# BÖLÜM 2. ALGORİTMA ANALİZİ

Bu bölümde ilk olarak algoritma analizi kavramının açıklanması gerekir. Algoritma analizi, bu algoritmanın icra edilmesi sırasında duyacağı kaynak miktarının tahmin edilmesine **Algoritma Analizi** denir. Aslında, kaynak denildiğinde, bellek, iletişim bant genişliği, mantık kapıları akla gelebilir, fakat en önemli kaynak algoritmanın icra edilebilmesi için gerekli olan zamanın tahmin edilmesidir. Bilgisayarlar ilk olarak kullanılmaya başlandığı zaman donanım kaynakları (bellek, bant genişliği, mantık kapıları) çok önemliydi; fakat günümüzde problemlerin boyutlarında çok hızlı bir artış olmazken, donanımın çok hızlı bir şekilde artması bu kaynakların önemini azaltırken, zaman kaynağı hala önemini korumaktadır.Bir problem için birden fazla aday algoritma varsa, bunlar arasında tercih yapılmak istendiği zaman bütün kaynaklara göre bu algoritmaların analizleri yapılır ve en az kaynağı kullanan algoritma tercih edilir. Algoritma analizinden sonra bir problem için birden fazla algoritma olabilir (performansları arasında önemli bir farkın olmadığı algoritmalar).

Bir algoritma, analiz edilmeden önce kullanılacak kaynağın modeli ve maliyeti belli olmalıdır. Bundan dolayı, bu kitapta tek işlemcili rastgele erişimli makine (RAM) üzerinde programın yapıldığı kabul edilecektir. RAM makinalarda komutlar ardışıl olarak icra edilirler ve komutların icra edilmesi sırasında zaman paylaşımı yoktur.

Bir algoritmanın analizinin yapılabilmesi için matematiksel bilgilere (ayrık kombinatorik, temel olasılık, kümeler, cebir, v.b.) ihtiyaç duyulduğu gibi bazı terimlerin formül olarak ifade edilmesi gereklidir. Çünkü her giriş için algoritmanın davranışı farklı olabilir.

Örneğin Öklid algoritmasının çalışmasının performansının değerlendirilmesi için ortalama alacağı zaman ve bu algoritmanın kullandığı verilerin hafızada kapladığı alan hesabı yapılır. Hesaplama sonucunda elde edilen değerlere göre algoritmanın iyiliği hakkında bir yargıya varılır. Öklid algoritması üç tane tamsayı kullandığından hafıza yönünde kapladığı alan çok küçüktür. Aldığı ortalama zamanı

hesaplamak için n sayısı belli olarak alınır ve m sayısı için de belli bir aralık seçilir (1' den n' ye kadar olan sayılar). Bu aralıktaki bütün sayılar için zaman hesabı yapılır. Her m sayısı için hesaplanan zaman değerleri toplanır ve farklı m değerlerinin sayısına bölünür. Elde edilen zaman ortalama zamandır ve $T_n$ ile gösterilir. Bu $T_n$ değeri algoritmanın zaman bakımından performansını gösteren bir kriterdir.

Önemli noktalardan biri $T_n$ değerinin alacağı değerlerle ilgili bir formülasyon gerçekleştirmektir. Algoritmanın ortalama zaman değeri algoritmaya verilen veriler kümesine bağlı olduğundan kesin bir formül verilmesi hatalı olur. Bundan dolayı algoritma analizinde zaman hesabı için yeni notasyonlar kullanılmaktadır. Bu notasyonlara daha sonraki bölümlerde detaylı olarak değinilecektir.

İkinci örnek olarak ArayaEkleSırala (Inseriton Sort) algoritmasının analizi yapılabilir. Daha önce vurgulandığı gibi bir algoritmanın taşıması gereken beş tane temel özellik vardır. Bunlar, algoritmanın mutlaka sınırlı zamanda icra edilip bitirilebilmesi gerekir; algoritmanın içinde bütün durumların düşünülerek buna göre kuralların tanımlanması gerekir; her algoritmanın giriş ve çıkış veriler kümelerinin belirtilmesi gerekir ve aynı zamanda aynı alanda yazılmış olan en etkili algoritmanın kullanılması gerekir.

**Algoritma 2.1. ArayaEkleSırala**

Kendisine giriş verisi olarak bir dizi verilir ve bu dizinin sıralanmış şekli çıkış verisi olarak alınır.

```
ArayaEkleSırala(A)
1. j←2,3,...,n
2. anahtar←A[j]
3. i←j-1
4.        (i>0 ve A[i]>anahtar) oldukça
5.              A[i+1]←A[i]
6.              i←i-1
7.         A[i+1]←anahtar
```

Bir algoritmanın analizi nasıl yapılır, bunu göstermek için ArayaEkle sıralama algoritmasından faydalanılacaktır.

Sıralama işlemi verilen dizinin içindeki elemanların birbiri ile olan ilişkisi ile yakından ilgilidir ve aynı zamanda sıralanacak eleman sayısı da sıralama zamanını etkiler. Çünkü iki sayıyı sıralamak için bir tane karşılaştırma gerekirken, dört tane sayıyı sıralamak için altı tane karşılaştırma işlemi gereklidir. Bundan dolayı dizinin boyutu algoritmaların zaman ve uzay analizinde çok önemli bir yere sahiptir. Dizinin içindeki elemanların birbirlerine olan yakınlıkları sıralama zamanını etkiler. Aynı sayıda iki tane dizinin sıralama zamanları aynı olmayabilir, çünkü bu dizilerin içindeki elemanların durumları algoritmanın işleyişini etkiler. Bundan dolayı algoritmanın çalışma zamanı giriş veri boyutunun bir bağıntısı olarak elde edilebilir.

Bir algoritmanın çalışma zamanı, belli bir giriş verisi için o algoritmanın kaç tane temel operasyon yaptığı ile ilgilidir. Her operasyon (adım) mümkün mertebe makineden bağımsız olarak tanımlanmalıdır. Bu kitapta bir algoritmanın her satırının icra edilme zamanı sabit kabul edilecektir ve her satırın çalışma zamanı bir diğerinden farklı olabilir.

Bir algoritmanın analizinin nasıl yapıldığı biraz daha detaylı olarak ArayaEkleSıralama algoritması üzerinde gösterilecektir. Bu algoritmada sıralanmak istenen dizi A olsun ve n=boyut(A) olmak üzere bu algoritmanın analizi üç şekilde yapılabilir.

- En iyi durum analizi
- En kötü durum analizi
- Ortalama durum analizi

En iyi durum daha matematiksel olarak tanımlanacak olursa,

***Tanım 2.1***

*(En iyi durum) O anda çözülecek problem için boyutu n olan giriş verilerinin kümesi $D_n$ olsun ve I verisi $D_n$ kümesinin bir elemanı olsun. I verisi üzerinde algoritma tarafından uygulanan temel operasyon sayısı t(I) olmak üzere W fonksiyonu*

*$W(n)=min\{t(I) \mid I \in D_n\}$*

*şeklinde tanımlanır ve bu fonksiyona algoritmanın en iyi durum karmaşıklığı (kompleksliği) denir. W(n) fonksiyonu, algoritma tarafından n boyutlu herhangi bir veri üzerine uygulanan minimum temel operasyon sayısıdır.*

En kötü durumun tanımı aşağıdaki gibi yapılabilir.

***Tanım 2.2***
*(En kötü durum) O anda çözülecek problem için boyutu n olan giriş verilerinin kümesi $D_n$ olsun ve I verisi $D_n$ kümesinin bir elemanı olsun. I verisi üzerinde algoritma tarafından uygulanan temel operasyon sayısı t(I) olmak üzere W fonksiyonu*

*$W(n)=max\{t(I) \mid I \in D_n\}$*

*şeklinde tanımlanır ve bu fonksiyona algoritmanın en kötü durum karmaşıklığı (kompleksliği) denir. W(n) fonksiyonu, algoritma tarafından n boyutlu herhangi bir veri üzerine uygulanan maksimum temel operasyon sayısıdır.*

Ortalama durum tanımı ise Tanım 2.3' de görüldüğü gibidir.

***Tanım 2.3***
*(Ortalama durum) P(I) fonksiyonu I verisinin giriş verisi olma olasılığı olmak üzere algoritmanın ortalama durum davranışı*

$$A(n)= \sum_{I \in D_n} P(I)t(I)$$

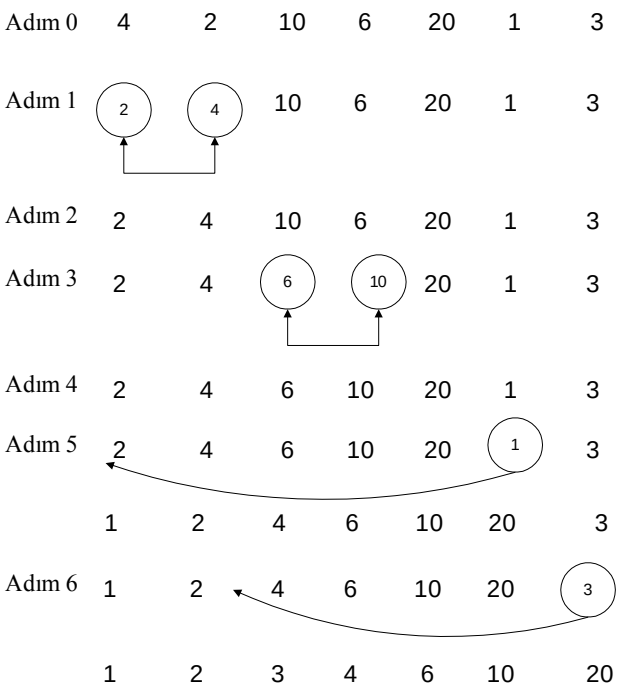
*şeklinde tanımlanır.*

Bu durumların analizi yapılmadan önce bu algoritmanın bir algoritma olduğu ve nasıl çalıştığı üzerinde durulabilir. Bu algoritmanın özelliklerine sırasıyla bakılacak olursa; bu algoritmanın içinde iki tane döngü var ve bu döngüler sınırlı zaman sonunda bitime giderler. Her durum için algoritmanın ne yapacağı kesin olarak belirtilmiştir. Bu

algoritmanın giriş verisi A dizidir ve çıkış verisi ise sıralanmış A dizisi olacaktır.

Bu algoritmanın etkililik özelliğini irdelemeden önce, bu algoritmanın <4,2,10,6,20,1,3> dizisi üzerinde nasıl çalıştığına bakılabilir.

İlk olarak dizinin ilk iki elemanı kontrol edilmektedir ve bu işlem adım 1' de gerçekleştirilmektedir. Kontrol edilen elemanlar sıralı olmadıklarından yer değiştiriliyorlar. Bundan sonra dizinin ilk üç elemanı kontrol ediliyor ve bu işlem adım 2' de gerçekleştirilmektedir. Fakat ilk üç eleman sıralı olduğundan herhangi bir işlem yapılmaz. Adım 3' de dördüncü ve üçüncü elemanların sırayı bozdukları görülmektedir. Bundan dolayı bu iki eleman yer değiştirir ve sıra sağlanmış olur. Bu işleme dizinin sonuna kadar devam edilir. Dizinin sonuna gelindiğinde dizi sıralanmış olur.




**Şekil 2.1.** *ArayaEkle sıralama algoritmasının <4, 2, 10, 6, 20, 1, 3> dizisi üzerinde çalışması.*

ArayaEkleSırala(A) algoritmasının her adımının kaç sefer icra edildiği hesaplanarak algoritmanın zaman bağıntısı elde edilir. Aşağıda her adımın kaç sefer icra edildiği görülmektedir ve her satırın çalışmasının zamanını o satırın indeksini taşıyan $s_i$ gibi sabitler olsun. $s_i$ sabiti i. satırın bir sefer çalışma zamanını göstermektedir.

| ArayaEkleSırala(A) | |
|---|---|
| 1) j←2,3,...,n | n |
| 2) anahtar←A[j] | n-1 |
| 3) i←j-1 | n-1 |
| 4) (i>0 ve A[i]>anahtar) oldukça | $\sum_{j=2}^{n} t_j$ |
| 5) A[i+1]←A[i] | $\sum_{j=2}^{n} (t_j - 1)$ |
| 6) i←i-1 | $\sum_{j=2}^{n} (t_j - 1)$ |
| 7) A[i+1]←anahtar | n-1 |

Bu durumda ArayaEkleSırala(A) algoritmasının zaman bağıntısı T(n)

$$T(n)=s_1n+s_2(n-1)+s_3(n-1)+s_4\sum_{j=2}^{n} t_j +s_5\sum_{j=2}^{n} (t_j - 1) +s_6\sum_{j=2}^{n} (t_j - 1) +s_7(n-1)$$

olur. Burada $t_j$ değeri o satırın j değeri için kaç kez çalıştığını gösterir. Zaman bağıntısından da anlaşılacağı gibi birinci satır n kez çalışır, ikinci ve üçüncü satırların her biri n-1 kez çalışırlar. Dördüncü satır

$$\sum_{j=2}^{n} t_j$$

kez çalışır ve beşinci ve altıncı satırlar

$$\sum_{j=2}^{n} (t_j - 1)$$

kez çalışırlar. Son olarak yedinci satır n-1 kez çalışır. Her satırın çalışma sayısı ile zamanının çarpılmasının toplanması ile algoritmanın zaman bağıntısı elde edilir.

Dördüncü satırın çalışma sayısı giriş verisi olarak verilen dizinin elemanlarının birbiri ile olan ilişkisine bağlıdır. Bu sıralama algoritmasının çalışma zamanının minimum olması için A dizisinin sıralı olması gerekir. Bu durumda j=2,3,...,n değerlerinin hepsi için $t_j$=1 olur ve T(n) zaman bağıntısı

$$T(n)=s_1n+s_2(n-1)+ s_3(n-1)+s_4(n-1)+s_7(n-1)$$
$$=s_1n+(s_2+s_3+s_4+s_7)(n-1)$$
$$=(s_1+s_2+s_3+s_4+s_7)n-(s_2+s_3+s_4+s_7)$$
$$=an-b$$

şeklinde lineer bir bağıntı olur ve bu durum bu algoritmanın en iyi durumudur. $t_j$=1 alınmasının anlamı 4. adımdaki döngü her seferinde bir kez çalışır anlamına gelmektedir ve bundan dolayı zaman lineer çıkar. Bunun diğer bir anlamı da dizi sıralı durumdadır. Eğer dizi tam tersi yönde sıralı ise, en kötü durum oluşur ve bu durumda T(n) zaman bağıntısı lineer olmayacaktır.

Dizi tersten sıralı olduğundan her eleman sıralı alt dizinin içindeki her eleman ile karşılaştırılacağından $t_j$=j olur. Buradan

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

ve

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

bağıntıları elde edilir. T(n) zaman bağıntısı tekrar yazılırsa,

$$T(n)=s_1n+s_2(n-1)+s_3(n-1)+s_4\left(\frac{n(n+1)}{2}-1\right)$$

$$+s_5\left(\frac{n(n-1)}{2}\right)+s_6\left(\frac{n(n-1)}{2}\right)+s_7(n-1)$$

$$=\left(\frac{s_4}{2}+\frac{s_5}{2}+\frac{s_6}{2}\right)n^2+(s_1+s_2+s_3+\frac{s_4}{2}-\frac{s_5}{2}-\frac{s_6}{2}+s_7)n$$

$$-(s_2+s_3+s_4+s_7)$$

$T(n)=an^2+bn+c$

olur. Dikkat edilirse, elde edilen bağıntı kuadratik bir bağıntıdır. Bunun anlamı, eğer dizi tersten sıralı ve boyutu arttıkça icra zamanı çok daha hızlı artar. Bir algoritma için her zaman en iyi durum veya en kötü durum her zaman oluşmaz. Bundan dolayı, algoritmanın analizinin yapılması için her zaman geçerli olan bir yöntemin kullanılması gerekir. Bütün durumların ortalaması yöntemi kullanılabilir. En kötü durum analizinde elde edilen sonuç o algoritmanın zaman bağıntısı için bir üst sınır teşkil ederken, en iyi durum analizinde elde edilen sonuç o algoritmanın çalışma zamanı için bir alt sınır teşkil eder.

Algoritma analizinde en kötü durum analizi sonucu algoritmanın alabileceği zamanın en fazla ne kadar olacağını garanti eder. Bazı algoritmalarda, özellikle arama algoritmalarında (büyük veritabanı aramalarında, v.b.) aranan kayıt mevcut değilse (yapılan arama işlemi başarısız bir arama ise), en kötü durum oluşur. Bir algoritmada en kötü durum çok sık oluşuyorsa, ortalama durumu ile en kötü durumu aynı olur. Örneğin, sıralama algoritmalarında A[j] alt dizisinde bir elemanı doğru yerine yerleştirmek için ortalama olarak j/2 tane karşılaştırma yapılır. Çünkü eklenecek değer A[j] alt dizisinde ortalama olarak dizinin elemanlarının yarısından küçük ve geriye kalan yarısından büyüktür. Bu değer T(n) zaman bağıntısında yerine yazıldığında, elde edilecek bağıntı yine kuadratik olur.

ArayaEkleSırala(A) algoritmasının analizinde en iyi durum ve en kötü durum analizleri yapıldı. Algoritmaların analizlerinde genellikle en kötü durum analizi yapılır ve bu sonuç verilir. Bu sonucun verilmesinin sebepleri

a) En kötü durum zaman bağıntısı, o algoritmanın zaman bağıntısının üst sınırını verir.
b) Bazı algoritmalarda en kötü durum çok sık oluştuğundan dolayı, zaman bağıntısı için en kötü durum zaman bağıntısı kullanılır.
c) Bir algoritmanın ortalama zaman bağıntısı genellikle en kötü durum zaman bağıntısına yaklaştığından dolayı, doğrudan en kötü durum zaman bağıntısının kullanılmasının bir sakıncası olmaz.

şeklinde sıralanabilir.

Algoritma analizi teriminden anlaşılan kavram, bir algoritma alınır ve bunun ortalama zaman ve uzay hesabı yapılır. Ondan sonra bu algoritmanın zaman ve uzay hesabı bakımından optimal olup olmadığına karar verilir. Diğer önemli bir nokta da verilen bir problem için etkili bir algoritma tanımlanabilir olup olmadığının araştırılmasıdır.

Algoritma analizinde elde edilen T(n) için bazı sadeleştirmeler yapılabilir ve T(n) bağıntısının bütün sabitleri ile değil, bu bağıntı, n arttıkça nasıl arttığı önemlidir. Bundan dolayı, T(n) zaman bağıntısının artışının nasıl olacağının belirlenmesi yeterlidir. İlk olarak bütün satırların maliyetlerinin ihmal edilmesi ve kod içerisinde baskın olan (daha çok zaman alan) kısımların göz önüne alınması sonucunda, bu algoritma için T(n) zaman bağıntısının nasıl artacağı elde edilmiş olur.

Algoritma analizinde T(n) zaman bağıntısının ne olduğu değil, nasıl arttığı önemlidir. Bundan dolayı da T(n) zaman bağıntısının en yüksek dereceli terimi önemlidir. Bunun için

$$T(n)=a_k n^k+a_{k-1}n^{k-1}+...+a_1 n+a_0$$

Bağıntısı elde edildiğinde $a_k n^k$ terimi göz önüne alınır; çünkü T(n) zaman bağıntısının davranışını belirleyen terim budur. Hatta $a_k n^k$ teriminde zaman içerisinde hiç değişmeyecek olan kısım $a_k$ olduğundan bu kısımda fazlaca önemsenmez ve $n^k$ kısmı n değişkeninin artış ve azalışında çok hızlı değişeceğinden dolayı T(n) zaman bağıntısının davranışı bu terime göre vurgulanır ve $T(n)=\Theta(n^k)$

şeklinde gösterilir. Θ-notasyonu daha sonraki bölümlerde detaylı olarak açıklanacaktır. Burada sadece ne anlama geldiğinin vurgulanması yeterlidir. $T(n)=\Theta(n^2)$ algoritmasının zamanı $T(n)=\Theta(n^3)$ algoritmasının zamanından daha az olur. Yani, ilk algoritma daha iyi bir algoritmadır.

Bu bölüm hesapsal yöntemin tanımı yapılarak kapatılabilir. Hesapsal yöntem HY=(Q,I,O,f) şeklinde ifade edilir ve

Q : Hesaplama durumları
I : Algoritmanın giriş verileri kümesi
O : Algoritmanın çıkış verileri kümesi
f : Hesaplama kuralları

olarak tanımlanırlar. Q, I ve O kümelerinin alt kümelerinin bir kümesidir. f bağıntısı Q' den Q' ye tanımlıdır ve O kümesinin elemanları için birim fonksiyondur. I kümesi içindeki her giriş $x_0$, $x_1$, …, gibi bir dizi oluşturur ve

$x_0=x$ ve $x_{k+1}=f(x_k)$, $k\geq 0$.

Eğer minimum k değeri için $x_k$ O kümesinin bir elemanı ise, hesapsal dizi k adım sonunda biter denir. Bu durumda x verisinden $x_k$ verisi üretilir denir. Bazı hesapsal yöntemler hiçbir zaman bitime gitmezler. Bir algoritma sınırlı adım sonunda bitime gider.

Öklid algoritmasının hesapsal yöntem tanımı şu şekilde yapılabilir. Q kümesinin elemanları, tekliler (n), sıralı ikililer (m,n), dörtlüler (m,n,r,1), (m,n,r,2), (m,n,p,3) şeklinde olurlar. m, n ve p pozitif tamsayılardır ve r ise negatif olmayan bir tamsayıdır. I kümesi (m,n) şeklindeki bütün ikililerin bir alt kümesidir ve O kümesi ise (n) şeklinde olan bir teklidir. f bağıntısı şu şekilde tanımlanabilir.

f(m,n)=(m,n,0,1); f(n)=(n)

f(m,n,r,1)=(m,n, m÷n kalanı, 2)

$$f(m,n,r,2)=\begin{cases}(n) & r=0\\ (m,n,r,3) & r\neq 0\end{cases}$$

f(m,n,p,3)=(n,p,p,1).

Verilen bu hesapsal yöntem tanımı ile Öklid algoritması arasındaki ilişki aşikardır.

Bu tanımlamada etkililik kriteri göz önüne alınmamıştır. Q kümesi sınırlanmamıştır ve Q kümesi sınırsız sayıda eleman içerebilir ve f bağıntısının yapacağı işlemler insanın kağıt-kalem ile yapamayacağı işlemler gerçekleştirebilir. Hesapsal yönteme sınırlamalar getirilebilir, fakat bu sınırlamalar bu kitabın konusu dışındadır.