1.2. Algoritmalar

"Algoritma" kelimesi 9. yüzyılda yaşamış Horasan doğumlu matematikçi Ebu Cafer Muhammed İbn-i Musa el Harezmi'nin (al-Khowarizm) adından gelmektedir. M.S. 825 yıllarında "Kitab el cebr ve'l mukabele" başlığıyla çok etkili olmuş bir matematik ders kitabı yazmıştır. Eskiden kullanılmakta olan "algorizma" teriminin yerini bugünkü kullanımda "algoritma" kelimesinin almış bulunması "aritmetik" kelimesi ile kurulan bir ilişkiye bağlı olmalıdır. ("Cebir" kelimesinin de aynı kitabın başlığında yer alan Arapça el-cebr'den gelmiş olması ilginçtir). Aslında algoritmalar el Harezmi'nin kitabında çok önce de bilinmekteydiler. Eski Yunan döneminden (yaklaşık M.Ö. 300 den) beri bilinen ve *Eukleides algoritması* adıyla tanınan en iyi örnek, iki sayının en büyük ortak bölenini bulma yöntemidir (Tepedelenlioğlu 1993, Penrose 1997). Bu yöntem ve algoritma tasarımı ile ilgili bilgiler daha sonra ayrıntılı biçimde verilecektir.

Algoritma, belirli bir görevi yerine getiren sonlu sayıdaki işlemler dizisidir. Her algoritma aşağıdaki koşulları sağlamalıdır.

* **Girdi:** Sıfır veya daha fazla değer dışarıdan verilmeli.
* **Çıktı:** En azından bir değer üretilmeli.
* **Açıklık:** Her işlem (komut) açık olmalı ve farklı anlamlar içermemeli.
* **Sonluluk:** Her türlü durum göz önünde bulundurularak algoritma sonlu adımda bitmeli.
* **Etkinlik:** Her komut herhangi birisinin kalem ve kağıt ile yürütebileceği kadar basit olmalıdır.

Algoritma, bir problemin çözümünün adımlarını gösteren formal bir dilde yazılmış tasarımdır. Kullanılan dil Türkçe ve İngilizce gibi doğal bir dildir ama sınırlı bir yapısı vardır. Bir algoritmada bulunması gereken temel öğeler sıra, karar verme yapıları ve yinelemedir.

Sıra

Çözüm adımları bir sıraya dizilmiş olarak verilmelidir. Bu sıra çözüm adımlarının hangi sırayla uygulanacağını gösterir. Bir adımın içerdiği alt adımlar numaralama düzeniyle açıkça belirlenir.

Karar Verme Yapıları

Bazı çözüm adımları seçeneğe bağlı olarak uygulanır. Her adımın hangi koşulla uygulanacağı açıkça belli olmalıdır. Adım numaralarından seçeneğe bağlı olarak uygulanacak adımın konulan koşulun sağlanması ya da sağlanmaması durumunda mı uygulanacağı belli olmalıdır.

Yineleme

Bazı adımların birkaç kez yinelenmesi gerekebilir. Bu amaçla yinelenecek adımlar yineleme adımının alt adımları olmak zorundadır. Yineleme adımı, alt adımların hangi koşullarda yineleneceğini gösteren bir mantıksal ifade kullanmalıdır. Yineleme adımının yapısından yinelenecek adımların hangi koşullarda yineleneceği ve yineleme işleminin ne zaman duracağı belli olmalıdır.

Algoritmaların Numaralanması

Yukarıdaki temel öğeler aşağıdaki gibi algoritma adımlarını numaralama yöntemi ile başarılabilir:

Sıra

Sırayla çözülecek adımlara, birbirini izleyen sıra numaraları verilebilir. Örneğin, A, B ve C adımlarının sırayla uygulanacağını gösterir. Eğer bir adım, alt adımları kapsayacaksa, alt adımlar, o

adımın numarasını da kapsayacak şekilde numaralanır. Örneğin, A, B, B1, B2 ve C adımlarının sırayla uygulanacağını gösterir.

Karar Verme Yapıları

Seçeneğe bağlı uygulanacak adımlar aşağıdaki gibi numaralanabilir. Bu yapı, seçeneğe bağlı olarak, A adımının alt adımlarının uygulanacağını gösteren bir yapıdır. Bu adımı uygulamak için, A. adımdaki mantıksal-ifadenin doğru olup olmadığı bulunur. Eğer mantıksal-ifade doğruysa A.D. adımı uygulanır; doğru değilse A.Y. adımı uygulanır. Bazı karar verme yapılarında A.Y. adımı bulunmayabilir, bu durumda A adımındaki mantıksal-ifade yanlış ise uygulamak için A'dan sonraki A+1 adımına gidilir ve adımların uygulanması oradan devam eder. Bazen A.D. (veya A.Y.) birden fazla adım içerebilir, bu durumda bu adımlar A.D.1, A.D.2, ... gibi numaralandırılabilir.

Yineleme

Yineleme adımı için aşağıdaki gibi bir yapı kullanabiliriz. Bu yapıda mantıksal-ifade doğru olduğu sürece A1, A2,..., An adımları sırayla yinelenir. Her yinelemeden sonra mantıksal-ifade kontrol edilir ve eğer doğru ise yineleme işlemi devam eder; doğru değilse bir sonraki A+1. adıma gidilir.

1.3. Algoritma tasarımına örnekler

Bazı durumlarda verilen bir problemin çözümüne yönelik herhangi bir bilgimiz olmayabilir, bu durumda çözümü kendimizin oluşturması gerekir. Bazı durumlarda da daha önceden verilen bir çözüm yönteminin algoritma olarak tasarlanması gerekebilir. Bazı durumlarda aynı problemin çözümüne yönelik birden fazla algoritma tasarlanabilir. Bu durumlarda çözümden istenen şey ne ise ona göre algoritma seçimine karar vermek gerekir. Doğruluk (hassaslık) ya da hız önem kazanabilir veya her ikisi birden istenebilir. Bu kısımda çeşitli algoritma tasarımı örnekleri üzerinde durulup okuyucunun algoritma tasarımında nelere dikkat etmesi gerektiği kısaca anlatılacaktır.

Problem-1.1. İki Bilinmeyenli Denklem Sisteminin Köklerini Bulma

$a_1, a_2, b_1, b_2, c_1, c_2$ verildiğinde

$$a_1 x + b_1 y = c_1$$
$$a_2 x + b_2 y = c_2$$

denklem sisteminin çözümünü veren algoritmayı geliştirelim. Algoritmanın oluşturulabilmesi için girdilerin alınmasından sonra bir çözüm yönteminin geliştirilmesi gerekmektedir. Bu amaçla $a_1 x + b_1 y = c_1$ eşitliğinden

$$x = \frac{(c_1 - b_1 y)}{a_1}$$

olarak bulunan bu değer $a_2 x + b_2 y = c_2$ eşitliğinde yazılırsa,

$$a_1 b_2 - a_2 b_1 \neq 0$$

olmak üzere,

$$y = \frac{(a_1 c_2 - a_2 c_1)}{(a_1 b_2 - a_2 b_1)}$$

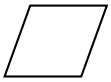
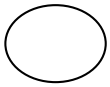
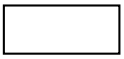
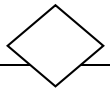
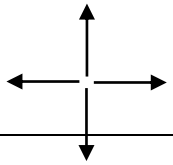
bulunur. Benzer şekilde bulunan bu değer $a_1 x + b_1 y = c_1$ eşitliğinde yazılırsa,

$$x = \frac{(c_1 b_2 - b_1 c_2)}{(a_1 b_2 - a_2 b_1)}$$

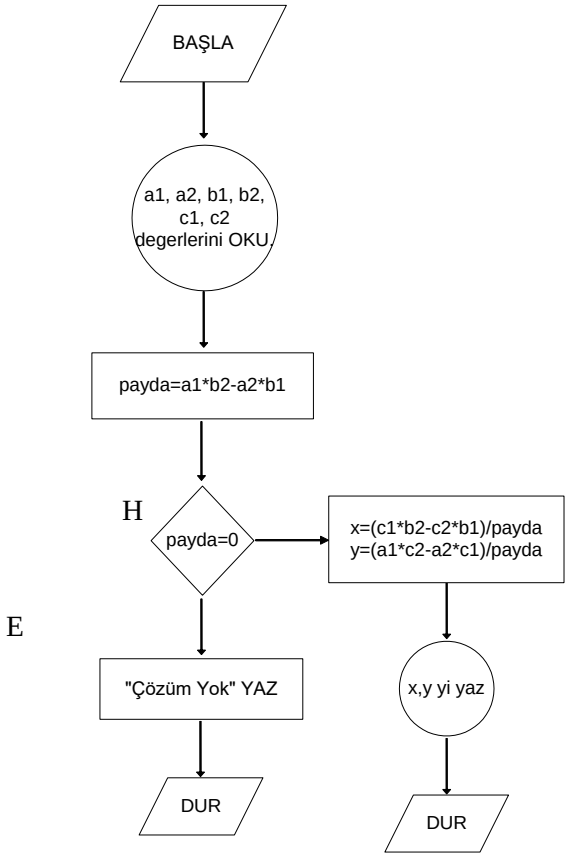
elde edilir. Burada dikkat edilmesi gereken nokta tek çözümün olabilmesi için $a_1 b_2 - a_2 b_1 \neq 0$ koşulunun sağlanması gerektiğidir.

Bazı durumlarda algoritmanın daha açık anlaşılabilmesi için akış şemaları kullanılmaktadır. Algoritma tasarımında biraz ustalaştıktan sonra akış şeması verilmeden doğrudan programlamaya geçilecektir. Akış şeması için basitçe Tablo 1.1.'de verilen şekillerden yararlanılır.

Tablo 1.1. Bir Algoritmanın akış şeması için kullanılan şekiller ve tanımları.

Şekil	Şekil Tanımı
	Algoritmanın *BAŞLA* ve *DUR* gösterimi için kullanılmaktadır.
	Algoritmada kullanılacak değişkenlerin *OKU*nması ve ekrana *YAZ*dırılması için kullanılmaktadır.
	Aritmetiksel İşlemler ve/veya değişken tanımlamalarında (atama işlemlerinde) kullanılmaktadır.
	Aritmetiksel ve mantıksal ifadeler için karar verme ve/veya karşılaştırma durumunu göstermektedir.
	Algoritmanın akış yönünü göstermek için kullanılmaktadır.

Bu şekiller kullanılarak Problem-1.1. ile verilen denklem sistemi çözümünün akış şeması aşağıdaki gibidir.



Algoritma adımları aşağıdaki gibidir.

A1. Girdilerin alınması (OKU)

a1, a2, b1, b2, c1 ve c2 değerlerini OKU

A2. payda=a1*b2-a2*b1

A3. EĞER payda=0 İSE
"Çözüm yok" iletisini YAZ
DUR

A4. x=(c1*b2-c2*b1)/payda
y=(a1*c2-a2*c1)/payda

A5. x ve y değerlerini YAZ

A6. DUR

Algoritmanın adımlarında görüldüğü gibi bazı kelimeler büyük harflerle yazılmıştır. Basic programlama dilinde,

OKU kelimesi INPUT deyimine
YAZ kelimesi PRINT deyimine
EĞER kelimesi IF deyimine
DUR kelimesi END deyimine

karşılık gelmektedir. Daha sonraki konularda bu deyimlerin ayrıntılarına girilecektir. Sadece yukarıdaki deyimler kullanılarak algoritmanın Basic programı aşağıdaki şekilde yazılabilir.

```
INPUT "a1=", a1
INPUT "a2=", a2
INPUT "b1=", b1
INPUT "b2=", b2
INPUT "c1=", c1
INPUT "c2=", c2
payda=a1 * b2 - a2 * b1
IF payda= 0 THEN
   PRINT "TEK ÇÖZÜM YOK"
   END
END IF
x = (c1 * b2 - c2* b1) / payda
y = (a1 * c2 - a2 * c1) / payda
PRINT "BULUNAN x DEĞERİ=", x
PRINT "BULUNAN y DEĞERİ=", y
END
```

INPUT ve PRINT deyimlerinde görülen " " işaretleri arasına ekranda görüntülenmesi istenilen iletiler yazılır. Programın ilk satırında yer alan ' işareti programa açıklama satırı eklemek için kullanılır. a1, a2, b1, b2, c1, c2, x ve y programda kullanılan değişkenlerdir.

IF …*deyimler*… END IF deyimi sorgulama yapmak için kullanılır. Bu sorgulama deyiminin basitten karmaşığa doğru kullanım biçimleri aşağıdaki gibidir.

a) IF *koşul(lar)* THEN *deyim(ler)*

b) IF *koşul(lar)* THEN *deyim(ler)* ELSE *deyim(ler)*

c) IF *koşul* THEN
    deyim bloğu-1
  ELSE
    deyim bloğu-2
  END IF

d) IF *koşul* THEN
    deyim(ler)
  ELSEIF
    deyim(ler)
  ELSE
    deyim(ler)
  END IF

e) IF *koşul-1* THEN
    IF *koşul-1.1* THEN
      deyim bloğu-1.1
    ELSE
      deyim bloğu-1.2
    END IF
  ELSE
    IF *koşul-2.1* THEN
      deyim bloğu-2.1
    ELSE
      deyim bloğu-2.2
    END IF
  END IF

IF…END IF deyiminin farklı kullanım biçimleri ile ileride verilen problemlerde karşılaşılacaktır.