Chapter 6
COUNTING

Topics

THE BASICS OF COUNTING

Introduction

- Assume we have a set of objects with certain properties.
- Counting is used to determine the number of these objects.

Examples.

- Suppose that a password on a computer system consists of six, seven, or eight characters. Each of these characters must be a digit or a letter of the alphabet. Each password must contain at least one digit. How many such passwords are there?
- Count the number of operations used by an algorithm to study its time complexity.

…

Basic Counting Principles

Two basic counting principles:

The product rule. Suppose that a procedure is carried out by performing the tasks $T_1, T_2, \dots, T_m$ in sequence. If each task T_i, i = 1, 2, … , n, can be done in n_i ways, regardless of how the previous tasks were done, then there are $n_1 \cdot n_2 \cdot \cdots \cdot n_m$ ways to carry out the procedure.

The sum rule. Suppose that a task can be done in one of n_1 ways, in one of n_2 ways, … , or in one of n_m ways, where none of the set of n_i ways of doing the task is the same as any of the set of n_j ways, for all pairs i and j with $1 \le i < j \le m$.
hen the number of ways to do the task is $n_1 + n_2 + \cdots + n_m$.

Product Rule

Example. A new company with just two employees, Sanchez and Patel, rents a floor of a building with 12 offices. How many ways are there to assign different offices to these two employees?

Solution.

The procedure of assigning offices to these two employees consists of assigning an office to Sanchez, which can be done in 12 ways, then assigning an office to Patel different from the office assigned to Sanchez, which can be done in 11 ways. By the product rule, there are 12 · 11 = 132 ways to assign offices to these two employees.

Remark. The number of different subsets of a finite set S is $2^{|S|}$.

Product Rule

Example 1. The chairs of an auditorium are to be labeled with an uppercase English letter followed by a positive integer not exceeding 100. What is the largest number of chairs that can be labeled differently?

Example 2. There are 32 computers in a data center in the cloud. Each of these computers has 24 ports. How many different computer ports are there in this data center?

Example 3. How many different bit strings of length seven are there?

Example 4. **(Counting functions)** How many functions are there from a set with m elements to a set with n elements?

Example 5. **(Counting one-to-one functions)** How many one-to-one functions are there from a set with m elements to one with n elements?

Product Rule

Example 6. What is the value of k after the following code, where $n_1, n_2, \dots, n_m$ are positive integers, has been executed?

$k := 0$
for $i_1 := 1$ **to** n_1
  for $i_2 := 1$ **to** n_2
  .
  .
  .
    for $i_m := 1$ **to** n_m
      $k := k + 1$

Solution.

The initial value of k is zero. Each time the nested loop is traversed, 1 is added to k. Let T_i be the task of traversing the ith loop. Then the number of times the loop is traversed is the number of ways to do the tasks $T_1, T_2, \dots, T_m$. The number of ways to carry out the task T_j, j = 1, 2, … , m, is n_j, because the jth loop is traversed once for each integer ij with $1 \le i_j \le n_j$. By the product rule, it follows that the nested loop is traversed $n_1 n_2 \cdots n_m$ times. Hence, the final value of k is $n_1 n_2 \cdots n_m$.

Product Rule

If $A_1, A_2, \ldots, A_m$ are finite sets, then the number of elements in the Cartesian product of these sets is

$$|A_1 \times A_2 \times \cdots \times A_m| = |A_1| \cdot |A_2| \cdot \cdots \cdot |A_m|.$$

Sum Rule

Example. A student can choose a computer project from one of three lists. The three lists contain 23, 15, and 19 possible projects, respectively. No project is on more than one list. How many possible projects are there to choose from?

Solution.

The student can choose a project by selecting a project from the first list, the second list, or the third list. Because no project is on more than one list, by the sum rule there are 23 + 15 + 19 = 57 ways to choose a project.

Sum Rule

Example. What is the value of k after the following code, where $n_1, n_2, \ldots, n_m$ are positive integers, has been executed?

Solution.

The initial value of k is zero. This block of code is made up of m different loops. Each time a loop is traversed, 1 is added to k. To determine the value of k after this code has been executed, we need to determine how many times we traverse a loop. Note that there are n_i ways to traverse the ith loop. Because we only traverse one loop at a time, the sum rule shows that the final value of k, which is the number of ways to traverse one of the m loops is $n_1 + n_2 + \cdots + n_m$.

```
k := 0
for i1 := 1 to n1
    k := k + 1
for i2 := 1 to n2
    k := k + 1
    .
    .
    .
for im := 1 to nm
    k := k + 1
```

Sum Rule

If $A_1, A_2, \ldots, A_m$ are pairwise disjoint finite sets, then the number of elements in the union of these sets is

$$|A_1 \cup A_2 \cup \cdots \cup A_m| = |A_1| + |A_2| + \cdots + |A_m| \text{ when } A_i \cap A_j = \ \text{for all } i, j.$$

More Complex Counting Problems

Many complicated counting problems can be solved using both of these rules in combination.

Example. n a version of the computer language BASIC, the name of a variable is a string of one or two alphanumeric characters, where uppercase and lowercase letters are not distinguished. (An alphanumeric character is either one of the 26 English letters or one of the 10 digits.) Moreover, a variable name must begin with a letter and must be different from the five strings of two characters that are reserved for programming use. How many different variable names are there in this version of BASIC?

The Subtraction Rule (Inclusion–Exclusion for Two Sets)

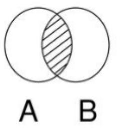
Used in counts where the decomposition yields two dependent count tasks with overlapping elements.

If we used the sum rule some elements would be counted twice.

Inclusion-exclusion principle: uses a sum rule and then corrects for the overlapping elements.

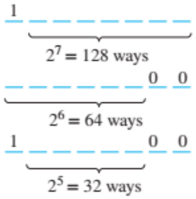
We used the principle for the cardinality of the set union.

$$|A \cup B| = |A| + |B| - |A \cap B|$$



The Subtraction Rule (Inclusion–Exclusion for Two Sets)

Example. How many bit strings of length eight either start with a 1 bit or end with the two bits 00?



The Subtraction Rule (Inclusion–Exclusion for Two Sets)

Example. How many positive integers not exceeding 1000 are divisible by 7 or 11?

The Principle of Inclusion-Exclusion

THE PRINCIPLE OF INCLUSION–EXCLUSION Let $A_1, A_2, \ldots, A_n$ be finite sets. Then

$$|A_1 \cup A_2 \cup \cdots \cup A_n| = \sum_{1 \le i \le n} |A_i| - \sum_{1 \le i < j \le n} |A_i \cap A_j| + \sum_{1 \le i < j < k \le n} |A_i \cap A_j \cap A_k| - \cdots + (-1)^{n+1} |A_1 \cap A_2 \cap \cdots \cap A_n|.$$

Counting Onto Functions

Theorem.

Let m and n be positive integers with $m \ge n$. Then, there are

$$n^m - C(n,1)(n-1)^m + C(n,2)(n-2)^m - \cdots + (-1)^{n-1} C(n, n-1) \cdot 1^m$$

onto functions from a set with m elements to a set with n elements.

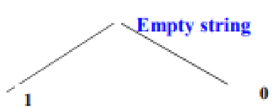
Example. How many onto functions are there from a set with 6 elements to a set with three elements?

Tree Diagrams

Tree: is a structure that consists of a root, branches and leaves.

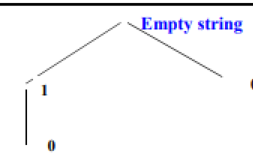
• Can be useful to represent a counting problem and record the choices we made for alternatives. The count appears on the leaf nodes.

Example. How many bit strings of length four do not have two consecutive 1s?



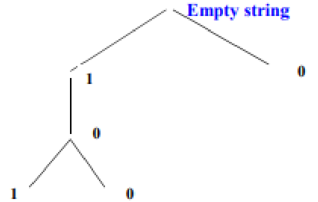
Tree Diagrams

Example. How many bit strings of length four do not have two consecutive 1s?



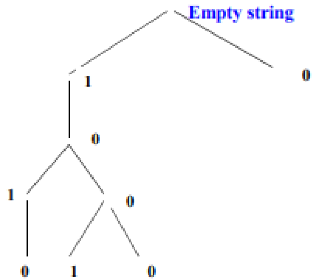
Tree Diagrams

Example. How many bit strings of length four do not have two consecutive 1s?



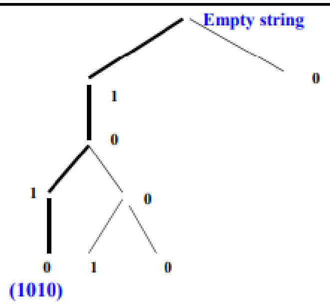
Tree Diagrams

Example. How many bit strings of length four do not have two consecutive 1s?



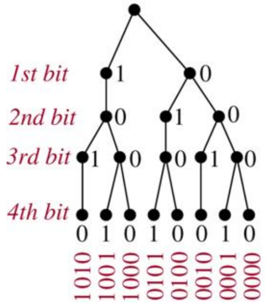
Tree Diagrams

Example. How many bit strings of length four do not have two consecutive 1s?



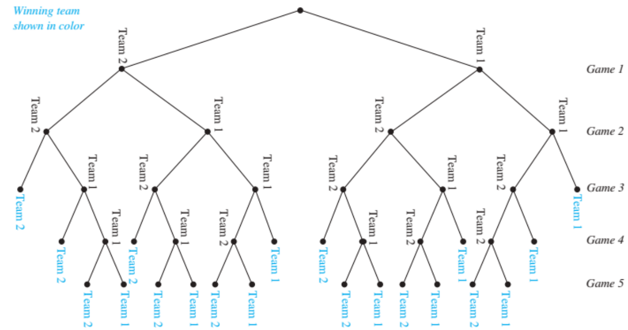
Tree Diagrams

Example. How many bit strings of length four do not have two consecutive 1s?



Tree Diagrams

Example. A playoff between two teams consists of at most five games. The first team that wins three games wins the playoff. In how many different ways can the playoff occur?



Exercises

1. An office building contains 27 floors and has 37 offices on each floor. How many offices are in the building?

2. How many bit strings are there of length eight?

3. How many bit strings of length ten both begin and end with a 1?

4. How many bit strings are there of length six or less, not counting the empty string?

5. How many positive integers between 5 and 31

a) are divisible by 3? Which integers are these?

b) are divisible by 4? Which integers are these?

c) are divisible by 3 and by 4? Which integers are these?

Exercises

6. How many positive integers less than 1000

a) are divisible by 7?

b) are divisible by 7 but not by 11?

c) are divisible by both 7 and 11?

7. How many different functions are there from a set with 10 elements to sets with the following numbers of elements?

a) 2 b) 3 c) 4 d) 5

8. How many one-to-one functions are there from a set with five elements to sets with the following number of elements?

a) 4 b) 5 c) 6 d) 7

Exercises

9. How many bit strings of length seven either begin with two 0s or end with three 1s?

10. How many bit strings of length eight contain either three consecutive 0s or four consecutive 1s?

11. How many positive integers not exceeding 100 are divisible by 6 or 9?

A. 17 B. 22 C. 26 D. 27 E. None of the other choices is correct

12. How many functions are there from the set {1, 2, 3, 4, 5} to the set {0, 1, 2}?

A. 125 B. 60 C. 243 D. 15 E. None of the other choices is correct

13. How many one-to-one functions are there from the set {1, 2, 3} to the set {1, 2, 3, 4, 5, 6}?

A. $6 \cdot 5 \cdot 4$ B. 6^3 C. 0 D. 18 E. None of the choices is correct

Exercises

14. Find the number of positive integers not exceeding 100 that are not divisible by 5 or by 7.

15. How many bit strings of length eight do not contain six consecutive 0s?

16. How many onto functions are there from a set with seven elements to one with five elements?

THE PIGEONHOLE PRINCIPLE

Pigeonhole principle

- Assume you have a set of objects and a set of bins used to store objects.
- The pigeonhole principle states that if there are more objects than bins then there is at least one bin with more than one object.

Example. 7 balls and 5 bins to store them

At least one bin with more than 1 ball exists.

Pigeonhole principle

Theorem. If there are k+1 objects and k bins. Then there is at least one bin with two or more objects.

k bins

Example. Assume 367 people. Are there any two people who has the same birthday?

The generalized Pigeonhole principle

Theorem. If N objects are placed into k boxes, then there is at least one box containing at least $\lceil N/k \rceil$ objects.

Example 1. Assume 100 people. Can you tell something about the number of people born in the same month?

Example 2. How many people must be selected to guarantee that there are at least 10 people born in the same month?

Example 3. How many cards must be selected from a standard deck of 52 cards to guarantee that there are at least 3 cards of the same suit? at least 3 hearts?

Example 4. How many students, each of whom comes from one of the 50 states, must be enrolled in a university to guaranteed that there are at least 100 who come from the same state?

PERMUTATIONS AND COMBINATIONS

Permutations

- A permutation of a set S of distinct elements is an **ordered** sequence that contains each element in S exactly once.

Example. S = {A, B, C} → Permutations of S: ABC, ACB, BAC, BCA, CAB, CBA

Number of permutations

- Assume we have a set S with n elements. S={a_1 a_2 … a_n}.
- **Question:** How many different permutations are there?

- In how many different ways we can choose the first element of the permutation? **n** (**either** a_1 or a_2 … or a_n)
- Assume we picked a_2.
- In how many different ways we can choose the remaining elements? **n-1** (**either** a_1 or a_3 … or a_n **but not** a_2**)**
- **Assume** we picked a_j.
- In how many different ways we can choose the remaining elements? **n-2** (**either** a_1 or a_3 … or a_n **but not** a_2 **and not** a_j)

P(n,n) = n.(n-1)(n-2)…1 = n!

r - permutations

- An **ordered** arrangement of r distinct elements of S is called an r-permutation of S.
- The number of r-permutations of a set with n = |S| elements is

$$\boxed{P(n,r) = n(n-1)(n-2)\cdots(n-r+1) = \frac{n!}{(n-r)!}\ ,\ 0 \le r \le n\,.}$$

Note.

$$P(n,n) = \frac{n!}{(n-n)!} = n!$$

Permutations

Examples.

1. Let S = {1, 2, 3}.

The arrangement 3, 1, 2 is a permutation of S (3! = 6 ways)

The arrangement 3, 2 is a 2-permutation of S (3·2=3!/1! = 6 ways)

2. There is an armed nuclear bomb planted in your city, and it is your job to disable it by cutting wires to the trigger device. There are 10 wires to the device. If you cut exactly the right three wires, in exactly the right order, you will disable the bomb, otherwise it will explode! If the wires all look the same, what are your chances of survival?

→ P(10,3) = 10×9×8 = 720, so there is a 1 in 720 chance that you'll survive.

3. How many permutations of the letters ABCDEFGH contain the string ABC?

→ ABC must occur as a block, i.e. consider it as one object Then, it'll be the number of permutations of six objects (ABC, D, E, F, G, H), which is 6! = 720

Combinations

- An r-combination of elements of a set S is an **unordered** selection of r elements from the set. Thus, an r-combination is simply a subset T ⊆ S with r members, |T| = r.

Example. S = {1, 2, 3, 4}, then {1, 3} is a 2-combination from S

- The number of r-combinations of a set with n = |S| elements, $0 \le r \le n$, is

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

- Corrolary:

$$C(n,k) = C(n,n-k)$$

Binomial coefficients

- The number of k-combinations out of n elements C(n,k) is often denoted as:

$$\binom{n}{k}$$

and reads **n choose k**. The number is also called **a binomial coefficient.**

- Binomial coefficients occur as coefficients in the expansion of powers of binomial expressions such as

$$(a+b)^n$$

- **Definition:** a binomial expression is the sum of two terms *(a+b)*.

Binomial theorem

THE BINOMIAL THEOREM Let x and y be variables, and let n be a nonnegative integer. Then

$$(x+y)^n = \sum_{j=0}^{n} \binom{n}{j} x^{n-j} y^j = \binom{n}{0} x^n + \binom{n}{1} x^{n-1} y + \cdots + \binom{n}{n-1} x y^{n-1} + \binom{n}{n} y^n.$$

Combinations

Example. How many bit strings of length n contain exactly r 1s?

Solution. The positions of r 1s in a bit string of length n form an r-combination of the set {1, 2, 3, … , n}. Hence, there are C(n, r) bit strings of length n that contain exactly r 1s.

Permutations with Repetition

Theorem. The number of r-permutations of a set of n objects with repetition allowed is n^r.

Example. How many strings of length r can be formed from the uppercase letters of the English alphabet?

Solution. By the product rule, because there are 26 uppercase English letters, and because each letter can be used repeatedly, we see that there are 26r strings of uppercase English letters of length r.

Combinations with Repetition

Theorem. There are C(n + r − 1, r) = C(n + r − 1, n − 1) r-combinations from a set with n elements when repetition of elements is allowed.

Example. Suppose that a cookie shop has four different kinds of cookies. How many different ways can six cookies be chosen? Assume that only the type of cookie, and not the individual cookies or the order in which they are chosen, matters.

Solution. The number of ways to choose six cookies is the number of 6-combinations of a set with four elements.

$$C(4+6-1,6) = C(9,6) = C(9,3) = 84$$

Exercises

1. How many bit strings of length 10 contain
a) exactly four 1s?
b) at most four 1s?
c) at least four 1s?
d) an equal number of 0s and 1s?
2. How many bit strings of length 10 have
a) exactly three 0s?
b) more 0s than 1s?
c) at least seven 1s?
d) at least three 1s?

Exercises

3. How many permutations of the letters ABCDEFG contain
a) the string BCD?
b) the strings BA and GF?
c) the strings ABC and DE?
d) the strings ABC and CDE?

RECURRENCE RELATIONS

Recurrence Relations

Definition. A **recurrence relation** for the sequence $\{a_n\}$ is an equation that expresses a_n in terms of one or more of the previous terms of the sequence, namely, $a_0, a_1, \ldots, a_{n-1}$, for all integers n with $n \geq n_0$, where n_0 is a nonnegative integer.

A sequence is called a **solution** of a recurrence relation if its terms satisfy the recurrence relation.

The **initial (base) conditions** for a sequence specify the terms that precede the first term where the recurrence relation takes effect.

Recurrence Relations

Example. A sequence $\{a_n\}$:
Recurrence relation $a_n = a_{n-1} - a_{n-2}$ for $n = 2, 3, 4, \ldots$
$a_0 = 3$ and $a_1 = 5$.
What are a_2 and a_3?
Initial conditions: $a_0 = 3$ and $a_1 = 5$

Recurrence Relations

Example. Consider the recurrence relation $a_n = 2a_{n-1} - a_{n-2}$ for $n = 2, 3, 4, \ldots$
Which of the following are solutions?
a) $a_n = 3n$ b) $a_n = 2^n$ c) $a_n = 5$
Solution.

Modeling With Recurrence Relations

Example (Compound interest). A person deposited $10,000 in a saving account at the rate of 11% a year with interest compounded annually. How much will be in the account after 30 years?

Modeling With Recurrence Relations

Example (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.

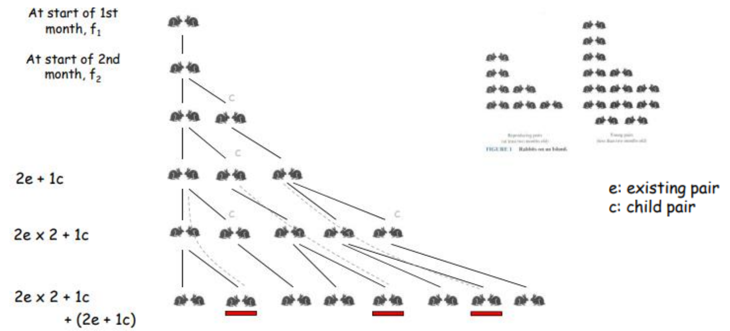
Modeling With Recurrence Relations

Example (Rabbits and the Fibonacci Numbers).

Reproducing pairs (at least two months old)	Young pairs (less than two months old)	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

Modeling With Recurrence Relations

Example (Rabbits and the Fibonacci Numbers).



Modeling With Recurrence Relations

Example (Rabbits and the Fibonacci Numbers).

Solution.

f_n = the number of pairs of rabbits at the start of nth months

$f_1 = 1, f_2 = 1$

$f_n = f_{n-1} + f_{n-2}$ for $n \geq 3$

Modeling With Recurrence Relations

Example (The tower of Hanoi).

In the late nineteenth century, the French mathematician Édouard Lucas invented a puzzle consisting of three pegs on a board with disks of different sizes. Initially all of the disks are on the first peg in order of size, with the largest on the bottom.

Rules: You are allowed to move the disks one at a time from one peg to another as long as a larger disk is never placed on a smaller.

Goal: Using allowable moves, end up with all the disks on the second peg in order of size with largest on the bottom.

Play game: https://mathsisfun.com/games/towerofhanoi.html

Modeling With Recurrence Relations

Example (The tower of Hanoi).

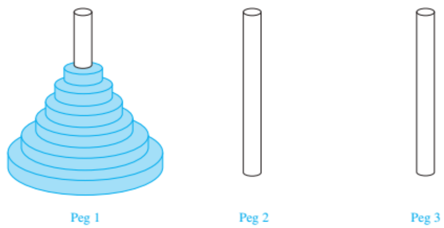


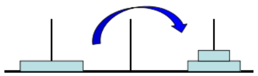
FIGURE 2 **The initial position in the Tower of Hanoi.**

Modeling With Recurrence Relations

Example (The tower of Hanoi).

- Let H_n denote the number of moves needed to solve the Tower of Hanoi problem with n disks.
- $n = 1$: $H_1 = 1$

- $n > 1$: Move n disks from peg 1 to peg 2
  - Move the top $(n-1)$ disks to peg 3. During these moves, the bottom disk on peg 1 stays fixed.

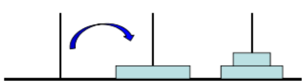


case: $n = 3$

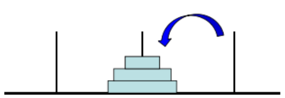
Modeling With Recurrence Relations

Example (The tower of Hanoi).

- Move the remaining disk on peg 1 to peg 2.



- Move the (n-1) disks on peg 3 to peg 2.



Thus, $H_n = 2H_{n-1} + 1$ and $H_1 = 1$

Modeling With Recurrence Relations

- $H_n = 2H_{n-1} + 1$
  $= 2(2H_{n-2} + 1) + 1 = 2^2H_{n-2} + 2 + 1$
  $= 2^2(2H_{n-3} + 1) + 2 + 1 = 2^3H_{n-3} + 2^2 + 2 + 1$
  $\cdots$
  $= 2^{n-1}H_1 + 2^{n-2} + 2^{n-3} + \cdots + 2^1 + 1$
  $= 2^{n-1} + 2^{n-2} + 2^{n-3} + \cdots + 2^1 + 1$
  $= 2^n - 1$

  $\therefore\ H_n = 2^n - 1\ ,\ n \geq 1$

- $H_{64} = 2^{64} - 1 = 18{,}446{,}744{,}073{,}709{,}551{,}615$ (moves)
  (500 billion years if one move per second)

Modeling With Recurrence Relations

Example (Counting bit strings). 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?

Solution. Let a_n denote the number of bit strings of length n without two consecutive 0s. To obtain a recurrence relation for $\{a_n\}$ note that the number of bit strings of length n that do not have two consecutive 0s is the number of bit strings ending with a 0 plus the number of such bit strings ending with a 1.

Modeling With Recurrence Relations

Example (Counting bit strings).

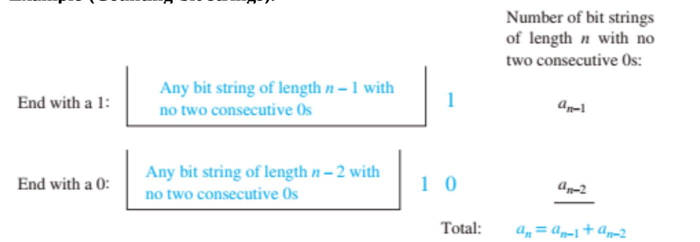


FIGURE 4 **Counting bit strings of length n with no two consecutive 0s.**

Modeling With Recurrence Relations

Example (Counting bit strings). Now assume that $n \geq 3$:

- The bit strings of length n ending with 1 without two consecutive 0s are the bit strings of length n −1 with no two consecutive 0s with a 1 at the end. Hence, there are a_{n-1} such bit strings.
- The bit strings of length n ending with 0 without two consecutive 0s are the bit strings of length n −2 with no two consecutive 0s with 10 at the end. Hence, there are a_{n-2} such bit strings.

We conclude that $a_n = a_{n-1} + a_{n-2}$, for $n \geq 3$

Modeling With Recurrence Relations

Example (Counting bit strings). The initial conditions are:

$a_1 = 2$, since both the bit strings 0 and 1 do not have consecutive 0s.

$a_2 = 3$, since the bit strings 01, 10, and 11 do not have consecutive 0s, while 00 does.

To obtain a_5, we use the recurrence relation three times to find that:

$a_3 = a_2 + a_1 =$

$a_4 = a_3 + a_2 =$

$a_5 = a_4 + a_3 =$

Exercises

1. A vending machine dispensing books of stamps accepts only one-dollar coins, $1 bills, and $5 bills.

a) Find a recurrence relation for the number of ways to deposit n dollars in the vending machine, where the order in which the coins and bills are deposited matters.

b) What are the initial conditions?

c) How many ways are there to deposit $10 for a book of stamps?

2. A country uses as currency coins with values of 1 peso, 2 pesos, 5 pesos, and 10 pesos and bills with values of 5 pesos, 10 pesos, 20 pesos, 50 pesos, and 100 pesos. Find a recurrence relation for the number of ways to pay a bill of n pesos if the order in which the coins and bills are paid matters.

Exercises

3. a) Find a recurrence relation for the number of bit strings of length n that contain a pair of consecutive 0s.

b) What are the initial conditions?

c) How many bit strings of length seven contain two consecutive 0s?

4. a) Find a recurrence relation for the number of bit strings of length n that do not contain three consecutive 0s.

b) What are the initial conditions?

c) How many bit strings of length seven do not contain three consecutive 0s?

5. What is the solution of the recurrence relation $a_n = a_{n-1} + 2a_{n-2}$ with $a_0 = 2$ and $a_1 = 7$?

DIVIDE-AND-CONQUER ALGORITHMS AND RECURRENCE RELATIONS

Divide-and-Conquer

Very important strategy in computer science:

- Divide problem into smaller parts
- Independently solve the parts
- Combine these solutions to get overall solution

Divide-and-Conquer

The most-well known algorithm design strategy:

1. **Divide** the problem into sub-problems that are similar to the original but smaller in size.

2. **Conquer** the sub-problems by solving them **recursively**.

3. **Combine** the solutions to create a solution to the original problem.

Remark:

- If the subproblems are relatively large, then divide-and-conquer is applied again.
- If the subproblems are small, they are solved without splitting.

Divide-and-Conquer Recurrence Relations

We can use recurrence relations to analyze the computational complexity of divide-and-conquer algorithms.

- Suppose that a recursive algorithm divides a problem of size n into a subproblems
- Each subproblem is of size $n\ /\ b$ (for simplicity, assume that n is a multiple of b).
- A total of $g(n)$ extra operations are required in the conquer step of the algorithm to
- combine the solutions of the subproblems into a solution of the original problem.
- $f(n)$ represents the number of operations required to solve the problem of size n.

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

This is called a **divide-and-conquer recurrence relation**.

Some Examples

Binary search.

Problem. Find the location of an element in a sorted collection of n items

Algorithm steps:

1. Divide: Find the middle term a_m of the list, where $m = \lfloor(n+1)/2\rfloor$. If $x > a_m$, the search for x is restricted to the second half of the list, which is $a_{m+1}, a_{m+2}, \dots, a_n$. If x is not greater than a_m, the search for x is restricted to the first half of the list, which is $a_1, a_2, \dots, a_m$.
2. Conquer: The search has now been restricted to a list with no more than $\lceil n/2\rceil$ elements. (Recall that $\lceil x\rceil$ is the smallest integer greater than or equal to x.) Using the same procedure, compare x to the middle term of the restricted list. Then restrict the search to the first or second half of the list. Repeat this process until a list with one term is obtained. Then determine whether this term is x.

Some Examples

Binary Search.

Break list into 1 sub-problem (smaller list) $(a = 1)$ of size $\le \lceil n/2\rceil$ $(b = 2)$. Two comparisons are needed to implement this reduction (one to determine which half of the list to use and the other to determine whether any terms of the list remain) $(g(n) = 2)$.

$\Rightarrow f(n) = f(n/2) + 2$ where n is even

ALGORITHM 3 The Binary Search Algorithm.

procedure *binary search* (x: integer, $a_1, a_2, \dots, a_n$: increasing integers)
$i := 1${i is left endpoint of search interval}
$j := n$ {j is right endpoint of search interval}
while $i < j$
  $m := \lfloor(i+j)/2\rfloor$
  if $x > a_m$ **then** $i := m + 1$
  else $j := m$
if $x = a_i$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript i of the term a_i equal to x, or 0 if x is not found}

Some Examples

Merge Sort.

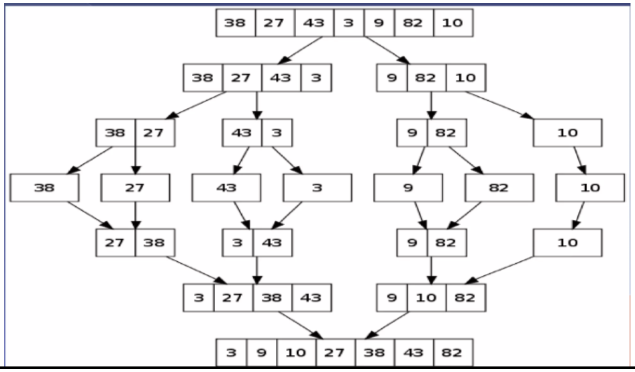
Problem. Sort a sequence of n elements into order

Algorithm steps:

1. Divide: Divide the n-element sequence to be sorted into 2 subsequences of n/2 elements each.
2. Conquer: Sort the 2 subsequences recursively until we have list sizes of length 1, in which case the list itself is returned.
3. Combine: Merge the 2 sorted subsequences to produce the sorted answer.

Some Examples

Merge Sort.



Some Examples

Merge Sort.

Split a list to be sorted with n items, where n is even, into two lists (a = 2) with n⁄2 elements each (b = 2) and uses fewer than n comparisons (g(n) = n) to merge the two sorted lists of n⁄2 items each into one sorted list.

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

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

Divide-and-conquer recurrence relation

Suppose that f satisfies this recurrence relation whenever n is divisible by b. Let $n = b^k$, where k is a positive integer. Then

$$\begin{aligned}
f(n) &= af(n/b) + g(n) \\
&= a^2f(n/b^2) + ag(n/b) + g(n) \\
&= a^3f(n/b^3) + a^2g(n/b^2) + ag(n/b) + g(n) \\
&\vdots \\
&= a^kf(n/b^k) + \sum_{j=0}^{k-1} a^j g(n/b^j).
\end{aligned}$$

Because $\frac{n}{b^k} = 1$, it follows that

$$f(n) = a^k f(1) + \sum_{j=0}^{k-1} a^j g(n/b^j).$$

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 $C_2 = -c/(a-1)$.

Example

Let $f(n) = 5f(n/2) + 3$ and $f(1) = 7$. Find $f(2^k)$, where k is a positive integer. Also, estimate $f(n)$ if f is an increasing function.

Solution: From the proof of Theorem 1, with $a = 5$, $b = 2$, and $c = 3$, we see that if $n = 2^k$, then

$$\begin{aligned} f(n) &= a^k[f(1) + c/(a-1)] + [-c/(a-1)] \\ &= 5^k[7 + (3/4)] - 3/4 \\ &= 5^k(31/4) - 3/4. \end{aligned}$$

Also, if $f(n)$ is increasing, Theorem 1 shows that $f(n)$ is $O(n^{\log_b a}) = O(n^{\log 5})$. ◀

Example. Complexity of Binary Search

Give a big-O estimate for the number of comparisons used by a binary search.

Solution. it was shown that $f(n) = f(n/2) + 2$ when n is even, where f is the number of comparisons required to perform a binary search on a sequence of size n. Hence, from Theorem, it follows that $f(n)$ is $O(\log n)$.

The Master Theorem

MASTER THEOREM Let f be an increasing function that satisfies the recurrence relation

$$f(n) = af(n/b) + cn^d$$

whenever $n = b^k$, where k is a positive integer, $a \geq 1$, b is an integer greater than 1, and c and d are real numbers with c positive and d nonnegative. Then

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

Example. Complexity of Merge Sort

The number of comparisons used by the merge sort to sort a list of n elements is less than M(n), where $M(n) = 2M(n/2) + n$. By the master theorem, we find that M(n) is $O(n \log n)$, which agrees with the estimate found in the last chapter.

Exercises

1. How many comparisons are needed for a binary search in a set of 64 elements?

2. Suppose that $f(n) = f(n/3) + 1$ when n is a positive integer divisible by 3, and $f(1) = 1$. Find

a) $f(3)$ b) $f(27)$ c) $f(729)$

3. Suppose that $f(n) = 2f(n/2) + 3$ when n is an even positive integer, and $f(1) = 5$. Find

a) $f(2)$ b) $f(8)$ c) $f(64)$ d) $f(1024)$

4. Suppose that $f(n) = f(n/5) + 3n^2$ when n is a positive integer divisible by 5, and $f(1) = 4$. Find

a) $f(5)$ b) f(125) c) $f(3125)$

Exercises

5. Find $f(n)$ when $n = 2^k$, where f satisfies the recurrence relation $f(n) = f(n/2) + 1$ with $f(1) = 1$.

6. Give a big-O estimate for the function f in Exercise 5 if f is an increasing function.

7. Find $f(n)$ when $n = 3^k$, where f satisfies the recurrence relation $f(n) = 2f(n/3) + 4$ with $f(1) = 1$.

8. Give a big-O estimate for the function f in Exercise 7 if f is an increasing function.

Thanks