Algoritma

Bilgisayar adını verdiğimiz hızlı, dayanıklı, yorulmak usanmak bilmez, ama alabildiğine de aptal emir kulu, 1950'li yıllardan başlayarak insanoğlunun yaşamına girdiğinden beri, insanoğlu bilim ve teknolojide karşılaştığı sorunların çözümünde güvenilir bir yardımcıya kavuşmuş oldu. Böylece, eskiden her "işi" kendisi yaparken, şimdi insanoğlu, zamanını daha çok bu hamarat ama zekâca gelişmemiş emir kuluna işlerin nasıl yapılacağını onun anlayacağı bir dille tarif etme yöntemlerini araştırmaya ayırmaya başladı.

İşte çok kaba bir tanımlamayla "bir işin nasıl yapılacağını tarif etme yöntemi" anlamına gelen ***algoritma*** sözcüğü de 1950'li yıllardan başlayarak daha sık kullanılır oldu. Biz de bu bölümde yerimizin elverdiği ölçüde algoritmanın ne olduğunu, nasıl yazıldığını incelemeye çalışacağız.

Algoritma sözcüğüyle ilk kez bu bölümde karşılaşan bir kimse "Ah şu dizgi yanlışları yok mu, bak 400 yıllık logaritma sözcüğü ne hallere düşmüş!" diye kızabilir; ama bu kez dizgi yanlışı yok. Sözcük, Hindu-Arap rakamlarını kullanarak aritmetik hesap yapmak anlamına gelen "algorism" sözcüğünden geliyor. Ortaçağ Avrupası'nda algoristler algorism yöntemiyle, abacistler (çörkücüler) de abacusla (çörküyle) hesap yapıyorlardı. Algorism

sözcüğü ise ünlü İslam matematikçisi Harzem'li Muhammed Bin Musa'nın adından türemiş. "Al Hvorizmi (= Harzemli)" sözcüğü zamanla değişerek alhorism ve sonra da algorism olmuş[1]. Daha sonra da anlam değiştirerek algorithm (algoritma) olmuş.

Algoritmayı yukarda, çok kaba olarak, "tarif etme yöntemi" olarak tanımladık. Ancak modern anlamda algoritmayı öteki "yöntem", "yordam", "tarifname"ya da "reçete"lerden ayıran bazı özellikler var. Bu özelliklere geçmeden önce bir örnek görelim. Bu örnek Öklid algoritması adıyla bilinen ve iki tamsayının en büyük ortak bölenini bulan algoritma:

Öklid Algoritması. Bu algoritma *m* ve *n* gibi sıfırdan büyük iki tamsayısının en büyük ortak bölenini bulur.

1. *m*'yi ve *n*'yi oku.
2. *m*'yi *n*'ye böl. Kalana *r* de.
3. *r* = 0 ise dur. Yanıt *n*'dir; değilse 4'üncü adıma git.
4. *m*'nin yerine *n*'yi koy, *n*'nin yerine de *r*'yi koy.
5. 2'nci adıma git.

Şimdi kendimizi bir bilgisayar yerine koyarak *m* = 24, *n* = 18 için algoritmamızı izleyelim. Birinci adımdaki okuma işlemini yaptığımıza göre ikinci adıma geçerek 24'ün 18'e bölünmesinden kalanı bulacağız. Kalan 6'dır. Şu halde *r* = 6. 3'üncü adımda "*r* = 0 ise" koşulu var. Bu koşul sağlanmadığına göre dördüncü adıma geçeceğiz ve *m* = 18, *n* = 6 yapacağız. Şimdi be-

1 "Algorism'in temeli olan onlu sayı düzeni ve sıfır kavramını Batı'ya ilk tanıtan kitap, Harzemli'nin Latinceye "Numero Indorum" adıyla çevrilen "Kitab-ül Muhtasar fi Hesab-ül Hindi" adlı kitabıdır. Onlu sayı düzeninin Araplardan İtalyanlar aracılığıyla Avrupa'ya geçişiyle ilgili ek bilgi için "Ortaçağ Arapları" ve "Leonardo Fibonacci" bölümlerine bakınız. Ayrıca Güney Gönenç'in Elektrik Mühendisliği dergisinin Temmuz -Ağustos 1978 sayısında yayımlanan "Matematik Tarihine Kısa Bir Bakış" adlı yazısını da salık veririm.

şinci adım uyarınca ikinci adıma geçeceğiz ve $r = 0$ bulacağız. Bu kez $r = 0$ olduğuna göre üçüncü adım uyarınca duracağız ve yanıtın 6 olduğunu açıklayacağız. Gerçekten de 24'ü ve 18'i ortak bölen sayılar içinde en büyüğü 6'dır.

Şimdi Öklid algoritmasını da gözönünde tutarak herhangi bir yöntemin algoritma olabilmesi için sağlaması gereken koşullara bakalım. Herşeyden önce algoritmanın sonlu sayıda kuraldan oluştuğunu gözleyelim. Bu kurallar dizisinin en küçük numaradan başlayarak birer birer izlenmesi gerekiyor.

Sonlu sayıda kuraldan oluşmak bir yöntemin algoritma olması için yeterli değil. Aşağıdaki reçel tarifi de sonlu sayıda kuraldan oluşuyor ama algoritmaların sağlaması gereken ilerde göreceğimiz bazı özelliklerden yoksun[2].

Ayva Reçeli Tarifi. Bu tarif aşağıdaki malzemeyle size ayva reçeli yaptırır.

Malzeme: 1 kg ayva, 5-6 su bardağı şeker, 1 çay kaşığı limon tuzu.

1. Ayvayı dilimleyip soy.
2. Ortasını çıkarıp istediğin biçimde doğra (portakal dilimi, kuşbaşı veya rende).
3. Ayvayı yumuşayıncaya kadar iki su bardağı su içinde iyice haşla (suyu süzme).
4. Şeker ve limon tuzunu ilave et.
5. Yavaş yavaş kaynat.
6. Kıvama geldiyse dur. Gelmediyse 5'inci adıma git.

Bir algoritma başka ne gibi özelliklere sahip olmalı? Bunları şöyle, sıralayabiliriz:

---

2 Reçel tarifini Gönül Candaş'ın "Bereketli Olsun" adlı yemek kitabından aynen aldım. Ancak bir algoritma edası verebilmek için her kuralı ayrı bir madde başı yaptım, bir de fiil kiplerini değiştirdim.

1. Sonlu Olma Özelliği, Bir algoritma sonlu sayıda işlem yapıldıktan sonra sona ermeli. Burada sonlu sayıda kuraldan oluşmakla, sonlu sayıda işlem yapılmak birbiriyle karıştırılmamalı. Örneğin aşağıdaki kurallar dizisi sonlu sayıda kural içerdiği halde sonlu sayıda işlemden sonra sona ermez:

1. *n*'yi 0 yap.
2. *n*'yi 1 artır.
3. *n*'nin değerini yaz.
4. *n* = 0 ise dur.
5. 2'nci adıma dön.

Oysa Öklid algoritmasında 2'nci adımda hesaplanacak *r* değeri daima *n*'den küçük olduğu için (neden?) ve bu *r* değeri sıfırdan farklı olduğu sürece 4'ncü adımda *n*'nin yerine konacağı için, *n*'nin değeri sürekli küçülecek ve 2'nci adımdan sonlu sayıda geçişten sonra *r* = 0 bulunacak ve böylece 3'üncü adım gereğince durulacaktır.

Bu özellik ayva reçeli için de geçerlidir. Belli bir (sonlu) zaman sonra reçel "kıvama" gelecek (ya da yanacak!) ve durulacaktır.

2. Kesin Olma Özelliği. Bir algoritmanın her adımı açık ve kesin bir biçimde tanımlanmış olmalıdır. Bu şekilde algoritmayı izlerken hiçbir yerde yoruma meydan verilmemelidir (unutmamalı ki algoritmayı okuyup izleyecek emir kulu, hamarat ama zekâca gelişmemiş bir yaratıktır). Şu halde algoritmanın hiçbir yerinde "*n*'nin değerini biraz arttır", "yumuşayıncaya kadar haşla", "yavaş yavaş kaynat", "kıvama geldiyse dur" türünden kurallar olamaz. "Biraz" nedir? *n*'nin değerini 1 mi artıracağız, yoksa 2 mi? Ayvalar ne kadar yumuşayacak, ateş ne kadar yavaş olacak? Hele reçelin kıvama geldiğini nasıl anlayacağız[3]?

---

3 Burada hem Sayın Candaş'ın hakkını yememek hem de bu bölümü hiç olmazsa bir reçel tarifi edinmek amacıyla okuyan okurların merakını tatmin etmek için belirteyim, bu konu reçel tarifinin hemen altında bir dipnotta şöyle açıklanıyor: Reçelden bir miktar bir tabağa koyacaksınız. Tabağı yavaşça bir sağa bir de sola eğeceksiniz. Eğer reçel tabağa sıvaşıp kalıyorsa kıvama gelmiş demektir.

Öte yandan açıklık ve kesinliği sağlamak için algoritma mutlaka bilgisayarın anlayacağı bir dille yazılmalıdır. Örneğin yukarıdaki Öklid algoritması alabildiğince açık ve kesin bir dille yazıldığı halde (umarım böyledir) Türkçe bilmeyen bir kişi tarafından anlaşılmayacaktır. Ayrıca algoritmayı izleyecek bilgisayarın yeteneği algoritmada geçen komutları anlayacak düzeyde olmalıdır.

Örneğin bilgisayar bölme bilmiyorsa, Öklid algoritmasında 2'nci adımdaki "böl" komutu bilgisayarca anlaşılmayacaktır. Bölme bilmeyen, ama çıkarma bilen bir bilgisayar için Öklid algoritmasını şu şekilde yazmak gerekecektir:

Öklid Algoritması (bölme bilmeyen bilgisayarlar için)

1. *m*'yi ve *n*'yi oku.
2. *m*'den *n*'yi çıkar. Farka *r* de.
3. *r* negatifse *m*'nin yerine *n*'yi, *n*'nin yerine *m*'yi koy. 2'nci adıma git; değilse 4'ncü adıma git.
4. $r < n$ ise 7'nci adıma git; değilse, 5'inci adıma git.
5. *r*'den *n*'yi çıkar. Farkı *r*'nin yerine koy.
6. 4'üncü adıma git.
7. $r = 0$ ise dur. Yanıt *n*'dir; değilse 8'inci adıma git.
8. *m*'nin yerine *n*'yi koy, *n*'nin yerine de *r*'yi koy.
9. 2'nci adıma git.

Bu ikinci Öklid algoritmasının kutu içine alınmış bölümünün yalnızca çıkarma işlemi kullanan bir bölme işlemi olduğu okurun dikkatinden kaçmayacaktır.

Yukarda belirttiğim, insanla bilgisayarın birbirlerini açık ve kesin bir biçimde anlaması sorununun çözümü için adına "programlama dili" ya da "bilgisayar dili" denilen "resmi" diller tasarımlanmış bulunuyor. Her bilgisayarın kendi büyüklüğüne ve yetenek düzeyine göre, anladığı bir ya da birkaç "resmi" dili var. Algoritmaların bu dillerle yazılması gerekiyor.

3. Algoritmanın Girdisi. Bazı algoritmaların bir ya da birkaç girdisi, yani algoritmayı izleyecek bilgisayarın istenileni yapabilmesi için işe başlamadan okuması gereken nicelik ya da nicelikler söz konusu olabilir. Örneğin Öklid algoritmasında, 1'nci adımda bilgisayara m ve n sayılarını okuması komutu veriliyor. Şu halde Öklid algoritmasının iki girdisi var.

Bu girdilerin belirli bir nicelik kümesinden seçilmesi gerekiyor. Örneğin gene Öklid algoritmasında m ve n'nin pozitif tamsayılar olması gerekli. Aksi takdirde algoritma doğru çalışmayacak.

4. Algoritmanın Çıktısı. Bir algoritmanın bir ya da birkaç çıktısı (ürünü) olacaktır. Öklid algoritmasında çıktı m ile n'nin en büyük ortak böleni olan tamsayıdır.

Yukardaki bütün bu özelliklere ek olarak bir algoritmanın mümkün olduğu kadar geniş bir problem kümesine uygulanabilir olması istenir. Örneğin aşağıdaki algoritma yukarda sıraladığımız tüm özellikleri sağladığı halde çok özel bir problemi çözdüğü için tümüyle yararsızdır:

Denklem Çözümü I: Bu algoritma $x + 2 = 5$ denklemini çözer.

1. 3 yaz.
2. Dur.

Bu algoritmanın biraz yararlı olabilmesi için çözebildiği problemlerin kümesini genişletmek gerekir. Şimdi öyle bir algoritma yazalım ki, sıfır ya da sıfırdan büyük olmak üzere herhangi bir a, b tamsayı çifti için $x + a = b$ denklemini sağlayan x değerini bulsun ve bu değeri yazsın. Böyle bir x değeri yoksa "yanıt yok" yazsın. Bu yeni algoritma, $a = 2$, $b = 5$ için yukarıdaki algoritmayı da bir özel hal olarak içereceği için çok daha esnek ve kullanışlı olacaktır.

Yazacağımız algoritmayı izleyecek bilgisayar çıkarma biliyorsa sorun çok kolaydır:

Denklem çözümü II: Bu algoritma $x + a = b$ denklemini çözer.

1. a'yı ve b'yi oku.

2. b'den a'yı çıkar. Farka x de.

3. x'in değerini yaz.

4. Dur.

Eğer bilgisayar çıkarma bilmiyor da yalnızca toplama biliyorsa aşağıdaki kurallar dizisini deneyebiliriz:

1. a'yı ve b'yi oku.

2. x'e 0 değerini ver.

3. x'le a'yı topla. Toplama t de.

4. $t = b$ ise dur. Yanıt x'tir; değilse 5'nci adıma git.

5. x'in değerini 1 arttır.

6. 3'üncü adıma git.

Görüldüğü gibi yukarıdaki kurallar dizisini izleyen bilgisayar sırayla a, $a + 1$, $a + 2$, ... sayılarını hesaplayacak ve bunları b ile karşılaştıracaktır. Eğer bu sayılardan biri b'ye eşitse aranan x değeri odur.

Yukarıdaki kurallar dizisine inatla "kurallar dizisi" dediğim, ama algoritma demediğim okurun dikkatini çekmiştir sanırım. Çünkü a ve b, denklemi sağlayan sıfır ya da sıfırdan büyük bir x değeri bulunamayacak şekilde verildiğinde (örneğin $a = 4$, $b = 3$), dizi bu durumu sezemeyecek ve 4, 5, 6, ... sayılarının 3'e eşit olup olmadığına bakmayı sürdürecek ve hiç durmayacaktır (dizi sonlu olma özelliğini sağlamamaktadır).

Eğer bilgisayar iki sayıyı karşılaştırıp bunlardan birinin diğerinden büyük olup olmadığını anlayabilme yeteneğine sahipse yukarıdaki diziyi kolayca algoritma haline getirebiliriz:

Denklem Çözümü III (çıkarma bilmeyen bilgisayarlar için)**:**

1. a'yı ve b'yi oku.

2. x'e 0 değerini ver.

3. *x*'le *a*'yı topla, toplama *t*'de.
4. $t > b$ ise dur, "yanıt yok" yaz; değilse 5'nci adıma git.
5. $t = b$ ise dur. Yanıt x'tir; değilse 6'ncı adıma git.
6. *x*'in değerini 1 arttır.
7. 3'üncü adıma git.

Bilgisayar büyük-küçük karşılaştırılması yapamıyorsa da bu diziyi algoritma haline getirebiliriz, ama bu biraz daha zor olur:

Denklem Çözümü IV (çıkarma bilmedikleri gibi büyük-küçük karşılaştırması da yapamayan bilgisayarlar için)**:**

1. *a*'yı ve *b*'yi oku.
2. *x*'e 0 değerini ver.
3. *x*'le *a*'yı topla, toplama *t* de.
4. $t = b$ ise dur, yanıt *x*'tir;değilse 5'nci adıma git.
5. *x*'le *b*'yi topla, toplama *s* de.
6. $s = a$ ise dur. "Yanıt yok" yaz; değilse 7'nci adıma git.
7. *x*'in değerini 1 arttır.
8. 3'üncü adıma dön.

Bu bölümü bitirirken, algoritma kavramının en önemli sorununun, kesinlik sorunu olduğunu bir daha vurgulamak isterim. Bilgisayarın hiçbir belirsizliğe yol açmadan bizi anlayabilmesi için bilgisayarın düzeyini (hangi işlemi yapıp hangi işlemi yapamayacağını) ve dilini iyi bilmek ve algoritmayı ona göre yazmak gerekir. Denklem çözümü algoritmaları bunu yeterince gösteriyor sanırım.

Öte yandan yemek ustası bir ev hanımına çok açık seçik gelecek ve unutulmaz ayva reçelleri yememizi sağlayacak bir reçel tarifi aptal bilgisayarın hiç içinden çıkamayacağı kadar belirsiz ve karmaşık olabilir. Bu nedenle algoritma yorum unsurunu tümüyle ortadan kaldıran kesin olma özelliğine sahip olmalıdır.

Dilerim, ayva reçeli pişirebilen bilgisayarlarınız olsun.

EKLER

Öklid algoritmasının, nasıl olup da m ve n gibi iki tamsayının en büyük ortak bölenini verdiğini kolayca kanıtlayabiliriz. Aşağıdaki kanıtta geçen adım numaraları yukarda ilk verilen Öklid algoritmasındaki adım numaralarıdır.

İkinci adımdan sonra $m = bn + r$ yazılabilir. Burada b bölüm, r'de m'nin n'ye bölünmesinden kalandır ($m < n$ ise $b = 0$, $r = m$ olacaktır). Şimdi, eğer $r = 0$ ise m, n'ye tam bölünüyor demektir ki, bu durumda en büyük ortak bölen gerçekten n'dir. Eğer $r \neq 0$ ise, m'nin ve n'nin herhangi bir ortak böleni $r = m - bn$'yi de böler (neden?) Öte yandan hem n'yi hem de r'yi bölen bir sayı $m = bn - r$'yi de böler (neden?) Şu halde m ile n'nin ortak bölenlerinin kümesi ile n ile r'nin ortak bölenlerinin kümesi aynıdır; öyleyse en büyük ortak bölenler de aynıdır. Bu nedenle 4'üncü adımdaki yer değiştirme, problemin yanıtını değiştirmez.