BİLGİSAYARLARLA PROBLEM ÇÖZÜMÜ VE ALGORİTMA

Emrehan HALICI

Problemi tam olarak anlamak, onu yarı yarıya çözmek demektir. Bilgisayar kullanarak problem çözerken de aynı durum geçerlidir. İlk yapılacak iş problemi anlayarak analiz etmek ve çözüm yollarını ortaya koymaktır. Bilgisayarlar problemlerin nasıl çözüleceği konusunda insanlara henüz yardımcı olamamakta, insanların çözüm için gösterdiği yollardan giderek komutları hatasızca uygulamaktadır. Bilgisayar, kendisine gösterilen çözüm yolunda hiçbir belirsizlikle ve karışıklıkla yüz yüze gelmemelidir. Bu yüzden hazırlanacak çözüm yolunda her türlü detay bulunmalı ve karşılaşabilecek değişik durumlarda bilgisayarın çözüme nasıl devam edeceği bildirilmelidir.

Problem çözümü için bilgisayara verilen çözüm yöntemine "Algoritma" denir. Algoritma, "bir problemin nasıl çözüleceğini tarif etme yöntemi" şeklinde tanımlanabilir. Ancak herhangi bir tarifin algoritma olabilmesi için :

1) Sınırlı sayıda kurallardan oluşması,
2) Belirsizliklere yer vermemesi,
3) Sonlu sayıda adımdan sonra sona ermesi gerekir.

Algoritma örnekleri vermeden önce, algoritma olmayan bir örnek verelim :

Sayıların karesini almak :

1. A'ya O değerini ver
2. A'yı kendisiyle çarp ve B'ye bulduğun değeri ver.
3. B'yi yaz
4. A'nın değerini 1 arttır.
5. 2. adıma dön.

Yukarıdaki tarifte adımlar kolay anlaşılsın diye uzun uzun yazıyla açıklanmıştır. Oysa aynı tarif aşağıdaki gibi de yapılabilir :

1. A = O
2. B = A x A
3. B'yi yaz
4. A = A + 1
5. 2. adıma dön.

Burada 4. adımda A = A + 1 terimi anlamsız gibi görünebilir. Oysa bu cebirsel bir terim olmayıp, A'nın değerinin 1 fazlasının A'ya yükleneceği, yani A'nın değerinin 1 artırılacağı anlamına gelmektedir. Karışıklığa sebep olmaması için bu A ← A + 1 şeklinde de gösterilebilir.

Şimdi üstteki tarifin neden "algoritma" olmadığına bakalım. Tarif sonlu sayıda kuraldan oluşmasına rağmen hiçbir zaman sona eremeyecektir. Bu tarifi uygulamaya çalışan bir kişi ya da bilgisayar O'dan başlayarak sonsuza kadar tüm sayıların karesini almaya çalışacaktır.

Belli bir yerde durmayı sağlamak için tarife yeni bir adım ekleyelim. Acaba bu sefer "algoritma" özelliklerini gerçekleştirebilecek miyiz?

1. A = O
2. B = A x A
3. B'yi yaz
4. A = A + 1
5. A çok büyük bir sayı ise dur
6. 2. adıma dön.

Üstteki tarifi uygulamaya çalışan bir insan, kendi büyük sayı kavramına göre belirli bir yerde duracaktır. Bu sayı kimisi için 500.000, kimisi için 1.538 vb. olabilir. Ancak bilgisayar için "çok büyük bir sayı" nedir? Bu belirsizlikten dolayı üstteki tarifte bir algoritma değildir. Tıpkı bunun gibi 4. adımı da "A'nın değerini biraz arttır" şeklinde değiştirirsek, bir insan kendi yorumuna göre bu adımı uygulayabilir. Ama bilgisayar bu adımda da ne yapacağını bilemeyecek ve belirsizliğin içine düşecektir. Örnekteki tarifimizi algoritma haline getirmek için "çok büyük sayı"yı 1.000 olarak seçelim.

0'dan 1.000'e kadar olan sayıların karesini alan algoritma :

1. A = 0
2. B = A x A
3. B'yi yaz
4. A = A + 1
5. A > 1.000 ise dur
6. 2. adıma dön.

Algoritmalar içinde en bilineni büyük matematikçi Eucilid'in iki pozitif tam sayısının en büyük ortak bölenini bulan algoritmadır. İki pozitif tam sayının en büyük ortak böleni (EBOB) bu iki sayıyı birden bölen en büyük tam sayıdır. Örneğin 24 ve 30'un EBOB'i 6'dır.

Euclid'in EBOB algoritması :

1. Sayıları oku, sırasıyla A ve B'ye koy,
2. A'yı ve B'yi karşılaştır. Eğer sayılar eşitse EBOB bu sayılardan herhangi biridir. Sonucu yaz.
3. Eğer A, B'den küçükse sayıları değiştir.

(A'ya B'nin, B'ye de A'nın değerini koy.)

4. B'yi A'dan çıkar. A'ya B'nin değerini, B'ye de çıkarmanın sonucunu koy.
5. 2. adıma git.

Algoritmanın 2 ve 4 sayıları için nasıl çalıştığını aşağıdaki tabloyla inceleyelim :

Adım no	İşlem	İlk sayı (A)	İkinci sayı (B)
1.	Sayıları oku	2	4
2.	A ≠ B olduğu için devam et	2	4
3.	A < B olduğu için sayıları değiştir	4	2
4.	A←B, B←A—B	2	2
5.	2. adıma git	2	2
2.	A = B olduğu için dur.	(EBOB : 2)	

Dikkat edilirse bu algoritmada iki sayı devamlı olarak kullanılmakta, kendilerine birinci ve ikinci sayı (A ve B) denmektedir. Bunlar değişkendir ve algoritmanın değişik kullanışlarında değişik değerler alabilirler. Ayrıca algoritmaya dış dünyadan değerler girebilir ve bazı değerler dış dünyaya verilebilir. Girdi/çıktı işlemleri diyebileceğimiz bu işlemlere üstteki örnekte de rastlanmaktadır. İlk adımda iki sayı okunmakta ve bu sayılara A ve B denmektedir. Algoritmanın başka bir uygulamasında A ve B değişkenlerine değişik girdiler yapılabilecektir. İkinci adımda ise EBOB bulunduktan sonra sonuç yazılacaktır. Yani algoritma çıktı yapacaktır. Algoritmaların girdileri olmayabilir ama en azından bir çıktısı kesinlikle bulunmalıdır.

Karşılaşılan bir problemi algoritma yardımıyla çözmek için şu aşamalar takip edilir :

1. Problemin tanımlanması ve analizi
2. Algoritmada kullanılacak girdilerin ve sonuçta üretilecek çıktıların belirlenmesi
3. Algoritmada kullanılacak değişkenlerin belirlenmesi
4. Kurallara uygun olarak algoritmanın yazılması
5. Çeşitli değerler vererek algoritmanın test edilmesi

Herhangi bir problemin algoritmik olarak çözülebilmesi demek, bir bilgisayarın bu problemi, yeterli zaman ve yerin verilmesi koşulu ile çözebilmesi demektir. Bazı problemler vardır ki teorik olarak bilgisayar yardımıyla çözülebilmelerine karşın pratikte bu çözüm imkânsızdır. Örneğin çok iyi satranç oynayan hatta yenilmeyen bir bilgisayar programı teorik olarak yazılabilir. Satranç taşlarının tahta üzerinde kombinasyonları çok büyük olmasına rağmen sonlu bir sayıdır. Ve sonlu sayıda hamleden sonra oyun sona ermektedir. Program teorik olarak rakibin yapabileceği her hamleyi, bu hamlelere kendisinin verebileceği değişik cevapları, her cevaba rakibin yapabileceği değişik hamleleri vb. şekilde oyun sonuna kadar varacak tüm olasılıkları değerlendirerek en iyi hamleyi bulabilir. Çünkü her hamlesinin oyun sonuna kadar giden yollarını görmektedir. Satrançta oynanabilecek hamle dizilerinin sayısı yaklaşık 10^{20} (1 ve yanında 20 adet 0) olarak hesaplanmıştır. Bilgisayarın bir saniyede 1 milyon hamleyi deneyebileceğini varsaysak, bir senede ancak yaklaşık 3×10^{13} hamleyi deneyebilecektir (10^6 hamle/saniye x 3 x 10^7 saniye/yıl). Böylece tüm hamleleri deneyebilmesi için 3×10^6 (3 milyon) yıl gerekmektedir.

Dolayısıyla yenilmez bir satranç programının bilgisayarda çalışması günümüz için imkânsızdır.

Bilgisayarla problem çözerken yer ve zaman limitleri daima göz önünde bulundurulmalıdır. Çözümü gerçekleştirecek algoritma sayısı kuşkusuz birden fazladır. Ancak bunlar içinde en az yer ve en kısa zaman kullanan algoritmayı bulmaya çalışmalıdır. Teknolojik ilerlemelerle bilgisayarların hızı ve kapasitesi fiziksel olarak giderek artmaktadır. Bu gelişmelere sırt dayayarak algoritma ve programların üzerinde uğraşmadan rastgele yazılması çok yanlış olur. Yapılacak iş fiziksel gelişmelerin sınırlarını düşünerek algoritmalarında geliştirilmesi için çalışmak olmalıdır. ■