Kıyametin Kopacağı Gün
(Hanoi Bilmecesi)

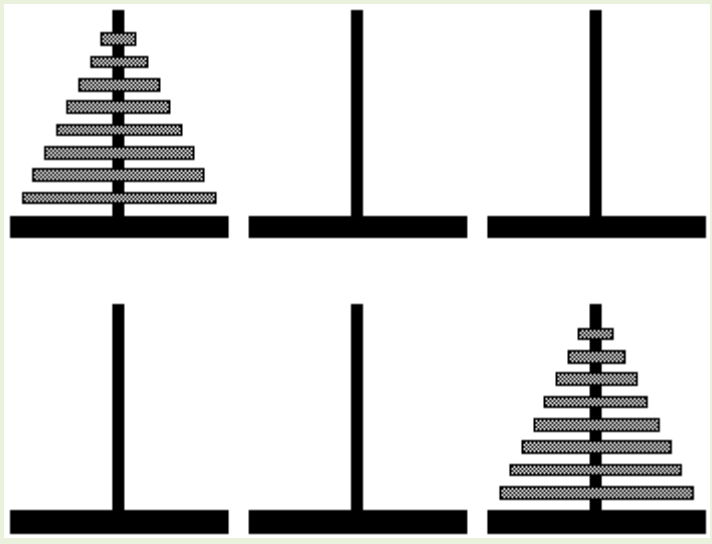
Timur Karaçay
tkaracay@baskent.edu.tr

Çok eskiden Hanoi'deki bir tapınakta başrahip tapınağın bahçesine üç sütun diktirmiş. Yanyana duran sütünlardan soldakine, ortası delik 64 disk (halka) geçirmiş. Yarıçapları birbirlerinden farklı olan disklerin en büyüğünü en alta, en küçüğünü en üste koymuş.

Sonra, başrahip, genç rahipleri bahçede toplayıp onlara demiş ki;

> " -Şu sütünlara ve disklere bakın. Sol sütundaki diskleri, size söyleyeceğim kurallara uyarak sağdaki sütuna geçireceksiniz. Bunu ilk başaran kişi doğrudan cennete gidecektir. Ama aynı anda kıyamet kopacak, dünyanın sonu gelecektir!...
>
> Kurallar şunlardır: "

1. Her hamle bir sütunda en üstteki diski alıp başka bir sütuna geçirme işlemidir.
2. Her hamlede yalnızca bir disk taşınabilir.
3. Küçük bir disk asla büyük bir diskin altına konamaz.
4. Üç sütunu dilediğiniz gibi kullanabilirsiniz.



Rivayet ederler ki, binlerce yıldır tapınaktaki genç rahipler cennete gitmek için gece gündüz demeden başrahibin verdiği işi bitirmeye çabalıyorlar. Yaşlanıp ölenlerin yerine yeni gençler geliyor ve çalışma aksamadan yürüyor. Tapınağın bahçesine gidip canla başla verilen uğraşı görenler diyor ki;

"-Bu işin bitmesi daha milyarlarca yıl alır!..."

Biz başrahibin dediği gibi, diskler taşınınca kıyametin kopacağına inanmıyoruz ama eğlence olsun diye denemeye kalkıştık. Çok uğraştık, ama işi bitiremedik. Sonunda vazgeçip, bu işin ne kadar zamanda bitebileceğini merak etmeye başladık.

Rahiplerin hamleleri çok hızlı yaptığını varsayalım. Bir hamleyi 1 saniyede bitirseler, 64 diski ne kadar zamanda soldaki sütundan sağdaki sütuna taşıyabilirler?

İçinizde bu merakımızı kim giderebilir?

Yol gösterme:

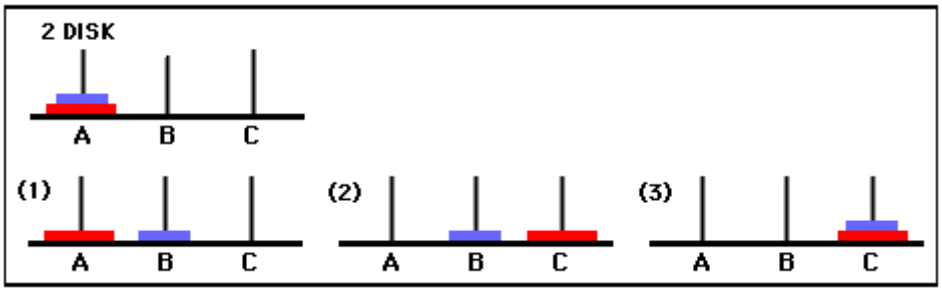
n = 1 durumu:

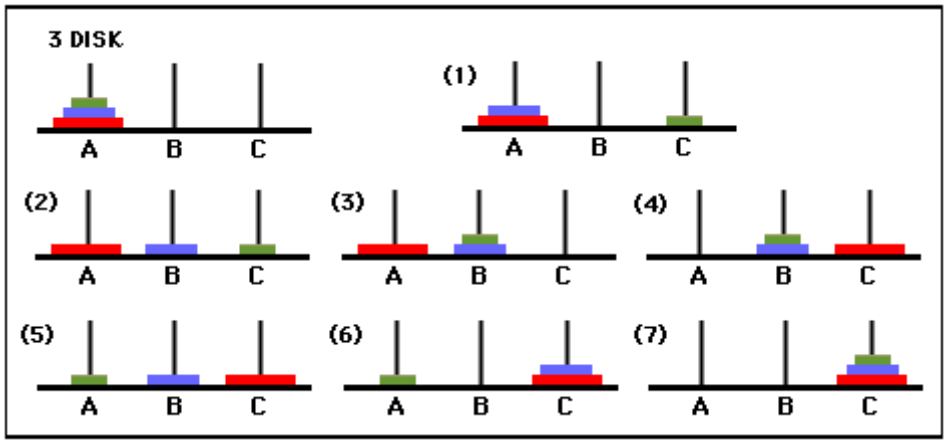
Kaynak sütunda 1 disk vardır. Bir hamlede kaynaktan al, doğrudan hedef sütuna geçir. Bu hamlede yedek sütunu kullanmaya gerek yoktur.

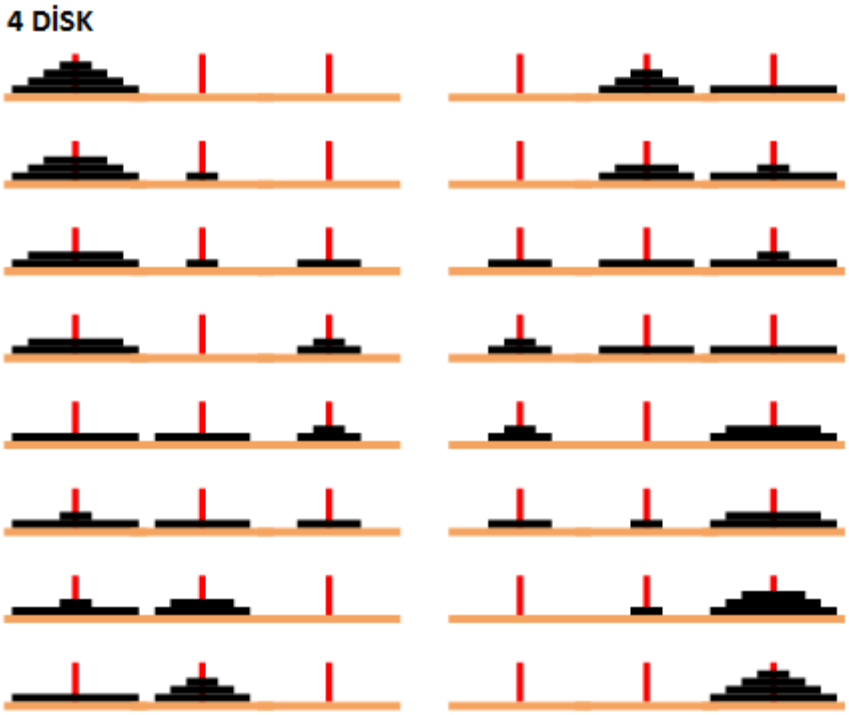
n = 2 durumu:

İlk hamlede, kaynak sütunda en üstte duran küçük diski yedek sütuna tak. İkinci hamlede kaynak sütundaki büyük diski hedef sütuna tak. Üçüncü hamlede yedek sütundaki diski hedef sütuna tak.

2,3 ve 4 disk için yaptığımız hamleleri aşağıdaki şekillerden görebilirsiniz.







8 diske kadar gerekli hamle sayısı aşağıdaki tabloda görülmektedir. Disk sayısını artırarak 64 diske çıkınız ve işin ne kadar zamanda biteceğini hesaplayınız.

Disk Sayısı	Hamle sayısı
1	1
2	3
3	7
4	15
5	31
6	63
7	127
8	255

Hanoi Kuleleri Bilmecesinin Genel Çözümü

Bu bilmece 1883 yılında Fransız matematikçi *E.Lucas* tarafından ortaya atılmıştır. Sonraları matematiğin çetin problemlerinden birisi haline geldi. *1957* yılında *Gardner*, problemin *n-hiperküp* üzerinde bir *Hamilton yolu* bulma problemine eşdeğer (isomorphic) olduğunu gösterdi. Matematikte *en kısa yol* problemi diye adlandırılan zor bir probleme örnek oluşturdu. Problemin farklı çözümleri için Hanoimania! web adresine bakılabilir:

http://www.kernelthread.com/projects/hanoi//

Burada, özyineli bir algoritma ile bilmeceyi çözen bir bilgisayar programı yazacağız.

Her hangi bir hamlede üstünden diskin alınacağı sütuna *kaynak* sütun, diskin götürüleceği sütuna *hedef* sütun, geri kalan sütuna da *yedek* sütun diyelim. Bir hamlede *yedek* sütun ancak gerekirse yardımcı olarak kullanılacak, gerekmezse kullanılmayacaktır. Öyleyse, her hangi bir hamlede üç

sütunumuz vardır: *kaynak, hedef* ve *yedek.* Farklı hamlelerde sütunların rolleri değişebilir. Disk sayısını *n* ile gösterelim.

Yukarıdaki şekillerde gösterilen hamleler, problemin genel çözümü için bir ipucu veriyor. Disk sayısının *1,2,…,n* olması durumlarını *1* den *n* 'ye doğru artırarak düşünmek yerine *n,(n-1), (n-2), …,3,2,1* gibi *n* 'den *1* 'e doğru azaltarak çözmeyi düşünelim. Bu düşünce bizi şuna götürür:

> *n* disk olması durumunda bulacağımız algoritma bir sütundaki *n* diski başka bir sütuna (kurallar uyarınca) taşıyacaktır.
>
> *n-1* disk olması durumunda bulacağımız algoritma bir sütundaki *(n-1)* diski başka bir sütuna (kurallar uyarınca) taşıyacaktır.
>
> *n-2* disk olması durumunda bulacağımız algoritma bir sütundaki *(n-2)* diski başka bir sütuna (kurallar uyarınca) taşıyacaktır.
>
> …
>
> *2* disk olması durumunda bulacağımız algoritma bir sütundaki *2* diski başka bir sütuna (kurallar uyarınca) taşıyacaktır.
>
> *1* disk olması durumunda bulacağımız algoritma bir sütundaki *1* diski başka bir sütuna (kurallar uyarınca) taşıyacaktır.

Bu algoritmaların hepsi birbirine benziyor. Öyleyse bu adımların hepsini içeren tek bir algoritma ile problemi çözebiliriz. Bu düşünce, bizi *özyineli (recursive)* bir algoritma yaratmaya götürür.

Taşıma işini yapacak özyineli bu algoritma için *yalancı kodları (pseudo codes)* kolayca yazabiliriz.

1. n = 1 ise
   - 1.1. Kaynaktaki tek diski doğrudan hedefe taşı
2. n > 1 ise
   - 2.1. Kaynaktan (n-1) diski yedek sütuna taşı
   - 2.2. Kaynakta geri kalan tek diski hedefe taşı
   - 2.3. Yedekteki (n-1) diski hedefe taşı
3. Dur (iş bitti)

Dikkat ederseniz 2.1-inci adımı ile 2.3-üncü adımının işlevleri aynıdır; yalnızca sütunların rolleri değişmiştir. 2.3-üncü adım, algoritmanın kendi kendisini iş bitene kadar çağırması demektir. Algoritmanın özyineli (recursive) olmasının nedeni budur.

Şimdi yukarıdaki özyineli algoritmayı biraz daha açık yazalım:

```
hanoi(n, kaynak, yedek, hedef)
    if n is 0
      exit
    else
      hanoi (n - 1, kaynak, hedef, yedek)
      taşı(kaynak, hedef); //Hamlenin nereden nereye olduğunu yazar
      hanoi (n - 1, yedek, kaynak, hedef)
```

```
	// Sütunlar : a kaynak, b yedek, c hedef
	static void hanoi(int diskSayısı, int a, int b, int c) {
		if (diskSayısı > 0) {
			hanoi(diskSayısı-1,  a, c, b);
			taşı(a, c);
			hanoi(diskSayısı-1,  b, a, c);
		}
	  }
```

Hanoi() metodu bilmeceyi çözer. Bilmecenin nasıl çözüldüğünü görmek istersek, her hamlede diskin nereden alınıp nereye taşındığını yazan bir metot ekleyebiliriz. Bu metoda *taşın()* diyelim:

```
	static void taşın(int nerden, int nereye) {
		System.out.print("Nereden nereye : ");
		System.out.print(nerden);
		System.out.print(" --> ");
		System.out.print(nereye);
		System.out.println("   Hamle : " + ++sayaç);
	}
```

Gerekli metotlarımızı hazırladığımıza göre, artık onları bir java uygulama programı haline getirebiliriz:

```
class HanoiKuleleri {
	static int diskSayısı = 4;
	static int sayaç;

	public static void main(String args[]) {
		hanoi(diskSayısı, 1, 2, 3);
		System.exit(0);
	}

	// Sütunlar : a kaynak, b yedek, c hedef
	static void hanoi(int diskSayısı, int a, int b, int c) {
		if (diskSayısı > 0) {
			hanoi(diskSayısı - 1,a, c, b);
			taşın(a, c);
			hanoi(diskSayısı - 1, b, a, c);
		}
	}
```

```
    static void taşın(int nerden, int nereye) {
        System.out.print("Nerden nereye : ");
        System.out.print(nerden);
        System.out.print(" --> ");
        System.out.print(nereye);
        System.out.println("   Hamle : " + ++sayaç);
    }
}
```

```
/*
Çıktı:
Nereden nereye : 1 --> 2   Hamle : 1
Nereden nereye : 1 --> 3   Hamle : 2
Nereden nereye : 2 --> 3   Hamle : 3
Nereden nereye : 1 --> 2   Hamle : 4
Nereden nereye : 3 --> 1   Hamle : 5
Nereden nereye : 3 --> 2   Hamle : 6
Nereden nereye : 1 --> 2   Hamle : 7
Nereden nereye : 1 --> 3   Hamle : 8
Nereden nereye : 2 --> 3   Hamle : 9
Nereden nereye : 2 --> 1   Hamle : 10
Nereden nereye : 3 --> 1   Hamle : 11
Nereden nereye : 2 --> 3   Hamle : 12
Nereden nereye : 1 --> 2   Hamle : 13
Nereden nereye : 1 --> 3   Hamle : 14
Nereden nereye : 2 --> 3   Hamle : 15
*/
```

Algoritmanın etkinliği

Bilmecenin ne kadar zamanda çözülebileceğini hesaplamak için, yukarıda yazdığımız algoritmanın etkinliğini bulmalıyız. Bunun için, algoritmadaki hamle (bir diski bir sütundan başkasına taşıma işlemi) sayısını hesaplamalıyız.

Kaç hamle yapıldığını hesaplayalım. *hanoi()* metodu iki kez kendisini çağırmaktadır. İki çağrı arasındada bir sabit birim zaman geçtiğini varsayalım. Buna göre, n tane diskin taşınması için geçen zaman T(n) ise

$T(0) = 0$

$T(n) = 2*T(n-1) + 1 , \; n>0$

olur. Buradan recursive olarak

$T(n) = 2*T(n-1) + 1$

$= 2 * [2 * T(n-2) + 1] + 1 = 2^2 * T(n-2) + 3 = 2^2 * T(n-2) + (2^2 - 1)$

$= 2^2 * [2 * T(n-3) + 1] + (2^2 - 1) = 2^3 * T(n-3) + 7 = 2^3 * T(n-3) + (2^3 - 1)$

$= 2^3 * [2 * T(n-4) + 1] + (2^3 - 1) = 2^4 * T(n-4) + 15 = 2^4 * T(n-4) + (2^4 - 1)$

$= \ldots$

$= 2^k * T(n-k) + (2^k - 1)$

$= \ldots$

$= 2^n * T(0) + (2^n - 1)$
$= 2^n - 1$

çıkar[1]. Ohalde, bu algoritma $O(2^n)$ grubuna aittir. Bu sonuç, algoritmanın zaman karmaşasının (time complexity) çok yüksek olduğunu gösterir.

Şimdi, en başta sorduğumuz soruya yanıt verebiliriz. Rahipler başladıkları işi ne zaman bitirecekler?

Rahipler olağanüstü bir hızla çalışıyor olsunlar. Bir hamleyi (bir diski bir sütundan başka bir sütuna taşıma işlemini) 1 saniyede yaptıklarını varsayalım. Bu durumda 64 diskin yerine ulaşması için gerekli hamle sayısı T8n) = 2^{64} -1 dir. Her hamle 1 saniyede oluyorsa, işin bitmesi için 2^{64} -1 saniye gerekir. Bunu 60*6**24*365 sayısına bölersek, işin kaç yılda biteceğini buluruz. Bu sayı yaklaşık olarak 585.000.000.000 yıl; yani 585.000.000 milenyum (bin yıl) olur.

[1] T(n) = 2^n -1 eşitliği tümevarım yöntemiyle de ispatlanabilir.