Büyük O Notasyonu
(Big O Notation)

Algoritma Analizi Nedir

- Yazdığımız bir algoritmanın doğru çalıştığından emin olmakla birlikte bu algoritmayı, daha önce yazılmış ve aynı sonucu veren başka algoritmalarla karşılaştırmak isteyebilirsiniz.

- Burada teknik olarak değerlendirilecek başlıca iki başlık söz konusudur.

- Birincisi algoritmaların bellek kullanım miktarı, ikincisi ise algoritmaların hesaplama yapmak için harcadığı süredir.

- Mesela bir algoritma aynı işi yapan diğer bir algoritmadan daha hızlı çalışmasına rağmen çoğu bilgisayar için bellek aşımı gerçekleştiriyorsa bu pek uygun olmayacaktır.

Algoritma Analizi Nedir

- Elbette diğer algoritmalarla karşılaştırma yapmak yerine, bir algoritmanın tek başına analizi de yapılabilir.

- Bunun için algoritmalar tek tek çalıştırılıp üzerinde hız ve bellek testi yapılabilir.

- Ama bu işlem hem zaman açısından sıkıntı yaratır hem de elde edilen veriler donanımsal ve sistemsel değişikliklerden dolayı bilimsel olmaz.(Bu gibi işlemleri performans testi olarak da düşünebiliriz) .

Algoritma Analizi Nedir

- Bu durumda matematiksel olarak ifade edebileceğimiz, donanımsal ve sistemsel bağımlılığı olmayan bir yönteme ihtiyacımız olacaktır.

- Bu yöntemle algoritmamıza girdi olarak verilen verilerin miktarına bağlı olarak sonuçlar üretiriz.

- İşte elde edilen bu sonuçlar ilgili algoritmanın *karmaşıklığı* olarak tanımlanır.

- Bir algoritmanın karmaşıklığı performansını etkiler ama karmaşıklık ile performans farklı kavramlardır.

Büyük O Notasyonu

- O notasyonu ilk olarak 1894 yılında Alman matematikçi Bachmann tarafından kullanılmış ve Landau tarafından da yaygınlaştırılmıştır.

- Bu yüzden adına Landau notasyonu veya Bachmann–Landau notasyonu da denmektedir.

- Algoritmanın en kötü durum analizini yapmak için kullanılan notasyondur. Matematiksel olarak şöyle tanımlanır:

Büyük O Notasyonu

f(x) ve g(x) reel sayılarda tanımlı iki fonksiyon olmak üzere, x > k olacak şekilde bir k vardır öyle ki,

|f(x)| < C*|g(x)| dir ve $f(x) \in O(g(x))$ şeklinde gösterilir. Burada C ve k sabit sayılardır ve x'ten bağımsızdırlar.

Büyük O Notasyonu

$$\begin{aligned} |5x^3 - 2x^2 + 3| &\leq 5x^3 + |2x^2| + 3 \\ &\leq 5x^3 + 2x^3 + 3x^3 \\ &\leq 10x^3 \\ &\leq 10|x^3| \end{aligned}$$

Burada k = 1 (x'in 1'den büyük olduğu tüm durumlarda) ve C = 10 olarak alınmıştır.

Büyük O Notasyonu

- Sabit zamanlı ifadeler O(1) ile gösterilirler. Örnek, atama işlemleri.

- if - else ifadelerinde, ifadenin if veya else bloğundaki hangi ifade karmaşıklık olarak daha büyükse O fonksiyonu o değeri döndürür. (Çünkü biliyorsunuz ki O fonksiyonu her zaman en kötü durumu analiz eder) Yani bunu şöyle ifade edebiliriz:

Maks (*if ifadesinin çalışma zamanı, else ifadesinin çalışma zamanı***)**

Örneğin if bloğu içi O(1) else bloğunun içi O(n) ise if – else bloğu O(n) olarak ele alınır.

Büyük O Notasyonu

```
//aşağıdaki if-else ifadesi O(n)'dir
if (ifade)
{
   //birinci ifade O(1) olsun


}
else
{
   //ikinci ifade O(n) olsun
}
```

Büyük O Notasyonu

- Bir döngü ifadesinin içindeki bir ifade, döngünün dönme sayısı kadar çalışacağı için, eğer döngü N kez dönüyorsa ve döngü içindeki ifadenin çalışma zamanı C ise, toplam çalışma zamanı N*C’dır.

```
//aşağıdaki for döngüsü O(C.N)'dir.
for (int i = 0; i < N; i++)
{
   //buradaki ifade C zamanda çalışsın
}
```

Büyük O Notasyonu

- İç içe döngülerde içteki döngü N kez, dıştaki döngü ise K kez dönüyorsa ve iç döngünün içindeki ifadenin çalışma zamanı C ise, toplam çalışma zamanı N*K*C'dir.

```
//aşağıdaki for döngüsü O(C.N.K)'dir.
for (int i = 0; i < N; i++)
{
   for (int j = 0; j < K; j++)
   {
      //buradaki ifade C zamanda çalışsın
   }
}
```

Büyük O Notasyonu

Notasyon	**İsim**	**Açıklama**
O(1)	Sabit	Algoritmadaki icra sayısı belliyse sabit bir değerle gösterlir. Örneğin, bir sayının tek mi çift mi olduğunun bulunması.

Büyük O Notasyonu

O(logn)	Logaritmik	n değerinin büyüyen değerlerine karşın algoritmanız çok daha az yavaşlıyorsa logaritmik bir durum söz konusudur. Örneğin, binary search ile sıralı bir dizide değer aramak.
O(n)	Lineer	n değerinin büyümesine karşılık algoritmanın lineer bir şekilde yavaşlaması söz konusudur. Örnek, sırasız bir listeden bir değeri bulmak.

Büyük O Notasyonu

O(n logn)	Loglineer	Bir problemi alt problemlere bölüp bağımsız olarak çözen, daha sonra bu sonuçları birleştiren algoritmalarda görülür. Örnek, birliştirmeli sıralama (merge sort) algoritması.
O(n^2)	Karesel	İç-içe döngüler ile verileri ikişerli şekilde inceleyen algoritmalarda görülür. Örnek, seçmeli sıralama (selection sort)
O(2^n)	Üstel	Girilen veriye göre iki kat yavaşlama görülen bu algoritmalar hiç pratik değildir. Örnek, seyyar satıcı problemi.

Büyük O Notasyonu

n	$\log n$	n	$n \log n$	n^2	2^n	$n!$
10	3×10^{-9} s	10^{-8} s	3×10^{-8} s	10^{-7} s	10^{-6} s	3×10^{-3} s
10^2	7×10^{-9} s	10^{-7} s	7×10^{-7} s	10^{-5} s	4×10^{13} yr	*
10^3	$1(0 \times 10^{-8}$ s	10^{-6} s	1×10^{-5} s	10^{-3} s	*	*
10^4	$1(3 \times 10^{-8}$ s	10^{-5} s	1×10^{-4} s	10^{-1} s	*	*
10^5	$1(7 \times 10^{-8}$ s	10^{-4} s	2×10^{-3} s	10 s	*	*
10^6	2×10^{-8} s	10^{-3} s	2×10^{-2} s	17 min	*	*

n boyutlu problemin çeşitli algoritmalarla çözüm hızı

Büyük O Notasyonu

Dizideki sayıların toplamını bulma

İşlem sayısı

```
int Topla(int A[], int N)
{
  int topla = 0;                 --> 1

  for (i=0; i < N; i++){         --> N
    topla += A[i];               --> N
  } //Bitti-for

  return topla;                  --> 1
} //Bitti-Topla
```

Toplam: $1 + N + N + 1 = 2N + 2$

- Çalışma zamanı: $T(N) = 2N+2$
  - N dizideki sayı sayısı

Büyük O Notasyonu

- Bilgisayar bilimlerinde bir algoritmanın incelenmesi sırasında sıkça kullanılan bu terim çalışmakta olan algoritmanın en kötü ihtimalle ne kadar başarılı olacağını incelemeye yarar.
- Bilindiği üzere bilgisayar bilimlerinde yargılamalar kesin ve net olmak zorundadır.
- Tahmini ve belirsiz karar verilmesi istenmeyen bir durumdur.
- Bir algoritmanın ne kadar başarılı olacağının belirlenmesi de bu kararların daha kesin olmasını sağlar.
- Algoritmanın başarısını ise çalıştığı en iyi duruma göre ölçmek yanıltıcı olabilir çünkü her zaman en iyi durumla karşılaşılmaz.

Büyük O Notasyonu

- Algoritma analizinde kullanılan en önemli iki ölçü hafıza ve zaman kavramlarıdır.

- Yani bir algoritmanın ne kadar hızlı çalıştığı ve çalışırken ne kadar hafıza ihtiyacı olduğu, bu algoritmanın performansını belirleyen iki farklı boyuttur.

- En iyi algoritma en hızlı ve en az hafıza ihtiyacı ile çalışan algoritmadır. İşte en kötü durum analizi olayın bu iki boyutu için de kullanılabilir.

- Yani en kötü durumdaki hafıza ihtiyacı ve en kötü durumdaki hızı şeklinde algoritma analiz edilebilir.

Büyük O Notasyonu

- Limit teorisindeki master teoremde büyük O ile gösterilen (big-oh) değer de bu en kötü durumu analiz etmektedir.

- Bu yüzden en kötü durum analizine, büyük O gösterimi (Big-O notation) veya algoritmanın sonsuza giderken nasıl değiştiğini anlatmak amacıyla büyüme oranı (growth rate) isimleri verilmektedir.

Büyük O Notasyonu

- Bir çok terimli fonksiyonun (polynomial function) big-o değerini hesaplamaya çalışalım.

- Örnek olarak fonksiyonumuz aşağıdaki şekilde olsun:

$$f(x) = 3x^4+2x^2+5$$

Büyük O Notasyonu

- Fonksiyonun üst asimtotik sınırı (asymptotic upper bound), her zaman için fonksiyona eşit veya daha yüksek değer veren ikinci bir fonksiyondur.

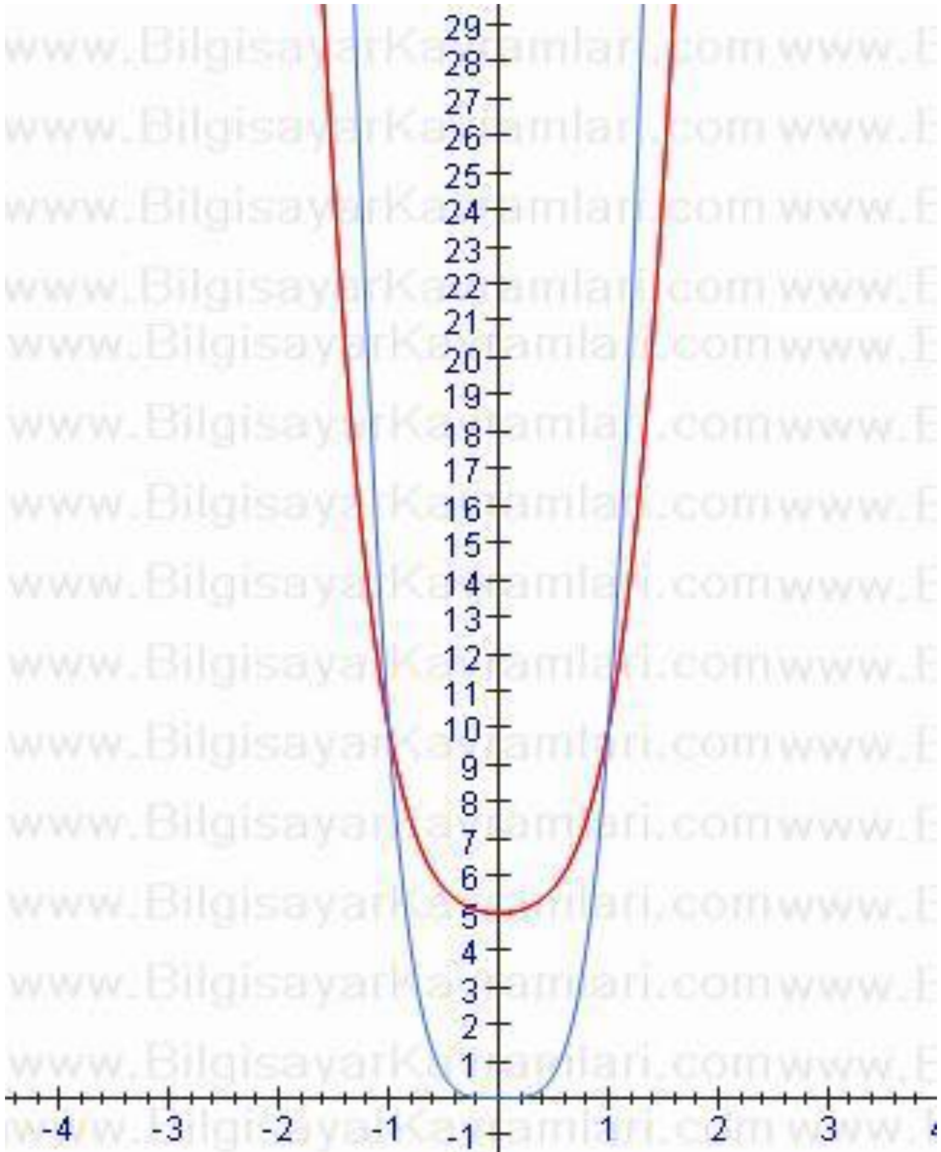
Bu durumda, yukarıdaki f(x) fonksiyonu için $O(x^4)$ denilmesinin anlamlı, herhangi bir sayı ile x^4değerinin çarpımının, f(x) fonksiyonuna eşit veya daha yüksek üreteceğidir. Bu durum aşağıdaki şekilde gösterilebilir:

$$f(x) \leq cx^4$$

Büyük O Notasyonu

- Gerçekten de bu değer denenirse, c=4 için aşağıdaki çizim elde edilebilir:

Büyük O Notasyonu



Büyük O Notasyonu

Yukarıdaki mavi renkte görülen fonksiyon $4x^4$ ve kırmızı renkte görülen fonksiyon da f(x) fonksiyonudur. Görüldüğü üzre $x>1$ için $4x^4$ fonksiyonu, f(x) fonksiyonundan büyüktür. Dolayısıyla tanımımıza $x>1$ koşulu eklenebilir.

Kısaca $f(x) = O(g(x))$

tanımı, $f(x) \leq c.g(x)$

şeklinde yorumlanabilir. Buradaki c değeri herhangi sabit bir sayıdır ve sonuçta elde edilen değer eşit veya daha büyüktür.

Büyük O Notasyonu

- Büyük-O (Big-O) gösterimi ile kardeş olan bir gösterim de küçük-o (small-o) gösterimidir.

- Bu iki gösterim arasındaki temel fark küçük-o gösteriminde asimptotik üst sınır (asymptotic upper bound) fonksiyonunun tamamından büyük olmasıdır.

- Yani büyük-O gösterimindeki eşitlik durumu yoktur.

- f(x) = og(x) için

Büyük O Notasyonu

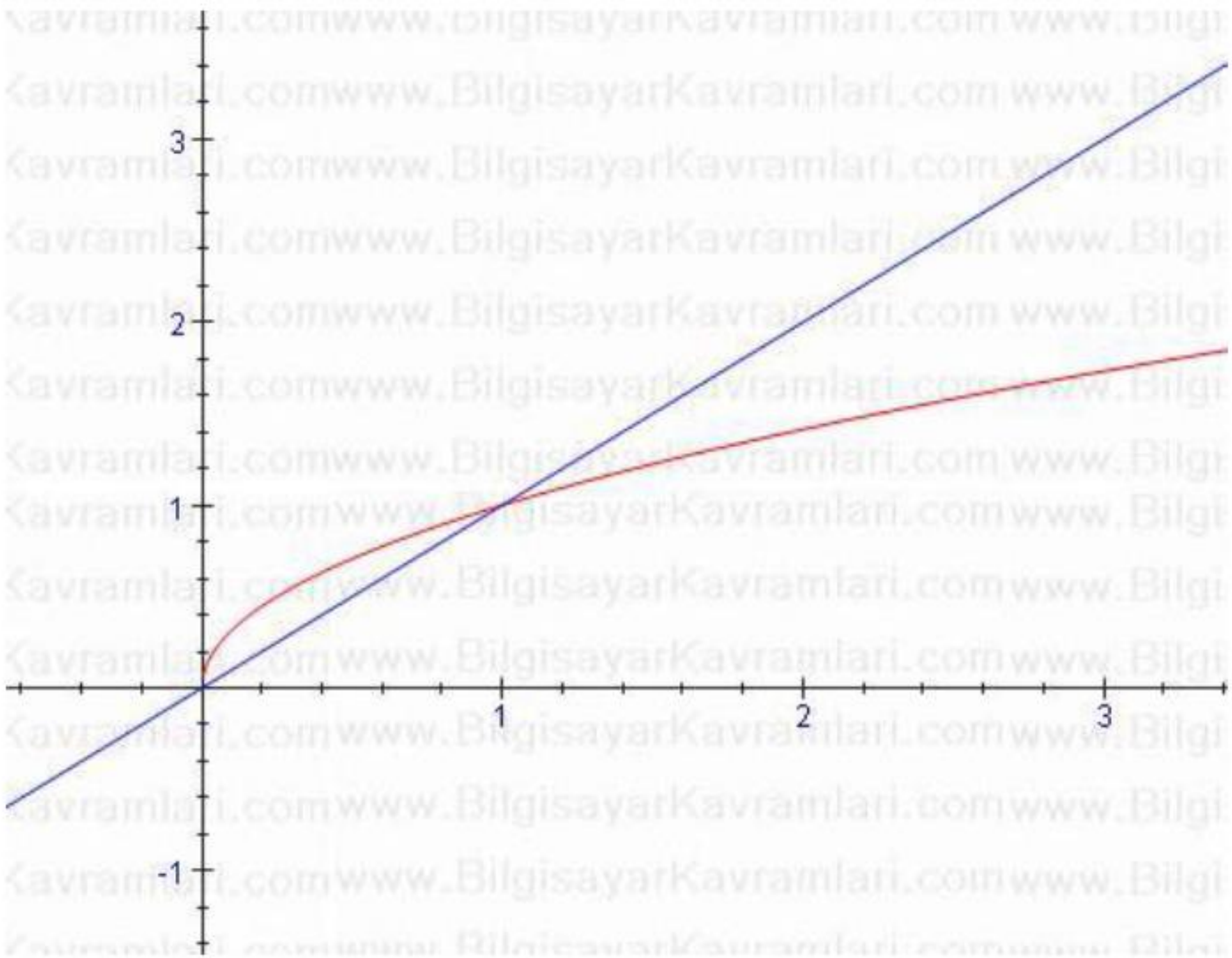
$f(x) < c\, g(x)$ şartı sağlanmalıdır.

Dolayısıyla $f(x) = O(f(x))$ tanımı doğru olurken $f(x)=o(f(x))$ tanımı hatalıdır.

Bazı örnekler aşağıda verilmiştir:

$\sqrt{x}=o(x)$

Büyük O Notasyonu

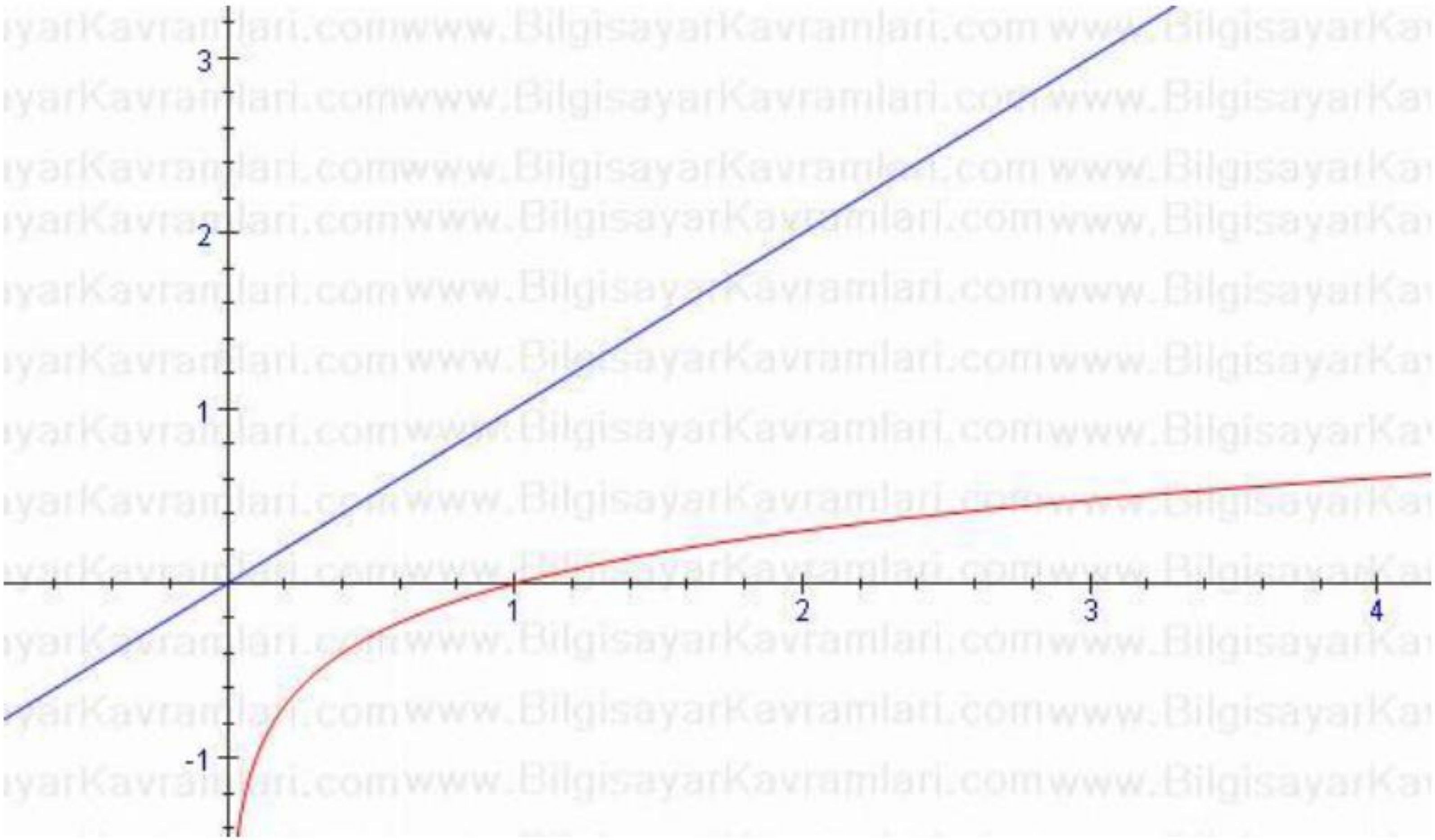


Büyük O Notasyonu

Yukarıda görüldüğü üzere $x>1$ için $\sqrt{x}=o(x)$ doğrudur. Ancak $x\geq1$ durumunda $\sqrt{x}=O(x)$ yazılmalıdır.

$\log(x) = o(x)$

Büyük O Notasyonu



Büyük O Notasyonu

- Yukarıdaki şekillerde krımızı ile gösterilen f(x) fonksiyonlarının small-o fonksiyonları mavi renk ile çizilmiştir.

Büyük O Notasyonu

Soru: $H(n)= 1+ 1/2 + ...+ 1/n$ ifadesinin algoritma karmaşıklığı nedir?

Cevap: bottom up approach kullanalim ve base line ile başlayalım:

$H(1)=1$

$H(2) = 1 + 1/2$

$H(2) = H(1) + 1/2$

$H(3) = 1 + 1/2 +1/3$

$H(3) = H(2) + 1/3$

demek ki $H(n) = H(n-1) + 1/n$