Bilgisayar Bilimi Köşesi

Chris Stephenson ve Ali Nesin*
cs@cs.bilgi.edu.tr, anesin@bilgi.edu.tr

Donald Knuth'un Bir Sorusu

Geçen sayımızda Donal Knuth'un şu problemminden sözetmiştik: 1'den 50'ye kadar olan tamsayıları öyle iki kümeye ayıralım ki, kümelerdeki sayıların kareköklerinin toplamı birbirine en yakın olsun. Yani problem, $A \cap B = \emptyset$ ve $A \cup B = \{1, 2, ..., 50\}$ koşullarını sağlayan A ve B kümeleri arasında, $|\sum_{a \in A} \sqrt{a} - \sum_{b \in B} \sqrt{b}|$ sayısını en küçük yapan A ve B'yi bulmak ve bunu en çağdaş bilgisayarlarla 10 saniye gibi kısa bir sürede yapmaktı...

Sayıların kareköklerini almak pek zor değildir, gerçek kareköklere kesirli sayılarla dilediğimiz kadar ve oldukça kısa sürede yaklaşabiliriz. Şimdi bu elli karekökü iki kümeye ayırmak gerekiyor. 1'in karekökü olan 1'i kümelerden birine koyarsak, geri kalan 49 sayıyı iki kümeye dağıtmamız lazım, ki bu da 2^{49} değişik biçimde yapılabilir, yani A ve B kümeleri için tam 2^{49} seçimimiz var. İşte Donald Knuth'un sorusunun ilk yarısının bir çözümü: Bilgisayarımıza tüm bu 2^{49} seçeneği teker teker buldurturuz, iki kümenin sayılarının kareköklerini ayrı ayrı toplarız, bu iki toplamın farkını alırız, farkları bir kenara yazarız ve nihayet en küçük farkı veren A ve B kümelerini seçeriz... Ancak bu çok zaman alır, çünkü 2^{49} çok büyük bir sayıdır, tam 14 rakamı vardır ve hiçbir çağdaş bilgisayar bu işlemleri 10 saniyede yapamaz.

1'den 50'ye kadar olan sayıların kareköklerini alacağımıza, n gerçel sayı alalım ve aynı problemi bu n gerçel sayı için çözmeye çalışalım. Bu n gerçel sayıdan herhangi ikisi birbirine eşit olabilsin, bu konuda bir varsayımımız olmasın.

Problemin mutlak olmasa da uygulamada oldukça çabuk yaklaşık sonuçlar veren çözümleri vardır. Yaklaşık çözümlerden en ilginci Karmarkar-Karp adıyla anılan KK-algoritmasıdır. Sayı adedi n arttıkça, yani problem zorlaştıkça KK-algoritması daha iyi sonuç verir; öte yandan n azaldığında algoritma çok kötü sonuçlar verebilir.

Sayı kümemize S diyelim. S'nin sayılarını iki kümeye ayıracağız. Bir anlamda, S'nin eleman sayısı n üzerine tümevarımla, $|\sum_{a \in A} a - \sum_{b \in B} b|$ sayısını en küçük yaptığını düşündüğümüz S'nın ayrık A ve B altkümelerini bulacağız.

Eğer $n = 1$ ise, çözüm belli: $A = S$, $B = \emptyset$ olsun. Bu kolaydı. Şimdi $n > 1$ durumunda ne yapacağımızı düşünelim.

S'nin en büyük iki sayısına s_0 ve t_0 diyelim. Varsayalım ki $s_0 \geq t_0$. En sonda, çözümü bulduğumuzda, bu sayılardan biri A'da biri B'de olacak. Şimdi S kümesinden s_0 ve t_0 sayılarını silip, bu sayıların yerine $s_0 - t_0$ sayısını koyalım, yani S yerine $(S \setminus \{s_0, t_0\}) \cup \{s_0 - t_0\}$ sayı kümesini alalım. Bu son kümenin S'den bir eksik elemanı vardır. Tümevarımı uygulayalım. $S_1 = (S \setminus \{s_0, t_0\}) \cup \{s_0 - t_0\}$ olsun. Benzer biçimde S_2, S_3, ... kümelerini de bulabiliriz. Her seferinde eleman sayısı bir azalır.

Bunu böyle sürdüre sürdüre, S'nin eleman sayısını birer birer indirerek tek elemanlı bir S_n kümesine varırız ki, bu küme için A_n ve B_n altkümelerinin ne olması gerektiği belli, yukarda görmüştük bunu. Şimdi algoritma geri dönüp, A_n ve B_n altkümelerinden hareketle, aradığımız A ve B altkümelerini bulmalı. A_1 ve B_1'den A ve B'yi bulmasını bilmek yeterli elbet.

S_1 kümesinde S'den bir eksik eleman olduğundan, (daha bilmediğimiz!) KK-algoritmasının inancına göre, S_1 kümesini, sayılarının toplamlarının farkının en az çıktığını düşündüğümüz A_1 ve B_1 diye iki ayrık altkümeye ayırdığımızı varsayabiliriz. Şimdi, S_1, A_1 ve B_1 altkümelerinden A ve B altkümelerini inşa etmeliyiz. Yazacağımız program, S'den çıkardığı s_0 ve t_0 sayılarını da aklında tutsun. Özetle:

Bildiklerimiz: S_1, A_1, B_1, s_0, t_0.

Bulmak istediklerimiz: A, B.

A_1 ve B_1 kümelerinden birinde mutlaka $s_0 - t_0$ sayısı vardır. Diğerinde olabilir de olmayabilir de.

* İstanbul Bilgi Üniversitesi öğretim üyesi.

Diyelim bu eleman A_1'de ama B_1'de değil. O zaman $A = (A_1 \setminus \{s_0 - t_0\}) \cup \{s_0\}$ ve $B = B_1 \cup \{t_0\}$ olmalı. Eğer $s_0 - t_0 \in B_1 \setminus A_1$ olsaydı, $A = A_1 \cup \{t_0\}$ ve $B = (B_1 \setminus \{s_0 - t_0\}) \cup \{s_0\}$ olacaktı. Şimdi $s_0 - t_0$ sayısının hem A_1'de hem de B_1'de olduğunu varsayalım. O zaman

$$\sum_{a \in A_1} a \text{ ve } \sum_{b \in B_1} b$$

sayılarını karşılaştıralım. Eğer

$$\sum_{a \in A_1} a \le \sum_{b \in B_1} b$$

ise s_0'ı A_1'e t_0'ı B_1'e koyalım ve A_1'deki $s_0 - t_0$'ı atalım, aksi halde tam tersini yapalım. Demek ki A_1 ve B_1 altkümelerinden A ve B kümelerini bulabiliyoruz.

Aynı biçimde A_{i+1} ve B_{i+1} altkümelerinden A_i ve B_i altkümelerini de bulabiliriz:

$s_i - t_i \in A_{i+1} \setminus B_{i+1}$ ise $A_i = (A_{i+1} \setminus \{s_i - t_i\}) \cup \{s_i\}$ ve $B_i = B_{i+1} \cup \{t_i\}$,
$s_i - t_i \in B_{i+1} \setminus A_{i+1}$ ise $A_i = A_{i+1} \cup \{t_i\}$ ve $B_i = (B_{i+1} \setminus \{s_i - t_i\}) \cup \{s_i\}$,

$s_i - t_i \in A_{i+1} \cap B_{i+1}$ ve $\sum_{a \in A_{i+1}} a \le \sum_{b \in B_{i+1}} b$ ise, $A_i = (A_{i+1} \setminus \{s_i - t_i\}) \cup \{s_i\}$ ve $B_i = B_{i+1} \cup \{t_i\}$,

$s_i - t_i \in A_{i+1} \cap B_{i+1}$ ve $\sum_{a \in A_{i+1}} a > \sum_{b \in B_{i+1}} b$ ise, $A_i = A_{i+1} \cup \{t_i\}$ ve $B_i = (B_{i+1} \setminus \{s_i - t_i\}) \cup \{s_i\}$.

Bunu böyle devam ettirirsek, en son bulacağımız A_0 ve B_0 kümeleri aradığımız A ve B kümeleri olur.

Bir örneğe bakmanın zamanı geldi.

$S = \{10, 9, 8, 6, 5\}$ olsun. Demek ki $s_0 = 10$, $t_0 = 9$ (en büyük iki eleman) ve $S_1 = \{8, 6, 5, 1\}$. Bir sonraki aşamada $s_1 = 8$, $t_1 = 6$ ve $S_2 = \{5, 2, 1\}$ olacak. Bir sonraki aşamada $s_2 = 5$, $t_2 = 2$ ve $S_3 = \{3, 1\}$ olacak. En son aşamada $s_3 = 3$, $t_3 = 1$ ve $S_4 = \{2\}$ olacak:

$S = \{10, 9, 8, 6, 5\}$
$s_0 = 10$, $t_0 = 9$, $S_1 = \{8, 6, 5, 1\}$
$s_1 = 8$, $t_1 = 6$, $S_2 = \{5, 2, 1\}$
$s_2 = 5$, $t_2 = 2$, $S_3 = \{3, 1\}$
$s_3 = 3$, $t_3 = 1$, $S_4 = \{2\}$

Yazacağımız program s_i ve t_i elemanlarını ($i = 0, 1, 2, 3$) aklında tutacak ve S_4 kümesinden başlayarak yavaş yavaş A ve B kümelerini bulacak.

Şimdi yukarıdaki örnekte, tersten, yani S_4 kümesinden başlayarak A ve B'yi bulalım, aşağıdaki karede yaptığımız gibi.

$S_4 = \{2\}$,	$A_4 = \{2\}$,	$B_4 = \emptyset$,	$s_3 = 3$,	$t_3 = 1$.
$S_3 = \{3, 1\}$,	$A_3 = \{3\}$,	$B_3 = \{1\}$,	$s_2 = 5$,	$t_2 = 2$.
$S_2 = \{5, 2, 1\}$,	$A_2 = \{5\}$,	$B_2 = \{1, 2\}$,	$s_1 = 8$,	$t_1 = 6$.
$S_1 = \{8, 6, 5, 1\}$,	$A_1 = \{5, 6\}$,	$B_1 = \{1, 8\}$,	$s_0 = 10$,	$t_0 = 9$.
$S_0 = \{10, 9, 8, 6, 5\}$,	$A_0 = \{5, 6, 9\}$,	$B_0 = \{8, 10\}$.		

Böylece $A = A_0 = \{5, 6, 9\}$ ve $B = B_0 = \{8, 10\}$ bulunur. A'daki sayıların toplamı $5 + 6 + 9 = 20$, B'deki sayıların toplamı $8 + 10 = 18$, aradaki fark da $20 - 18 = 2$, yani sonuncu $S_4 = \{2\}$ kümesinin tek elemanı...

Görüldüğü gibi oldukça çabuk bir algoritma, ve binlerce seçenekten en iyisini seçmiyor, tek çırpıda "yanıt"ı buluyor.

Ama bu algoritma her zaman en iyi sonucu vermiyor. Örneğin yukarda bulduğumuz sonuç en iyi sonuç değil, $A = \{10, 9\}$ ve $B = \{8, 6, 5\}$ daha iyi bir sonuç, fark tam 0.

Öte yandan S'nin eleman sayısı arttıkça, yani problem zorlaştıkça bu algoritma çok daha mutlak sonuca yakın sonuçlar veriyor.

Tim Peters, en son moda Python dilinde KK-algoritmasının bir uygulaması yazdı (http://uselesspython.com/Karp.py).

Örneğin $S = \{\sqrt{0}, \sqrt{1}, \sqrt{2}, ..., \sqrt{49}\}$ ise A'yı {0, 3, 5, 6, 8, 11, 13, 14, 17, 19, 20, 22, 25, 27, 28, 30, 33, 34, 37, 39, 40, 43, 44, 46, 49} sayılarının karekökü olarak, B'yi de geri kalan {1, 2, 4, 7, 9, 10, 12, 15, 16, 18, 21, 23, 24, 26, 29, 31, 32, 35, 36, 38, 41, 42, 45, 47, 48} sayılarının karekökü olarak alıyor. Birinci kümedeki sayıların toplamı 119,517919732..., ikinci kümedeki sayıların toplamı 119,517880871... çıkıyor. Aradaki fark görüldüğü gibi çok çok küçük.

Ancak Peters'ın programında birkaç sorun bulduk. Peters'ın programı bizim yukarda bulduğumuz programdan biraz daha karmaşık, çizgeleri boyamak gibi oldukça karmaşık bir işlemden geçiyor; ayrıca uygulamada $O(n \log n)$ yerine $O(n^2)$ zamanda sonuç veriyor. Peters'in programının daha basitini yazdık ve böylece 50 satırlık program 23 satıra indi. Ayrıca programımız $O(n \log n)$ zamanda çalışıyor (www.cs.bilgi.edu.tr). Elbette her iki program da aynı sonuçları buluyor. ♦