We will cover these parts of the book (8th edition):

5.2.1-5.2.3
5.3.1-5.3.3 (up to page 371)
5.4.1-5.4.3
6.1, 6.2

Strong Induction(Second principle)

- There is another proof technique that is very similar to the principle of mathematical induction. It is called the second principle of mathematical induction. It can be used to prove that a propositional function $P(n)$ is true for any natural number n.
- We should complete two steps:

1. **Basis step**: Verify $P(1)$ (or $P(0)$) is true.
2. **Inductive step**: Show that the conditional statement $[P(1) \wedge P(2) \wedge \cdots \wedge P(k)] \rightarrow P(k+1)$ is true for all positive integers k.

Strong Induction(Second principle)

- **Example**: Show that every integer greater than 1 can be written as the product of primes.

1. Show that $P(2)$ is true.

   P(2) is true, because 2 is the product of one prime: itself

2. Show that if $P(2), P(3), \ldots, P(n)$ is true, then $P(n+1)$ is true

   Two possible cases:

   1. $(n+1)$ is prime. So $P(n+1)$ is true.

   2. $(n+1)$ is composite that can be written as the product of two integers a and b, $2 \leq a \leq b < n+1$. So by the induction hypothesis, both a and b can be written as the product of primes. So $n+1 = a * b$ can be written as the product of primes.

Recursive Definition

- Recursion is a principle closely related to mathematical induction.
- In a recursive definition, an object is defined in terms of itself.
- We can recursively define sequences, functions and sets.
- Example: The sequence a_n of powers of 2 is given by $a_n = 2^n$ for $n = 0,1,2, \ldots$.
- The same sequence can also be defined recursively:
  - $a_0 = 1$
  - $a_{n+1} = 2a_n$ for $n = 1,2, \ldots$
- Obviously, induction and recursion are similar principles.

Recursively Defined Functions

- We use two steps to define a function with the set of nonnegative integers as its domain:
  1. **Basis step**: Specify the value of the function at zero.
  2. **Recursive step**: Give a rule for finding its value at an integer from its values at smaller integers.
- Such a definition is called a **recursive** or **inductive definition**.

Recursively Defined Functions

- Example: Find $f(1), f(2), f(3)$, and $f(4)$ if $f(0) = 3$ and $f(n+1) = 2f(n) + 3$.
  - $f(1) = 2f(0) + 3 = 9$
  - $f(2) = 2f(1) + 3 = 21$
  - $f(3) = 2f(2) + 3 = 45$
  - $f(4) = 2f(3) + 3 = 93$
- How can we define the factorial $f(n) = n!$
- Remember $f(0) = 1$
- $f(n+1) =?$

Recursively Defined Functions

- $f(1) = 1f(0) = 1$
- $f(2) = 2f(1) = 2$
- $f(3) = 3f(2) = 6$
- $f(4) = 4f(3) = 24$

- So we can say

- $f(n+1) = (n+1)f(n)$
- $f(0) = 1$

Recursively Defined Sets

- Like functions, we have two steps:
  1. **Basis step**: An initial collection of elements is specified(boundary conditions).
  2. **Recursive step**: Rules for construction of additional elements from elements in the set.

Recursively Defined Sets

- Example: Let S recursively defined by:
  - $3 \in S$
  - $(x + y) \in S$ if $x \in S$ and $y \in S$
- Find some elements of S.
- $3 \in S \Rightarrow 6 \in S$
- $3,6 \in S \Rightarrow 9 \in S$
- $3,6,9 \in S \Rightarrow 12,15 \in S$
- So it seems S is the set of positive integers divisible by 3.

Recursively Defined Sets

- **Proof:**
- Let A be the set of all positive integers divisible by 3. To show that $A = S$, we must show $A \subseteq S$ and $S \subseteq A$
- **Part 1.** To prove $A \subseteq S$, we must show that every positive integer divisible by 3 is in S.
- We will use mathematical induction to show this. Let $P(n)$ be the statement "$3n \in S$"
  - $P(1)$ is true because $3 \in S$.
  - Assume $P(n)$ is true. So $3n \in S$. And we know $3 \in S \Rightarrow 3n + 3 \in S \Rightarrow 3(n + 1) \in S \Rightarrow P(n + 1)$ is true.
  - So $A \subseteq S$

Recursively Defined Sets

- **Part 2.** To prove $S \subseteq A$, must show $\forall x \in S \rightarrow x \in A$
  - **Basis step**: All initial elements of S are in A. $3 \in A$
  - **Inductive step**: To show $(x + y) \in A$ whenever $x \in S$ and $y \in S$ are in A.
  - $x, y \in S \Rightarrow 3|x$ and $3|y$, so $3|(x + y) \Rightarrow (x + y) \in A$
  - So $S \subseteq A$

- Overall conclusion: $A = S$
- This form of induction, is called **Structural Induction.**

Recursively Defined Sets

- **Definition I:** The set $\sum^*$ of strings over the alphabet $\sum$ is defined recursively by:
  - Basis step: $\lambda \in \sum^*$ (where λ is the empty string containing no symbols)
  - Recursive step: If $w \in \sum^*$ and $x \in \sum$, then $wx \in \sum^*$.

Recursively Defined Sets

- **Definition II:** Two strings can be combined via the operation of concatenation. Let $\sum$ be a set of symbols and $\sum^*$ the set of strings formed from symbols in $\sum$. We can define the concatenation of two strings, denoted by $\cdot$, recursively as follows:
  - Basis step: If $w \in \sum^*$, then $w.\lambda = w$, where λ is the empty string
  - Recursive step: If $w_1 \in \sum^*$ and $w_2 \in \sum^*$ and $x \in \sum$, then $w_1.(w_2x) = (w_1.w_2)x$.

Recursively Defined Sets

- Another example: The well-formed formulas of variables, numerals and operators from $\{+, -, *, /, \hat{}\}$ are defined by:
  - x is a well-formed formula if x is a numeral or variable.
  - $(f + g), (f - g), (f * g), (f / g), (f \hat{} g)$ are well-formed formulas if f and g are.

- Find some elements of S.
- For example:
- $(x - y)$
- $((z/3) - y)$
- $((z/3) - (6 + 5))$

Recursive Algorithms

- An algorithm is called recursive if it solves a problem by reducing it to an instance of the same problem with smaller input.

- **Example1:** Recursive Euclidean Algorithm

- **procedure** gcd(a, b: positive integers $a < b$)
- **if** $a == 0$ **then**
  - result = b
- **else**
  - result **=** gcd(b mod a, a)
- **endif**
- **return** result {result is gcd(a, b)}

Recursive Algorithms

- The Fibonacci sequence:
- $f(0) = 1$
- $f(1) = 1$
- $f(n) = f(n-1) + f(n-2)$

  - $f(0) = 1$
  - $f(1) = 1$
  - $f(2) = 2$
  - $f(3) = 3$
  - $f(4) = 5$
  - $f(5) = 8$

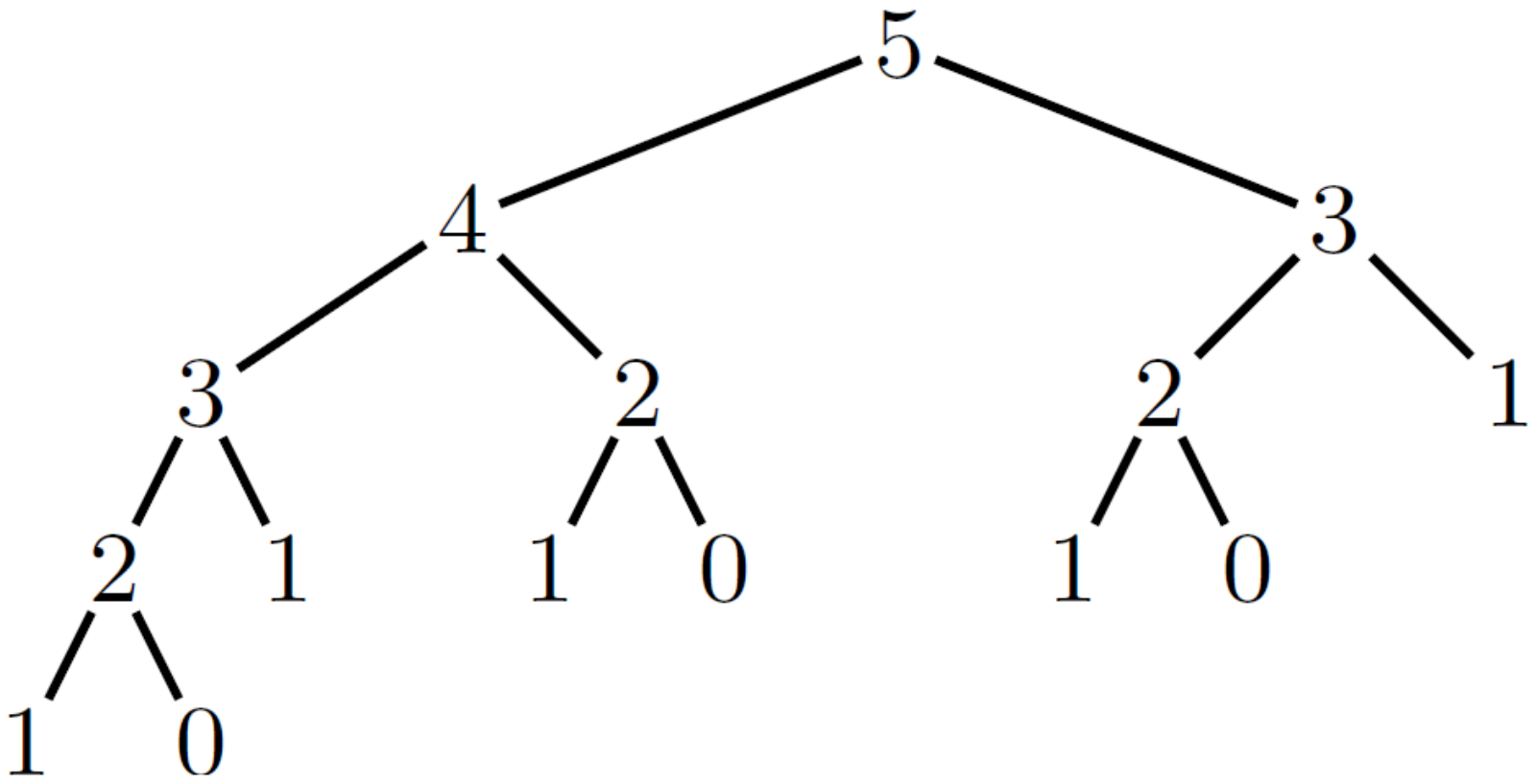
Recursive Algorithms

- **Example2:** Recursive Fibonacci Algorithm

- **procedure** fibo(n: positive integer)
- **if** $(n == 0)$ **then**
  - result = 1
- **else if** $(n == 1)$ **then**
  - result = 1
- **else**
  - result = fibo(n-1) + fibo(n-2)
- **endif**
- **return** result {result is fibo(n)}

Recursive Algorithms

- **Example2:** Recursive Fibonacci Algorithm
- Exponential Complexity!



Recursive Algorithms

- Linear Complexity:
- **procedure** iterative-fibo(n: positive integer)
- **if** $(n == 0)$ **then**
  - y = 0
- **else**
  - $x = 0, y = 1$
  - **for** i = 1 to n-1 **do**
    - z = x + y
    - x = y
    - y = z
  - **endfor**
- **endif**
- **return** y {y is fibo(n)}

Recursive Algorithms

- For every recursive algorithm, there is an equivalent iterative algorithm.
- Recursive algorithms are often shorter, more elegant, and easier to understand than their iterative counterparts.
- However, iterative algorithms are usually more efficient in their use of space and time.

Recursive Algorithms

- **Example3:** Recursive Algorithm for a^n

- **procedure** power(a: positive real number, n: positive integer)
- **if** $(n == 0)$ **then**
  - result = 1
- **else**
  - result = a*power(a, n-1)
- **endif**
- **return** result {result is a^n}

Recursive Algorithms

- **Example3:** Recursive Algorithm for a^n
- **Proof:** We use mathematical induction on n
  - **Basis step**: $power(a, 0) = 1$. It's correct because $a^0 = 1$
  - **Inductive step**: Suppose $power(a, k) = a^k$. The algorithm computes $power(a, k+1) = a * power(a, k) = a * a^k = a^{k+1}$ which is correct

- Generally, we need to use strong induction to prove that recursive algorithms are correct

Now let us do some…

- Counting

Basic Counting Principles

- **Counting problems** are of the following kind:
- "**How many** different 8-letter passwords are there?"
- "**How many** possible ways are there to pick 11 soccer players out of a 20-player team?"
- Most importantly, counting is the basis for computing **probabilities of discrete events**.
- ("What is the probability of winning the lottery?")

Basic Counting Principles

- **The product rule:**
- Suppose that a procedure can be broken down into two successive tasks. If there are n_1 ways to do the first task and n_2 ways to do the second task after the first task has been done, then there are $n_1 n_2$ ways to do the procedure.
- **Generalized product rule:**
- If we have a procedure consisting of sequential tasks $T_1, T_2, \ldots, T_m$ that can be done in $n_1, n_2, \ldots, n_m$ ways, respectively, then there are $n_1 \cdot n_2 \cdot \ldots \cdot n_m$ ways to carry out the procedure.

Basic Counting Principles

- **Example1:**
- How many different license plates are there that contain exactly three English letters ?
- **Solution:**
- There are 26 possibilities to pick the first letter, then 26 possibilities for the second one, and 26 for the last one.
- So there are 26·26·26 = **17576** different license plates.

Basic Counting Principles

- **Example2:**
- 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:**
- There are 12 offices for the first one to choose. When he chose his office, the number of remaining offices to be chosen is 11. So the answer is $12 \cdot 11 = 132$

Basic Counting Principles

- **Example3:** 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? $26 * 100 = 2600$
- **Example4:** 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? $24 * 32 = 768$
- **Example5:** How many different bit strings of length seven are there? $2^7 = 128$
- **Example6:** Use the product rule to show that the number of different subsets of a finite set S is $2^{|S|}$.
- There are $2^{|S|}$ bit strings of length $|S|$.

Basic Counting Principles

- **The sum rule:**
- If a task can be done in n_1 ways and a second task in n_2 ways, and if these two tasks cannot be done at the same time, then there are $n_1 + n_2$ ways to do either task.

- **Generalized sum rule:**
- If we have tasks $T_1, T_2, \ldots, T_m$ that can be done in $n_1, n_2, \ldots, n_m$ ways, respectively, and no two of these tasks can be done at the same time, then there are $n_1 + n_2 + \ldots + n_m$ ways to do one of these tasks.

Basic Counting Principles

- **Example1:**
- The department will award a free computer to either a CS student or a CS professor. How many different choices are there, if there are 530 students and 15 professors?

There are 530 + 15 = 545 choices.

- **Example2:**
- 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? 23+15+19=57

Basic Counting Principles

- The sum and product rules can also be phrased in terms of **set theory**.

- **Sum rule:** Let $A_1, A_2, \ldots, A_m$ be disjoint sets. Then the number of ways to choose any element from one of these sets is $|A_1 \cup A_2 \cup \ldots \cup A_m| = |A_1| + |A_2| + \ldots + |A_m|$.

- **Product rule:** Let $A_1, A_2, \ldots, A_m$ be finite sets. Then the number of ways to choose one element from each set in the order $A_1, A_2, \ldots, A_m$ is $|A_1 \times A_2 \times \ldots \times A_m| = |A_1| \cdot |A_2| \cdot \ldots \cdot |A_m|$.

Basic Counting Principles

- **More complex example:**
- Each user on a computer system has a password, which is six to eight characters long, where each character is an uppercase letter or a digit. Each password must contain at least one digit. How many possible passwords are there?
- **Solution:**
- Let P be the total number of possible passwords, and let $P6$, $P7$, and $P8$ denote the number of possible passwords of length $6, 7$, and 8, respectively. By the sum rule, $P = P6 + P7 + P8$. We will now find $P6, P7$, and $P8$. Finding $P6$ directly is difficult. To find $P6$ it is easier to find the number of strings of uppercase letters and digits that are six characters long, including those with no digits, and subtract from this the number of strings with no digits.

Basic Counting Principles

- By the product rule, the number of strings of six characters is 36^6, and the number of strings with no digits is 26^6. Hence,
- $P6 = 36^6 - 26^6 = 2176782336 - 308915776 = 1867866560$
- $P7 = 36^7 - 26^7 = 78364164096 - 8031810176$

$= 70332353920$

- $P8 = 36^8 - 26^8 = 2821109907456 - 208827064576$

$= 2612282842880$

- So
- $P = P6 + P7 + P8 = 2684483063360$

Basic Counting Principles

- **The subtraction rule:**

- If a task can be done in either n_1 ways or n_2 ways, then the number of ways to do the task is $n_1 + n_2$ minus the number of ways to do the task that are common to the two different ways.

- $|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2|$

Inclusion-Exclusion (The Subtraction Rule)

- **Example1:** How many bit strings of length 8 either start with a 1 or end with 00?
- **Task 1:** Construct a string of length 8 that starts with a 1.
- There is one way to pick the first bit (1),
- two ways to pick the second bit (0 or 1),
- two ways to pick the third bit (0 or 1),
- .
- .
- .
- two ways to pick the eighth bit (0 or 1).
- **Product rule:** Task 1 can be done in $1 \cdot 2^7 = 128$ ways.

Inclusion-Exclusion (The Subtraction Rule)

- **Task 2:** Construct a string of length 8 that ends with 00.
- There are two ways to pick the first bit (0 or 1),
- two ways to pick the second bit (0 or 1),
- .
- .
- .
- two ways to pick the sixth bit (0 or 1),
- one way to pick the seventh bit (0), and
- one way to pick the eighth bit (0).
- **Product rule:** Task 2 can be done in $2^6 = 64$ ways.

Inclusion-Exclusion (The Subtraction Rule)

- Since there are 128 ways to do Task 1 and 64 ways to do Task 2, does this mean that there are 192 bit strings either starting with 1 or ending with 00 ?
- No, because here Task 1 and Task 2 can be done **at the same time**.
- When we carry out Task 1 and create strings starting with 1, some of these strings end with 00.
- Therefore, we sometimes do Tasks 1 and 2 at the same time, so **the sum rule does not apply**.

Inclusion-Exclusion (The Subtraction Rule)

- If we want to use the sum rule in such a case, we have to subtract the cases when Tasks 1 and 2 are done at the same time.

- How many cases are there, that is, how many strings start with 1 **and** end with 00?

- There is one way to pick the first bit (1),
- two ways for the second, …, sixth bit (0 or 1),
- one way for the seventh, eighth bit (0).

- **Product rule:** In $2^5 = 32$ cases, Tasks 1 and 2 are carried out at the same time.

Inclusion-Exclusion (The Subtraction Rule)

- Since there are 128 ways to complete Task 1 and 64 ways to complete Task 2, and in 32 of these cases Tasks 1 and 2 are completed at the same time, there are
- 128 + 64 – 32 = 160 ways to do either task.

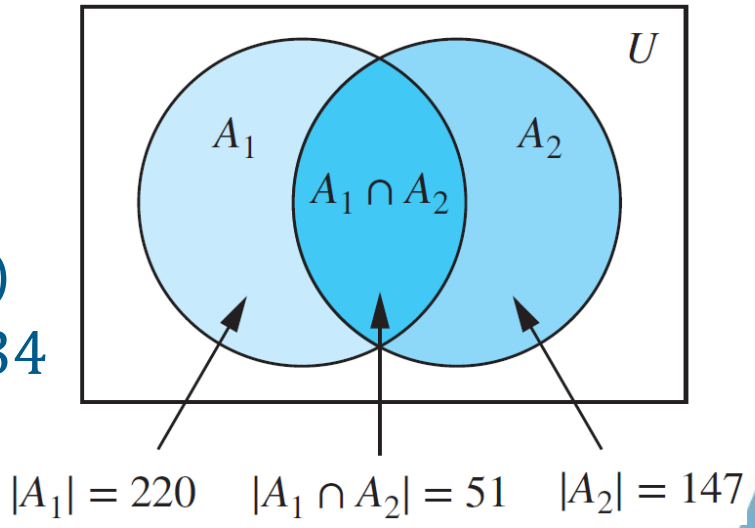
- In set theory, this corresponds to sets A_1 and A_2 that are **not** disjoint. Then we have:
- $|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2|$

- This is called the **principle of inclusion-exclusion.**

Inclusion-Exclusion (The Subtraction Rule)

- **Example2:** A computer company receives 350 applications from college graduates for a job planning a line of new web servers. Suppose that 220 of these applicants majored in computer science, 147 majored in business, and 51 majored both in computer science and in business. How many of these applicants majored neither in computer science nor in business?

- **Solution:**
- $|\overline{A_1 \cup A_2}| = |U| - |A_1 \cup A_2| =$
  $|U| - (|A_1| + |A_2| - |A_1 \cap A_2|)$
  $= 350 - (220 + 147 - 51) = 34$



Basic Counting Principles

- **The division rule:**

- There are n/d ways to do a task if it can be done using a procedure that can be carried out in n ways, and for every way w, exactly d of the n ways correspond to way w.

- The division rule comes in handy when it appears that a task can be done in n different ways, but it turns out that for each way of doing the task, there are d equivalent ways of doing it. Under these circumstances, we can conclude that there are n/d inequivalent ways of doing the task.

Basic Counting Principles

- **The division rule:**

- **Example1:** Suppose that an automated system has been developed that counts the legs of cows in a pasture. Suppose that this system has determined that in a farmer's pasture there are exactly 572 legs. How many cows are there is this pasture, assuming that each cow has four legs and that there are no other animals present? 572/4=143

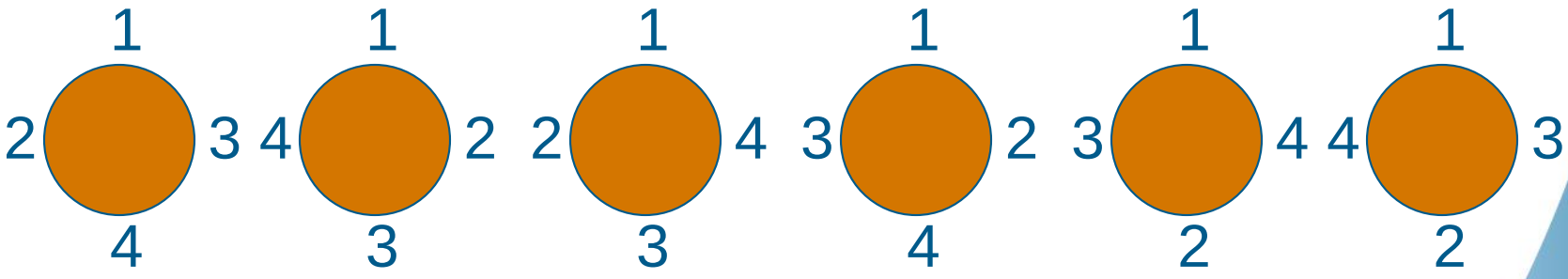
- **Example2:** How many different ways are there to seat four people around a circular table, where two seatings are considered the same when each person has the same left neighbor and the same right neighbor?

- **Solution:** We arbitrarily select a seat at the table and label it seat 1. We number the rest of the seats in numerical order, proceeding clockwise around the table.

Basic Counting Principles

- **The division rule:**

- Note that are four ways to select the person for seat 1, three ways to select the person for seat 2, two ways to select the person for seat 3, and one way to select the person for seat 4. Thus, there are 4! = 24 ways to order the given four people for these seats. However, each of the four choices for seat 1 leads to the same arrangement. Because there are four ways to choose the person for seat 1, by the division rule there are 24⁄4 = 6 different seating arrangements of four people around the circular table.

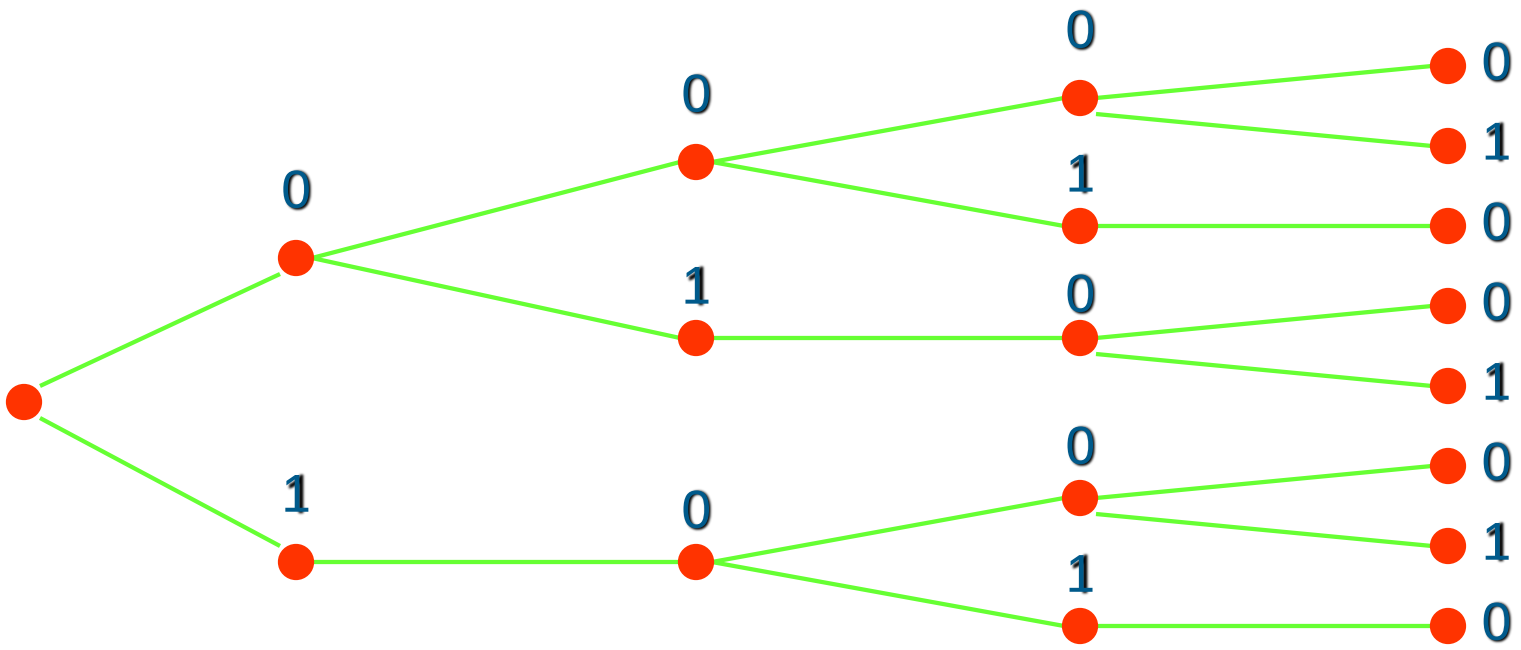


Tree Diagrams

- Counting problems can be solved using **tree diagrams**.
- A tree consists of a root, a number of branches leaving the root, and possible additional branches leaving the endpoints of other branches.
- To use trees in counting, we use a branch to represent each possible choice.
- We represent the possible outcomes by the leaves, which are the endpoints of branches not having other branches starting at them.

Tree Diagrams

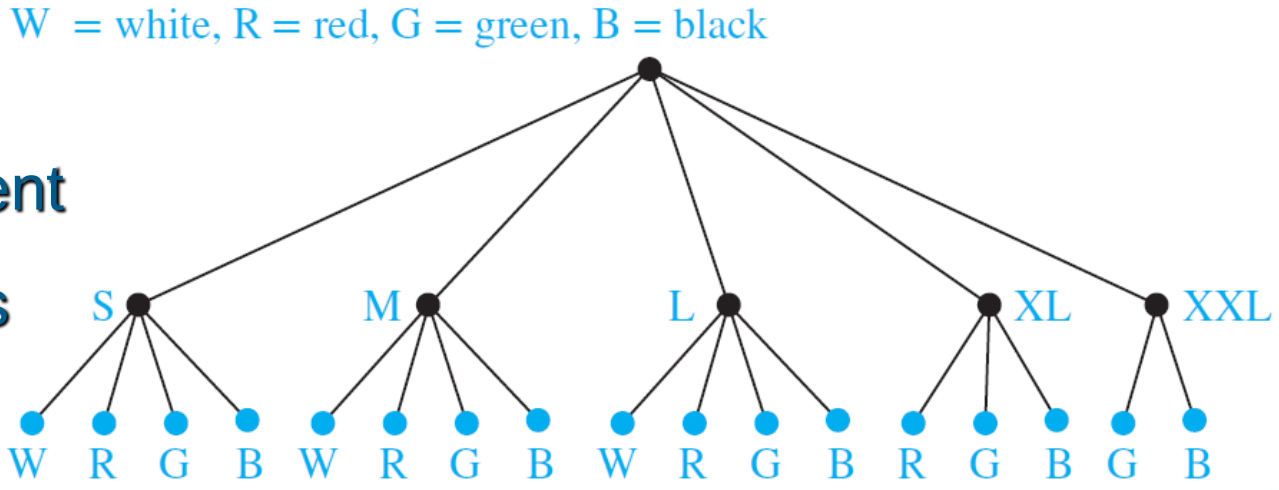
- **Example1:** How many bit strings of length four do not have two consecutive 1s?



There are 8 strings.

Tree Diagrams

- **Example2:** Suppose that "I Love Boston" T-shirts come in five different sizes: S, M, L, XL, and XXL. Further suppose that each size comes in four colors, white, red, green, and black, except for XL, which comes only in red, green, and black, and XXL, which comes only in green and black. How many different shirts does a souvenir shop have to stock to have at least one of each available size and color of the T-shirt?



17 different T-shirts

The Pigeonhole Principle

- **The pigeonhole principle:** If (k + 1) or more objects are placed into k boxes, then there is at least one box containing two or more of the objects.

- **Example 1:** If there are 11 players in a soccer team that wins 12-0, there must be at least one player in the team who scored at least twice (assuming there are no own goals!).

- **Example 2:** If you have 6 classes from Monday to Friday, there must be at least one day on which you have at least two classes.

The Pigeonhole Principle

- **Example 3:** Assume you have a drawer containing a random distribution of a dozen brown socks and a dozen black socks. It is dark, so how many socks do you have to pick to be sure that among them there is a matching pair?

- There are two types of socks, so if you pick at least 3 socks, there must be either at least two brown socks or at least two black socks.

- Generalized pigeonhole principle: $\lceil 3/2 \rceil = 2$.

- In general, if N objects are placed into k boxes, then there is at least one box containing at least $\lceil N/K \rceil$ objects (can be easily shown by contradiction).

The Pigeonhole Principle

▸**Example 4:** During a month with 30 days, a baseball team plays at least one game a day, but no more than 45 games. Show that there must be a period of some number of consecutive days during which the team must play exactly 14 games in that period.

▸**Solution:** Let a_j be the number of games played on or before the jth day of the month. Then $a_1, a_2, \ldots, a_{30}$ is an increasing sequence of distinct positive integers, with $1 \leq a_j \leq 45$.

The Pigeonhole Principle

- Moreover, $a_1 + 14, a_2 + 14, \ldots, a_{30} + 14$ is also an increasing sequence of distinct positive integers, with $15 \leq a_j + 14 \leq 59$. The 60 positive integers $a_1, a_2, \ldots, a_{30}, a_1 + 14, a_2 + 14, \ldots, a_{30} + 14$ are all less than or equal to 59.
- Hence, by the pigeonhole principle two of these integers are equal. Because the integers $a_j, j = 1, 2, \ldots, 30$ are all distinct and the integers $a_j + 14, j = 1, 2, \ldots, 30$ are all distinct, there must be indices i and j with $a_i = a_j + 14$. This means that exactly 14 games were played from day $j + 1$ to day i.