kifmd

Kilis 7 Aralık Üniversitesi Fen ve Mühendislik Dergisi
2017 Cilt 1 Sayı 2

Kombinasyon Sayımlama İçin Bir Özet Fonksiyon Önerisi

Mehmet Oğuz YARDIMCI
Bilgisayar Bilimleri Bölümü, Fen Fakültesi, Dokuz Eylül Üniversitesi, İzmir, Türkiye
moguzyardimci@gmail.com

Zeynep Nihan BERBERLER
Bilgisayar Bilimleri Bölümü, Fen Fakültesi, Dokuz Eylül Üniversitesi, İzmir, Türkiye
zeynep.berberler@deu.edu.tr

Özet

Kombinasyon n elemanlı bir kümenin r elemanlı alt kümelerinin kaç farklı şekilde düzenlenebileceğini bulmak için kullanılan matematiksel bir ifadedir. Sadece bu düzenlemelerin sayısı ile ilgilenilmeyip aynı zamanda yazdırılması da gerekirse literatürde çözüme yönelik birçok uygulama örneği bulunabilen, özyinelemeli algoritma tekniği ile bu amaca ulaşılabilir ve hatta istenirse düzenlemeler alfabetik bir sırada da yazdırılabilir. Bu çalışmada literatüre katkı olarak r elemanlı alt kümelerden herhangi birisinin alfabetik sırada kaçıncı olduğu sayımlama yapılmadan, önerilen özet fonksiyon yardımıyla hesaplanmıştır ve özyinelemeli algoritma tekniğinin kullanıldığı geleneksel yöntemle karşılaştırılmalı hesaplama denemeleri yapılarak önerilen özet fonksiyonun etkinliği test edilmiştir.

Giriş

Kombinasyon hesabı yani *n* elemanlı bir kümenin *r* elemanlı alt kümelerinin sayısının bulunması bilgisayar bilimlerinde temel algoritma ve programlama konusuna ilgi duymuş hemen herkesin örneğini gördüğü ve belki de programladığı formülü (1) denkleminde verilen en basit örneklerden birisidir. [1][2]

$$C(n,r) = \binom{n}{r} = \frac{n!}{(n-r)!r!} \tag{1}$$

Eğer aynı örnek daha ileri seviyede karşılaşılan algoritmaların tasarımı ve analizi konsepti içinde ele alınırsa bu kez hem sayısal değerinin hesaplanması için daha az karmaşıklık gerektirecek yaklaşımlar tartışılır hem de *r* elemanlı alt kümelerin yazdırılması için özyinelemeli algoritma tekniklerinden bahsedilir. [3][4]

Daha az karmaşıklık için (1) denkleminde sadeleşen terimlerin olduğu gerçeği atlanmadığı taktirde kesrin payında yer alan ifadenin nasıl hesaplanacağına karar verilmesi gerektiği önem kazanmaktadır. Bunun için de *n!* ifadesini hesaplamak yerine paydada yer alan en büyük terime kadar olan kısmı hesaplamak işlem karmaşıklığını azaltacaktır. Örneğin *(n-r)!* teriminin *r!* teriminden daha büyük olduğu kabul edilsin, böylece kesrin pay kısmında yer alan *n!* ifadesi *n(n-1)(n-2)...(n-r+1)(n-r)!* şeklinde yazılarak kesrin paydasında yer alan *(n-r)!* terimiyle denklem (2)'de görüldüğü gibi sadeleştirilebilir. Öte yandan eğer *r!* terimi *(n-r)!* teriminden daha büyük ise yine çok benzer şekilde (2) denklemindeki hesaplamanın yapılacağı aşikardır.

$$\binom{n}{r} = \frac{n(n-1)(n-2)\ldots(n-r+1)\cancel{(n-r)!}}{\cancel{(n-r)!}r!}$$

Eğer *n* elemanlı bir kümenin *r* elemanlı alt kümelerinin sadece sayısı ile ilgilenilmeyip aynı zamanda bu kümelerin yazdırılması da isteniyorsa literatürde çözüme yönelik birçok yaklaşım mevcuttur ve hatta bu kümelerin alfabetik sırada yazdırılması da mümkündür [5][6][7]. Alfabetik sırada yazdırılan küme listesi içinde özel olarak ilgilenilen kümenin kaçıncı sırada olduğunu kümeleri yazdırmadan bulan bir yönteme yazarlar tarafından literatürde rastlanmamıştır. Bu çalışmada önerilecek özet fonksiyon ile alt kümelerin yazdırılmasına gerek olmadan özel olarak ilgilenilen kümenin alfabetik sıralamada kaçıncı sırada olduğu etkin bir şekilde bulunabilecektir. Örneğin *n=49* ve *r=6* seçildiğinde *A*={2,3,5,7,11,13} alt kümesinin tüm kombinasyonlar içinde alfabetik sırada kaçıncı olduğu "hızlı" bir şekilde bulunabilir ki burada cevap 1727519 'dur. Ayrıca hesaplama denemeleri bölümünde de önerilen özet fonksiyon ile geleneksel özyinelemeli yöntemin karşılaştırılması yapılmış ve etkinliği CPU zamanı ölçüt alınarak nesnel olarak test edilmiştir, Tablo 4: n=49 ve r=6,7,8,9,10,11,12 için süreler ve Şekil 1. özyinelemeli algoritma için CPU süreleri'de özetlenmiştir.

1. Yöntem

Yöntemi basitçe anlatabilmek için *n=4* seçildiğinde olası *r* değerlerine karşılık elde edilecek kümeler aşağıda Tablo 1'de görülmektedir. Burada *r=1* ve *r=4* alındığında özel durumlar oluşacağı ve yapı çok basit olacağı için açıklamaya gerek duyulmamıştır. Oysa *r=2* seçildiğinde alfabetik sırayla oluşacak **{{1,2},{1,3},{1,4},{2,3},{2,4},{3,4}}** 6 adet alt kümenin kendi içinde birinci terime göre sırasıyla 3, 2 ve 1 eleman içeren 3 gruba ayrıldığı görülmektedir. Benzer şekilde *r=3* seçildiğinde alfabetik sırayla oluşacak **{{1,2,3},{1,2,4},{1,3,4},{2,3,4}}** 4 adet alt kümenin kendi içinde birinci terime göre sırasıyla 3 ve 1 eleman içeren 2 gruba ve ikinci terime göre de sırasıyla 2 ve 1 eleman içeren 2 gruba ayrıldığı görülmektedir. Yöntemin esası bu gözlem üzerine kuruludur, şöyle ki; öncelikle ilgili elemanın bulunduğu grubun toplam eleman sayısı kadar pozitif katkısı ve elemanın grup içinde yer aldığı pozisyon kadar negatif katkısı hesaplanmalıdır. Bu hesaplamalar *A* alt kümesinin tüm elemanları için yapıldığında ve kümülatif toplandığında istenen değer elde edilir.

Tablo 1: n=4 için alt kümeler

	r									
#	1	2		3			4			
1	1	1	2	1	2	3	1	2	3	4
2	2	1	3	1	2	4				
3	3	1	4	1	3	4				
4	4	2	3	2	3	4				
5		2	4							
6		3	4							

Örneğin, r=2 ve A={2,4} seçildiğinde, *A* alt kümesinin birinci elemanı olan 2'nin bulunduğu gruptaki eleman sayısı 6 olduğundan pozitif katkı olarak alınmıştır ve 2'nin grup içindeki başlangıç pozisyonuna ulaşmak için de 3 değeri çıkarılmıştır. 3 sayısı ise 2'den önce gelen eleman olan 1'in bulunduğu grupta kaç adet yer aldığı Pascal Üçgeni'nden, yani yine kombinasyon yardımıyla hesaplanmıştır. Benzer şekilde *A* alt kümesinin ikinci elemanı olan 4'ün bulunduğu gruptaki eleman sayısı 2 olduğundan pozitif katkı olarak alınmıştır ve 4'ün grup içindeki başlangıç pozisyonuna ulaşmak için de 1 değeri çıkarılmıştır. Sonuç olarak edilen sayılar toplandığında ve sonuca 1 sayısı ilave edildiğinde istenen 6-3+2-1+1=5 değeri bulunmuştur. *r=2* seçildiğinde tüm *A* alt kümelerinin alfabetik sıralamasının bulunmasında kullanılan sayısal değerler aşağıdaki Tablo 2'de görülmektedir.

Tablo 2: n=4 , r=2 için alt kümeler

			I			II					
I	II		+	-		+	-		+		#
1	2		6	6		3	3		1		1
1	3		6	6		3	2		1		2
1	4		6	6		3	1		1		3
2	3		6	3		2	2		1		4
2	4		6	3		2	1		1		5
3	4		6	1		1	1		1		6

Benzer şekilde *r=3* seçildiğinde tüm *A* alt kümelerinin alfabetik sıralamasının bulunmasında kullanılan sayısal değerler de Tablo 3'te görülmektedir.

Tablo 3: n=4 , r=3 için alt kümeler

				I			II			III					
I	II	III		+	-		+	-		+	-		+		#
1	2	3		4	4		3	3		2	2		1		1
1	2	4		4	4		3	3		2	1		1		2
1	3	4		4	4		3	1		1	1		1		3
2	3	4		4	1		1	1		1	1		1		4

Sonuç olarak tüm bu hesaplamalar matematiksel gösterimle ifade edilecek olursa aşağıdaki (3) denklemi elde edilir. Hesaplama sürecinde *A* kümesinin elemanlarına sıralı erişebilmek için *A* kümesi dizi olarak *f* fonksiyonuna gönderilmektedir ve A[0]=0 olarak kabul edilmiştir.

$$f(n,r,A[\,]) = \sum_{i=1}^{r}\left[\binom{n-A[i-1]}{r+1-i} - \binom{n+1-A[i]}{r+1-i}\right] + 1 \qquad (3)$$

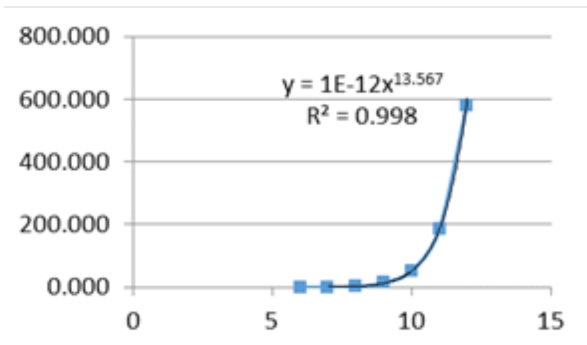
2. Hesaplama Denemeleri

Hem bu çalışmada önerilen özet fonksiyon önerisi, hem de özyinelemeli yaklaşım C programlama dilinde kodlanmış olup üzerinde 64 bit Windows 7 işletim sistemi bulunan i5 3.3 GHz işlemciye ve 8 GB belleğe sahip bir bilgisayar üzerinde hesaplama denemeleri yapılmıştır. Denemelerde kullanılan tüm kodlar http://kisi.deu.edu.tr/zeynep.berberler/ksibofo/ adresinde yer almaktadır.

İki yöntemin CPU zamanı kriteri yönünden karşılaştırmasını etkin bir şekilde yapabilmek için çalışma süreleri en çok 10 dk ile sınırlandırılmış ve örnekler buna göre seçilmiştir. *n*=49 alınarak *r*=6,7,8,9,10,11,12 için denemeler yapılmış ve sonuçlar aşağıdaki Tablo 4'te ve Şekil 1'de raporlanmıştır. En kötü durum analizini yapabilmek için seçilen *r* değerine ait tüm alt kümelerden alfabetik sıranın en sonundaki alt küme A olarak seçilmiştir. Örneğin *r*=6 için *A*={44,45,46,47,48,49}, *r*=7 için *A*={43,44,45,46,47,48,49} v.b. şekilde seçilmektedir.

Tablo 4: n=49 ve r=6,7,8,9,10,11,12 için süreler

						t (sn)
r		#A		f		özyineleme
6		13.983.816		0		0,055
7		85.900.584		0		0,371
8		450.978.066		0		1,907
9		2.054.455.634		0		15,283
10		8.217.822.536		0		52,733
11		29.135.916.264		0		184,071
12		92.263.734.836		0		579,740



Şekil 1. özyinelemeli algoritma için CPU süreleri

3. Sonuç ve Değerlendirme

Özet fonksiyon *f* içinde yer alan kombinasyon sayısını hesaplayan algoritma karmaşıklığı düşürecek şekilde tasarlandığından denemelerde ele alınan örnekler için hesaplama süreleri 0 sn görünmektedir, oysa özyinelemeli yaklaşım ile elde edilen sürelerin üstel olduğu gözlenmiştir. Sonuç olarak önerilen özet fonksiyon literatürde yazarların karşılaştığı diğer geleneksel yöntemler olan özyinelemeli sayımlama v.b. tekniklere göre çok daha etkindir.

REFERANSLAR

[1] Liu, C.L., An Introduction to Combinatorial Mathematics, McGraw-Hill, New York, 1968.

[2] Dewdeney, A. K., The Turing Omnibus, Computer Science Press, New York, 1993.

[3] Papadimitriou, C. H., Computational Complexity, Addison-Wesley, Reading, Mass, 1994.

[4] Dexter C Kozen, The Design and Analysis of Algorithms, Cornell University, Ithaca NY, 1990.

[5] Skiena, Steven S, The Algorithm Design Manual, Springer, New York, 2012.

[6] Cormen T. H., Leiserson C. E., Rivest R.L., Stein C., Introduction to Algorithms, 3rd Edition, MIT Press, 2009.

[7] Kleinberg J., Tardos É, Algorithm Design, Pearson/Addison-Wesley, 2006.