Permütasyonları Sıralamak!

Olcay Coşkun
Boğaziçi Üniversitesi
olcay.coskun@gmail.com

n nesnenin tüm permütasyonlarını tekrarsız olarak nasıl sıralarız? Rasgele bir permütasyon nasıl seçeriz? İlk problem zor görünmese de $n!$ hızında zorlaşmaktadır. Bir çoğumuzun hiç düşünmeden aklına gelen yöntem şöyledir: Diyelim ki $1, 2, \ldots, n$ kümesinin permütasyonlarını sıralamak istiyoruz. Biliyoruz ki ilk terim bu sayılardan herhangi biri olabilir. Dolayısıyla bir sayı seçtikten sonra geriye kalan terimler seçilen sayıdan sonra kümede kalan elemanların permütasyonları olacak. Örneğin $n = 3$ ise, ilk terimi 1 olan permütasyonlar (1 2 3) ve (1 3 2)'dir. Burada (2 3) ve (3 2) terimleri $\{2, 3\}$ kümesinin permütasyonlarıdır. Şimdi tümevarımla istediğimiz permütasyon listesine ulaşabiliriz.

Ama bu biraz acemice oldu! Her adımda eleman sayısı bir tane az olan kümelerin permütasyonlarını listelememiz gerekiyor. Ayrıca seçme özgürlüğümüz olduğu için her defasında farklı listeler elde etmek de olasılık dahilinde.

Tabii ki çok daha etkin algoritmalar kurmak mümkün. Bu yazımızda klasik sayılabilecek bir yöntemden söz edeceğiz ve aynı zamanda bu yöntemin ikinci sorumuza da cevap vereceğini göreceğiz. Bu yöntemin güzel bir özelliği permütasyonları sıralı olarak vermesidir. Ne demek istediğimiz biraz sonra netleşecek. Öncelikle bu tür algoritmalar için temel oluşturacak kavramları tanımlayalım.

Faktoriyel Taban

İlk okuldan itibaren sayıları onluk tabanda gösteririz: $321 = 1 \cdot 10^0 + 2 \cdot 10^1 + 3 \cdot 10^2$. İlerleyen yıllarda 10 tabanı yerine herhangi bir sayının da taban olarak kullanılabileceğini öğreniriz. Sümerlerin 60 tabanını kullandığını öğrenmek çoğumuz için sürpriz olmuştur. 0 ve 1 dizilerinin bilgisayarların ortaya çıkmasında oynadığı rol ise mutluluk verici bir uygulama örneğidir.

Permütasyonlarla ilgili işler içinse daha yaratıcı, ama içgüdülerimize biraz ters gelebilecek bir çözümleme var: Faktoriyel taban. Alışageldiğimizden biraz farklı olarak, faktoriyel tabanda verilen bir n sayısının k'inci faktoriyel rakamı 0 ile k arasındadır ve n'yi

$$n = f_1 \cdot 1! + f_2 \cdot 2! + \ldots$$

olarak yazarız (tabii ki sadece sonlu sayıda f_k sıfırdan farklı olacaktır). Bu durumda n'nin faktoriyel tabandaki gösterimi de aşağıdaki gibidir.

$$n = f_1 : f_2 : \ldots$$

İlk bakışta rakam sayısı belirsiz görünse de aslında $k! \leq n < (k+1)!$ eşitsizliklerini sağlayan k sayısından sonraki tüm rakamlar sıfır olmalıdır.

Örnek 1. $n = 10$ ise

$$10 = 0 \cdot 1! + 2 \cdot 2! + 1 \cdot 3!$$

olur. Bu durumda 10'un faktoriyel tabandaki gösterimini şöyle yazarız:

$$10 = 0 : 2 : 1$$

Örnek 2. $n = 81$ ise

$$81 = 1 \cdot 1! + 1 \cdot 2! + 1 \cdot 3! + 3 \cdot 4! = 1 : 1 : 1 : 3$$

elde ederiz.

Her n için böyle bir çözümlemenin var olacağını içgüdüsel olarak görebiliriz. Matematiksel bir kanıt için faktoriyel rakamları veren bir algoritma tanımlayacağız.

Öncelikle çözümlemedeki biricik tek sayı 1!'den gelebileceği için f_1 katsayısı n tekse 1, çiftse 0 olmalı. Bir diğer deyişle

$$f_1 \equiv n \pmod 2.$$

Şimdi

$$n = 2 \cdot n_1 + f_1$$

olarak yazabiliriz. Burada n_1, n'nin yarısıdır. Algoritmanın devamında tümevarım kullanacağız. Öncelikle $n_0 = n$ yazalım. Eğer $f_1, f_2, \ldots, f_{k-1}$'i bulmuşsak f_k'yi

$$n_{k-1} = (k+1) \cdot n_k + f_k$$

eşitliğiyle tanımlayalım. Yani f_k, $k-1$'inci adımda elde ettiğimiz bölümün k'ye oranıdır. Algoritma $n_l = 0$ olduğunda sonlanır.

Örnek 3. $n = 81$ örneğinde, sayı tek olduğu için $f_1 = 1$ olur. 2'ye bölüm 40 olduğundan $n_1 = 40$ alırız. Böylece

$$40 = 3 \cdot 13 + 1$$

eşitliğini elde ederiz. Bu eşitlik $f_2 = 1$'i verir. Dikkatli okuyucu,

$$f_k \equiv n_{k-1} \bmod (k+1)$$

olduğunu fark etmiş olmalı. Şimdi, $n_2 = 13$ ve dolayısıyla $f_3 = 1$, $n_3 = 3$, $f_4 = 3$ ve $n_4 = 0$ olacağı açıktır.

Faktoriyel tabanda aritmetik yapmanın inceliklerini keşfetmeyi okuyucuya bırakıyoruz. Örneğin $n, m \in \mathbb{N}$ verildiğinde n, m ve $n+m$'nin faktoriyel rakamları arasında nasıl bir ilişki vardır? Küçük örneklerle başlayıp bir kural bulabilir misiniz?

Ürettiğimiz bu algoritma, her doğal sayının faktoriyel tabanda çözümlenebileceğini kanıtlıyor. Peki, ama bu çözümleme biricik midir? Bir diğer deyişle, herhangi bir doğal sayı için iki farklı faktoriyel gösterim mümkün müdür?

Cevap tabii ki hayır olacak. Kanıt için önemli olacak bir sonuçla başlayalım. Faktoriyel tabandaki k basamaklı en büyük sayının $1:2:3:\ldots:k$ olduğu açıktır. İlk sonucumuz bu sayının onluk tabandaki karşılığını belirliyor.

Önsav 1. *Her $k \in \mathbb{N}$ için*

$$1 \cdot 1! + 2 \cdot 2! + \ldots + k \cdot k! = (k+1)! - 1$$

eşitliği sağlanır.

Kanıt. Bu iddiayı k üzerinden tümevarımla kanıtlayalım. $k = 1$ olduğunda kanıtlayacak bir şey yok. O yüzden $k > 1$ olsun ve iddianın k'den küçük tüm doğal sayılar için doğru olduğunu varsayalım. O zaman

$$1 \cdot 1! + 2 \cdot 2! + \ldots + (k-1) \cdot (k-1)! + k \cdot k! = k! - 1 + k \cdot k!$$

olur ki bu eşitlik kanıtı bitirir. □

Şimdi faktoriyel gösterimin biricik olduğunu kanıtlayabiliriz.

$$m = f_1 : f_2 : \ldots : f_k \quad \text{ve} \quad n = g_1 : g_2 : \ldots : g_l$$

verilsin. Bu gösterimde en sağdaki f_k ve g_l rakamlarımızın 0 olmadığını kabul ediyoruz, yani $f_k \neq 0, g_l \neq 0$. Amacımız $m = n$ olduğunda $k = l$ ve her i için $f_i = g_i$ olduğunu göstermek. O zaman $m = n$ olduğunu varsayalım. Öncelikle eğer $k \neq l$ ise, diyelim ki $k < l$ olsun. Bu durumda

$$m < (k+1)! \leq l! \leq n$$

çelişkisini elde ederiz. İlk eşitsizlik için bir önceki önsavı, son eşitsizlik içinse g_l'nin sıfır olmadığı varsayımını kullandık. Dolayısıyla $k = l$ olmalı.

Şimdi eğer gösterimler aynı değilse $f_i \neq g_i$ eşitsizliğinin sağlandığı bir i göstergesi vardır. Bu göstergelerden en küçüğüne r diyelim. (Yani, $i < r$ ise $f_i = g_i$ olacak.) r'nin k'den küçük olması gerektiği açık. Bu durumda

$$\begin{aligned} 0 &= m - n \\ &= \sum_{i=r}^{k} (f_i - g_i) \cdot i! \\ &= (f_r - g_r) \cdot r! + \sum_{i=r+1}^{k} (f_i - g_i) \cdot i! \\ &= (f_r - g_r) \cdot r! + (r+1)! \cdot K \end{aligned}$$

elde ederiz. Üçüncü eşitliği yazarken $i > k$ olduğunda $i!$'in $(k+1)!$'e bölünebilmesini kullandık ve $K \in \mathbb{N}$ sayısını

$$K = \frac{\sum_{i=r+1}^{k} (f_i - g_i) \cdot i!}{(k+1)!}$$

olarak tanımladık. Hatırlarsanız r'inci faktoriyel rakam r'den küçük olmalıydı. O zaman $f_r - g_r$ farkı r'den küçük, $(f_r - g_r) \cdot r!$ ise $(r+1)!$'den küçük olur. Yani yukarıdaki toplamın 0'a eşit olabilmesi için tek yol, $K = 0$ ve $f_r = g_r$ olmasıdır. Ama f_r'nin g_r'den farklı olduğunu varsaymıştık. Bulduğumuz çelişki bize $f_i = g_i$'nin tüm i göstergeleri için sağlandığını gösterir. Elde ettiğimiz sonucu bir teorem olarak not edelim.

Teorem 1. *Her n doğal sayısı için*

$$n = f_1 \cdot 1! + f_2 \cdot 2! + \ldots, \quad 0 \leq f_i \leq i$$

eşitliğini sağlayan biricik $f_1, f_2, \ldots$ doğal sayılar listesi vardır.

Yukarıda bahsettiğimiz gibi f_1, n'nin tek-çift olmasıyla belirlenir. n'nin sıfır olmayan en sağ rakamının göstergesi ise $k! \leq n < (k+1)!$ eşitsizliğini sağlayan k değeridir.

Peki faktoriyel tabanı permütasyonlarla nasıl ilişkilendireceğiz? Öncelikle Önsav 1'den, 0 ile birlikte düşünüldüğünde, r basamaklı $r!$ tane sayı yazılabileceği açıktır. Burada 0'ı her sayıya eklemlemiş gibi düşünüyoruz. Örneğin,

$$10 = 0 \cdot 0! + 0 \cdot 1! + 2 \cdot 2! + 1 \cdot 3!$$

olarak yazacağız. Algoritmamız da 0!'in katsayısını her zaman sıfır verecektir.

Permütasyonlardan Faktoriyel Tabana

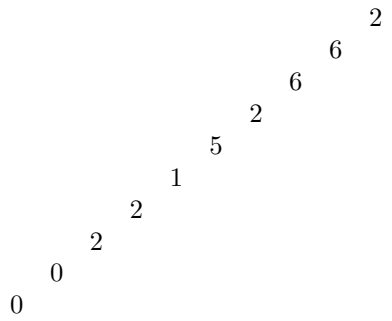
Bu gözlem n'li permütasyonlar kümesiyle faktoriyel tabanda n rakamlı olan sayılar arasında eşlemeler olduğunu verir. Bu tür bir eşleme bulmanın çok sayıda yolu biliniyor. Aşağıda Lehmer'in iyi bilinen eşlemesini tanıtacağız. Bu eşleme faktoriyel gösterimden başlayıp bir permütasyon kurduğu

için aynı zamanda permütasyonları sıralamış oluyor. Böylece, örneğin "10'lu permütasyonların bir milyonuncusu" anlamlı bir söz haline geliyor. Eşlemenin kurulumunu bu örnek üzerinde yapalım.

Öncelikle 1 milyonun faktoriyel tabandaki çözümlemesini yapmalıyız. Yukarıdaki algoritmayı uyguladığımızda

$$10^6 = 0:0:2:2:1:5:2:6:6:2$$

buluruz. İşlemin sağlamasını okuyucuya bırakıyoruz. Bu eşleme için en sola fazladan bir 0 eklediğimizi hatırlatalım. Şimdi bu rakamları bir ters köşegen oluşturacak şekilkde, soladan başlayıp yukarıya giderek dizelim:



Amacımız bu köşegeni aşağı yönde bir üçgene tamamlamak. Bunun için soldan ikinci sütundan başlıyoruz ve sütundaki eksik sayıları şu kurala göre belirliyoruz:

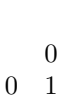
k'inci sütundaki r'inci girişe $f_{k,r}$ diyelim. O zaman, eğer k'inci sütuna kadar r'inci satırdaki tüm girişler bulunmuşsa, $f_{k,r}$'yi aşağıdaki fonksiyon ile belirleriz:

$$f_{k,r} = \begin{cases} f_{k-1,r} & \text{eğer} \quad f_{k-1,r} < f_k \quad \text{ise} \\ f_{k-1,r}+1 & \text{eğer} \quad f_{k-1,r} \geq f_k \quad \text{ise} \end{cases}$$

Bir diğer deyişle, girdi yapacağımız noktanın solundaki sayı bulunduğumuz sütunun en tepe girdisinden küçükse o sayıyı, büyükse ya da eşitse bir fazlasını yazıyoruz.

Örnek 4. Yukarıda verilen örnekte ikinci sütunu bulmak için

ile başlarsak $f_{2,2} = 1$ olur. Benzer şekilde



bulunduğunda $f_{3,2} = 0, f_{3,3} = 1$ buluruz. Dikkat edilirse ikinci sütundaki sayılar üçüncü sütunun tepe girdisi olan 2'den küçük olduklarından aynı sütünu yazdık. Şimdi bu kurala göre üçgeni tamamlayalım. Diğer adımları okuyucuya bırakıyoruz.

									2
							6	7	
						6	7	8	
					2	2	2	3	
				5	6	7	8	9	
			1	1	1	1	1	1	
		2	3	3	4	4	4	5	
	2	3	4	4	5	5	5	6	
0	0	0	0	0	0	0	0	0	
0	1	1	1	2	2	3	3	3	4

Algoritmamızın iddiasına göre son sütun bize $0, 1, \ldots, 9$ kümesinin bir permütasyonunu verecektir. Standart gösterime ulaşmak için, her sayıya bir eklersek $1, 2, \ldots, 10$ kümesinin bir permütasyonunu buluruz. Bir milyonla başladığımızdan, bu permütasyon bir milyonuncu olacak:

$$\sigma_{10^6} = (3\ 8\ 9\ 4\ 10\ 2\ 6\ 7\ 1\ 5)$$

Bu yöntemle elde ettiğimiz dizinin bir permütasyon olduğunu ve her permütasyonu bu yolla bir ve yalnızca bir şekilde elde edebileceğimizi göstermemiz gerekir.

Öncelikle, seçimimizden dolayı, k'inci sütunun en tepesinde n'nin k'inci faktoriyel rakamı bulunuyor, yani en çok $k-1$ olabilir (0 eklediğimizi tekrar hatırlayın!). Algoritmamız k'inci satırda sağa doğru ilerlerken bu rakama ya bir ekliyor ya da aynı bırakıyor. Dolayısıyla sonuncu sütunda k'inci satır en çok $k-1+(r-k) = r-1$ olabilir. Bir diğer deyişle r'inci sütundaki sayılar 0 ile $r-1$ arasındadır. Tabii bu sonuç permütasyon elde etmek için yeterli değil. Ayrıca sayıların tekrar etmediğini de göstermeliyiz.

Aslında elde ettiğimiz her sütun bir permütasyon olmalı. Gerçekten de $0 : f_1 : f_2 : \ldots : f_r$ dizisini soldan bir rakamdan kesersek, örneğin, $s < r$ olmak üzere $0 : f_1 : f_2 : \ldots : f_s$ dizisini alırsak, bu dize de bir sayının faktoriyel gösterimi olacağından, s'inci sütunun da bir permütasyon olmasını bekliyoruz.

O zaman ilk tekrarın son sütunda, yani r'inci sütunda olduğunu varsayabiliriz. Öyleyse

$$f_{r,s} = f_{r,t} := a$$

olsun. İlk tekrar bu sütunda olduğundan $f_{r-1,s} \neq f_{r-1,t}$ de doğrudur. Ama her yeni sütunda satırlar ya bir artıyordu ya da aynı kalıyordu. Dolayısıyla

$$f_{r-1,s} = a, \quad f_{r-1,t} = a-1$$

ya da

$$f_{r-1,s} = a-1, \quad f_{r-1,t} = a$$

olmalıdır. Şimdi algoritmayı uygularsak, ilk durumda, s'inci satırdaki sayı değişmediğinden,

$$a = f_{r-1,s} < f_r, \quad a - 1 = f_{r-1,t} \geq f_r$$

olur ki bu mümkün değildir. İkinci durumda da benzer bir çelişki ortaya çıkar. Bu çelişkili durum r'inci sütunda tekrar olmadığını kanıtlar.

Faktoriyel Tabandan Permütasyonlara

Yukarıdaki kanıt, faktoriyel tabanda r rakamlı sayılarla r'li permütasyonlar arasında iyi tanımlı bir fonksiyon olduğunu gösterdi. Fonksiyonun bir eşleme olduğunu görmek için algoritmanın tersinin olduğunu fark etmek yeterli.

Şöyle ki $\{1, 2, \ldots, r\}$'nin verilen bir permütasyonunu, önce her sayıdan 1 çıkarıp $a_1, a_2, \ldots, a_r$ dizisini elde edelim. Bu sayıları bir sütun olarak yazalım, dolayısıyla $f_r = a_1$ olacak. Eğer $a_1 < a_2$ ise $f_{r-1} = a_2 - 1$, değilse $f_{r-1} = a_2$ olmalı. Şimdi bu sütundaki diğer girdileri de benzer yöntemle bulabiliriz: k'inci satırda, eğer $a_1 < a_k$ ise $f_{r-1,k} = a_k - 1$, değilse $f_{r-1,k} = a_k$ olmalı. $r-1$'inci sütun tamamlandığında aynı yöntemin tekrar uygulamaları ile üçgeni tamamlayabiliriz.

Ortaya çıkan üçgenin tepe girdileri bize bir sayının faktoriyel tabandaki rakamlarını verecektir. Elde ettiğimiz bu sayıya bir önceki algoritmayı uyguladığımızda başladığımız permütasyonu üreteceği de açıktır. Detaylı sağlamayı okuyucuya bırakıyoruz.

Şimdi ikinci sorumuzun yanıtı da ortaya çıktı. Rasgele bir permütasyon için önce rasgele bir sayı üretiriz (bu işlem oldukça standarttır). Sonrasında yukarıdaki algoritmayı uygulayarak bu sayıya karşılık gelen permütasyonu buluruz. İşte size rasgele bir permütasyon!

Lehmer Kodu

Her ne kadar uygulaması zor olmasa da yukarıdaki algoritmayı uygulamak zaman alır. Aslında ortaya çıkan fonksiyonu tarif etmenin çok daha kısa yolları da vardır. Burada yine Lehmer'in bir sonucunu verelim.

Tanım 1. $\sigma = (a_1\ a_2\ \ldots a_n)$ permütasyonunun *Lehmer kodu* $l_\sigma = (l_1,\ l_2,\ \ldots, l_n)$ ile gösterilir. Burada l_k, $(a_1\ a_2\ \ldots a_n)$ gösteriminde k'inci koordinatın sağında olan ama a_k'dan küçük olan terimlerin sayısıdır.

Örnek 5. $\sigma = (6\ 3\ 1\ 4\ 2\ 5)$ ise σ'nın Lehmer kodu $l_\sigma = (5, 2, 0, 1, 0, 0)$ olur.

Ters yönde, $l = (l_1,\ l_2,\ \ldots, l_n)$ Lehmer kodu verildiğinde karşılık gelen permütasyon şöyle bulunur:

- a_1 küçükten büyüğe dizilen

  $$[n] = \{1, 2, \ldots, n\}$$

  kümesindeki $l_1 + 1$'inci elemandır.

- $a_1, a_2, \ldots, a_{k-1}$ belirlendikten sonra a_k, küçükten büyüğe sıralanmış olan

  $$[n] \backslash \{a_1, a_2, \ldots, a_{k-1}\}$$

  kümesinin $l_k + 1$'inci elemanıdır.

Örnek 6. Yukarıda bulduğumuz $l = \{5, 2, 0, 1, 0, 0\}$ kodunu alalım. Yukarıdaki sürece göre

- $[6]$ kümesindeki 6'ncı eleman, yani 6, a_1 olur.
- $\{1, 2, 3, 4, 5\}$ kümesindeki 3'üncü eleman, yani 3, a_2 olur.
- $\{1, 2, 4, 5\}$ kümesindeki 1'inci eleman, yani 1, a_3 olur.

Diğer terimleri $a_4 = 4, a_5 = 2, a_6 = 5$ olarak belirlenir. Okuyucunun bu adımları yapmasını öneririz.

Bu iki algoritma permütasyonlarla Lehmer kodları arasında bir eşleme kurdu. Şimdi can alıcı nokta şu: Eğer σ'nın Lehmer kodunu tersten okursak σ'ya karşılık gelen faktoriyel tabandaki sayıyı buluruz. Yukarıdaki örnekte bu sayı

$$0 : 0 : 1 : 0 : 2 : 5 = 2 + 2 \cdot 4! + 5 \cdot 5! = 650$$

olur. Bir önceki algoritmanın uygulamasını okuyucuya bırakıyoruz.

Lehmer koduyla yapılabilecek daha çok şey var! Yukarıdaki iddianın kanıtını ve daha fazlasını bir sonraki yazımıza bırakıyoruz.

Not 1. Rasyonel sayıları da faktoriyel tabanda göstermek mümkündür. İlgili okuyucuya 1995-4 sayımızdan Oya Tabağ'ın "Faktöriyel Taban ile Hatasız İşlemler" yazısını öneririz.