Bir problem için algoritma tasarlarken

bu programı bilgisayarda çalıştırdığımızda, problemin çözümüne ulaşılmasının ne kadar

zaman alacağı önemli bir husustur. Bilindiği üzere, bir problemin farklı algoritmalarla

birden fazla çözümü olabilir. Bir problemi çözmek istediğimizde, olası algoritmalardan

çalışma zamanı ve bellek gereksinimi açısından uygun olanını seçmek oldukça önemlidir.

Örneğin, sıralama problemi için birçok algoritma kullanılabilir. Sıralama algoritmasını

küçük bir veri seti üzerinde çalıştıracaksak, programcının en kolay tasarlayacağı algorit-

mayı seçmesi en basit yaklaşımdır. Ancak sıralama probleminde milyonlarca veri olması

söz konusuysa çözüme makul bir zaman içinde ulaşılması tercih edilecektir. Böyle du-

rumlarda, çalışma zamanı en kısa olan algoritmayı seçmemiz kendi yararımıza olacaktır.

Ayrıca, kullanılacak algoritmanın bellek gereksinimi de üzerinde dikkatle düşünülmesi

gereken konulardan biridir.

Algoritmaları çalışma zamanı açısından karşılaştırmak istediğimizde aklımıza gelebi-

lecek temel ölçütlerden biri, ilgili algoritmanın çözümüne kaç saniyede ulaşıldığıdır. An-

cak, saniye, dakika, saat vb. zaman birimlerinin bu ölçümde ölçüt olarak kullanılması çok

uygun değildir. Herhangi bir problemin kaç saniyede çözüme ulaşabileceği, kullandığımız

bilgisayarın konfigürasyonuna ve kullandığımız derleyicinin ürettiği makine koduna bağ-

lı olarak değişebilmektedir. Çalışma zamanı karşılaştırması için kullanılabilecek diğer bir

yaklaşım ise algoritmadaki bütün operasyonları tek tek saymaktır. Ancak bu işlem hem

zor hem de gereksizdir. Bunun yerine, algoritmanın içerisinde temel işlemin yapıldığı

yeri bulup buradaki operasyon sayısını hesaplamak çok daha uygun bir yaklaşım olacak-

tır. Dikkatle incelendiği takdirde, bir algoritmanın temel operasyonunu bulmak çok zor

değildir. Örneğin bir sıralama algoritmasında temel operasyon, sayıların karşılaştırıldığı

kısımdır. Sıralama algoritmasının ne kadar karşılaştırma operasyonu yaparak ana sonuca

ulaştığı, temel operasyonu oluşturmaktadır.

Çalışma Zamanı (running time): Tasarlanan algoritma ile problemin çözümüne ulaşa-

bilmek yapılan toplam temel operasyon sayısıdır. T (n) ile ifade edilir. Temel operasyonlar,

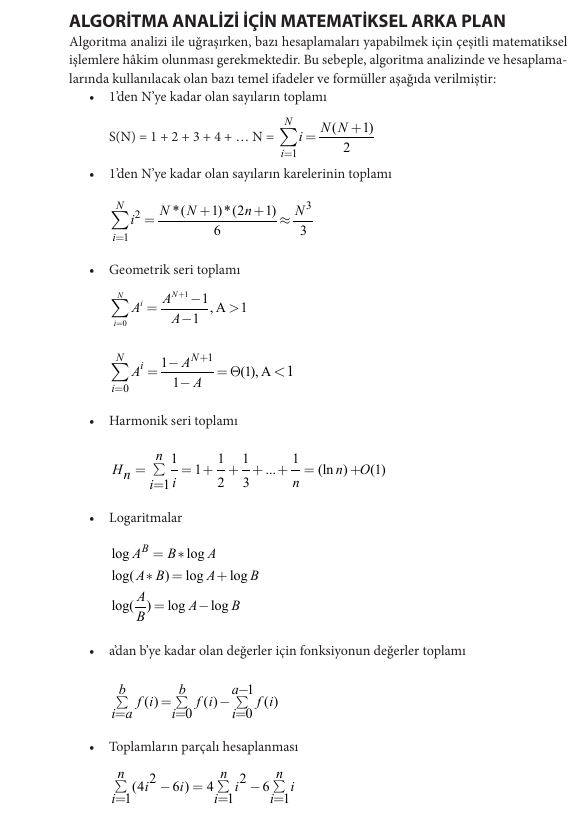
karşılaştırma sayısı, döngü içerisinde dönme sayısı vb. işlemler olabilir.

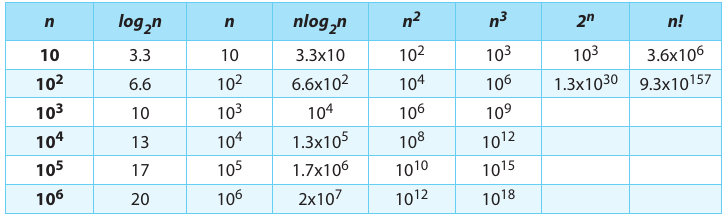
Zaman Karmaşıklığı (time complexity): Bir algoritmanın verilen asimptotik gösterime

göre karmaşıklık derecesini gösterir.

Alan Karmaşıklığı: Algoritmanın eleman sayısının çok büyük olduğu durumlarda,

problemin çözümünü ulaşabilmeye yönelik bellek gereksinimidir.





ALGORİTMALARIN EN KÖTÜ DURUM, EN İYİ DURUM VE

ORTALAMA DURUM VERİMLİLİKLERİ

Algoritmaların durum verimlilikleri verilen girdiye göre değişebilmektedir. Örneğin, bir

lineer sıralama algoritmasını ele alalım. Bu algoritma, verilen bir sayının dizideki eleman-

lardan biri olup olmadığının sonucunu döndürmektedir. Bu algoritmada verilen sayı, di-

zideki ilk elemandan başlayarak sırasıyla dizinin tüm elemanlarıyla karşılaştırılmaktadır.

Bulmak istediğimiz değerin dizideki ilk elemana eşit olması, arama probleminde karşıla-

şabileceğimiz en iyi durumdur. Böyle bir durumda, dizinin ilk elemanı ile karşılaştırmayla

hemen sonuca ulaşıp dizinin diğer elemanlarıyla karşılaştırmaya gerek kalmayacaktır. Bu

durum, lineer arama algoritmasının optimal girdi için verimliğidir.

Bir algoritmanın en iyi durum verim-

liliği, en optimal girdi durumundaki algoritma karmaşıklığıdır. Ancak en iyi durum ve-

rimliliği çok fazla anlam ifade etmemektedir. Çünkü çoğu zaman verilen girdiler sürekli

değişmektedir ve optimal girdinin denk gelme olasılığı oldukça düşüktür. Karşılaşabilece-

ğimiz olası bir başka durum ise girdi verisinin algoritma için en kötü olduğu durumdur.

Lineer arama algoritmasını düşünecek olursak aradığımız elemanın dizinin son elemanı

olması veya dizide hiç bulunmaması karşılaşabileceğimiz en kötü girdi durumlarıdır. Bu

durumlarda, lineer arama algoritması ilk elemandan son elemana kadar sırasıyla bütün

elemanlar için arama işlemini gerçekleştirecektir. Başka bir ifadeyle, en kötü durum ve-

rimliliği söz konusudur. Algoritmaların karmaşıklığını hesaplarken genellikle en kötü du-

rumu göz önüne almaktayız.

Son olarak bakacağımız durum ise ortalama durum verimliliğidir. Burada ise algorit-

mamızı farklı girdiler için pek çok kez çalıştırarak çalıma sürelerinin ortalamasını alırız.

Böylece, ortalama durum verimliliğini elde etmiş oluruz. Lineer arama algoritmasını dü-

şünecek olursak verilen girdiye göre 1., 2., 3., ..., N. gibi adımlarda aradığımız sayıyı bula-

biliriz. Bu olasılıkların hepsinin ortalamasını aldığımızda (N/2)’nci adımda, yani dizinin

orta elemanında sonucu bulmamız gerekmektedir. Bu değer ise lineer arama algoritması-

nın ortalama durum verimliliğidir.

ASİMPTOTİK GÖSTERİMLER

Bir algoritma çalıştığında kaç birim adımda sonuca ulaşacağı hesaplanarak aynı problemi

çözen farklı algoritmaların verimliliği karşılaştırılabilir. Bilgisayar bilimcileri, fonksiyon-

ların büyümesini de göz önünde bulundurarak algoritmaları karşılaştırırken kullanılmak

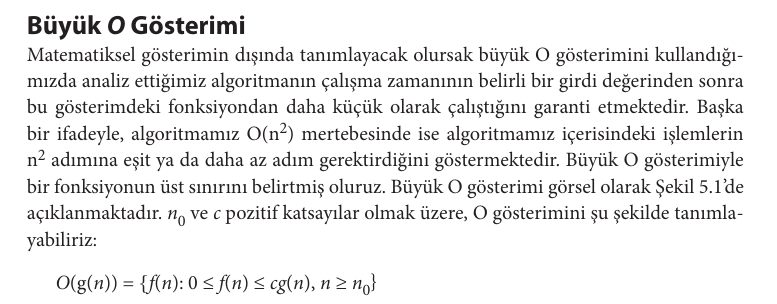
üzere üç tane gösterim tanımlamıştır:

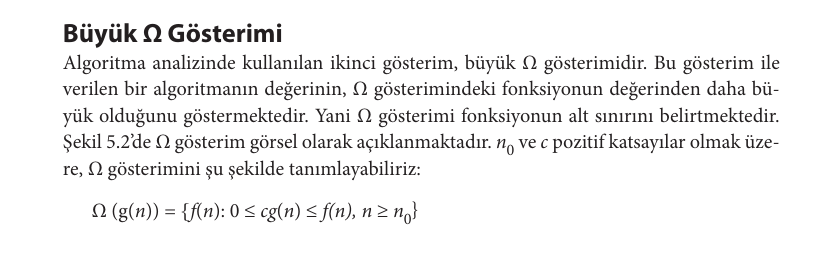
• Büyük O Gösterimi

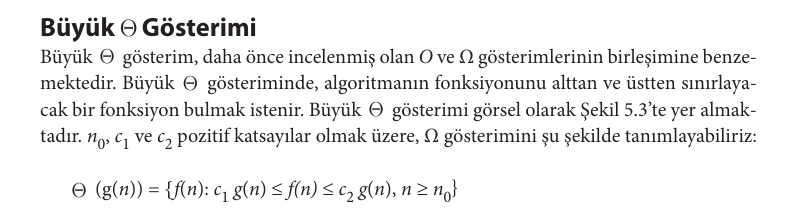
• Büyük Ω Gösterimi

• Büyük Θ Gösterimi

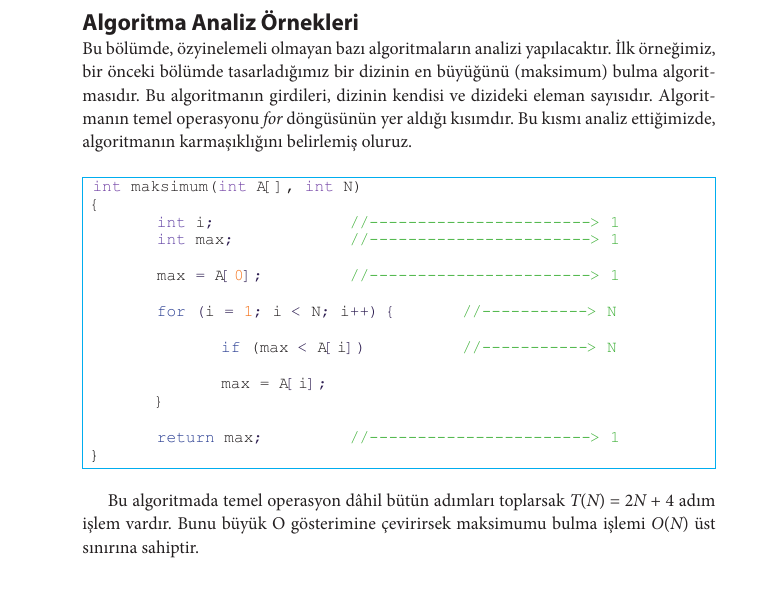
Bu gösterimlere göre, algoritmaları kendi arasında verimlilik açısından karşılaştırabiliriz.

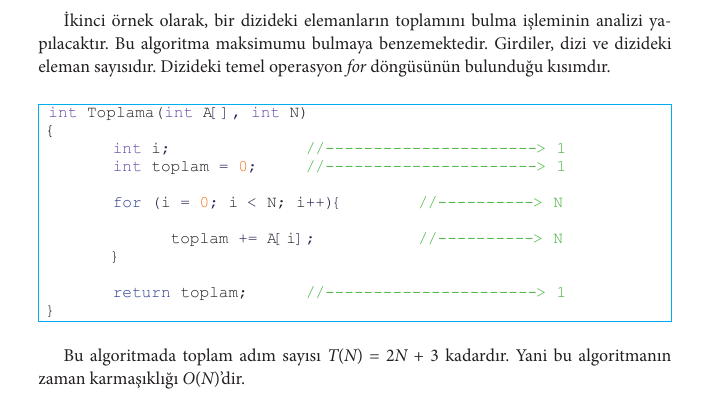


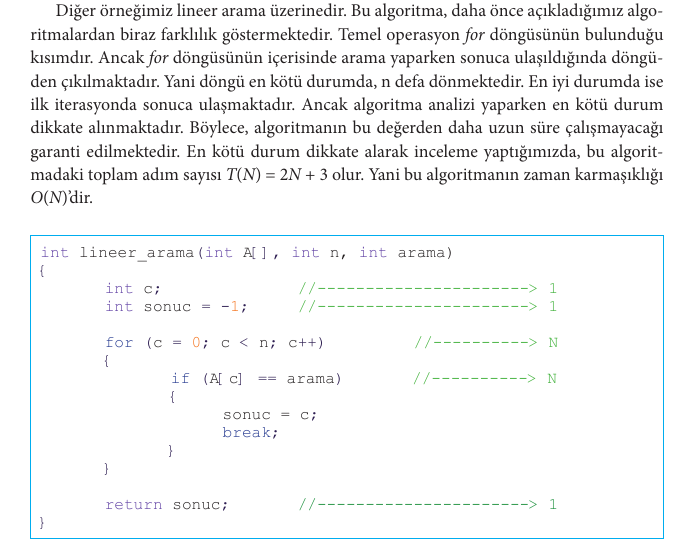


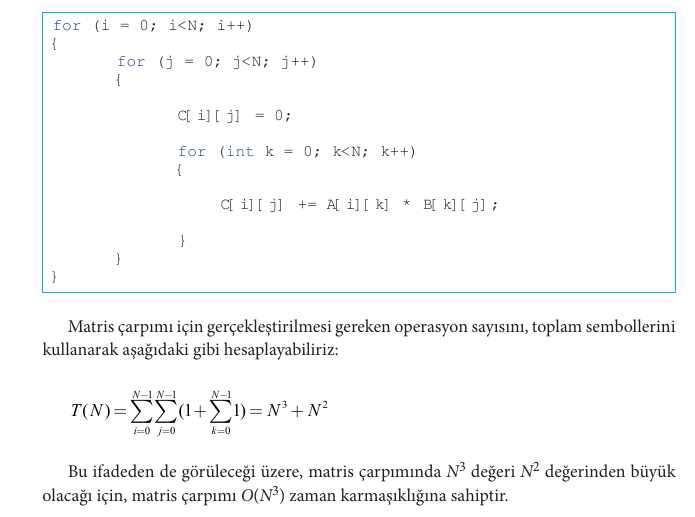


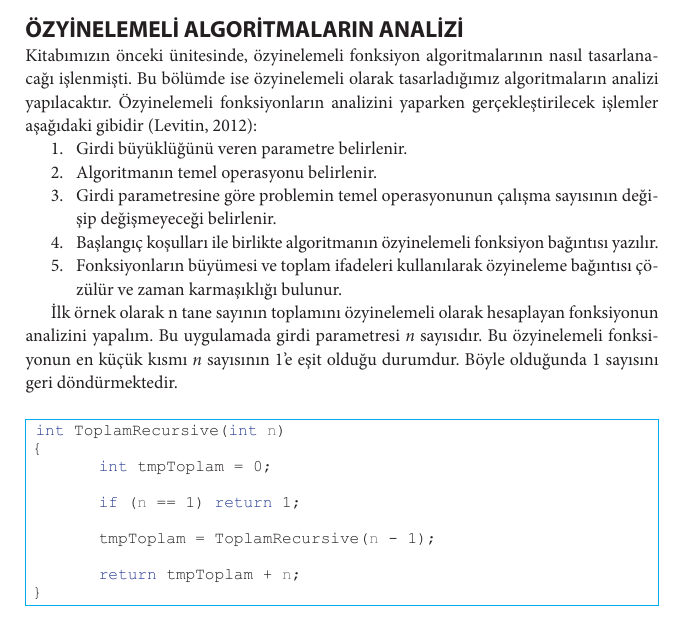


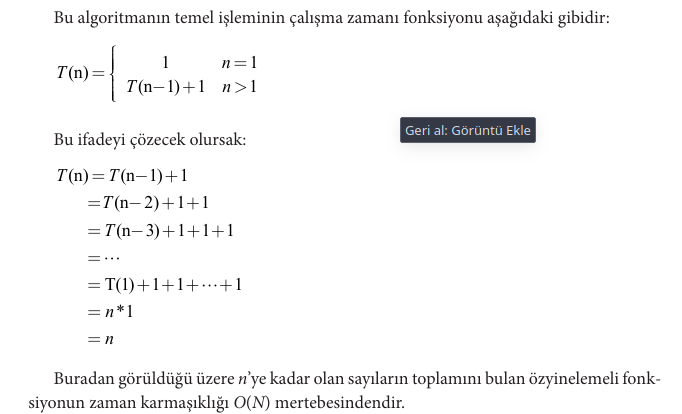


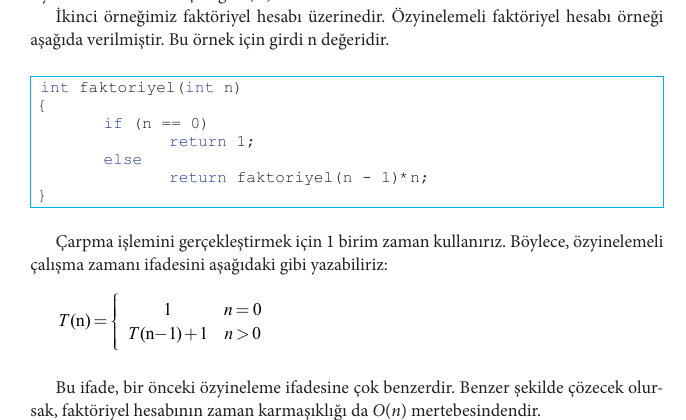


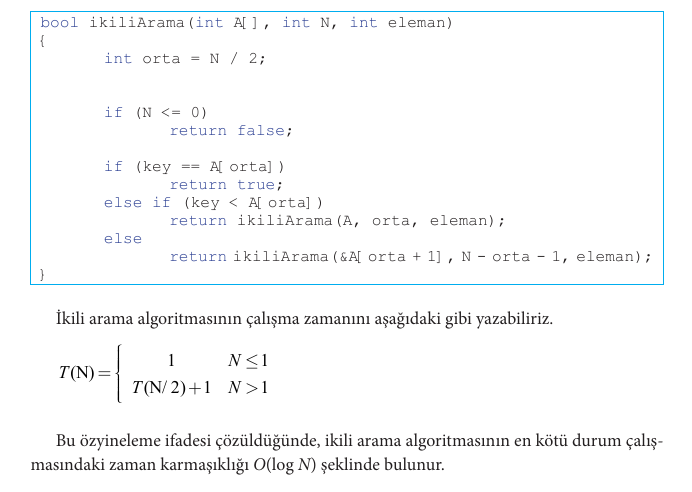












Hbj