Introduction

- Consider a set of objects and a property of interest
- Often, we want to know how many of these objects have the desired property
- Example: How many possible passwords are there if a password must contain 6-8 characters?
- These kinds of problems arise frequently in computer science
- Combinatorics (counting) deals with these questions

Basic Counting Rules

- Counting problems can be hard ⇒ useful to decompose
- Two basic very useful decomposition rules:
  1. Product rule: useful when task decomposes into a sequence of independent tasks
  2. Sum rule: decomposes task into a set of alternatives

Example 1

- New company with 12 offices and 2 employees Kate and Jack
- How many ways to assign different offices to Kate and Jack?
- Decomposition: First assign office to Kate, then to Jack
- How many ways to assign office to Kate?
- How many ways to assign office to Jack?
- How many ways to assign offices to Kate and Jack?

Example 2

- Chairs in auditorium labeled with a letter (A-Z) and an integer ∈ [1, 100].
  - Examples: D32, G4, Z99
- Under this labeling scheme, what is the maximum number of chairs in auditorium if we want every chair to be labeled?
- Observe: Max # of labeled chairs = # of different labelings
  - Same as "How many different labelings for a chair?"
- Decomposition: First assign letter, then integer to chair
Example 2, cont.

Chairs in auditorium labeled with a letter (A-Z) and an integer \( \in [1, 100] \). How many different labels?

- How many ways to assign a letter?
- How many ways to assign digit?
- How many ways to assign label?

Extended Product Rule

- We formulated product rule for decomposition into 2 tasks
- But generalizes to decomposition into any \( k \) tasks
- Suppose task \( A \) decomposes to sequence of tasks \( A_1, \ldots, A_k \)
- If there are \( n_1 \) ways of doing \( A_1 \), \( \ldots \), \( n_k \) ways of doing \( A_k \), then there are \( n_1 \cdot n_2 \cdot \ldots \cdot n_k \) ways of doing \( A \)

Example 3

- A bitstring is a string where each character is either 0 or 1
- How many different bit strings of length 7 are there?
- Decomposition: First assign bit to first character, then to second, \ldots, then to seventh
- How many ways to assign bit to each character?
- How many different bitstrings?
- By extended product rule: \( 2 \cdot 2 \cdot \ldots \cdot 2 \) (7 times)
- This is \( 2^7 = 128 \)

Example 4

- Each license plate consists of 3 letters followed by 3 digits [0-9]
- How many different license plates are there?
- Decomposition:
  - How many ways to assign each letter?
  - How many ways to assign each digit?
  - How many different labels for each license plate?

Counting One-to-One Functions

- How many one-to-one functions are there from a set with 3 elements to a set with 5 elements?
- Recall: In a one-to-one function, different elements in domain cannot be assigned to same element in codomain
- Decomposition: Assign first element in domain, then second element, then third element
- How many ways to assign first element?
- How many ways to assign second element?
- How many ways to assign third element?
- How many different functions?

Sum Rule

- Two basic very useful decomposition rules:
  1. Product rule ✓
  2. Sum rule
- Suppose a task \( A \) can be done either in way \( B \) or in way \( C \)
- Suppose there are \( n_1 \) ways to do \( B \), and \( n_2 \) ways to do \( C \)
- Sum rule: There are \( n_1 + n_2 \) ways to do \( A \).
Example 1

- Suppose either a CS faculty or CS student must be chosen as representative for a committee
- There are 14 faculty, and 50 majors
- How many ways are there to choose the representative?

Note: Just like the product rule, the sum rule can be extended to more than two tasks

Extended sum rule: If task can be done in one of \( n_1 \) or \( n_2 \), or \( \ldots \), \( n_k \) ways, then there are \( n_1 + n_2 + \ldots + n_k \) ways of doing it

Example 2

- A student can choose a senior project from one of three lists
- First list contains 23 projects; second list has 15 projects, and third has 19 projects
- Also, no project appears on more than one list
- How many different projects can student choose?

What if some of the projects appeared on both lists?
Caveat: For sum rule to apply, the possibilities must be mutually exclusive

More Complex Counting Problems

- Problems so far required either only product or only sum rule
- But more complex problems require a combination of both!
- Example: In a programming language, a variable name is a string of one or two characters.
  - A character is either a letter [a-z] or a digit [0,9].
  - First character in the string must be a letter.
  - How many possible variable names are there?

Example, cont.

- First, decompose using sum rule.
  - Variable name is either one character or two characters
  - Compute number \( n_1 \) of one character names
  - Compute number \( n_2 \) of two character names
  - By sum rule, total number of variable names = \( n_1 + n_2 = 26 + 936 = 962 \)

Another Example

- A password must be six to seven characters long
- A character is upper case letter or digit
- Each password must contain at least one digit
- How many possible passwords?
- First decompose using sum rule: either six or seven characters
  - Thus, number of possible passwords is \( n_6 + n_7 \) where \( n_i \) is number of passwords with \( i \) characters
Example, cont.

- Computing $n_6$: How many six-character passwords are there containing at least one digit?
- For this, compute the total number of six-character passwords; then subtract number of passwords without any digits
- Total # of 6-char passwords:
  - # of 6-char passwords without any digits:
  - Thus, $n_6 = 36^6 - 26^6$

Example 3

- How many bitstrings are there of length 6 that do not have two consecutive 1’s?
- Let $F(n)$ denote the number of bitstrings of length $n$ that do not have two consecutive 1’s
- We’ll first derive a recursive equation to characterize $F(n)$
- By the sum rule, $F(n)$ is the sum of:
  1. # of $n$-bit strings starting with 1 not containing 11
  2. # of $n$-bit strings starting with 0 not containing 11
- How many ways of forming “the rest”?
  - # of bitstrings of length $n - 1$ not containing 11:
  - By product rule, number of $n$-bit strings starting with 0, not containing 11 is $1 \cdot F(n - 1) = F(n - 1)$

Example, cont.

- Computing $n_7$: How many seven-character passwords are there containing at least one digit?
- $n_7 =$ Total - # without any digits
- Total # of 7-char passwords =
- Those without any digits:
  - $n_7 = 36^7 - 26^7$
  - Total number of passwords = $n_6 + n_7 = 72,200,220,480$

Example, cont.

- How many $n$-bit strings are there starting with 1 and not containing 11?
- Since we don’t want two consecutive 1’s, second bit must be 0
- But the rest can be anything as long as it doesn’t contain two consecutive 1’s
- Thus, the rest = number of bitstrings of length $n - 2$ not containing two consecutive 1’s
  - But this is precisely $F(n - 2)$!
  - Thus, by product rule, there are $1 \cdot 1 \cdot F(n - 2) = F(n - 2)$
- $n$-bit strings starting with 1 and not containing 11

Example, cont.

- Recall: $F(n)$ is sum of:
  1. # of $n$-bit strings starting with 1 not containing 11
  2. # of $n$-bit strings starting with 0 not containing 11
- We determined that (1) is $F(n - 2)$
- We determined that (2) is $F(n - 1)$
- Therefore, $F(n) = F(n - 1) + F(n - 2)$
- This is a recursive definition, but what are the base cases?
Example 3, cont.
- What is $F(1)$?
- What is $F(2)$?
- Original question asks for $F(6)$
- Using base cases and recursive definition:
  - What is $F(3)$?
  - What is $F(4)$?
  - What is $F(5)$?
  - What is $F(6)$?
- Recall: Sum Rule
  - Recall: Sum rule only applies if a task is as disjunction of two mutually exclusive tasks
  - What do we do if the tasks aren’t mutually exclusive?
    - Example: You can choose from set $A$ or set $B$, but they have some elements in common
      - In such cases, the sum rule is not applicable because common elements counted twice
    - Fortunately, there is a generalization of the sum rule called the inclusion-exclusion principle
- The Inclusion-Exclusion Principle
  - Suppose a set $A$ can be written as union of sets $B$ and $C$
  - Inclusion-Exclusion Principle:
    $$|A| = |B| + |C| - |B \cap C|$$
- Example, cont.
  - First compute $|B|$, bitstrings of length 8 that start with 1
  - What is $|B|$?
  - Now compute $|C|$, bitstrings of length 8 ending in 00
  - What is $|C|$?
  - Now compute $|B \cap C|$, those starting with 1 and ending in 00
  - What is $|B \cap C|$?
  - Number of 8-bit strings that start with 1 and end in 00:
    $$128 + 64 - 32 = 160$$
- Inclusion-Exclusion Principle Example
  - How many bit strings of length 8 either start with 1 or end with two bits 00?
  - Let $B$ be the set of bitstrings that start with 1
  - Let $C$ be the set of bitstrings that end with 00
  - We want $|B \cup C|$.
  - By the inclusion-exclusion principle,
    $$|B \cup C| = |B| + |C| - |B \cap C|$$
    Thus, compute $|B|$, $|C|$ and $|B \cap C|$.
- Another Example
  - A company receives 350 applications for job positions
  - 220 of applicants are CS majors, 147 of applicants are business majors, and 51 are double CS and business majors
  - How many are neither CS nor business majors?
  - # of non-CS/non-business applicants = Total applicants - # of CS or business majors
  - What is $|B \cup C|$?
  - The remaining applicants are neither business nor CS:
    $$350 - 316 = 34$$
The Pigeonhole Principle

- Suppose there is a flock of 36 pigeons and a set of 35 pigeonholes
- Each pigeon wants to sit in its own hole
- But since there are less holes than there are pigeons, one pigeon is left without a hole.

**The Pigeonhole Principle:** If \( n + 1 \) or more objects are placed into \( n \) boxes, then at least one box contains 2 or more objects.

Examples

- Consider an event with 367 people. Is it possible no pair of people have the same birthday?
- Consider function \( f \) from a set with \( k + 1 \) or more elements to a set with \( k \) elements. Is it possible \( f \) is one-to-one?
- Consider \( n \) married couples. How many of the \( 2n \) people must be selected to guarantee there is at least one married couple?

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:** (by contradiction) Suppose every box contains less than \( \lceil n/k \rceil \) objects
  - Then, there must be less than \( k \cdot (\lceil n/k \rceil - 1) \) objects
  - From earlier, we know \( \lceil n/k \rceil < n + 1 \)
  - Adding one and multiply both sides by \( k \): \( k \cdot (\lceil n/k \rceil - 1) < n \)
  - Thus, this would imply there are less than \( n \) objects, a contradiction

Examples

- If there are 30 students in a class, at least how many must be born in the same month?
- What is the minimum # of students required to ensure at least 6 students receive the same grade (A, B, C, D, F)?
  - Want \( \min n \) such that \( \lceil n/5 \rceil \geq 6 \Rightarrow \)
  - What is the min # of cards that must be chosen to guarantee three have same suit? (suits: ♠, ♥, ♦, ♣)
  - Want \( \min n \) such that \( \lceil n/4 \rceil = 3 \Rightarrow \)

More Examples

- Suppose there are 25 million phones in a state
  - Each phone number has seven digits, where first digit is \([2−9]\) and other digits are \([0−9]\)
  - What is the minimum number of area codes necessary to guarantee each phone has a unique number?
  - First, compute # of seven-digit phone numbers:
  - Let \( k \) be the number of area codes
  - Want to find minimum \( k \) such that \( \lceil \frac{25 \times 10^6}{k} \rceil \leq 8 \times 10^6 \Rightarrow \)