CS 5002: Discrete Structures **Fall 2018**

Lecture 8: November 1, 2018 [1]

Instructors: Tamara Bonaci, Adrienne Slaughter

Intro to Counting Techniques

Readings for this week:
Rosen, Chapter 6.1, 6.2, 6.3, 6.4, 6.5

8.1 Overview

1. Review: linear data structures
2. Introduction to counting techniques
3. The Pigeonhole Principle
4. Permutations and Combinations
5. Binomial Coefficients and Identities
6. Generalized Permutations and Combinations

8.2 Introduction:

In today's lecture, we turn our focus to **combinatorics**, a brach of discrete mathematics that studies arrangement of objects. More specifically, we will focus on **enumeration**, the part of combinatorics that explores the ways we can count objects with certain properties.

Many of us have heard a saying: *It's as easy as 1-2-3!*. So, even a proverb states that counting is easy. Before our reading assignment, and our today's lecture, you may have thought something similar yourself: "Counting is straightforward and easy... What could we possibly be talking about for three hours today, that is related to counting?"

The proveb is right - generally speaking, counting is easy, **as long as you know what to count**. Let's consider an example, to expand this idea further. In our everyday life, all of us use passwords. Often, when we create a new account, we need to create a new password too. When we do that, we often get an indicator of the password's strength (typically, `weak, moderate or strong`). How do we determine the strength of a password? A meaningful answer, obviously, involves counting. But what should we count:

- The number of symbols in your password?
- The total number of passwords that could possible be generated, using the same symbols as you used in your password?
- The number of people that are already registerd with the same system, and have the same password as you?
- The number of valid English words in your password?

- The number of special characters in your password?
- The number of digits followed by a special character in your password?
- The number of combinations `letter-number-letter` in your password?

There are many different ways in which we could count something related to passwords. Concievably, we could claim that any such metric is a proxy for the strength of a password[2] So, as we claimed earlier, counting something related to passwords is easy. The hard part is determining what to count, and how is that relevant to the problem we are trying to solve.

Today, we will start our conversation about couting by getting familiar with **basic rules of counting**, and we will show how these simple rules can be used to model and solve a variety of problems we encounter every day. We will then explore the **pigeonhole principle**. In its essence, the pigoenhole principle states, that when some objects are placed in boxes, and there are more objects than boxes, then we know that there will exist at least one box that will contain at least two elements.

We will then explore how to arrange some (ordered or unordered) sequences of objects, when those objects are drawn from the set with and without repetitions. We will introduce the notions of **combinations** and **permutations**, and show how those can be used when modeling and solving many real life problems.

But, first let's quickly review what did we talk about in our last lecture (before the midterm).

8.3 Review - Linear Data Structures

In the last lecture, we explored the notion of a **data structure**, generally defined as some collection of items, where we typically want to perform some basic operations on such a collection efficiently. We motivated the need for different data structures by focusing on two criteria, **memory** and **efficiency**.

We then explored two major groups of data structures, divided based upon on the arrival of elements into the data structure:

- The **LIFO** (Last-In, First-Out) data structure, and
- The **FIFO** (First-In, First-OUt) data structure

We analyzed a **stack** as an example of a LIFO data structure, and defined the basic operations on a stack to be:

- `pop()` : Get the top item from the stack, removing that item
- `push()` : Put a new item on the stack
- `peek()` : Look to see what the top element is
- `create()` : Create the stack
- `destroy()` : Destroy the stack

We showed that a **queue** is an example of a FIFO data structure, with the following basic operations:

- `enqueue()` : Put an element at the back of the queue
- `dequeue()` : Get the element that is at the front of the queue
- `front()` : Look at the element at the front
- `back()` : Look at the element at the back/last inserted
- `create()` : Create the queue
- `destroy()` : Destroy the queue

We also introduced **deque** (**double ended queue**) as a data structure that operates as a hybrid of a stack

[2]Some of these metrics are, or have until recently been used to measure passwords' strengths.

and a queue, and allows the data to be add to the collection at either the front or the back. Similarly, a deque allows the data to be removed at either the front or the back, and its typically operations include:

- `insertFront()` : Put an element at the front of the deque
- `insertBack()` : Put an element at the back of the deque
- `removeFront()` : Get the element at the front of the deque
- `front()` : Look at the element at the front
- `back()` : Look at the element at the back/last inserted
- `create()` : Create the queue
- `destroy()` : Destroy the queue

We next explored memory issues related to instatiation and usage of different data structures, and we distinguished between **contiguously-allocated** and **linked data** structures, defined as follows:

- **Contiguously-Allocated:** composed of a single chunk of memory. Examples: arrays, matrices, heaps, hash tables.
- **Linked data structure:** composed of distinct chunks of memory connected by pointers. Examples: Lists, trees, graphs adjacency lists.

We introduced an **array** as a simple contiguous chunk of memory set, that can typically contain only one type of data (e.g., an int, or a float, or a char).

Table below represents a summary of differences between various data structures we encountered thus far. We analyze our data structures with respect to the memory used for storing data, and with respect to the time complexity of some typically operations on those data structures .

	Array	Linked List	Stack	Queue	Tree
Type	Linear	Linear	Linear	Linear	Tree/graph
Insert action	Puts	Puts element on top	Puts element on top	Puts element in back	Puts element where specified
Insert Operatio	a[n] = i;	insert()	push()	enqueue()	...
Remove Action	a		Returns element at top	Returns element at front	...
Remove operatio			pop()	dequeue()	...
Notes			LIFO	FIFO	

8.4 Introduction to Counting Techniques

> *The whole of science is nothing more than a refinement of everday thingking.* *Albert Eninstein (1897-1955)*

Counting problems arise throughout our everyday life, as well as in mathematics and in computer science. For example, we may want to count:

- The number of all possible 7-symbols license plates, `xxxxxxx`, where symbols include letters `A-Z` and digits `0-9`, but such that the license plate always starts with a letter.
- The number of all possible 7-symbols phone numbers, `xxx-xxxx` that we can generate for some given area code, for example 206 or 425.
- The number of all possible ways an ALIGN student can choose their courses, such that they satisfy all of the degree requirements.
- The degree of separation between any two persons, A and B, where the degree of separation is defined

as the number of people person A would have to engage with to reach person B. For example, if person A reaches to their friend C, who then reaches to their friend D, who then reaches to their friend E, who has a direct contact to person B, the degree of separation between A and B would be 4 ($A \to B \to C \to D \to E \to B$).

8.4.1 Basic Counting Principles

Often times, we can solve problems like these by relying on two principles: **the product rule** and **the sum rules**. Let's explore those rules.

Definition 8.1 (*The Product Rule*) *Suppose that some procedure can be broken down into a sequence of two tasks. If there are n_1 to execute the first task, and n_2 ways to execute the second task, then there are $n_1 \cdot n_2$ ways to perform the whole procedure.*

The stated **product rule** can further be generalized to more than two tasks, as follows.

Definition 8.2 (Generalized Product Rule) *Suppose that some procedure can be broken down into a sequence of k tasks. If there are n_1 ways to execute the first task, n_2 ways to execute the second task, all the way to n_k ways to execute the k-th task, then there are $n_1 \cdot n_2 \cdots n_k$ ways to perform the whole procedure.*

Let's see some examples.

Example 1: Number of PIN Numbers

A typical PIN (Personal Identification Number) is a sequence of any six symbols chosen from the the set of digits $\{0 - 9\}$. How many different PINs are there if:

- We allow the symbols to repeat?
- We do not allow the symbols to repeat?

We can solve this problem by thinking of forming a PIN as a six-step operation to fill each of the six symbols in a sequence:

$$\underbrace{\_\_}_{1}\ \underbrace{\_\_}_{2}\ \underbrace{\_\_}_{3}\ \underbrace{\_\_}_{4}\ \underbrace{\_\_}_{5}\ \underbrace{\_\_}_{6}$$

- **Step 1:** Choose the first digit.
- **Step 2:** Choose the second digit.
- **Step 3:** Choose the third digit.
- **Step 4:** Choose the fourth digit.
- **Step 5:** Choose the fifth digit.
- **Step 6:** Choose the sixth digit.

The question now boils down to counting how many different ways there are for us execute each of these steps.

If **we allow repetitions**, then every steps can be executed in the same number of ways - there are 10 possible digits, so there are 10 possible ways in which we can choose the first digit, and the second digit, and the third, and so on. Using the multiplication rule, we see that the total number of 6-digits PINs, when we are allowed to repeat symbols, is equal to $\prod_{i=1}^{10} i = 10 \cdot 10 \cdots 10 = 10^6$.

If **we do not allow symbols to repeat in the PIN**, then:

- The first symbol can be chosen in 10 possible way.
- The second symbol can now be drawn from the 9 remaining options. $\rightarrow$ 9 possible ways
- The second symbol can now be drawn in 8 possible ways.
- $\vdots$
- The sixth symbol can be chosen in 4 ways only.

Using the multiplication rule again, we see that the total number of 6-digits PINs, when we are do not allow symbol repetitions, is equal to $10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 = 604800$ [3].

Definition 8.3 (*The Sum Rule*) *If a task can be done either in one of n_1 ways or in one of n_2 ways, where none of the set of n_1 ways is the same as any of the set of n_2 ways , then there are $n_1 + n_2$ ways to do the task.*

Similar to the product rule, the sum rule can also be generalized, as follows.

Definition 8.4 (Generalized Sum Rule) *Consider some finite set A that can be divided in the union of k distinct and mutually disjoint subsets A_1, $A_2 \ldots, A_k$. Then the cardinality of set A can be found as:*

$$|A| = |A_{1]}| + |A_2| + \cdots + |A_k|$$

Let's see an example.

Example 2: Number of Passwords with Five or Less Symbols

Let's stay in the spirit of exploring how secure the systems around us are. Let's consider a system where a user gets authenticated using a password. The password can have from one to five symbols, chosen from the set of letters (alphabets) A–Z. How many different passwords can we generate for our system?

To solve this problem, let observe that we can partition the set of all possible passwords into five subsets:

- Subset of all passwords of length 1
- Subset of all passwords of length 2
- $\vdots$
- Subset of all passwords of length 5

Using the sum rule, the total number of passwords is equal to the sum of the cardinalities of these five subsets. So, let's find these cardinalities:

- Cardinalitiy of the subset of all passwords of length 1 - only 26 possiblities
- Cardinality of the subset of all passwords of length 2 - we use the product rule here, and since repetitions are allowed, the cardinality is equal to 26^2
- Cardinality of the subset of all passwords of length 3 - using the product rule again, the cardinality is equal to 26^3
- $\vdots$
- Cardinality of the subset of all passwords of length 5 - using the product rule again, the cardinality is equal to 26^5

[3] Moral of the story: 6-digit PIN is not very secure, but it is a bit more secure if we allow repetitions.

So, the total number of passwords consisting of up to five letters A–Z, can be computed as[4]:

$$\text{Number of passwords} = 26 + 26^2 + 26^3 + 26^4 + 25^5 = 26(1 + 26 + 26^2 + 26^3 + 26^4) = 12356630$$

8.4.2 The Subtraction Rule (Inclusion-Exclusion for Two Sets)

Suppose we have some task that can be done in one of two ways, but some of the ways to do it are common to both ways. In this situation, we cannot use the sume rule to count the total number of ways to do the task, because we would overcount, and include the common ways twice. To correctly count the total number of ways to do the task, we need to subtract the number of ways that were counted twice. This way of thinking leads to the subtraction rule, defined below.

Definition 8.5 (*The Subtraction Rule*) *If a task can be done in either n_1 ways or in n_2 ways, then the numbers of ways to do the taks is equal to $n_1 + n_2$ minus the number of ways to do the task that are common to the two different ways.*

The subtraction rule is also known as the **inclusion-excusion principle** that we saw earlier, when we talked about sets. As a reminder, here is a more general view of the subtaction rule.

Definition 8.6 (The Inclusion-Exclusion Principle for Two and Three Sets) *Let A, B and C be some finite sets. Then:*

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

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

Let's see some examples.

Example 3: Number of Python Identifiers of Eight or Fewer Characters

In Python, an identifier starts with one of 53 symbols: either one of the 52 upper- and lower- alphabets A–Z, or an underscore (_). The identifier can have only one character, or the initial character may be followed by any number of characters chose from a set of 63 symbols: the 53 symbols allowed as the initial character plus ten digits. Certain words, however, such as `and, it, print` are set aside as **keywords**, and cannot be used as identifiers. In one Python implementation, there are 31 keywords, none of which has more than eight characters.

How many Python identifiers are there that are less than or equal to eigth characters in length?

Similar to our approach in Example 2, the set of all Python identifiers with eight of less characters can be partitioned into eigth subset - one for each allowed length. The keywords can have variety of lengths, but we know that all such keywords have length less than 8.

So, let's find the cardinalities of our disjoint subsets:

- Cardinality of the subset of identifiers of length 1: There are 53 possible ways to choose the first character, so the cardinality is 53
- Cardinality of the subset of identifiers of length 2: $53 \cdot 63$
- Cardinality of the subset of identifiers of length 3: $53 \cdot 63^2$

[4]This number is misleadingly high, because it leave an impression that alphabet passwords of up to five symbols are secure, but that is not the case, because not all possible passwords are equally likely to be chosen as a password.

- ⋮
- Cardinality of the subset of identifiers of length 3: $53 \cdot 63^7$

The total number of identifiers is equal to the sum of cardinalities of the possible subsets minus the cardinality of the set of keywords. So, we can write:

$$\begin{aligned} \text{Number of identifiers} &= 53 + 53 \cdot 63^2 + 53 \cdot 63^3 + 53 \cdot 63^4 + 53 \cdot 63^5 + 53 \cdot 63^6 + 53 \cdot 63^7 - 31 \\ &= 212133167002849. \end{aligned}$$

8.4.3 The Division Rule

Definition 8.7 (*The Division Rule*) *There are $\frac{n}{d}$ 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, there exist exactly d of the n ways that correspond to that way w.*

Let's express the division rule in terms of sets and functions:

- **The division rule expressed in terms of sets:** If the finite set A is the union of n pairwise disjoint subsets, each with d elements, then $n = \frac{|A|}{d}$.
- **The division rule expressed in terms of functions:** If f is a funcition from A to B where A and B are finite sets, such that for every $y \in B$ there are exactly d values of $x \in A$ such that $f(x) = y$, then $|B| = \frac{|A|}{d}$. We say that function f is a $d - to - 1$ **function.**

Example 4: How many different ways there are to seat four people around a circular table, where two seatings are considered the same when each person has the same left and right neighbors?

To solve this problem, we arbitrarily select some seat and label it 1. We label all other seats, proceeding clockwise around the table. Note that there are four ways to select the person for seat 1, three ways to select a person for seat 2, two ways to selest a person for seat 3, and only one way to select a person for set 4. So, there are $4! = 24$ ways to seat four people around the table. However, each of the four choices for seat 1 leads to the same arrangement, because we distinguish two arrangements only when one of the people has a different immediate left or immediate right neighbor. Because there are four ways to choose the person for seat 1, by division rule, the total number of distinct arrangements is $\frac{24}{6} = 6$.

8.5 The Pigeonhole Principle

A pigeonhole principle is a powerful and important tool, used in many counting problems. This principle can be expressed in many different ways, and in this section, we analyze some of those definitions.

Definition 8.8 (*The Pigeonhole Principle*) *If k is a positive integer and $k+1$ or more objects are placed into k boxes, then there is at least one box containing two or more of the objects.*

The pigeonhole principle is sometimes also called the **the Dirichlet's box principle**, because it was first formally states by J. P. G. L. Dirichlet.

Let's consider what happens when our objects are **pigoens**, and our boxes are **pigeonholes**, as depicted in Figure 8.1. Let's assume we have n pigeons, and they fly into m pigeonholes, where $n > m$. Then, at least one of the holes has to contain two or more pigeons.

Figure 8.1: Pigeons depicting the pigeonhole principle.

Let's consider some example applications of the pigeonhole principle

Example 5: In a group of six people, must there be at least two persons who were born in the same month?

Example 6: Among any group of 367 people, must there be at least two persons with the same birthday?

Example 7: How many students must there be in a class to guarantee that at least two students will have the same score, if the exam is graded on a scale of 0 to 50 points?

Example 8: Let $A = \{1, 2, 3, 4, 5, 6, 7, 8\}$. If five integers are selected from A, are we guaranteed that at

least one pair of integers will have the sum of 9?

The pigenhole principle has a very interesting, and important corollary, stated below.

Corollary 8.9 *A function f from a set with $k+1$ or more elements to a set with k elements is not a one-to-one function.*

The pigeonhole principle can also be generalized as follows.

Theorem 8.10 *If N objects are placed into k boxes, then there is at least one box that contains at least $\lceil \frac{N}{k} \rceil$.*

Example 9: Among 100 people, what is the minimum number of people that were born in the same month?

Example 10: What is the minimum required number of students in some class to guarantee that at least ten will get the same grade, if there are four possible grades, A, B, C, D.

Example 11: What is the least number of area codes needed to guarantee that 25 million people will be assigned distinct 10-digit telephone numbers?

8.6 Permutations and Combinations

8.6.1 Permutations

Many counting problems can be solved by finding the numbers of ways to arrange a specified number of distinct elements of a set of a particular size, where the order of these elements matters. For example, we may ask in how many different ways can we select 13 topics from a selection of 35 topics to cover in a semester-long course.

Such problems lead to the concept of a permutation, defined below.

Definition 8.11 (*Permutation*) *A* **permuation** *of a set of distinct objects is an ordered arrangement of these objects. An* **r-permutation** *is an ordered arrangement of r elements of a set.*

The number of r-permutations of a set with n elements is typically denoted as $P(n, r)$.

Theorem 8.12 *If n is a positive integer and r is an integer such that $1 \leq r \leq n$, then there exists*

$$P(n, r) = n(n-1)(n-2)\dots(n-r+1) \tag{8.1}$$

r-permutations of a set with n distinct elements.

Proof: We use the product rule to prove the given permutation formula (8.1). The first element of the permutation can be chosen in n different ways, because there are n elements in the set. There are $n-1$ elements in the set after we fix the first element. Therefore, the second element can be chosen in $n-1$ different ways. Similarly, the third elements can be chosen in $n-2$ different ways, and so fort, until we reach the last element of our permutation, which can be chosen in $n-(r-1)$ different ways. Using now the product rule, we get the permutation formula (8.1). ■

Note 1: Note that $P(n, 0) = 1$, whenever n is a non-negative number, because there exist exactly one empty list, and therefore only one way to order zero elements.

This note, combined with theorem 8.12 leads to the following corollary.

Corollary 8.13 *If n and r are integers such that $0 \leq r \leq n$, then:*

$$P(n, r) = n(n-1)(n-2)\dots(n-r+1) = \frac{n!}{(n-r)!}$$

Let's see some examples:

Example 12: How many 5-permutations are there of a set of eight objects?

Example 13: How many 5-permutations are there of the set of five elements:

Example 14: How many ways are there to select a first-prize winner, a second-prize winner and a third-prize winner from 100 different competitors?

8.6.2 Combinations

There exist many problems that can be solved by finding the number of ways to select a particular number of elements from the set of a particular size, where the order of elements selected does not matter. For example, how many different 3-persons teams can be formed from a class of 25 students?

Solving these problems leads to the concept of combinations, defined below.

Definition 8.14 (*Combination*) *An r-combination of elements of a set is an unordered selection of r elements from the set. Therefore, the r-combination is simply a subset of the set with r elements.*

*The number of r-combinations of a set with n distinct elements is denoted as $C(n, r)$ or $\binom{n}{r}$, and it is typically referred to as **binomial coefficient**.*

Theorem 8.15 *The number of r-combinations of a set with n elements, where n is a non-negative integer and r is an integer such that $0 \leq r \leq n$ equals:*

$$C(n, r) = \frac{n!}{r!(n-r)!} \tag{8.2}$$

Proof: The $P(n, r)$ r-permutation of some set can be obtained by forming $C(n, r)$ r-combinations of the set, and then ordering the elements in every r-combination, which can be done in $P(r, r)$ ways. Therefore, using the product rule, we have:

$$P(n, r) = C(n, r) \cdot P(r, r)$$

This implies that:

$$C(n, r) = \frac{P(n, r)}{P(r, r)} = \frac{\frac{n!}{(n-r)!}}{\frac{r!}{(r-r)!}} = \frac{n!}{r!(n-r)!} \tag{8.3}$$

■

Corollary 8.16 *Let n and r be non-negative integers such that $r \leq n$. Then $C(n, r) = C(n, n - r)$.*

Let's see some examples:

Example 15: How many 2-combinations can be made from the set $\{1, 2, 3, 4, 5\}$?

Example 16: How many distinct ways there are to choose five members from a group of 12 people to work as a team on a special project?

Example 17: Let's now assume that two members of the group insist on working as a pair - any team must contain either both or neither. How many five-person teams can be formed?

8.7 Binomial Coefficients and Identities

8.7.1 The Binomial Theorem

When talking about combinations, we said that number $C(n, r)$ is typically referred to as **binomial coefficient**, and there is a reason for that.

Given some sum of two terms, typically referred to as a ***binomial expression***, we can use the **binomial theorem** to **find the coefficients of the expansion of powers of binomial expressions**.

Let's see one example before the state the theorem:

Example 18: Let's consider sum:

$$(x + y)^3$$

We could, for example, rewrite the given expression as:

$$\begin{aligned}(x+y)^2 = (x^2+2xy+y^2)(x+y) &= x^3+2x^2y+xy^2+x^2y+2xy^2+y^3 \\ &= x^3+3x^2y+3xy^2+y^3 \end{aligned} \tag{8.4}$$

Let's now see how would we use combinatorial reasoning to derive the same expansion. To do so, let's expanded $(x+y)^3$ as follows:

$$(x+y)^3 = (x+y)(x+y)(x+y)$$

We now clearly see that when all products of a term in the first sum, a term in the second sum, and a term in the third sum are added, terms of the form x^3, y^3, x^2y, xy^2 arise. The question is: **what are the coefficients (weights) associated with these terms?**

To answer this question, we observe that to obtain a term of the form x^3, x must be chosen in each of the sums, and there is only one way to chose x in every sum. So, the x^3 term in the product has a coefficient 1. We now proceed by observing that to obtain a term of the form x^2y, term x must be chosen from two of the three sums. So, the number of such terms is the number of 2-combinations of three objects, namely $\binom{3}{2}$. Similar argument can now be applied to x - the coefficient of xy^2 is the number of ways to pick term x from one of the three sums, and this can be done in $\binom{3}{1}$ ways. Lastly, the only way to obtain y^3 is to choose y for each of the three sums in the product.

So, with this analysis in mind, we get the following expansion of the given binomial expression:

$$(x+y)^3 = x^3+3x^2y+3xy^2+y^3$$

Let's now generalize this analysis, and state the **binomial theorem** as follows.

Theorem 8.17 (*The Binomial Theorem*) *Let x and y be variables, and let n be a non-negative integer. Then:*

$$(x+y)^n = \sum_{j=0}^{n} \binom{n}{j} x^{n-j}y^j = \binom{n}{0}x^n + \binom{n}{1}x^{n-1}y + \cdots + \binom{n}{n-1}xy^{n-1} + \binom{n}{n}y^n \tag{8.5}$$

Sketch of a proof: We prove the binomial theorem using the **combinatorial theorem**, defined as follows.

Definition 8.18 (*Combinatorial Proof*) *A* **combinatorial proof** *of some identity is a proof that uses counting arguments to prove that both sides of the identity count to the same object, but in different ways. Similarly, a* **combinatorial proof** *is a proof based on showing that there exists a bijection between the sets of objects counted by the two sides of the identity. These two types of proofes are often referred to as* ***double counting proofs*** *and* ***bijective proofs****, respectively.*

Let's see some examples.

Example 18: What is the expansions of $(x+y)^5$?

Example 19: What is the coefficient of $x^{12}y^{13}$ in the expression $(x+y)^{25}$?

Example 20: What is the coefficient of $x^{12}y^{13}$ in the expression $(2x-3y)^{25}$?

8.7.2 Pascal's Identity and Triangle

The binomial coefficients satisfy many identities, and one of the most important ones is the **Pascal's identity**, defined as follows.

Theorem 8.19 *Let n and k be positive integers with $n \geq k$. Then:*

$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$$

Note 2: Pascal's identity, combined with the initial conditions:

$$\binom{n}{0} = \binom{n}{n} = 1$$

for all integers n, can be used to recursively define binomial coefficients. This recursive definition is useful in the computation of binomial coefficients because we only need **addition operation**, and not **multiplication** operations.

Note 3: Pascal's identity is the basis for a geometric arrangement of the binomial coefficients in a triangle, known as ***the Pascal's triangle***, and depicted in Figure 8.2.

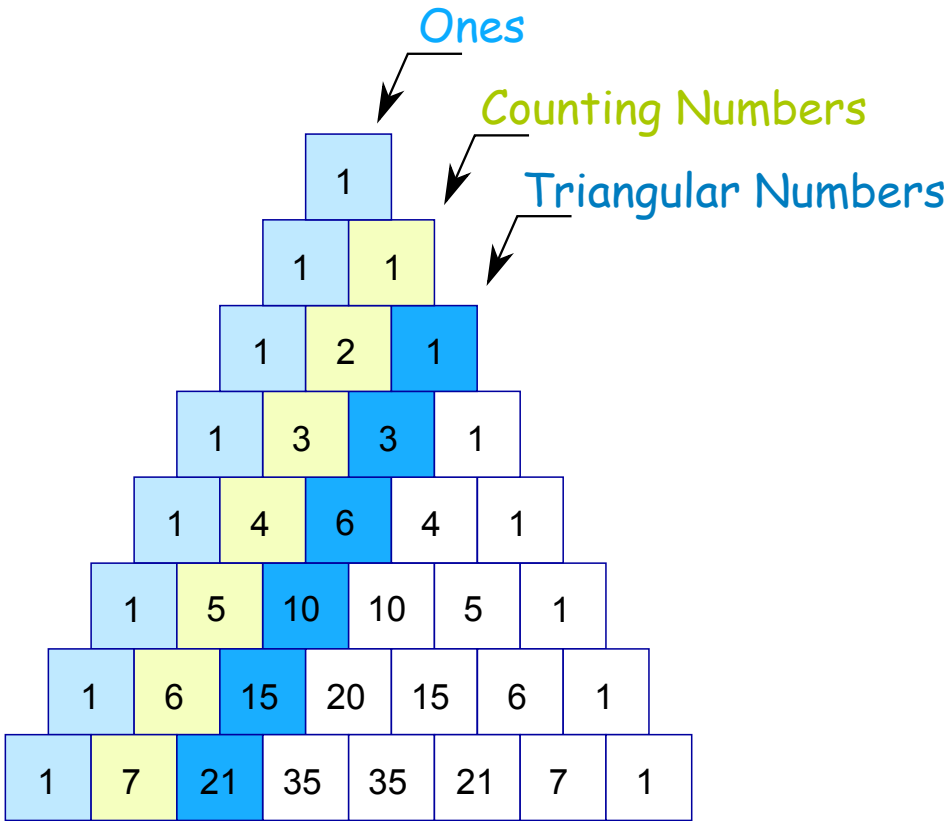

Figure 8.2: An example of a Pascal triangle

8.8 Generalized Permutations and Combinations

In many counting problems, elements may be used repeatedly. For example, an SSN number, a telephone number, a student ID may contain the same digit multiple times. This contrast with the counting problems we have discussed in the previous section, where we considered only permulations and combinations in which each element can be used at most once.

Let's see what happens with our counting problems when we allow repetitions.

8.8.1 Permutations with Repetition

Counting r-permutations of a set of n elements when repetitions are allowed can easily be done using the product rule. Let's see how.

Theorem 8.20 *The number of r permutations of a set of n objects with repetition allowd is equal to n^r.*

8.8.2 Combinations with Repetition

In this subsection we ask: **how many ways are there to choose r elements without regard to order from a set of n elements, if repetitions are allowed**? A good way to visualize this is to imagine the n elements as categories of objects from which multiple selestions may be made. For example, if categories are labeled 1, 2, 3 and 4, and we need to choose three elements, we can maybe choose two elements of type 3, and one elements of type 1. Or, in a different selection, we can choose all three elements of type 4.

Note that because order does not matter, selections [3, 1, 3] = [3, 3, 1] = [1, 3, 3].

This leads us to the following definition.

Definition 8.21 (*Multiset of size r*) *An r-combination with repetition allowed, or multiset of size r, chosen from a set X of n elements is an unordered selection of elements taken from X with repetition allowed.*

Theorem 8.22 *The number of r-combinations with repetition allowed (multiset of size r) that can be selected from a set of n elements is equal to:*

$$\binom{r+n-1}{r}$$

Theorem 8.23 *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.*

Let's see some examples.

Example 21: A mathematician is throwing a party, and she wants to set out 15 assorted cans of soft drinks for her quests. She shops at a store that sells five different types of soft drinks.

- How many different selections of cans of 15 soft drinks can she make?
- If root beer is one of the types of soft drinks, how many different selections include at least six cans of root beer?
- If the store has only five cans of root beer, but at least 15 cans of every other type of soft drink, how many different selections are possible?

Readings for NEXT week:

Rosen, Chapter 7.1, 7.2, 7.3, 7.4