For any disjoint sets, $A$ and $B$: $|A \cup B| = |A| + |B|$

In our example:

$A = \{\text{custom characters}\}$ \hspace{1cm} |A| = 96

$B = \{\text{preset characters}\}$ \hspace{1cm} |B| = 10

$A \cup B = \{m : m \text{ is a character that is either custom or preset}\}$

$|A \cup B| = \text{the number of possible characters}$
For any disjoint sets, A and B: \(|A \cup B| = |A| + |B|\)

More generally:

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.
For any disjoint sets, A and B: $|A \cup B| = |A| + |B|$

More generally:

To count the number of objects with a given property:
* Divide the set of objects into mutually exclusive (disjoint/nonoverlapping) groups.
* Count each group separately.
* Add up these counts.
Select which method lets us count the number of length n binary strings.

A. The product rule.
B. The sum rule.
C. Either rule works.
D. Neither rule works.
Length n binary strings

Select which method lets us count the number of length n binary strings.

A. The product rule. Select first bit, then second, then third …
B. The sum rule. \{0…\} U \{1…\} gives recurrence \( N(n) = 2N(n-1), N(0)=1 \)
C. Either rule works.
D. Neither rule works.
Memory: storing length $n$ binary strings

How many binary strings of length $n$ are there?

How many bits does it take to store a length $n$ binary string?
Memory: storing length $n$ binary strings

How many binary strings of length $n$ are there? $2^n$

How many bits does it take to store a length $n$ binary string? $n$

*General principle:* number of bits needed to store [the “unique ID” of] an object is

$$\lceil \log_2(\text{number of objects}) \rceil$$

Why the ceiling function?
Scenario: We want to store a user’s state of residence in our database. How many bits of memory do we need to allocate?

A. 2
B. 5
C. 6
D. 10
E. 50
Memory: storing integers

**Scenario:** We want to store a non-negative integer that has at most $n$ decimal digits. How many bits of memory do we need to allocate?

A. $n$
B. $2^n$
C. $10^n$
D. $n \log_2 10$
E. $n \log_{10} 2$
At an ice cream parlor, you can choose to have your ice cream in a bowl, cake cone, or sugar cone. There are 20 different flavors available.

How many single-scoop creations are possible?

A. 20  
B. 23  
C. 60  
D. 120  
E. None of the above.
At an ice cream parlor, you can choose to have your ice cream in a bowl, cake cone, or sugar cone. There are 20 different flavors available.

You can convert your single-scoop of ice cream to a sundae. Sundaes come with your choice of caramel or hot fudge. Whipped cream, and a cherry on top, are options. How many desserts are possible?

A. $20 \times 3 \times 2 \times 2$
B. $20 \times 3 \times 2 \times 2 \times 2$
C. $20 \times 3 + 20 \times 3 \times 2 \times 2$
D. $20 \times 3 + 20 \times 3 \times 2 \times 2 \times 2$
E. None of the above.
A scheduling problem

In one request, four jobs arrive to a server: J1, J2, J3, J4.

The server starts each job right away, splitting resources among all active ones.

Different jobs take different amounts of time to finish.

How many possible finishing orders are there?

A. $4^4$
B. 4+4+4+4
C. 4 * 4
D. None of the above.
A scheduling problem

In one request, four jobs arrive to a server: J1, J2, J3, J4.

The server starts each job right away, splitting resources among all active ones.

Different jobs take different amounts of time to finish.

**How many possible finishing orders are there?**

*Product rule analysis*

- 4 options for which job finishes first.
- Once pick that job, 3 options for which job finishes second.
- Once pick those two, 2 options for which job finishes third.
- Once pick first three jobs, only 1 remains.

\( (4)(3)(2)(1) = 4! = 24 \)

Which options are available will depend on first choice; but the **number** of options will be the same.
Permutations

**Permutation:** rearrangement / ordering of n distinct objects so that each object appears exactly once

**Theorem 1:** The number of permutations of n objects is

\[ n! = n(n-1)(n-2) \ldots (3)(2)(1) \]

*Convention: 0! = 1*
Traveling salesperson

Planning a trip to

New York
Chicago
Baltimore
Los Angeles
San Diego
Minneapolis
Seattle

Must **start in New York** and **end in Seattle**.

How many ways can the trip be arranged?

A. $7!$
B. $2^7$
C. None of the above.
Planning a trip to

New York
Chicago
Baltimore
Los Angeles
San Diego
Minneapolis
Seattle

Must **start in New York** and **end in Seattle**.
Must also **visit Los Angeles immediately after San Diego**.

*How many ways can the trip be arranged now?*
Traveling salesperson

Planning a trip to

New York
Chicago
Baltimore
Los Angeles
San Diego
Minneapolis
Seattle

Must **start in New York** and **end in Seattle**.
Must also **visit Los Angeles immediately after San Diego**.

How many ways can the trip be arranged now?

Treat LA & SD as a single stop.

\[(1)(4!)(1) = 24\text{ arrangements.}\]
Traveling salesperson

Planning a trip to

New York
Chicago
Baltimore
Los Angeles
San Diego
Minneapolis
Seattle

Must **start in New York** and **end in Seattle**.
Must also **visit Los Angeles and San Diego immediately after each other (in any order)**.

*How many ways can the trip be arranged now?*
Traveling salesperson

Planning a trip to

New York
Chicago
Baltimore
Los Angeles
San Diego
Minneapolis
Seattle

Break into two disjoint cases:
Case 1: LA before SD  
Case 2: SD before LA  

→ 48 arrangements in total

Must **start in New York** and **end in Seattle**.
Must also **visit Los Angeles and San Diego immediately after each other** (in any order).

*How many ways can the trip be arranged now?*
Traveling salesperson

Planning a trip to

New York
Chicago
Baltimore
Los Angeles
San Diego
Minneapolis
Seattle

Must start in New York and end in Seattle.
Must also visit Los Angeles and San Diego immediately after each other (in any order).

How many ways can the trip be arranged now?
Traveling salesperson

Planning a trip to

New York
Chicago
Baltimore
Los Angeles
San Diego
Minneapolis
Seattle

Is there an order of visiting the cities that stops at each city exactly once and minimizes the distance traveled?

<table>
<thead>
<tr>
<th></th>
<th>NY</th>
<th>Chicago</th>
<th>Balt.</th>
<th>LA</th>
<th>SD</th>
<th>Minn.</th>
<th>Seattle</th>
</tr>
</thead>
<tbody>
<tr>
<td>NY</td>
<td>0</td>
<td>800</td>
<td>200</td>
<td>2800</td>
<td>2800</td>
<td>1200</td>
<td>2900</td>
</tr>
<tr>
<td>Chicago</td>
<td>800</td>
<td>0</td>
<td>700</td>
<td>2000</td>
<td>2100</td>
<td>400</td>
<td>2000</td>
</tr>
<tr>
<td>Balt.</td>
<td>200</td>
<td>700</td>
<td>0</td>
<td>2600</td>
<td>2600</td>
<td>1100</td>
<td>2700</td>
</tr>
<tr>
<td>LA</td>
<td>2800</td>
<td>2000</td>
<td>2600</td>
<td>0</td>
<td>100</td>
<td>1900</td>
<td>1100</td>
</tr>
<tr>
<td>SD</td>
<td>2800</td>
<td>2100</td>
<td>2600</td>
<td>100</td>
<td>0</td>
<td>2000</td>
<td>1300</td>
</tr>
<tr>
<td>Minn.</td>
<td>1200</td>
<td>400</td>
<td>1100</td>
<td>1900</td>
<td>2000</td>
<td>0</td>
<td>1700</td>
</tr>
<tr>
<td>Seattle</td>
<td>2900</td>
<td>2000</td>
<td>2700</td>
<td>1100</td>
<td>1300</td>
<td>1700</td>
<td>0</td>
</tr>
</tbody>
</table>
Want a Hamiltonian tour
Given a set of cities and distances between them, determining a shortest-distance path that visits all of them is "NP-hard" (== “intractable”, “very hard”)

Is there any algorithm for this question?

A. No, it's not possible.
B. Yes, it's just very slow.
C. ?

In terms of real-world practice: means that no efficient algorithm is known…
Traveling salesperson

Exhaustive search algorithm

List all possible orderings of the cities.
For each ordering, compute the distance traveled.
Choose the ordering with minimum distance.

How long does this take?

Want a Hamiltonian tour
Traveling salesperson

Exhaustive search algorithm: given $n$ cities and distances between them.

List all possible orderings of the cities. For each ordering, compute the distance traveled. Choose the ordering with minimum distance. $O(\text{number of orderings})$

How long does this take?

Want a Hamiltonian tour
Traveling salesperson

Exhaustive search algorithm: given $n$ cities and distances between them.

List all possible orderings of the cities.
For each ordering, compute the distance traveled.

Choose the ordering with minimum distance.

O(number of orderings)

How long does this take?

A. O(n)
B. O($n^2$)
C. O($n^n$)
D. O(n!)
E. None of the above.

Want a Hamiltonian tour
Traveling salesperson

Exhaustive search algorithm: given $n$ cities and distances between them.

List all possible orderings of the cities. For each ordering, compute the distance traveled. Choose the ordering with minimum distance.

O(number of orderings)

How long does this take?

A. $O(n)$
B. $O(n^2)$
C. $O(n^n)$
D. $O(n!)$
E. None of the above.

Moral: counting gives upper bound on algorithm runtime.

$2^n < n! < n^n$ for large $n$
A complete bipartite graph is an undirected graph whose vertex set is partitioned into two sets $V_1$, $V_2$ such that
- there is an edge between each vertex in $V_1$ and each vertex in $V_2$
- there are no edges both of whose endpoints are in $V_1$
- there are no edges both of whose endpoints are in $V_2$

A. Yes
B. No
A complete bipartite graph is an undirected graph whose vertex set is partitioned into two sets $V_1$, $V_2$ such that

- there is an edge between each vertex in $V_1$ and each vertex in $V_2$
- there are no edges both of whose endpoints are in $V_1$
- there are no edges both of whose endpoints are in $V_2$
Claim: any complete bipartite graph with \(|V_1| = k, |V_2| = k+1\) is Hamiltonian.

How many Hamiltonian tours can we find?

A. \(k\)
B. \(k(k+1)\)
C. \(k!(k+1)!\)
D. \((k+1)!\)
E. None of the above.
Claim: any complete bipartite graph with $|V_1| = k$, $|V_2| = k+1$ is Hamiltonian.

How many Hamiltonian tours can we find?

A. $k$
B. $k(k+1)$
C. $k!(k+1)!$
D. $(k+1)!$
E. None of the above.

*Product rule!*
When product rule fails

How many Hamiltonian tours can we find?

A. 5!
B. 5!4!
C. ?
Which Hamiltonian tours start at e?
List all possible next moves.
Then count leaves.

Rosen p.394-395
Let $A = \{\text{people who know Java}\}$ and $B = \{\text{people who know C}\}$

How many people know Java or C (or both)?

A. $|A| + |B|$  
B. $|A| |B|$  
C. $|A|^{|B|}$  
D. $|B|^{|A|}$  
E. None of the above.
When sum rule fails

Let \( A = \{ \text{people who know Java} \} \) and \( B = \{ \text{people who know C} \} \)

\# people who know Java or C = \# people who know Java
When sum rule fails

Let A = \{ people who know Java \} and B = \{ people who know C \}

\# people who know Java or C = \# people who know Java + \# people who know C

Double counted!

Rosen p. 392-394
Let $A = \{ \text{people who know Java} \}$ and $B = \{ \text{people who know C} \}$

$\# \text{people who know Java or C} = (\# \text{people who know Java}) + (\# \text{people who know C}) - (\# \text{people who know both})$
Inclusion-Exclusion principle

Let $A = \{ \text{people who know Java} \}$ and $B = \{ \text{people who know C} \}$

$$|A \cup B| = |A| + |B| - |A \cap B|$$
Inclusion-Exclusion for three sets

\[|A \cup B \cup C| =?\]

Rosen p. 392-394
Inclusion-Exclusion for three sets

\[ |A \cup B \cup C| =? \]
Inclusion-Exclusion for three sets

\[ |A \cup B \cup C| =? \]
Inclusion-Exclusion for three sets

\[ |A \cup B \cup C| =? \]
Inclusion-Exclusion for three sets

\[ |A \cup B \cup C| =? \]

Rosen p. 392-394
Inclusion-Exclusion for three sets

\[ |A \cup B \cup C| = ? \]
Inclusion-Exclusion for three sets

$|A \cup B \cup C| = ?$

Rosen p. 392-394
Inclusion-Exclusion for three sets

\[ |A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |A \cap C| + |A \cap B \cap C| \]
Inclusion-Exclusion principle

If $A_1, A_2, \ldots, A_n$ are finite sets then

$$|A_1 \cup A_2 \cup \cdots \cup A_n| = \sum_{1 \leq i \leq n} |A_i| - \sum_{1 \leq i < j \leq n} |A_i \cap A_j| + \sum_{1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k| - \cdots + (-1)^{n+1}|A_1 \cap A_2 \cap \cdots \cap A_n|$$
How many four-letter strings have one vowel and three consonants?

There are 5 vowels: AEIOU and 21 consonants: BCDFGHJKLMNPQRSTVWXYZ.

A. $5 \cdot 21^3$
B. $26^4$
C. $5 + 52$
D. None of the above.
How many four-letter strings have one vowel and three consonants? There are 5 vowels: AEIOU and 21 consonants: BCDFGHJKLMNPQRSTVWXYZ.

<table>
<thead>
<tr>
<th>Template</th>
<th># Matching</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCCC</td>
<td>5 * 21 * 21 * 21</td>
</tr>
<tr>
<td>CVCC</td>
<td>21 * 5 * 21 * 21</td>
</tr>
<tr>
<td>CCVC</td>
<td>21 * 21 * 5 * 21</td>
</tr>
<tr>
<td>CCCV</td>
<td>21 * 21 * 21 * 5</td>
</tr>
</tbody>
</table>

**Total:** $4 * 5 * 21^3$
If \( A = X_1 \cup X_2 \cup \ldots \cup X_n \) and all \( X_i, X_j \) are disjoint \((i \neq j)\), and all \( X_i \) have same cardinality, then
\[
|X_i| = \frac{|A|}{n}
\]

“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 \( w \), \( d \) of the \( n \) ways give the same result as \( w \) did.
If \( A = X_1 \cup X_2 \cup \ldots \cup X_n \) and all \( X_i, X_j \) are disjoint \((i \neq j)\), and all \( X_i \) have same cardinality, then

\[
|X_i| = |A| / n
\]

“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 \( w \), \( d \) of the \( n \) ways give the same result as \( w \) did.
If $A = X_1 \cup X_2 \cup \ldots \cup X_n$ and all $X_i, X_j$ are disjoint ($i \neq j$), and all $X_i$ have same cardinality, then

$$|X_i| = \frac{|A|}{n}$$

Put another way …..

If objects are partitioned into categories of equal size, and we want to think of different objects as being the same if they are in the same category, then

$$\# \text{categories} = \frac{\text{(# objects)}}{\text{(size of each category)}}$$
An ice cream parlor has $n$ different flavors available. How many ways are there to order a two-scoop ice cream cone (where you specify which scoop goes on bottom and which on top, and the two flavors must be different)?

A. $n^2$
B. $n!$
C. $n(n-1)$
D. $2n$
E. None of the above.
An ice cream parlor has n different flavors available. How can we use our earlier answer to decide the number of cones, if we count two cones as the same if they have the same two flavors (even if they're in opposite order)?

A. Double the previous answer.
B. Divide the previous answer by 2.
C. Square the previous answer.
D. Keep the previous answer.
E. None of the above.
Ice cream!

An ice cream parlor has $n$ different flavors available. **How can we use our earlier answer to decide the number of cones, if we count two cones as the same if they have the same two flavors (even if they're in opposite order)?**

**Objects:**
**Categories:**
**Size of each category:**

# categories = (# objects) / (size of each category)
Ice cream!

An ice cream parlor has n different flavors available. How can we use our earlier answer to decide the number of cones, if we count two cones as the same if they have the same two flavors (even if they're in opposite order)?

**Objects**: cones
**Categories**: flavor pairs (regardless of order)
**Size of each category**:

\[
\# \text{ categories} = \frac{\# \text{ objects}}{\text{size of each category}}
\]
An ice cream parlor has \( n \) different flavors available. How can we use our earlier answer to decide the number of cones, if we count two cones as the same if they have the same two flavors (even if they're in opposite order)?

**Objects**: cones \( n(n-1) \)

**Categories**: flavor pairs (regardless of order)

**Size of each category**: 2

\[
\text{# categories} = \frac{(n)(n-1)}{2}
\]

Avoiding double-counting
How many different colored triangles can we create by tying these three pipe cleaners end-to-end?

A. $3!$
B. $2^3$
C. $3^2$
D. 1
E. None of the above.
How many different colored triangles can we create by tying these three pipe cleaners end-to-end?

**Objects:** all different colored triangles  
**Categories:** physical colored triangles (two triangles are the same if they can be rotated and/or flipped to look alike)  
**Size of each category:**

\[
\# \text{ categories} = \frac{\# \text{ objects}}{(\text{size of each category})}
\]
How many different colored triangles can we create by tying these three pipe cleaners end-to-end?

**Objects**: all different colored triangles \(3!\)

**Categories**: physical colored triangles (two triangles are the same if they can be rotated and/or flipped to look alike)

**Size of each category**: \((3)(2)\) three possible rotations, two possible flips

\[
\# \text{ categories} = \frac{\# \text{ objects}}{\text{size of each category}} = \frac{6}{6} = 1
\]
Object Symmetries
Announcements

**HW6 is assigned**
Due Thursday 2/23 at 11:59pm

**MT2 PPs and Review**
Improve your process from MT1?

**Office Hours and 1-1 sessions**
Are you using these well?
Lots on the course calendar!

HW6 is assigned
Due Thursday 2/23 at 11:59pm

MT2 PPs and Review
Improve your process from MT1?

Office Hours and 1-1 sessions
Are you using these well?
Lots on the course calendar!