'*dir*

→ Ör. T(n)= 2n^2 + 6n + 3 için O(f(n))? O(n^2)

yaygın Big-O'lar

f(n)	Name
1	Constant
$\log n$	Logarithmic
n	Linear
$n \log n$	Log Linear
n^2	Quadratic
n^3	Cubic
2^n	Exponential

Table 4.1: Common Functions for Big-O

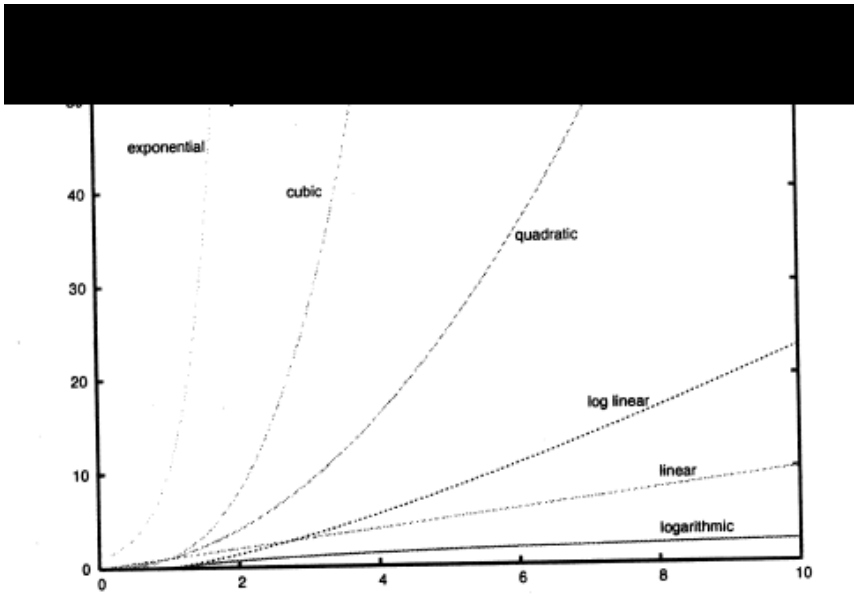

Figure 4.1: Plot of Common Big-O Functions

örnek

→ örnek

```
1       a=5
2       b=6
3       c=10
4       for i in range(n):
5           for j in range(n):
6               x = i * i
7               y = j * j
8               z = i * j
9       for k in range(n):
10          w = a*k + 45
11          v = b*b
12      d = 33
```

→ s1-3: 3

→ s6-8: 3 x n^2

→ s10-11: 2 x n

→ s12: 1

→ sonuçta: 3 + 3xn^2 + 2xn + 1

→ T(n)?

örnek

→ örnek

```
1       a=5
2       b=6
3       c=10
4       for i in range(n):
5           for j in range(n):
6               x = i * i
7               y = j * j
8               z = i * j
9       for k in range(n):
10          w = a*k + 45
11          v = b*b
12      d = 33
```

→ s1-3: 3

→ s6-8: 3 x n^2

→ s10-11: 2 x n

→ s12: 1

→ sonuçta: 3 + 3xn^2 + 2xn + 1

→ T(n)?=3n^2 + 2n + 4

→ O(f(n))?

örnek

→ örnek

```
1       a=5
2       b=6
3       c=10
4       for i in range(n):
5           for j in range(n):
6               x = i * i
7               y = j * j
8               z = i * j
9       for k in range(n):
10          w = a*k + 45
11          v = b*b
12      d = 33
```

→ s1-3: 3

→ s6-8: 3 x n^2

→ s10-11: 2 x n

→ s12: 1

→ sonuçta: 3 + 3xn^2 + 2xn + 1

→ T(n)?=3n^2 + 2n + 4

→ O(f(n))? O(n^2)