CHAPTER

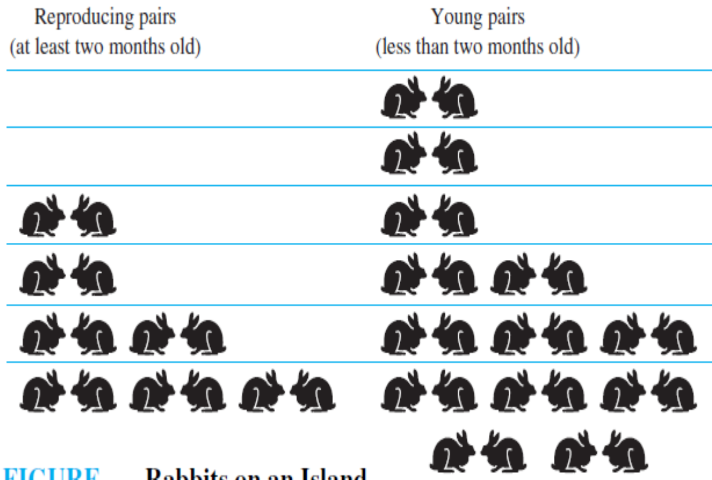
8 Advanced Counting Techniques

8.1 Applications of Recurrence Relations

8.3 Divide-and-Conquer Algorithms and Recurrence Relations

Applications of Recurrence Relations

Rabbits and the Fibonacci Numbers A young pair of rabbits (one of each sex) is placed on an island. A pair of rabbits does not breed until they are 2 months old. After they are 2 months old, each pair of rabbits produces another pair each month. Find a recurrence relation for the number of pairs of rabbits on the island after n months, assuming that no rabbits ever die.



Month	Reproducing pairs	Young pairs	Total pairs
1	0	1	1
2	0	1	1
3	1	1	2
4	1	2	3
5	2	3	5
6	3	5	8

FIGURE **Rabbits on an Island.**

$$f_n = f_{n-1} + f_{n-2}$$

$$f_0 = 0 \text{ and } f_1 = 1.$$

Let c_1 and c_2 be real numbers. Suppose that $r^2 - c_1 r - c_2 = 0$ has two distinct roots r_1 and r_2. Then the sequence $\{a_n\}$ is a solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ if and only if $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$ for $n = 0, 1, 2, \ldots$, where α_1 and α_2 are constants.

EXAMPLE 14

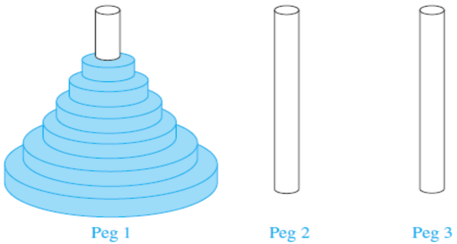
Find an explicit formula for the Fibonacci numbers.

Solution:

$$f_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n.$$

The Tower of Hanoi

Loop Technique:



The Initial Position in the Tower of Hanoi.

đĩa nhỏ luôn ở trên
mỗi lần chuyển 1 đĩa

BT chuyển n đĩa (1) → (3)
① chuyển n-1 đĩa (1) → (2)
số cặp ② chuyển đĩa n từ (1) → (3)
③ chuyển n-1 đĩa từ (2) → (3)

Solution:

Let H_n denote the number of moves needed to solve the Tower of Hanoi problem with n disks. Set up a recurrence relation for the sequence $\{H_n\}$.

$$\begin{aligned}
H_n &= 2H_{n-1} + 1 \\
&= 2(2H_{n-2} + 1) + 1 = 2^2 H_{n-2} + 2 + 1 \\
&= 2^2(2H_{n-3} + 1) + 2 + 1 = 2^3 H_{n-3} + 2^2 + 2 + 1 \\
&\ \vdots \\
&= 2^{n-1} H_1 + 2^{n-2} + 2^{n-3} + \cdots + 2 + 1 \\
&= 2^{n-1} + 2^{n-2} + \cdots + 2 + 1 \\
&= 2^n - 1.
\end{aligned}$$

EXAMPLE 16

Counting Subsets of a Finite Set Set up a recurrence relation

Solution:

gọi P_n là số tập con của tập n phần tử

thì $P_n = 2.P_{n-1}$

$= 2(2P_{n-2}) = 2^2 P_{n-2}$

$= 2^3 P_{n-3} = \ldots\ldots = 2^n P_{n-n} = 2^n.1$

Loop Technique:

$S = \{a_1, a_2, a_3\}$

$S = \{a, b, c\}$ $S' = \{a, b\}$

PowerSet(S) =

$\{\ \phi,\ \{a\},\ \{b\},\ \{c\},\ \{a, b\},\ \{a, c\},\ \{b, c\},\ \{a, b, c\}\ \}$

ϕ, $\{a\}$, $\{b\}$, $\{a, b\}$ (2) (2) (3) (3)

EXAMPLE 17

Loop Technique:

Compound Interest Suppose that a person deposits \$10,000 in a savings account at a bank yielding 11% per year with interest compounded annually. How much will be in the account after 30 years?

Solution: Có P_0 USD gửi ngân hàng

Mỗi năm có lãi suất r, $0 < r \le 1$ $0\% < \ \le 100\%$

Sau năm 1: gốc P_0 Lãi $P_0 \times r$ Tổng $P_0 + P_0 \times r = P_1$

2: P_1 $P_1 \times r$ $P_1 + P_1 \times r = P_2$

⋮

Hết năm n: P_{n-1} $P_{n-1} \times r$ $P_{n-1} + P_{n-1} \times r = \boxed{P_n}$

$P_n = (1+r)^{1} P_{n-1}$

$= (1+r)\left[(1+r) P_{n-2}\right]$ gọn $= (1+r)^{2} P_{n-2}$

$= (1+r)^2\left[(1+r) P_{n-3}\right] = (1+r)^{3} P_{n-3}$

$= \cdots\cdots = (1+r)^{n} P_{n-n}$

Nếu $n = 0$, $P_0 = 1000$ P_0

$\Rightarrow \boxed{P_n = (1+r)^n . P_0}$

Applications of Recurrence Relations

Find a recurrence relation and give initial conditions for the number of bit strings of length n that do not have two consecutive 0s. How many such bit strings are there of length five?

Tree Diagrams

Solution:

			Number of bit strings of length n with no two consecutive 0s:
End with a 1:	Any bit string of length $n-1$ with no two consecutive 0s	1	a_{n-1}
End with a 0:	Any bit string of length $n-2$ with no two consecutive 0s	1 0	a_{n-2}
		Total:	$a_n = a_{n-1} + a_{n-2}$

8.3 Divide-and-Conquer Algorithms and Recurrence Relations

Many recursive algorithms take a problem with a given input and divide it into one or more smaller problems. This reduction is successively applied until the solutions of the smaller problems can be found quickly.

These procedures follow an important algorithmic paradigm known as **divide-and-conquer**, and are called **divide-and-conquer algorithms**, because they *divide* a problem into one or more instances of the same problem of smaller size and they *conquer* the problem by using the solutions of the smaller problems to find a solution of the original problem, perhaps with some additional work.

MASTER THEOREM

Let f be an increasing function that satisfies the recurrence relation

$$f(n) = af(n/b) + c$$

whenever n is divisible by b, where $a \geq 1$, b is an integer greater than 1, and c is a positive real number. Then

$$f(n) \text{ is } \begin{cases} O(n^{\log_b a}) & \text{if } a > 1, \\ O(\log n) & \text{if } a = 1. \end{cases}$$

Furthermore, when $n = b^k$ and $a \neq 1$, where k is a positive integer,

$$f(n) = C_1 n^{\log_b a} + C_2,$$

where $C_1 = f(1) + c/(a-1)$ and $C_2 = -c/(a-1)$.

Binary Search

Tìm kiếm nhị phân

vị trí of x ≥ 1

ko có x → 0

$x \in \{a_1, a_2, \dots, a_n\}$

1 đầu — trái — x — a — phải — n cuối

giữa = (đầu + cuối)/2

complexity BS

- Tồi nhất → số phép toán so sánh tối đa (khi ko tìm thấy x)
- Tg bq: (KHPS)

(Tồi nhất)

$$f(n) = f(n/2) + 2$$

So sánh: $x < a[\text{giữa}]$ → tìm 1/2 TRÁI [đầu, giữa-1]

loại

$x > a[\text{giữa}]$ → 1/2 phải [giữa+1, cuối]

Finding the Maximum and Minimum of a Sequence

$$f(n) = 2f(n/2) + 2$$

Merge Sort The merge sort algorithm splits a list to be sorted with n items, where n is even, into two lists with $n/2$ elements each, and uses fewer than n comparisons to merge the two sorted lists of $n/2$ items each into one sorted list. Consequently, the number of comparisons used by the merge sort to sort a list of n elements is less than $M(n)$, where the function $M(n)$ satisfies the divide-and-conquer recurrence relation

$$M(n) = 2M(n/2) + n.$$

Fast Matrix Multiplication we showed that multiplying two $n \times n$ matrices using the definition of matrix multiplication required n^3 multiplications and $n^2(n-1)$ additions. Consequently, computing the product of two $n \times n$ matrices in this way requires $O(n^3)$ operations (multiplications and additions). Surprisingly, there are more efficient divide-and-conquer algorithms for multiplying two $n \times n$ matrices. Such an algorithm, invented by Volker Strassen in 1969, reduces the multiplication of two $n \times n$ matrices, when n is even, to seven multiplications of two $(n/2) \times (n/2)$ matrices and 15 additions of $(n/2) \times (n/2)$ matrices. (See [CoLeRiSt09] for the details of this algorithm.) Hence, if $f(n)$ is the number of operations (multiplications and additions) used, it follows that

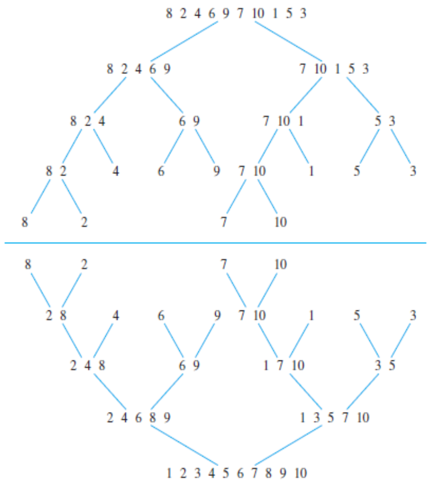
$$f(n) = 7f(n/2) + 15n^2/4$$

when n is even.

Merge Sort

The number of comparisons needed to merge sort a list with n elements is $O(n \log n)$.

assume that n, the number of elements in the list, is a power of 2, say 2^m.



At the end, there are 2^k lists at level k, each with 2^{m-k} elements.

$(k = 1, 2, \dots, m)$

$(k = m, m-1, m-2, \dots, 3, 2, 1)$

at level k, 2^k lists each with 2^{m-k} elements are merged into 2^{k-1} lists.

(each with 2^{m-k+1} elements.

at level $k-1$)

To do this a total of 2^{k-1} mergers of two lists, each with 2^{m-k} elements, are needed.

each of these mergers can be carried out using at most $2^{m-k} + 2^{m-k} - 1 = 2^{m-k+1} - 1$ comparisons.

$$\sum_{k=1}^{m} 2^{k-1}(2^{m-k+1} - 1) = \sum_{k=1}^{m} 2^m - \sum_{k=1}^{m} 2^{k-1} = m2^m - (2^m - 1) = n \log n - n + 1,$$

Binary Search

Tìm kiếm nhị phân

vị trí of x ≥ 1

không có x → 0

$x \in \{a_1, a_2, \dots, a_n\}$

1 đầu — trái — a — phải — n cuối

x

giữa = (đầu + cuối)/2

complexity BS

- Tồi nhất → số phép toán so sánh B_n (khi không tìm thấy x)
- Trung bình: (KTHS) (Tồi nhất)

So sánh: $x < a[\text{giữa}]$ → tìm 1/2 trái [đầu, giữa−1]

loại → 1 → 3

$x > a[\text{giữa}]$ → 1/2 phải [giữa+1, cuối]

$$B_n = B_{n/2} + 1$$

$$= (B_{n/4} + 1) + 1 = B_{n/2^2} + 2$$

$$= (B_{n/2^2 \cdot 2} + 1) + 2 = B_{n/2^3} + 3$$

$$= \cdots = \log_2 n + B_{n/2^{\log_2 n}}$$

$B_1 = B_{n/n} = 1$ so sánh

$(\log_2 n) + 1 =$

$B_n = O(\log_2 n)$

complexity logarit!