Part IV: Counting

- L11: Basics of Counting
- L12: Permutations and Combinations
- L13: Binomial Coefficients
- L14: Principle of Inclusion and Exclusion

L11: Basics of Counting

- Outline
  - Product Rule
  - Sum Rule
  - Using Both Product and Sum Rules
  - Tree Diagrams
  - The Pigeonhole Principle

- Reading: Rosen 6.1, 6.2

Introduction

Examples of when counting is needed in computer science:

- Determining whether there are enough Internet protocol (IP) addresses to meet the demand
- Determining the time or space complexity of algorithms
- Computing the probabilities of events

Two basic counting principles:

- **Product rule** (or **sequential rule**)
- **Sum rule** (or **disjunctive rule**)

Product Rule

- Suppose a procedure can be broken down into $m > 1$ tasks, $T_1, T_2, \dots, T_m$, and carried out by performing the tasks in sequence.
- If each task T_i can be done in n_i ways regardless of how the previous tasks were done, then there are $n_1 n_2 \dots n_m$ ways to carry out the procedure.

Example

- **Example**
  Ten chairs are numbered from 1 to 10. How many ways are there to assign different chairs to Joseph and Mary?
- **Solution**
  There are 10 ways to assign a chair to Joseph and 9 ways to assign a chair to Mary. By the product rule, there are $10 \times 9 = 90$ ways.

Example

- **Example**

Let U and V be two finite sets $U = \{1, 2, 3\}, V = \{a, b\}$. How many different functions $f : U \to V$ can be defined?

- **Solution**: $2^3 = 8$ possible functions.

	f(1)	f(2)	f(3)
f_1	a	a	a
f_2	a	a	b
f_3	a	b	a
f_4	a	b	b
f_5	b	a	a
f_6	b	a	b
f_7	b	b	a
f_8	b	b	b

Example

- **Example**
  Let U and V be two finite sets consisting of $|U| = m > 0$ and $|V| = n > 0$ elements, respectively. How many different functions $f: U \to V$ can be defined?

- **Solution:**
  We can map each of the elements in the domain U to $|V| = n$ possible values in the codomain V. So there are n^m different functions.

Example

- **Example**
  Let U and V be two finite sets consisting of $|U| = m > 0$ and $|V| = n > 0$ elements, respectively. How many different one-to-one (or injective) functions $f: U \to V$ can be defined?
- **Solution:**
- If $m > n$, no injective function could be formed.
- If $m \leq n$, then the first element in U can be mapped to n possible elements in V, the second element in U can be mapped to $n - 1$ possible elements in V, and so on. The k^{th} element of U can be mapped to $n - k + 1$ possible elements in V. Therefore, by the product rule, there are $n(n-1)(n-2)\ldots(n-m+1)$ different one-to-one functions.

Example

- **Example**
  Let S be a finite set of cardinality $|S| = n > 0$. How many different subsets of S are there?

- **Solution:**
  Note that there is a one-to-one correspondence between subsets of S and bit strings of length $|S|$. Therefore, we can count the number of bit strings of length $|S|$ instead.

Solution (cont'd)

- Let the elements of S be indexed from 1 to n.
- Each element of S could be either included or not included in a subset.
- A subset of S is associated with the bit string with a "1" in the i-th position if the i-th element is in the subset, and 0 in this position otherwise.
  - Example: Let $|S| = 6$, then {2, 3, 6} corresponds to 011001.
- By the product rule, there are $2^{|S|}$ bit strings of length $|S|$. Therefore there are $2^{|S|}$ subsets.

Example

- **Example**

Consider the following algorithm for matrix multiplication:

```
for $i$ = 1 to $m$
        for $j$ = 1 to $n$
                $s$ = 0
                for $k$ = 1 to $p$
                        $s = s + A[i, k] \times B[k, j]$
                $C[i, j]$ = s
        end for
end for
```

How many multiplications are executed?

Solution:

The multiplication operation is in the "loop k", so it will be executed as many times as the total number of iterations the program will enter "loop k" :

For **a fixed j value**, the program loops through *"loop k"* *p* times, performing the multiplication operation ***p* times**.

Since there are ***n*** different *j* values, the multiplication operation is performed **n*p times ,** considering the *"loop j" and "loop k"*.

Taking into account *"loop i"*, **for each fixed value *i***, the program iterates through the big *"loop j" and "loop k"* once (i.e. performing n*p multiplications). Since there are ***m*** values for *i*, therefore in total the multiplication is performed **m*n*p times**.

Outline

- Product Rule
- **Sum Rule**
- Using Both Product and Sum Rules
- Tree Diagrams
- The Pigeonhole Principle

Sum Rule

Suppose a procedure can be carried out in one of $m > 1$ methods, $T_1, T_2, \dots, T_m$. If each method T_i can be done in n_i ways and no two methods share at least one way (i.e., the methods are disjoint), then there are totally $\boldsymbol{\sum_{i=1}^{m} n_i}$ different ways to carry out the procedure.

Note that the sum rule cannot be applied if the tasks (or sets) are not disjoint. In that case, we need to apply the **inclusion-exclusion principle** which will be studied later.

Example

- **Example**

There are 15 boys and 18 girls in a class. How many ways are there to choose a representative for the class?

- **Solution**
  - There are 15 ways to choose a boy representative and 18 ways to choose a girl representative.
  - No one is both a boy and a girl.
  - By the sum rule it follows that there are 15+18=33 possible ways to pick the representative.

Outline

- Introduction
- Product Rule
- Sum Rule
- **Using Both Product and Sum Rules**
- Tree Diagrams
- The Pigeonhole Principle

Using Both Product and Sum Rules

- **Example**

In a programming language, a variable name is a string of one or two alphanumeric characters which are not case-sensitive, i.e., 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 the language?

Using Both Product and Sum Rules

- **Solution:**

  There are two cases: one to account for variables of length 1 (formed by a single alphanumeric character), and the other to account for variables of length 2 (formed by two alphanumeric characters).

  (case 1) For variables formed by a single alphanumeric character, we have a total of **26** possible variables (it cannot be a numerical digit, since variable names need to start with a letter)

Solution (cont'd):

(case 2) For variables formed by two alphanumeric characters, we will first apply the product rule to get the total number of possible combinations of 2-character strings, and then subtract the number by 5 (i.e. number of reserved 2-character strings).

For the first character, we can choose from the pool of 26 English letters, and for the second character we can choose from the pool of (26 letters +10 digits)=36 alphanumeric characters. By applying the product rule we have the total number of different variables being $26 \times 36 = 936$. Then we need to subtract the number by 5 in order to exclude the 5 reserved strings. Therefore we have $936 - 5 = \mathbf{931}$.

In total the number of variable names is $\mathbf{26 + 931 = 957}$

Using Both Product and Sum Rules

- **Example**

In a computer system, each user has a password of 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 P_6, P_7, P_8 denote the number of possible passwords of length 6, 7, 8 respectively. Then $P = P_6 + P_7 + P_8$.

We require each password to contain at least one digit (numerical value 0-9), this could be hard, because a legitimate password of length 6 could contain 1 digit, 2 digits,… 6 digits. For the passwords of length 6 with 1 digit, that digit could be in any of the 6 positions. For the passwords of length 6 with 2 digits, the two digits can be in any 2 positions within the 6 possible positions.

Solution (cont'd)

So direct calculation for the number of possible combinations is tricky. A shortcut we could apply is to first calculate the total number of passwords of length 6 formed by using the 26 letters and 10 digits. Then we subtract this number by the number of passwords with 0 digit to get the desired sum.

P_6 = total number of passwords formed by the 36 alphanumeric characters – total number of passwords formed by the 26 English letters only

$$= 36^6 - 26^6$$

$$= 1867866560$$

▪

Solution (cont'd)

Similarly, P_7, P_8 can be calculated as

$P_7 = 36^7 - 26^7 = 70332353920$

$P_8 = 36^8 - 26^8 = 2612282842880$

As a result we have P:

$P = P_6 + P_7 + P_8$

$= 1867866560 + 70332353920 + 2612282842880$

$=$ **2 684 483 063 360** (how big is this number? If we consider it to be the number of seconds, then this translates to ~85000 years!)

Outline

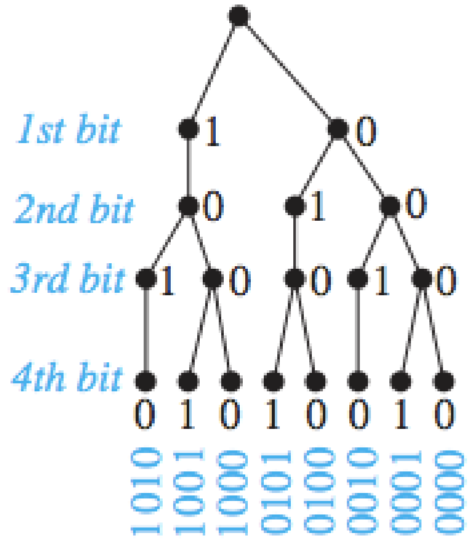
- Introduction
- Product Rule
- Sum Rule
- Using Both Product and Sum Rules
- **Tree Diagrams**
- The Pigeonhole Principle

Tree Diagrams

Some counting problems can be solved more easily using tree diagrams.

- **Example**

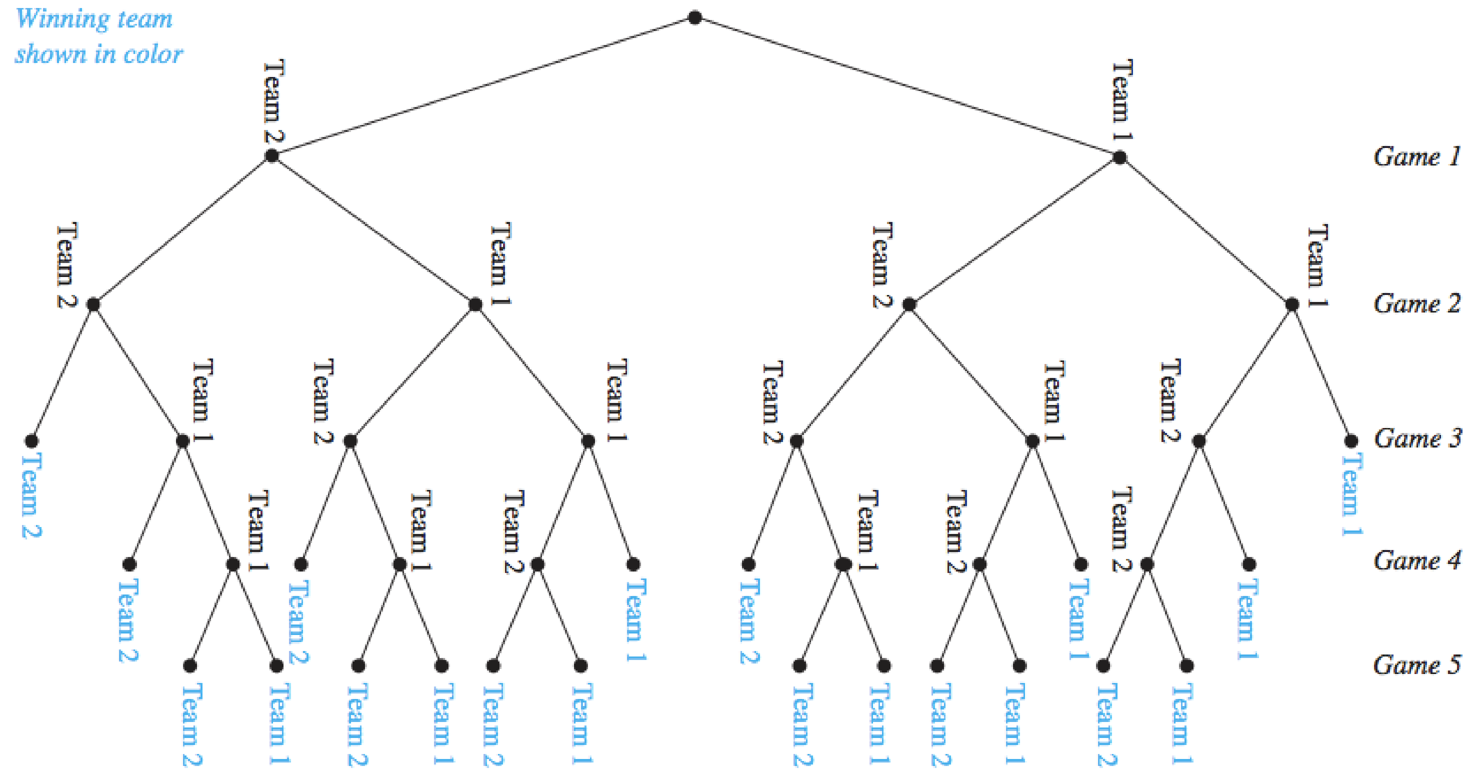
  How many bit strings of length four do not have two consecutive 1s?



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?



Outline

- Introduction
- Product Rule
- Sum Rule
- Using Both Product and Sum Rules
- Tree Diagrams
- **The Pigeonhole Principle**

Pigeonhole Principle

- **Pigeonhole principle**
  If $k + 1$ or more objects (pigeons) are placed into k boxes (pigeonholes) where k is a positive integer, then there is at least one box (pigeonhole) containing two or more of the objects (pigeons).
- **Proof**
  By contradiction
- **Corollary**
  A function f from a set with $k + 1$ or more elements to a set with k elements is not one-to-one.
- The challenge is to correctly determine what are the objects (pigeons) and the boxes (pigeonholes).

Examples

- **Example**
  Among any group of 367 people, there must be at least two with the same birthday.
- **Example**
  In any group of 27 English words, there must be at least two that begin with the same letter.
- **Example**
  How many students must be in a class to guarantee that at least two students receive the same score on the final exam, if the exam is graded on a scale from 0 to 100 points?

Example

- **Example**

Twenty-five points are marked inside the area bounded by a rectangle that is 6 cm long and 4 cm wide. Show that there are at least two points that are at most $\sqrt{2}$ cm apart.

- **Solution:**

Divide the area into squares of 1x1cm, then we have 24 squares, We consider the **squares as pigeonholes**, and the **points as pigeons**. By the pigeonhole Principle, at least 2 points will be in the same square. The biggest distance between 2 points in a 1x1 square is $\sqrt{2}$ cm (i.e. diagonal).

Example

- **Example**
  Let $a_1,\ a_2, \dots ,\ a_{1997}$ represent an arbitrary arrangement of the numbers 1, 2, … , 1997. Is the product $(a_1 - 1)(a_2 - 2)\dots(a_{1997} - 1997)$ an odd or even number? Justify your answer.
- **Solution:**
  The product will be odd only if all the terms are odd. There are 999 odd numbers and 998 even numbers from 1 to 1997. For the product to be odd, all the odd numbers need to perform the subtraction with even numbers to get odd numbers, but we only have 998 even numbers. We consider the **even numbers** as **pigeonholes**, and the **odd numbers pigeons**. By the Pigeonhole principle, at least one of the odd numbers will have a subtraction with another odd number, giving an even result. Therefore the product must be even.

Generalized Pigeonhole Principle

- **Generalized pigeonhole principle**

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

- **Proof**

We prove it bv contradiction. Suppose every box contains at most $\lceil n/k \rceil$ - 1 objects. Then, the total number of objects is at most $k(\lceil n/k \rceil - 1)$. Note that

$\lceil n/k \rceil < n/k + 1$. This implies

$$k(\lceil n/k \rceil - 1) < k\,((n/k\ + 1) - 1) = n,$$

which contradicts the fact that there are a total of n objects.

Examples

- **Example**
  Among 100 people there are at least $\lceil 100/12 \rceil$ = 9 who were born in the same month.
- **Example**
  What is the minimum number of students required in a class to be sure that at least six will receive the same grade, if there are five possible grades, A, B, C, D, and F?
- **Solution**
  It is the smallest integer n such that $\lceil n/5 \rceil$=6, which is $n = 5 \times 5 + 1 = 26$.

Example

- **Example**
  How many cards must be selected from a standard deck of 52 cards to guarantee that at least three cards of the same suit are chosen? How many cards must be selected to guarantee that at least three hearts are selected?
- **Solution:**
  Let the cards to be selected be pigeons and the suits be pigeonholes. By the generalized pigeonhole principle, to guarantee at least three cards of the same suit, we need to have $\lceil n/4 \rceil=3$, so we have $n=2*4+1=9$ cards.
  To guarantee that at least three hearts are selected, we might have to exhaust all the clubs, diamonds, and spades first, and then pick three more cards = 13*3+3=42 cards.

Example

- **Example**

Assume that phone numbers are of the form *NXX-NXX-XXXX*, where the first three digits form the area code, *N* represents a digit from 2 to 9 inclusive, and *X* represents any digit. What is the least number of area codes needed to guarantee that the 25 million phones in a state can be assigned distinct 10-digit phone numbers?

Example

- **Solution:**

Discarding the area code, by the product rule, there are $8*10^6$ different phone numbers of the form *NXX-XXXX*.

Applying the generalized Pigeonhole principle, we have $\lceil 25*10^6 / 8*10^6 \rceil = 4$, so at least one phone number of the form *NXX-XXXX* is shared by 4 phones.

In order to have $25*10^6$ phones assigned distinct numbers, 4 area codes are required in order to distinguish the phones sharing the same *NXX-XXXX* number into different areas.