Veri Yapıları ve Algoritmalar

DR. ÖGR. ÜYESİ MEHMET AKİF BÜLBÜL

2023-2024 GÜZ YARIYILI

PROGRAM ÇALIŞMA HIZI VE BELLEK GEREKSİNİMİ (ALGORİTMA ANALİZİ)

Algoritma Analizi

Algoritma analizi, bilgisayar programının performansı (başarım) ve kaynak kullanımı konusunda teorik çalışmalardır.

Bir başka ifadeyle, algoritmanın icra edilmesi sırasında duyacağı kaynak miktarının tahmin edilmesine **Algoritma Analizi** denir.

Algoritma analizi, farklı çözüm yöntemlerinin verimliğini analiz eder.

Algoritma Analizi

Performanstan daha önemli ne vardır ?

Algoritma Analizi

- modülerlik
- doğruluk
- bakım kolaylığı
- işlevsellik
- sağlamlık
- kullanıcı dostluğu
- programcı zamanı (fiyat)
- basitlik
- genişletilebilirlik
- güvenilirlik

Algoritma Analizi

Neden algoritmalar ve başarımla uğraşırız?

Algoritma Analizi

Başarım (performans) genelde yapılabilir olanla imkansızın arasındaki çizgiyi tanımlar.

Algoritmik matematik program davranışlarını açıklamak için ortak dil oluşturur.

Başarım **bilgi işleme**'nin para birimidir.

Program başarımından alınan dersler diğer bilgi işlemekaynaklarına genellenebilir.

Algoritmik Performans

Algoritmik performansın iki yönü vardır:

Zaman (Time)

- Yönergeler veya talimatlar zaman alabilir.
- Algoritma ne kadar hızlı bir performans gösteriyor?
- Algoritmanın çalışma zamanını (runtime) ne etkiler?
- Bir algoritma için gerekli olan zaman nasıl tahmin edilir?
- Gerekli olan zaman nasıl azaltılabilir?

Alan (Space)

- Veri yapıları yer kaplar.
- Ne tür veri yapıları kullanılabilir?
- Veri yapılarının seçimi çalışma zamanını nasıl etkiler?

Algoritma Analizi

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

Benzer problemi çözmek için iki algoritmanın zaman verimliliğini nasıl karşılaştırabiliriz?

Naif(Basit) yaklaşım: bir programlama dilinde bu algoritmaların uygulanması ve çalışma zamanlarının karşılaştırılması.

Algoritma Analizi

Algoritmalar yerine programların karşılaştırılmasında bazı zorluklar vardır:

Programın kullanabileceği veri nedir?

Analiz yöntemi veriye bağımlı olmamalıdır. Çalışma zamanı verinin büyüklüğü ile değişebilir.

Hangi bilgisayarı kullanmak gerekir?

Algoritmaların verimliliği belirli bir bilgisayara bağımlı olmadan karşılaştırılmalıdır. Çünkü, aynı algoritmanın işlemci hızları farklı iki bilgisayarda çalışma zamanı aynı olmaz.

Algoritma nasıl kodlanmalıdır?

Çalışma zamanını karşılaştırmak, uygulamaları karşılaştırmak anlamına gelir. Uygulamalar, programlama tarzına duyarlı olduğundan karşılaştıramayız. Programlama tarzı çok verimli bir algoritmanın çalışma zamanını bile etkileyebilir.

Programları karşılaştırmak, bir algoritmanın kesin ölçümü için uygun değildir.

Algoritma Analizi

Algoritma analizi, özel uygulamalardan, bilgisayarlardan veya veriden bağımsızdır.

Algoritma analizi, tasarlanan program veya fonksiyonun belirli bir işleme göre matematiksel ifadesini bulmaya dayanır.

Algoritmaları analiz etmek;

- İlk olarak, algoritmanın etkinliğini değerlendirmek için belirli bir çözümde anlamlı olan işlemlerin kaç adet olduğu sayılır.
- Daha sonra büyüme fonksiyonları kullanılarak algoritmanın verimliliği ifade edilir.

Algoritma Analizi

Anlamlı olan işlemler hakkında önemli not:

- Eğer problemin boyutu çok küçük ise algoritmanın verimliliğini muhtemelen ihmal edebiliriz.
- Algoritmanın zaman ve bellek gereksinimleri arasındaki ağırlığı dengelemek zorundayız.

Çalışma Zamanı fonksiyonu :T(n)

Çalışma zamanı veya koşma süresi (running time) fonksiyonu:

'n' boyutlu bir problemin algoritmasını çalıştırmak için gerekli zamandır ve **T(n)** ile gösterilir.

Başka bir ifadeyle **T(n)**: **bir programın veya algoritmanın işlevini yerine getirebilmesi için, döngü sayısı, toplama işlemi sayısı, atama sayısı gibi işlevlerden kaç adet yürütülmesini veren bir bağıntıdır**.

Çalışma Zamanı fonksiyonu :T(n)

Örnek: Basit ***if*** bildirimi

	Cost	Times
`if (n < 0)`	c1	1
`  absval = -n;`	c2	1
`else`		
`  absval = n;`	c3	1

Toplam maliyet <=c1+max(c2,c3)

Çalışma Zamanı fonksiyonu :T(n)

Döngüler (Loops)

Bir döngünün çalışma zamanı en çok döngü içindekideyimlerin çalışma zamanının iterasyon sayısıyla çarpılması kadardır.

İç içe döngüler (Nested Loops)

İç içe döngülerde grubunun içindeki deyimin toplam çalışma zamanı, deyimlerin çalışma sürelerinin bütün döngülerin boyutlarının çarpımı kadardır. Bu durumda analiz içten dışa doğru yapılır.

Ardışık deyimler

Her deyimin zamanı birbirine eklenir.

if/else

En kötü çalışma zamanı: test zamanına **then** veya **else** kısmındaki çalışma zamanının hangisi büyükse o kısım eklenir.

Çalışma Zamanı fonksiyonu :T(n)

Örnek: Basit bir döngü

	Maliyet	Tekrar
i = 1;	c1	1
sum = 0;	c2	1
while (i <= n) {	c3	n+1
i = i + 1;	c4	n
sum = sum + i;	c5	n
}		

Toplam maliyet=c1 + c2 + (n+1)*c3 + n*c4 + n*c5 =3n+3

T(n)=3n+3

Çalışma Zamanı fonksiyonu :T(n)

Örnek: İç içe döngü

	Maliyet	Tekrar
i=1;	c1	1
sum = 0;	c2	1
while (i <= n) {	c3	n+1
j=1;	c4	n
while (j <= n) {	c5	n*(n+1)
sum = sum + i;	c6	n*n
j = j + 1;	c7	n*n
}		
i = i +1;	c8	n
}		

Bu algoritma için gerekli zaman n^2 ile doğru orantılıdır.

Çalışma Zamanı fonksiyonu :T(n)

```
float ortalama_bul(float A[], int n)
{
        float ortalama, toplam=0;
        int k;
        for(k=0; k<n; k++)
                toplam=toplam+A[k];
        ortalama=toplam/n;
        return ortalama;
}
```

Çalışma Zamanı fonksiyonu :T(n)

Temel Hesap Birimi	İşlem	Tekrarı	Toplam
float ortalama_bul(float A[], int n)			
{			
float ortalama, toplam=0;			
int k;			
for(k=0; k<n; k++)	1,1,1	1,(n+1),n	2n+2
toplam=toplam+A[k];	1	n	n
ortalama=toplam/n;	1	1	1
return ortalama;	1	1	1
}			
			T(n)=3n+4

En iyi (Best), En kötü (Worst), Ortalama(Average) Durum Analizi

- **En iyi durum (best case):** Bir algoritma için, çalışma zamanı, maliyet veya karmaşıklık hesaplamalarında en iyi sonucun elde edildiği duruma “en iyi durum” denir. Bir giriş yapısında hızlı çalışan yavaş bir algoritma ile hile yapmak. (gerçek dışı) Ör: Bütün elemanların sıralı olduğu durum.

- **En kötü durum (worst case):** Tüm olumsuz koşulların oluşması durumunda algoritmanın çözüm üretmesi için gerekli maksimum çalışma zamanıdır. (genellikle). Ör: Bütün elemanlar ters sıralı.

- **Ortalama durum (avarage case):** Giriş parametrelerin en iyi ve en kötü durum arasında gelmesi ile ortaya çıkan durumda harcanan zamandır. Fakat bu her zaman ortalama durumu vermeyebilir. (bazen) Ör: Elemanların yaklaşık yarısı kendi sırasındadır.

Çalışma Zamanı fonksiyonu :T(n)

Temel Hesap Birimi	İşlem	Tekrarı	Toplam
float ortalama_bul(float A[], int n)			
{			
float ortalama, toplam=0;			
int k;			
for(k=0; k<n; k++)	1,1,1	1,(n+1),n	2n+2
toplam=toplam+A[k];	1	n	n
ortalama=toplam/n;	1	1	1
return ortalama;	1	1	1
}			
			T(n)=3n+4

Çalışma Zamanı fonksiyonu :T(n)

ÖRNEK: Bir dizi içerisindeki en küçük elemanı bulan ekbul adlı bir fonksiyonu C dilinde yazarak çalışma zamanı fonksiyonunu hesaplayınız? En iyi ve en kötü durum analizini yapınız?

Çalışma Zamanı fonksiyonu :T(n)

Çalışma Zamanı fonksiyonu :T(n)

Temel Hesap Birimi	İşlem	Tekrarı	Toplam