COMP170
Discrete Mathematical Tools for Computer Science

More Counting

Version 2.0: Last updated, JUL 19, 2007

Discrete Math for Computer Science
K. Bogart, C. Stein and R.L. Drysdale
Section 1.2, pp. 9-19

1.2 Counting Lists, Permutations, and Subsets

- Using the Sum and Product Principles
- Lists and Functions
- The Bijection Principle
- k-Element Permutations of a Set
- k-Element Subsets of a Set
  Binomial Coefficients

Some Simple Examples

- The entry code to our office is 4 characters long where each character is a number between $\{0 \ldots 9\}$.

  e.g., 1012, 2561.

  How many door codes are there?

Some Simple Examples

- The entry code to our office is 4 characters long where each character is a number between $\{0 \ldots 9\}$.

  e.g., 1012, 2561.

  How many door codes are there?

- A computer system's pasword must be 6 characters long, where each character is in $\{a \ldots z, A \ldots Z\}$.

  e.g., $abcdEF$, $DHrsAQ$, $Mickey$.

  How many passwords are there?

Some Simple Examples

- The entry code to our office is 4 characters long where each character is a number between $\{0 \ldots 9\}$.

  e.g., 1012, 2561.

  How many door codes are there?

- A computer system's pasword must be 6 characters long, where each character is in $\{a \ldots z, A \ldots Z\}$.

  e.g., $abcdEF$, $DHrsAQ$, $Mickey$.

  How many passwords are there?

- Hong Kong car plates are of the form $L_1L_2D_1D_2D_3D_4$ where the L_i are letters in $\{A \ldots Z\}$ and the D_i are digits in $\{0, \ldots 9\}$.

  e.g., $AB1234$, $EC1357$.

  How many car plates are there?

The entry code to our office is 4 characters long where each character is a number between $\{0 \ldots 9\}$.
E.G., 1012, 2561.
How many door codes are there?

The entry code to our office is 4 characters long where each character is a number between $\{0 \ldots 9\}$.

E.G., 1012, 2561.

How many door codes are there?

A door code is of the form $X_1 X_2 X_3 X_4$ where,

for $i = 1, 2, 3, 4, \;\; X_i \in \{0, 1, \ldots, 9\}$

($\in$ means *is in* or *is a member of*)

The entry code to our office is 4 characters long where each character is a number between $\{0 \ldots 9\}$.
E.G., 1012, 2561.
How many door codes are there?

A door code is of the form $X_1 X_2 X_3 X_4$ where,
for $i = 1, 2, 3, 4,\ \ X_i \in \{0, 1, \ldots, 9\}$
($\in$ means *is in* or *is a member of*)

There are
10 possibilities for X_1 10 possibilities for X_2
10 possibilities for X_3 10 possibilities for X_4

$\Rightarrow 10 \times 10 \times 10 \times 10 = 10^4 = 10,000$ possible door codes.

The entry code to our office is 4 characters long where each character is a number between $\{0 \ldots 9\}$.
E.G., 1012, 2561.
How many door codes are there?

A door code is of the form $X_1X_2X_3X_4$ where,
for $i = 1, 2, 3, 4, \;\; X_i \in \{0, 1, \ldots, 9\}$
($\in$ means *is in* or *is a member of*)

There are

10 possibilities for X_1 10 possibilities for X_2
10 possibilities for X_3 10 possibilities for X_4

$\Rightarrow 10 \times 10 \times 10 \times 10 = 10^4 = 10,000$ possible door codes.

Note; This is really the *product principle*
What are the sets being used?

A computer system's pasword must be 6 characters long, where each character is in $\{a \ldots z, A \ldots Z\}$.
E.G., $abcdEF$, $DHrsAQ$, $Mickey$.
How many passwords are there?

A computer system's pasword must be 6 characters long, where each character is in $\{a \ldots z, A \ldots Z\}$.
E.G., $abcdEF$, $DHrsAQ$, $Mickey$.
How many passwords are there?

A password is of the form $D_1D_2D_3D_4D_5D_6$ where
for $i = 1, 2, 3, 4, 5, 6, \quad D_i \in \{a \ldots z, A \ldots Z\}$.

Each D_i has 52 possible choices so,
the total number of passwords is

A computer system's pasword must be 6 characters long, where each character is in $\{a \ldots z, A \ldots Z\}$.
E.G., $abcdEF$, $DHrsAQ$, $Mickey$.
How many passwords are there?

A password is of the form $D_1D_2D_3D_4D_5D_6$ where
for $i = 1, 2, 3, 4, 5, 6,$ $\quad D_i \in \{a \ldots z, A \ldots Z\}$.

Each D_i has 52 possible choices so,
the total number of passwords is

$$52 \times 52 \times 52 \times 52 \times 52 \times 52 = 52^6$$

Hong Kong car plates are of the form $L_1L_2D_1D_2D_3D_4$
where the L_i are letters in $\{A \ldots Z\}$
and the D_i are digits in $\{0, \ldots 9\}$.

e.g., $AB1234$, $EC1357$.

How many car plates are there?

Hong Kong car plates are of the form $L_1 L_2 D_1 D_2 D_3 D_4$
where the L_i are letters in $\{A \ldots Z\}$
and the D_i are digits in $\{0, \ldots 9\}$.

e.g., $AB1234$, $EC1357$.

How many car plates are there?

Each L_i has 26 possibilities and
each D_i has 10 possibilities
so the total number of car plates is

$$26 \times 26 \times 10 \times 10 \times 10 \times 10 = 26^2 \times 10^4 = 6,760,000$$

We have just seen examples of

Product Principle, Version 2

If a set S of lists of length m has the properties that

1. there are i_1 different first elements of lists in S, and
2. for each $j > 1$
   and each choice of the first $j-1$ elements of a list in S,
   there are i_j choices of elements in position j
   of those lists,

$\Rightarrow$ there are $i_1 i_2 \cdots i_m = \prod_{k=1}^{m} i_k$ lists in S.

We have just seen examples of

Product Principle, Version 2

If a set S of lists of length m has the properties that

1. there are i_1 different first elements of lists in S, and

2. for each $j > 1$
   and each choice of the first $j-1$ elements of a list in S,
   there are i_j choices of elements in position j
   of those lists,

$\Rightarrow$ there are $i_1 i_2 \cdots i_m = \prod_{k=1}^{m} i_k$ lists in S.

$$\boxed{L_1} \quad \boxed{L_2} \quad \boxed{L_3} \quad \cdots \quad \boxed{L_m}$$

$$\uparrow \qquad \uparrow \qquad \uparrow \qquad\qquad \uparrow$$

$$i_1 \times i_2 \times i_3 \times \cdots \times i_m$$

Combining Sum and Product Principles

A password for a certain computer system is supposed to be between 4 and 8 characters long and composed of lowercase and/or uppercase letters.

Combining Sum and Product Principles

A password for a certain computer system is supposed to be between 4 and 8 characters long and composed of lowercase and/or uppercase letters.

How many passwords are possible?

Combining Sum and Product Principles

A password for a certain computer system is supposed to be between 4 and 8 characters long and composed of lowercase and/or uppercase letters.

How many passwords are possible?

This is a counting problem!

Should we use sum or product principle?

Combining Sum and Product Principles

A password for a certain computer system is supposed to be between 4 and 8 characters long and composed of lowercase and/or uppercase letters.
How many passwords are possible?

This is a counting problem!
Should we use sum or product principle?

Set of *all* passwords is *disjoint union* of
passwords with 4, 5, 6, 7, or 8 letters.

Combining Sum and Product Principles

A password for a certain computer system is supposed to be between 4 and 8 characters long and composed of lowercase and/or uppercase letters.

How many passwords are possible?

This is a counting problem!

Should we use sum or product principle?

Set of *all* passwords is *disjoint union* of
passwords with 4, 5, 6, 7, or 8 letters.

So *sum principle* might help us.

Let P_i be set of i-letter passwords
and P set of *all* possible passwords.

Let P_i be set of i-letter passwords
and P set of *all* possible passwords.

Clearly

$$P = P_4 \cup P_5 \cup P_6 \cup P_7 \cup P_8.$$

Let P_i be set of i-letter passwords
and P set of *all* possible passwords.

Clearly

$$P = P_4 \cup P_5 \cup P_6 \cup P_7 \cup P_8.$$

P_i are mutually disjoint, thus apply *sum principle*

$$|P| = \sum_{i=4}^{8} |P_i|.$$

Let P_i be set of i-letter passwords
and P set of *all* possible passwords.

Clearly
$$P = P_4 \cup P_5 \cup P_6 \cup P_7 \cup P_8.$$

P_i are mutually disjoint, thus apply *sum principle*
$$|P| = \sum_{i=4}^{8} |P_i|.$$

By *product principle*, $|P_i|$ is 52^i.

Let P_i be set of i-letter passwords
and P set of *all* possible passwords.

Clearly

$$P = P_4 \cup P_5 \cup P_6 \cup P_7 \cup P_8.$$

P_i are mutually disjoint, thus apply *sum principle*

$$|P| = \sum_{i=4}^{8} |P_i|.$$

By *product principle*, $|P_i|$ is 52^i.

$\Rightarrow$ total number of passwords is

$$52^4 + 52^5 + 52^6 + 52^7 + 52^8.$$

Lists and Functions

We've already seen *lists*.
Informally, a list of k elements from T is

$L = L_1 L_2 \ldots L_k$, where each $L_i \in T$.

Lists and Functions

We've already seen *lists*.
Informally, a list of k elements from T is
$L = L_1 L_2 \ldots L_k$, where each $L_i \in T$.

Rewriting a list in a different order yields a different list
(compare to sets, where changing the order doesn't change the set)

Lists and Functions

We've already seen *lists*.
Informally, a list of k elements from T is
$L = L_1 L_2 \ldots L_k$, where each $L_i \in T$.

Rewriting a list in a different order yields a different list
(compare to sets, where changing the order doesn't change the set)

Example: $MORE$ and $ROME$ are
two different 4-letter lists from $T = \{A, B, \ldots, Z\}$

$\{M, O, R, E\}$ and $\{R, O, M, E\}$ are
the same subset of T.

Lists and Functions (cont)

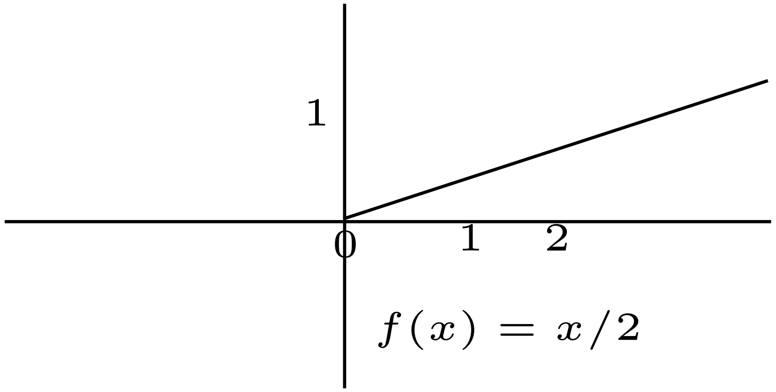
We now introduce functions.

A function f from set S to set T is a *relationship* that associates exactly one element of T with each element of S

Lists and Functions (cont)

We now introduce functions.

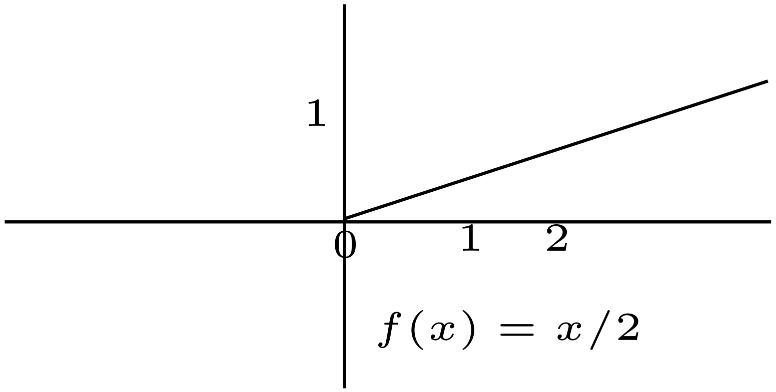
A function f from set S to set T is a *relationship* that associates exactly one element of T with each element of S



Lists and Functions (cont)

We now introduce functions.

A function f from set S to set T is a *relationship* that associates exactly one element of T with each element of S



In continuous math, e.g., basic calculus and algebra, you often have formulas, e.g., $f(x) = x/2$ that describe the function. In discrete math, where we often deal with finite sets, we can frequently state the full function explicitly: e.g.,

$$f(1) = \text{Sam},\ f(2) = \text{Mary},\ f(3) = \text{Sarah}$$

Lists and Functions (cont)

We often write a function from S to T as

$$f : S \to T$$

S is the domain of f; T is the range of f.

Lists and Functions (cont)

A k-element List, $L = L_1 L_2 \ldots L_k$ from set T can now be defined as a function

$$f : \{1, 2, \ldots, k\} \to T \text{ where } f(i) = L_i$$

Lists and Functions (cont)

A k-element List, $L = L_1 L_2 \ldots L_k$ from set T can now be defined as a function

$$f : \{1, 2, \ldots, k\} \rightarrow T \text{ where } f(i) = L_i$$

Example: 4-element lists from $\{A, \ldots, Z\}$

$MORE$ is the function

$$f(1) = M,\ f(2) = O,\ f(3) = R,\ f(4) = E$$

$ROME$ is the function

$$f(1) = R,\ f(2) = O,\ f(3) = M,\ f(4) = E$$

Exercises

- Write down all the functions from the 2-element set $\{1, 2\}$ to the 2-element set $\{a, b\}$.

Exercises

- Write down all the functions from the 2-element set $\{1, 2\}$ to the 2-element set $\{a, b\}$.

- How many functions are there from a 2-element set to a 3-element set?

Exercises

- Write down all the functions from the 2-element set $\{1, 2\}$ to the 2-element set $\{a, b\}$.
- How many functions are there from a 2-element set to a 3-element set?
- How many functions are there from a 3-element set to a 2-element set?

Write down all the functions from the two-element set $\{1, 2\}$ to the two-element set $\{a, b\}$.

Write down all the functions from the two-element set $\{1,2\}$ to the two-element set $\{a,b\}$.

Use $f_1, f_2, \ldots$ to denote the functions. To describe $f_i : \{1,2\} \to \{a,b\}$, we must specify $f_i(1)$ and $f_i(2)$.

Write down all the functions from the two-element set $\{1, 2\}$ to the two-element set $\{a, b\}$.

Use $f_1, f_2, \ldots$ to denote the functions. To describe $f_i : \{1, 2\} \to \{a, b\}$, we must specify $f_i(1)$ and $f_i(2)$.

$$\begin{array}{ll} f_1(1) = a & f_1(2) = b \\ f_2(1) = b & f_2(2) = a \\ f_3(1) = a & f_3(2) = a \\ f_4(1) = b & f_4(2) = b \end{array}$$

Write down all the functions from the two-element set $\{1, 2\}$ to the two-element set $\{a, b\}$.

Use $f_1, f_2, \ldots$ to denote the functions. To describe $f_i : \{1, 2\} \to \{a, b\}$, we must specify $f_i(1)$ and $f_i(2)$.

$$
\begin{array}{ll}
f_1(1) = a & f_1(2) = b \\
f_2(1) = b & f_2(2) = a \\
f_3(1) = a & f_3(2) = a \\
f_4(1) = b & f_4(2) = b
\end{array}
$$

How do we know we've found *all* of them?

Write down all the functions from the two-element set $\{1, 2\}$ to the two-element set $\{a, b\}$.

Use $f_1, f_2, \ldots$ to denote the functions. To describe $f_i : \{1, 2\} \to \{a, b\}$, we must specify $f_i(1)$ and $f_i(2)$.

$$\begin{array}{ll} f_1(1) = a & f_1(2) = b \\ f_2(1) = b & f_2(2) = a \\ f_3(1) = a & f_3(2) = a \\ f_4(1) = b & f_4(2) = b \end{array}$$

How do we know we've found *all* of them?

Set of *all* functions from $\{1, 2\}$ to $\{a, b\}$ is just the set of 2-element lists from $\{a, b\}$ which we already saw, by the product principle, has size $2 \times 2 = 4$.

How many functions are there from a two-element set to a three-element set?

How many functions are there from a two-element set to a three-element set?

Set of all functions is union of three sets,
one for each choice of $f_i(1)$.

How many functions are there from a two-element set to a three-element set?

Set of all functions is union of three sets,
one for each choice of $f_i(1)$.

Each of these sets has three elements,
one for each choice of $f_i(2)$.

How many functions are there from a two-element set to a three-element set?

Set of all functions is union of three sets,
one for each choice of $f_i(1)$.

Each of these sets has three elements,
one for each choice of $f_i(2)$.

Thus, by product principle,
we have $3 \cdot 3 = 9$ functions.

How many functions are there from a two-element set to a three-element set?

Set of all functions is union of three sets,
one for each choice of $f_i(1)$.

Each of these sets has three elements,
one for each choice of $f_i(2)$.

Thus, by product principle,
we have $3 \cdot 3 = 9$ functions.

Note: This is the same as number of 2-element lists from a 3-element set

How many functions are there from a three-element set to a two-element set?

How many functions are there from a three-element set to a two-element set?

This is same as number of 3-element lists from a 2-item set, which is $2 \times 2 \times 2 = 8$.

A function f is **one-to-one**, or an **injection**, if $(x \neq y) \Rightarrow (f(x) \neq f(y))$.

A function f is **one-to-one**, or an **injection**, if
$(x \neq y) \Rightarrow (f(x) \neq f(y))$.

$$
\begin{array}{lll}
f_1(1) = a & f_1(2) = b & \checkmark \\
f_2(1) = b & f_2(2) = a & \checkmark \\
f_3(1) = a & f_3(2) = a & \times \\
f_4(1) = b & f_4(2) = b & \times
\end{array}
$$

A function f is **one-to-one**, or an **injection**, if $(x \neq y) \Rightarrow (f(x) \neq f(y))$.

$$\begin{array}{lll} f_1(1) = a & f_1(2) = b & \checkmark \\ f_2(1) = b & f_2(2) = a & \checkmark \\ f_3(1) = a & f_3(2) = a & \times \\ f_4(1) = b & f_4(2) = b & \times \end{array}$$

A function $f : X \to Y$ is **onto**, or a **surjection**, if every element y in range Y is $f(x)$ for some x in domain X.

A function f is **one-to-one**, or an **injection**, if $(x \neq y) \Rightarrow (f(x) \neq f(y))$.

$$\begin{array}{llll} f_1(1) = a & f_1(2) = b & \checkmark \\ f_2(1) = b & f_2(2) = a & \checkmark \\ f_3(1) = a & f_3(2) = a & \times \\ f_4(1) = b & f_4(2) = b & \times \end{array}$$

A function $f : X \to Y$ is **onto**, or a **surjection**, if every element y in range Y is $f(x)$ for some x in domain X.

f_1 and f_2 are surjections

f_3 and f_4 are not surjections

A function f is **one-to-one**, or an **injection**, if $(x \neq y) \Rightarrow (f(x) \neq f(y))$.

$$\begin{array}{llc} f_1(1) = a & f_1(2) = b & \checkmark \\ f_2(1) = b & f_2(2) = a & \checkmark \\ f_3(1) = a & f_3(2) = a & \times \\ f_4(1) = b & f_4(2) = b & \times \end{array}$$

A function $f : X \to Y$ is **onto**, or a **surjection**, if every element y in range Y is $f(x)$ for some x in domain X.

In French *sur* $=$ "on"

f_1 and f_2 are surjections
f_3 and f_4 are not surjections

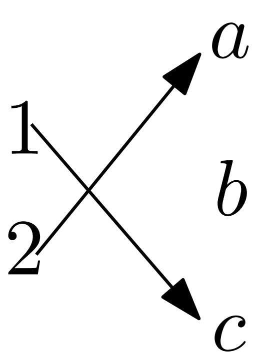
Using 2- or 3-element sets as domains & ranges, find an example of a *one-to-one* function that is not *onto*.

Using 2- or 3-element sets as domains & ranges, find an example of a *one-to-one* function that is not *onto*.

Function $f : \{1, 2\} \to \{a, b, c\}$
given by $f(1) = c$, $f(2) = a$.

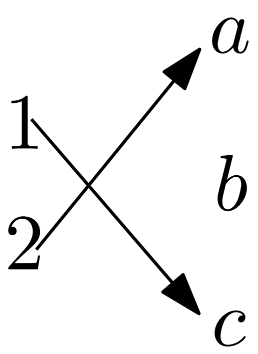
Using 2- or 3-element sets as domains & ranges, find an example of a *one-to-one* function that is not *onto*.

Function $f : \{1, 2\} \to \{a, b, c\}$
given by $f(1) = c$, $f(2) = a$.



Using 2- or 3-element sets as domains & ranges, find an example of a *one-to-one* function that is not *onto*.

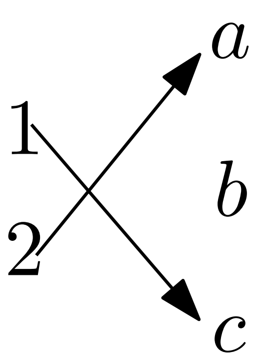
Function $f : \{1, 2\} \to \{a, b, c\}$
given by $f(1) = c$, $f(2) = a$.



Using 2- or 3-element sets as domains & ranges, find an example of a *onto* function that is not *one-to-one*.

Using 2- or 3-element sets as domains & ranges, find an example of a *one-to-one* function that is not *onto*.

Function $f : \{1, 2\} \to \{a, b, c\}$
given by $f(1) = c$, $f(2) = a$.

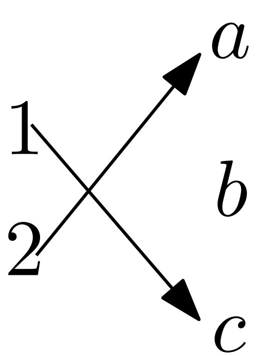


Using 2- or 3-element sets as domains & ranges, find an example of a *onto* function that is not *one-to-one*.

Function $f : \{1, 2, 3\} \to \{a, b\}$
given by $f(1) = a$, $f(2) = b$, $f(3) = a$.

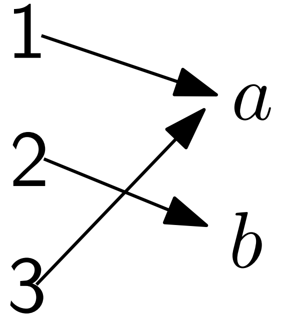
Using 2- or 3-element sets as domains & ranges, find an example of a *one-to-one* function that is not *onto*.

Function $f : \{1, 2\} \to \{a, b, c\}$
given by $f(1) = c$, $f(2) = a$.



Using 2- or 3-element sets as domains & ranges, find an example of a *onto* function that is not *one-to-one*.

Function $f : \{1, 2, 3\} \to \{a, b\}$
given by $f(1) = a$, $f(2) = b$, $f(3) = a$.



Bijections & Permutations

A function that is both one-to-one and onto is called a **bijection**, or a **one-to-one correspondence**.

A bijection from a set onto itself is called a **permutation**

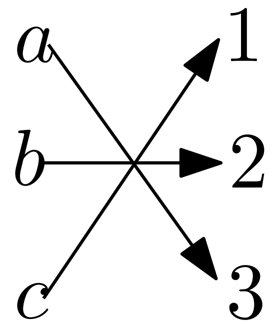
$f : \{a, b, c\} \to \{1, 2, 3\}$ defined by
$f(a) = 3, f(b) = 2, f(c) = 1$
is a bijection.

Bijections & Permutations

A function that is both one-to-one and onto is called a **bijection**, or a **one-to-one correspondence**.

A bijection from a set onto itself is called a **permutation**

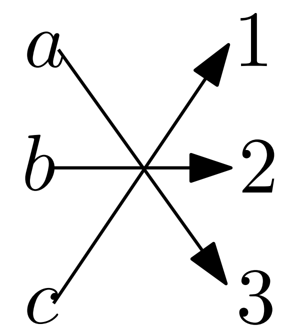
$f : \{a, b, c\} \to \{1, 2, 3\}$ defined by $f(a) = 3, f(b) = 2, f(c) = 1$ is a bijection.



Bijections & Permutations

A function that is both one-to-one and onto is called
a **bijection**, or a **one-to-one correspondence**.
A bijection from a set onto itself is called a **permutation**

$f : \{a, b, c\} \to \{1, 2, 3\}$ defined by
$f(a) = 3, f(b) = 2, f(c) = 1$
is a bijection.



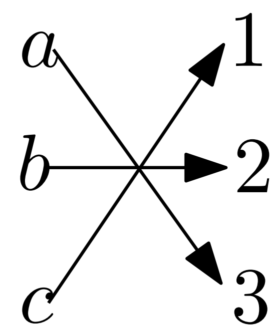
$f : \{1, 2, 3\} \to \{1, 2, 3\}$ defined by
$f(1) = 3,\ f(2) = 2, f(3) = 1$
is a permutation.

Bijections & Permutations

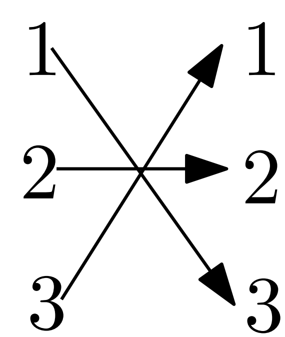
A function that is both one-to-one and onto is called a **bijection**, or a **one-to-one correspondence**.

A bijection from a set onto itself is called a **permutation**

$f : \{a, b, c\} \to \{1, 2, 3\}$ defined by
$f(a) = 3, f(b) = 2, f(c) = 1$
is a bijection.



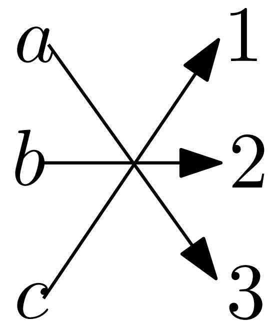
$f : \{1, 2, 3\} \to \{1, 2, 3\}$ defined by
$f(1) = 3,\ f(2) = 2, f(3) = 1$
is a permutation.

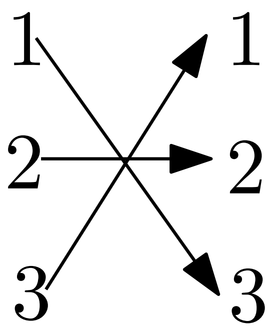


Bijections & Permutations

A function that is both one-to-one and onto is called
a **bijection**, or a **one-to-one correspondence**.
A bijection from a set onto itself is called a **permutation**

In a bijection
exactly one arrow
leaves each item on the left
and
exactly one arrow
arrives at each item on the right
so the left and right sides
must have the same size





The Bijection Principle

The Bijection Principle

The following loop is a part of program to determine the number of triangles formed by n points in the plane.

The Bijection Principle

The following loop is a part of program to determine the number of triangles formed by n points in the plane.

```
triangleCount = 0
  for i = 1 to n
    for j = i+1 to n
      for k = j+1 to n
        if points i, j, k are not collinear
          triangleCount = triangleCount + 1
```

The Bijection Principle

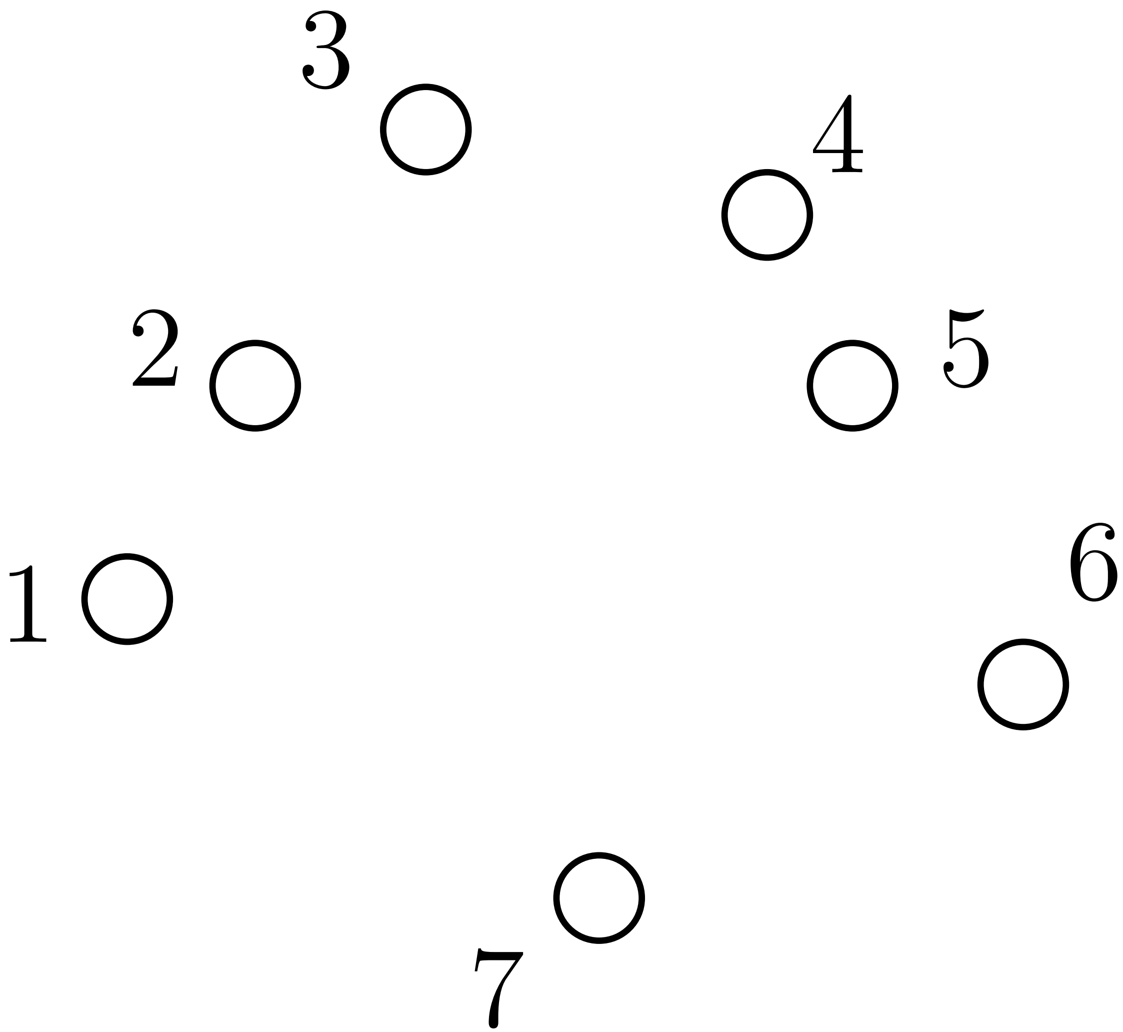
The following loop is a part of program to determine the number of triangles formed by n points in the plane.

```
(1) trianglecount = 0
(2)   for i = 1 to n
(3)     for j = i+1 to n
(4)       for k = j+1 to n
(5)         if points i, j, k are not collinear
(6)           trianglecount = trianglecount + 1
```

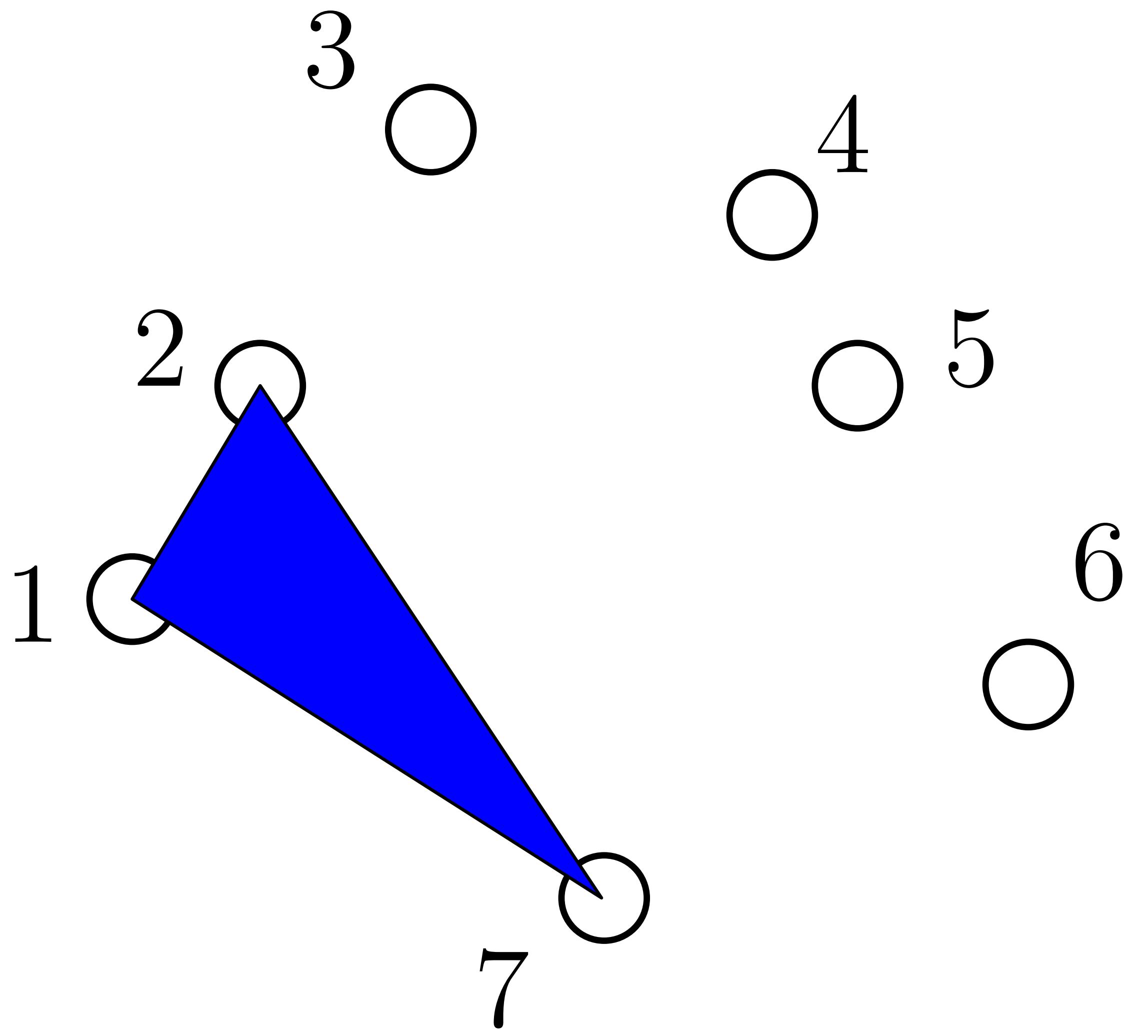
Among all iterations of line 5 in the pseudocode, what is the total number of times this line checks three points to see if they are collinear?

Counting Triangles: 3 points form a triangle if and only if they are not collinear (on a straight line)

Counting Triangles: 3 points form a triangle if and only if they are not collinear (on a straight line)

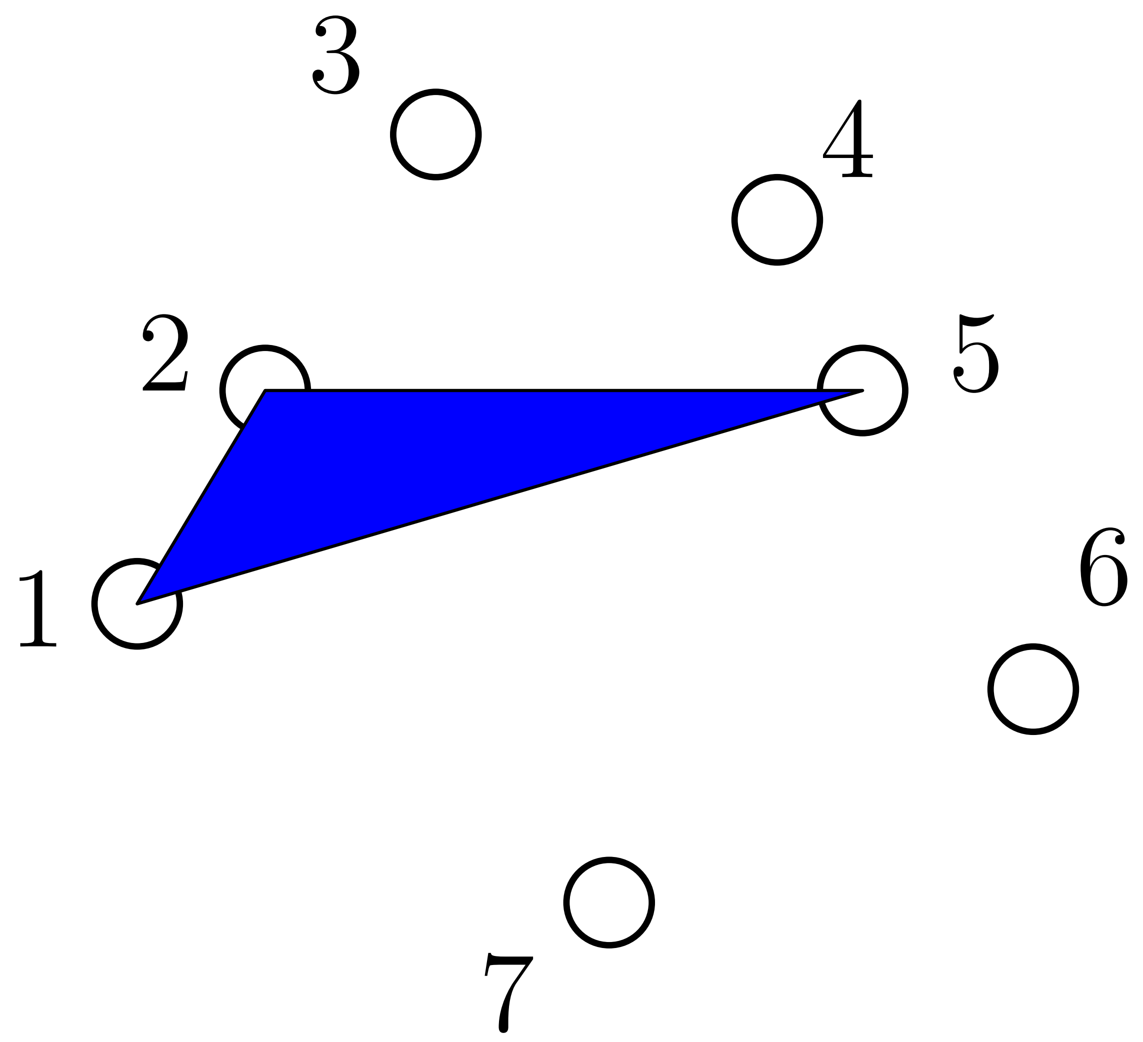


Counting Triangles: 3 points form a triangle if and only if they are not collinear (on a straight line)



$1 - 2 - 7$: yes

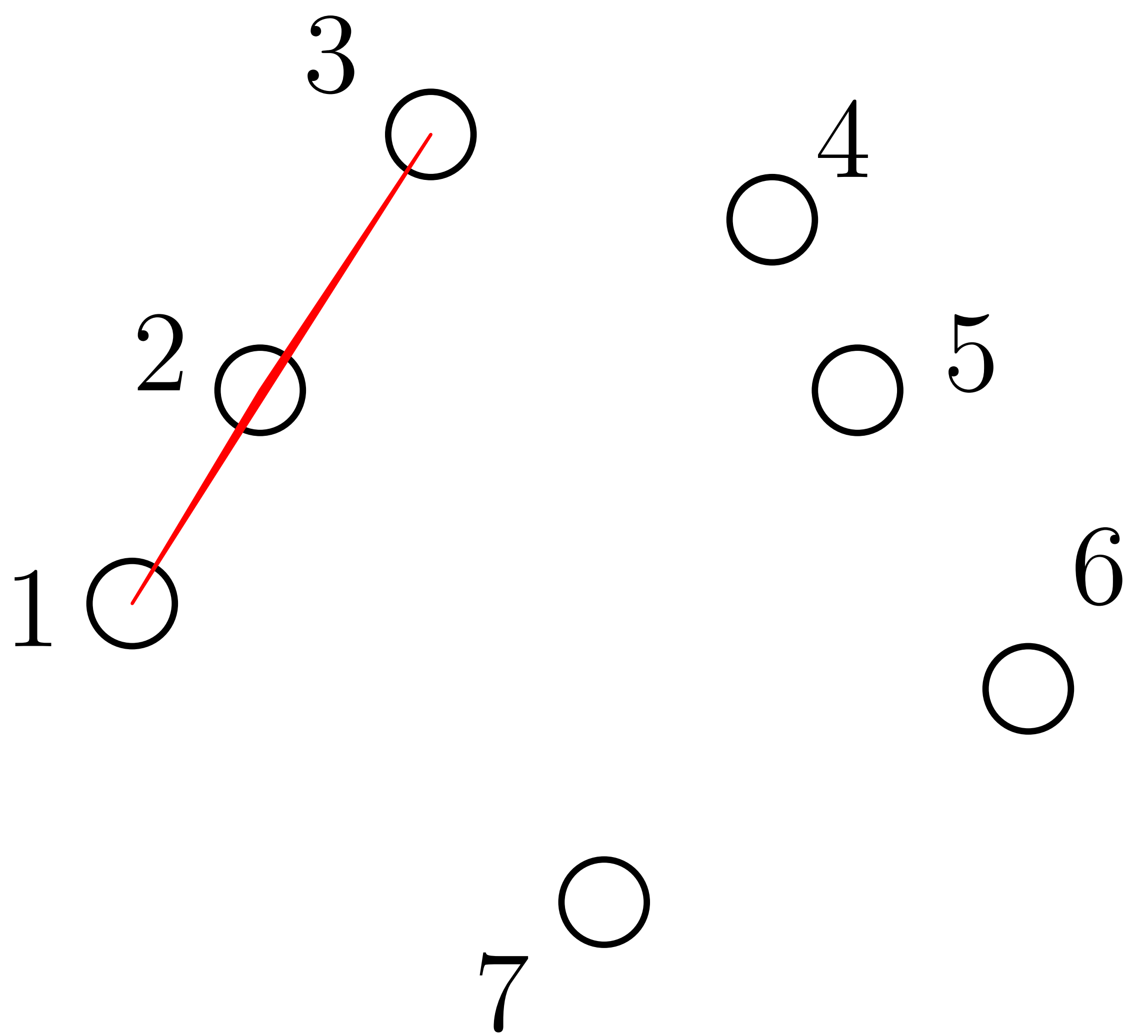
Counting Triangles: 3 points form a triangle if and only if they are not collinear (on a straight line)



$1 - 2 - 7$: yes

$1 - 2 - 5$: yes

Counting Triangles: 3 points form a triangle if and only if they are not collinear (on a straight line)

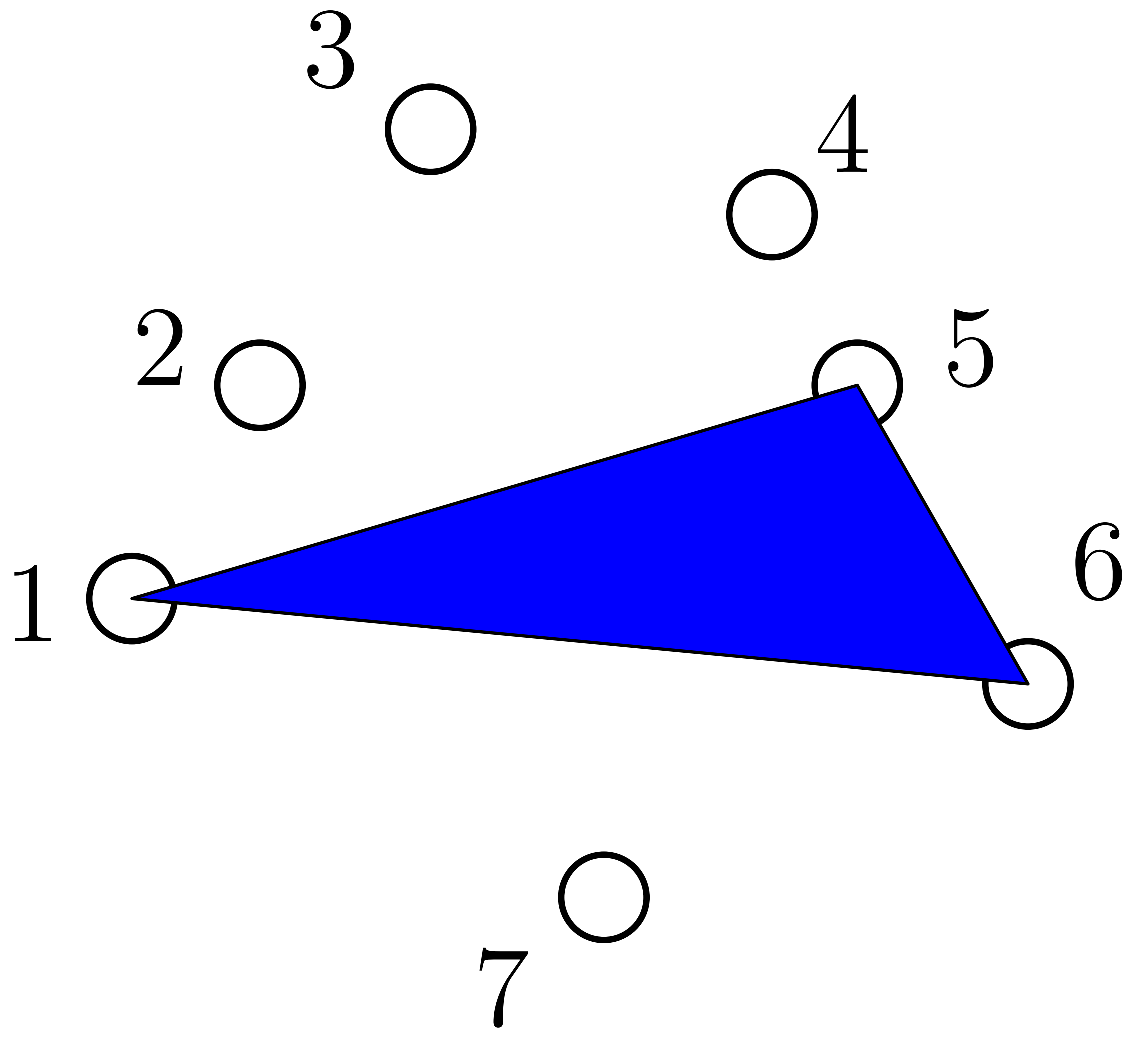


$1 - 2 - 7$: yes

$1 - 2 - 5$: yes

$1 - 2 - 3$: no

Counting Triangles: 3 points form a triangle if and only if they are not collinear (on a straight line)



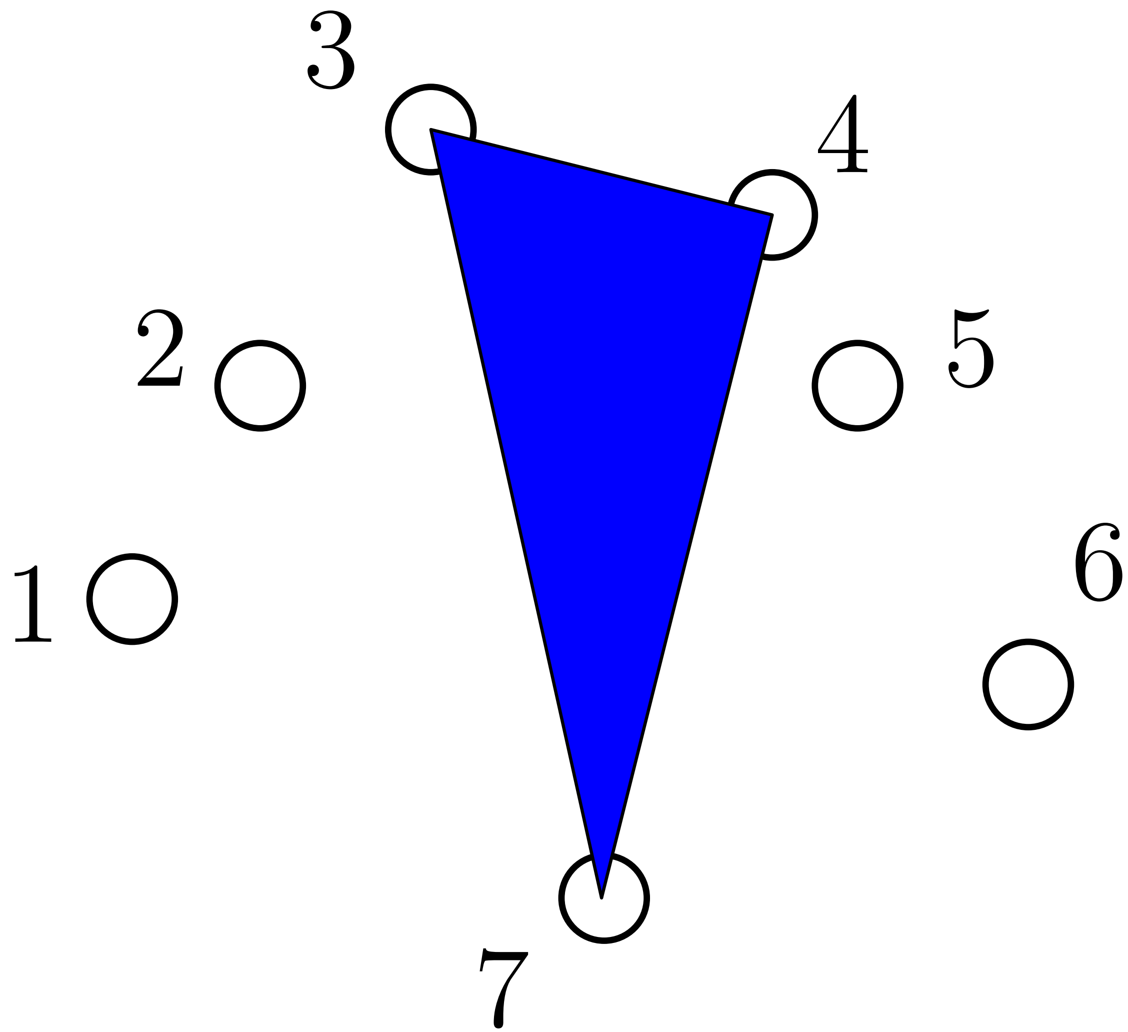
$1 - 2 - 7$: yes

$1 - 2 - 5$: yes

$1 - 2 - 3$: no

$1 - 5 - 6$: yes

Counting Triangles: 3 points form a triangle if and only if they are not collinear (on a straight line)



$1 - 2 - 7$: yes

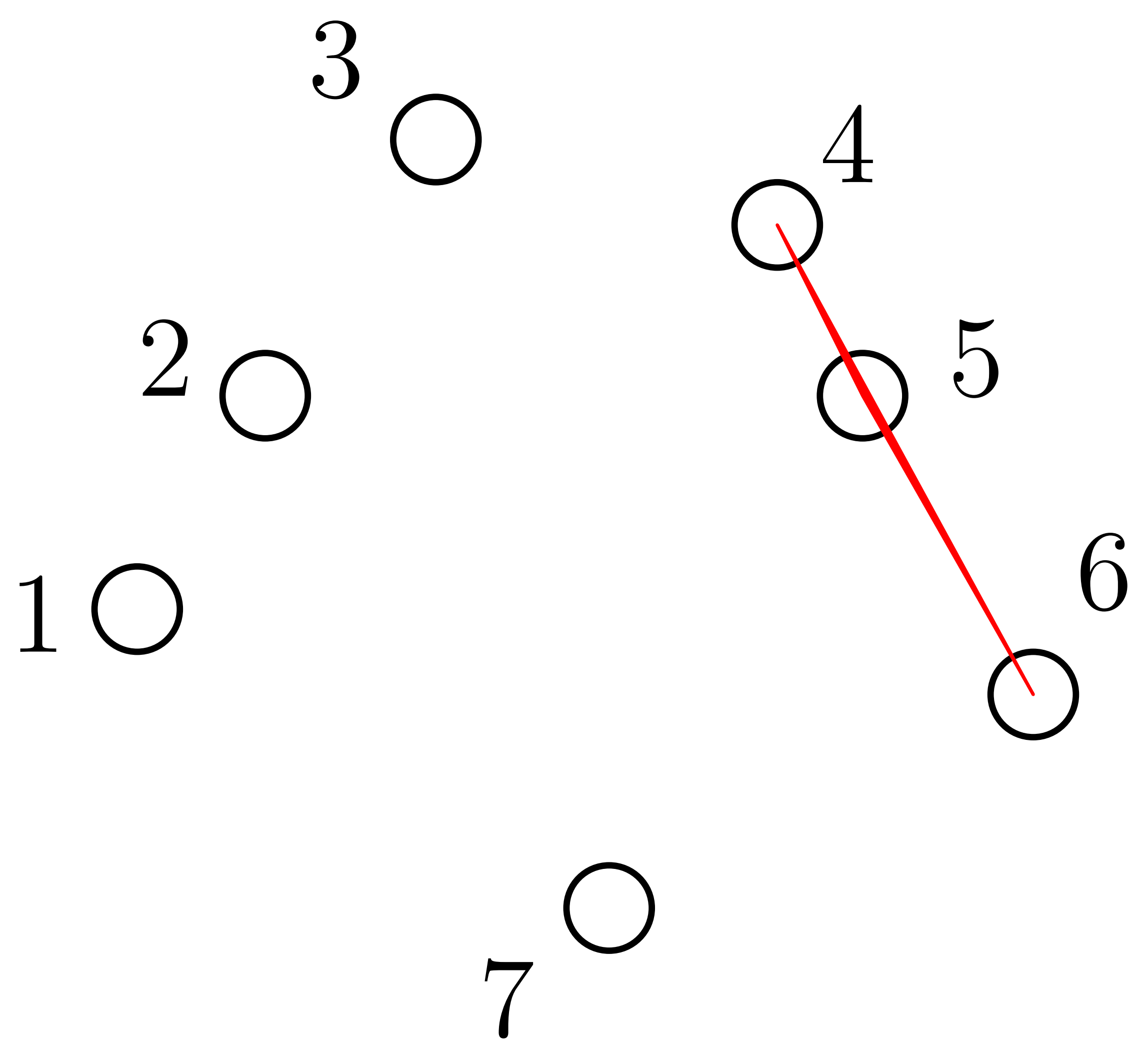
$1 - 2 - 5$: yes

$1 - 2 - 3$: no

$1 - 5 - 6$: yes

$3 - 4 - 7$: yes

Counting Triangles: 3 points form a triangle if and only if they are not collinear (on a straight line)



$1-2-7$: yes

$1-2-5$: yes

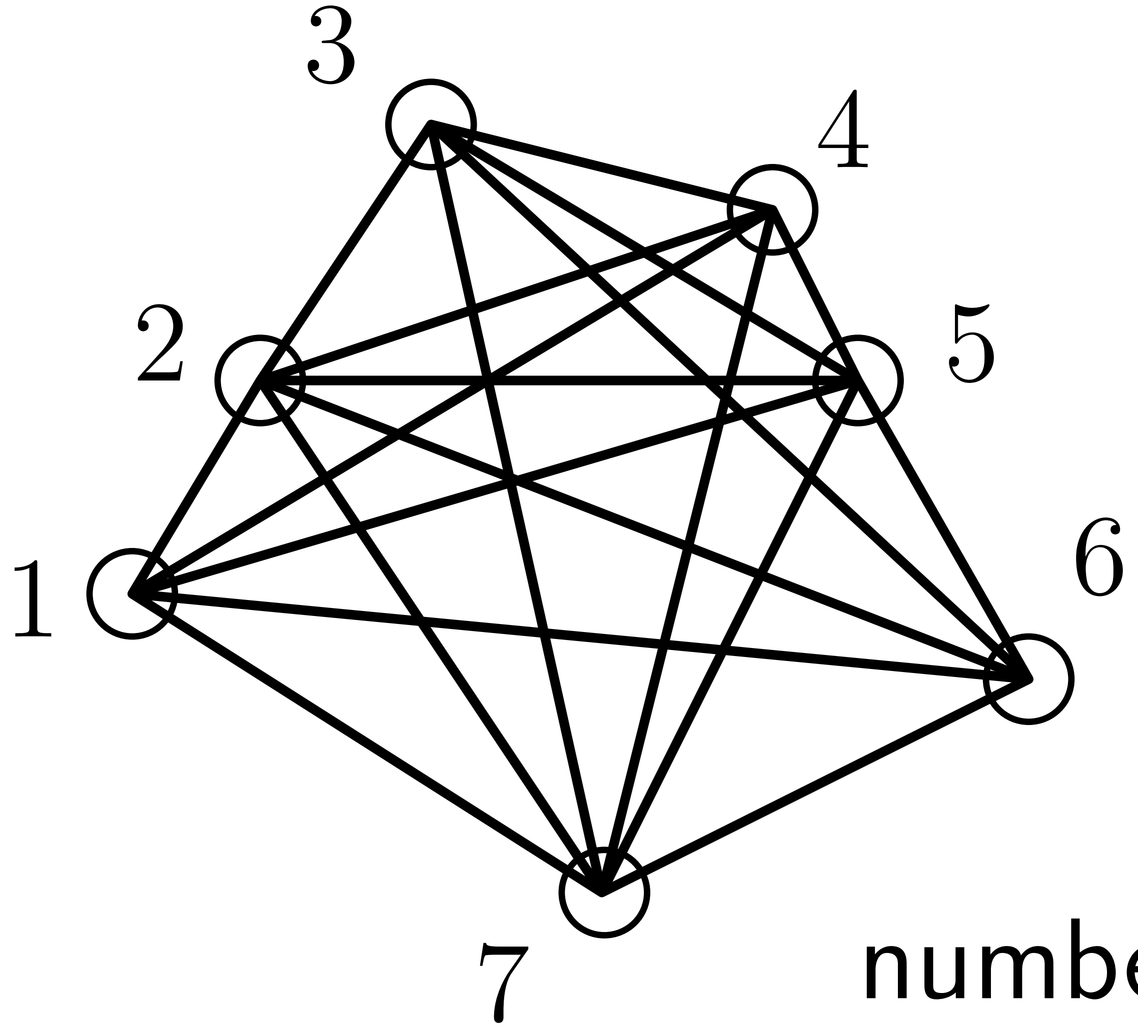
$1-2-3$: no

$1-5-6$: yes

$3-4-7$: yes

$4-5-6$: no

Counting Triangles: 3 points form a triangle if and only if they are not collinear (on a straight line)



$1 - 2 - 7$: yes

$1 - 2 - 5$: yes

$1 - 2 - 3$: no

$1 - 5 - 6$: yes

$3 - 4 - 7$: yes

$4 - 5 - 6$: no

number of triangles: 33

```
trianglecount = 0
  for i = 1 to n
    for j = i+1 to n
      for k = j+1 to n
        if points i, j, k are not collinear
          trianglecount = trianglecount + 1
```

```
trianglecount = 0
  for i = 1 to n
    for j = i+1 to n
      for k = j+1 to n
        if points i, j, k are not collinear
          trianglecount = trianglecount + 1
```

A loop

```
triangleccount = 0
  for i = 1 to n
    for j = i+1 to n
      for k = j+1 to n
        if points i, j, k are not collinear
          trianglecount = trianglecount + 1
```

A loop embedded in a loop

```
triangleCount = 0
  for i = 1 to n
    for j = i+1 to n
      for k = j+1 to n
        if points i, j, k are not collinear
          triangleCount = triangleCount + 1
```

A loop embedded in a loop embedded in another loop.

```
triangleCount = 0
  for i = 1 to n
    for j = i+1 to n
      for k = j+1 to n
        if points i, j, k are not collinear
          triangleCount = triangleCount + 1
```

A loop embedded in a loop embedded in another loop.

Second loop begins with $j = i + 1$ and j increases up to n.

```
triangleecount = 0
  for i = 1 to n
    for j = i+1 to n
      for k = j+1 to n
        if points i, j, k are not collinear
          trianglecount = trianglecount + 1
```

A loop embedded in a loop embedded in another loop.

Second loop begins with $j = i + 1$ and j increases up to n.
Third loop begins with $k = j + 1$ and k increases up to n.

```
trianglecount = 0
  for i = 1 to n
    for j = i+1 to n
      for k = j+1 to n
        if points i, j, k are not collinear
          trianglecount = trianglecount + 1
```

A loop embedded in a loop embedded in another loop.

Second loop begins with $j = i + 1$ and j increases up to n.
Third loop begins with $k = j + 1$ and k increases up to n.

Thus each triple i, j, k with $i < j < k$ is examined exactly once.

```
trianglecount = 0
  for i = 1 to n
    for j = i+1 to n
      for k = j+1 to n
        if points i, j, k are not collinear
          trianglecount = trianglecount + 1
```

A loop embedded in a loop embedded in another loop.

Second loop begins with $j = i + 1$ and j increases up to n.

Third loop begins with $k = j + 1$ and k increases up to n.

Thus each triple i, j, k with $i < j < k$ is examined exactly once.

For example, if $n = 4$, then triples (i, j, k) used by algorithm are (1,2,3), (1,2,4), (1,3,4), and (2,3,4).

```
trianglecount = 0
  for i = 1 to n
    for j = i+1 to n
      for k = j+1 to n
        if points i, j, k are not collinear
          trianglecount = trianglecount + 1
```

A loop embedded in a loop embedded in another loop.

Second loop begins with $j = i + 1$ and j increases up to n.
Third loop begins with $k = j + 1$ and k increases up to n.

Thus each triple $\boxed{i, j, k \text{ with } i < j < k}$ is examined exactly once.

For example, if $n = 4$, then triples (i, j, k) used by algorithm are (1,2,3), (1,2,4), (1,3,4), and (2,3,4).

Thus, compute number of increasing triples!

- Want to compute the number of
  increasing triples (i, j, k) with $1 \leq i < j < k \leq n$.

- Want to compute the number of
  increasing triples (i, j, k) with $1 \le i < j < k \le n$.
- Claim: Number of increasing triples is exactly the same as number of 3-element subsets from $\{1, 2, \ldots, n\}$

- Want to compute the number of *increasing triples* (i, j, k) with $1 \le i < j < k \le n$.
- Claim: Number of increasing triples is exactly the same as number of 3-element subsets from $\{1, 2, \ldots, n\}$
- Why? Let $X =$ set of increasing triples and $Y =$ set of 3-element subsets from $\{1, 2, \ldots, n\}$

- Want to compute the number of *increasing triples* (i, j, k) with $1 \le i < j < k \le n$.
- Claim: Number of increasing triples is exactly the same as number of 3-element subsets from $\{1, 2, \ldots, n\}$
- Why? Let $X =$ set of increasing triples and $Y =$ set of 3-element subsets from $\{1, 2, \ldots, n\}$
- Define $f : X \to Y$ by $f(\,(i, j, k)\,) = \{i, j, k\}$
  Claim: f is a **bijection** (why) so $|X| = |Y|$

- Want to compute the number of *increasing triples* (i, j, k) with $1 \leq i < j < k \leq n$.

- Claim: Number of increasing triples is exactly the same as number of 3-element subsets from $\{1, 2, \ldots, n\}$

- Why? Let $X =$ set of increasing triples and $Y =$ set of 3-element subsets from $\{1, 2, \ldots, n\}$

- Define $f : X \to Y$ by $f(\,(i, j, k)\,) = \{i, j, k\}$
  Claim: f is a **bijection** (why) so $|X| = |Y|$

  f is a bijection because
  f is one-to-one
  if $(i, j, k) \neq (i', j', k') \Rightarrow f(\,(i, j, k)\,) \neq f(\,(i', j', k'))$
  f is onto
  if γ is a 3-element subset then it can be written as $\gamma = \{i, j, k\}$
  where $i < j < k$ so $f(\,(i, j, k)\,) = \gamma$.

- We've already seen something very similar.
  The number of
  increasing pairs (i, j) with $1 \leq i < j \leq n$
  is the same as the number of
  2-sets from $\{1, 2, \ldots, n\}$

- We've already seen something very similar.
  The number of
  increasing pairs (i, j) with $1 \leq i < j \leq n$
  is the same as the number of
  2-sets from $\{1, 2, \ldots, n\}$

- Define $f : X \to Y$ by $f(\,(i, j)\,) = \{i, j\}$
  Claim: f is a **bijection** so $|X| = |Y|$

- We've already seen something very similar.
  The number of
  increasing pairs (i, j) with $1 \le i < j \le n$
  is the same as the number of
  2-sets from $\{1, 2, \ldots, n\}$

- Define $f : X \to Y$ by $f(\,(i, j)\,) = \{i, j\}$
  Claim: f is a **bijection** so $|X| = |Y|$

- We actually already saw that $|X| = |Y| = \binom{n}{2}$

(Bijection Principle)
Two sets have the same size if and only if there is a one-to-one function from one set onto the other.

(Bijection Principle)
Two sets have the same size if and only if there is a one-to-one function from one set onto the other.

- A standard first step in counting the size of a set is to use a bijection to show that it has the same size as a 2nd set, and then count the 2nd set instead.

(Bijection Principle)
Two sets have the same size if and only if there is a one-to-one function from one set onto the other.

- A standard first step in counting the size of a set is to use a bijection to show that it has the same size as a 2nd set, and then count the 2nd set instead.

- In practice, in real problems we often only *implicitly* use the bijection and don't *explicitly* describe it

(Bijection Principle)
Two sets have the same size if and only if there is a one-to-one function from one set onto the other.

- A standard first step in counting the size of a set is to use a bijection to show that it has the same size as a 2nd set, and then count the 2nd set instead.

- In practice, in real problems we often only *implicitly* use the bijection and don't *explicitly* describe it

- Currently, we started with the problem of counting the $\#$ of increasing triples and changed it to the problem of counting the $\#$ of 3-element sets from $\{1, 2, \ldots, n\}$

(Bijection Principle)
Two sets have the same size if and only if there is a one-to-one function from one set onto the other.

- A standard first step in counting the size of a set is to use a bijection to show that it has the same size as a 2nd set, and then count the 2nd set instead.

- In practice, in real problems we often only *implicitly* use the bijection and don't *explicitly* describe it

- Currently, we started with the problem of counting the $\#$ of increasing triples and changed it to the problem of counting the $\#$ of 3-element sets from $\{1, 2, \ldots, n\}$

- We will now see how to count the $\#$ of k-element permutations of $\{1, 2, \ldots, n\}$. From this we will derive how to count $\#$ of k-element sets.

k-Element Permutations of a Set

k-Element Permutations of a Set

- In how many ways can we choose an ordered triple of distinct elements from $\{1, 2, \ldots, n\}$?

k-Element Permutations of a Set

- In how many ways can we choose an ordered triple of distinct elements from $\{1, 2, \ldots, n\}$?

- More generally, in how many ways can we choose a list of k distinct elements from $\{1, 2, \ldots, n\}$?

k-Element Permutations of a Set

- In how many ways can we choose an ordered triple of distinct elements from $\{1, 2, \ldots, n\}$?

- More generally, in how many ways can we choose a list of k distinct elements from $\{1, 2, \ldots, n\}$?

- A list of k *distinct* elements chosen from a set N is called a k-**element permutation of** N.

k-Element Permutations of a Set

- In how many ways can we choose an ordered triple of distinct elements from $\{1, 2, \ldots, n\}$?

- More generally, in how many ways can we choose a list of k distinct elements from $\{1, 2, \ldots, n\}$?

- A list of k *distinct* elements chosen from a set N is called a k-**element permutation of** N.

- Note that the case of $k = n$ is special;
  An n-element permutation of a set N of size $|N| = n$ is what we earlier simply called a permutation.

k-Element Permutations of a Set

- In how many ways can we choose an ordered triple of distinct elements from $\{1, 2, \ldots, n\}$?

- More generally, in how many ways can we choose a list of k distinct elements from $\{1, 2, \ldots, n\}$?

- A list of k *distinct* elements chosen from a set N is called a k-**element permutation of** N.

  - To start, how many 3-element permutations of $\{1, 2, \ldots, n\}$ are there?

How many three-element permutations of $\{1, 2, \ldots, n\}$ are there?

How many three-element permutations of $\{1, 2, \ldots, n\}$ are there?

- n choices for first number

How many three-element permutations of $\{1, 2, \ldots, n\}$ are there?

- n choices for first number
  - For each way of choosing first number there are $n - 1$ choices for the second.

How many three-element permutations of $\{1, 2, \ldots, n\}$ are there?

- n choices for first number
  - For each way of choosing first number there are $n - 1$ choices for the second.
    - For each way of choosing first two numbers, there are $n - 2$ choices for the third number.

How many three-element permutations of $\{1, 2, \ldots, n\}$ are there?

- n choices for first number
  - For each way of choosing first number there are $n-1$ choices for the second.
    - For each way of choosing first two numbers, there are $n-2$ choices for the third number.

By version 2 of product principle, there are $n(n-1)(n-2)$ ways to choose the permutation.

By version 2 of product principle, there are $n(n-1)(n-2)$ ways to choose the permutation

Ex: When $n = 4$, there are $4 \times 3 \times 2 = 24$
3 -element permutations of $\{1, 2, 3, 4\}$

$$L = \{123, 124, 132, 134, 142, 143, 213, 214, 231, 234, 241, 243$$
$$312, 314, 321, 324, 341, 342, 412, 413, 421, 423, 431, 432\}.$$

By version 2 of product principle, there are $n(n-1)(n-2)$ ways to choose the permutation

Ex: When $n=4$, there are $4 \times 3 \times 2 = 24$ 3 -element permutations of $\{1,2,3,4\}$

$$L = \{123, 124, 132, 134, 142, 143, 213, 214, 231, 234, 241, 243\\ 312, 314, 321, 324, 341, 342, 412, 413, 421, 423, 431, 432\}.$$

Note: This type of "dictionary" ordering of tuples (assuming that we treat numbers the same as letters) is called a **lexicographic ordering** and is used quite often.

- We just saw that there are $n(n-1)(n-2)$ ways to choose a 3-element permutation from $\{1, 2, \ldots, n\}$

- We just saw that there are $n(n-1)(n-2)$ ways to choose a 3-element permutation from $\{1, 2, \ldots, n\}$

- How does this help us solve our original problem(from triangle program) of counting $\#$ of 3-element subsets?

- We just saw that there are $n(n-1)(n-2)$ ways to choose a 3-element permutation from $\{1, 2, \ldots, n\}$
- How does this help us solve our original problem(from triangle program) of counting $\#$ of 3-element subsets?
- Note that every 3-element subset $\{i, j, k\}$ can be made into exactly 6 3-element perms.

- We just saw that there are $n(n-1)(n-2)$ ways to choose a 3-element permutation from $\{1, 2, \ldots, n\}$
- How does this help us solve our original problem(from triangle program) of counting $\#$ of 3-element subsets?
- Note that every 3-element subset $\{i, j, k\}$ can be made into exactly 6 3-element perms.

$$\boxed{\begin{array}{ccc} (1,2,3) & (1,3,2) & (2,1,3) \\ (2,3,1) & (3,1,2) & (3,2,1) \end{array}}$$

- We just saw that there are $n(n-1)(n-2)$ ways to choose a 3-element permutation from $\{1, 2, \ldots, n\}$
- How does this help us solve our original problem(from triangle program) of counting $\#$ of 3-element subsets?
- Note that every 3-element subset $\{i, j, k\}$ can be made into exactly 6 3-element perms.

$(1,2,3)$	$(1,3,2)$	$(2,1,3)$
$(2,3,1)$	$(3,1,2)$	$(3,2,1)$

- $\Rightarrow$ ($\#$ 3-element perms) $=$ 6 $\times$ ($\#$ 3-element subsets)

- We just saw that there are $n(n-1)(n-2)$ ways to choose a 3-element permutation from $\{1, 2, \dots, n\}$

- How does this help us solve our original problem(from triangle program) of counting # of 3-element subsets?

- Note that every 3-element subset $\{i, j, k\}$ can be made into exactly 6 3-element perms.

$$\boxed{\begin{array}{ccc} (1,2,3) & (1,3,2) & (2,1,3) \\ (2,3,1) & (3,1,2) & (3,2,1) \end{array}}$$

- $\Rightarrow$ (# 3-element perms) $=$ 6 $\times$ (# 3-element subsets)

- Let $\binom{n}{3}$ denote # of 3-element subsets

- We just saw that there are $n(n-1)(n-2)$ ways to choose a 3-element permutation from $\{1, 2, \ldots, n\}$
- How does this help us solve our original problem(from triangle program) of counting # of 3-element subsets?
- Note that every 3-element subset $\{i, j, k\}$ can be made into exactly 6 3-element perms.

$$\boxed{\begin{array}{ccc} (1,2,3) & (1,3,2) & (2,1,3) \\ (2,3,1) & (3,1,2) & (3,2,1) \end{array}}$$

- $\Rightarrow$ (# 3-element perms) $=$ 6 $\times$ (# 3-element subsets)
- Let $\binom{n}{3}$ denote # of 3-element subsets

$$\Rightarrow \boxed{\frac{n(n-1)(n-2)}{6} = \binom{n}{3}}$$

General formula to compute the number of k-element permutations of the set $\{1, 2, \ldots, n\}$

General formula to compute the number of k-element permutations of the set $\{1, 2, \ldots, n\}$

- n choices for first element.

General formula to compute the number of
k-element permutations of the set $\{1, 2, \ldots, n\}$

- n choices for first element.
  - Independent of first choice, we have
    $n - 1$ choices for second number

General formula to compute the number of
k-element permutations of the set $\{1, 2, \ldots, n\}$

- n choices for first element.
  - Independent of first choice, we have
    $n - 1$ choices for second number
    - Independent of first and 2nd choices, we have
      $n - 2$ choices for third number

General formula to compute the number of k-element permutations of the set $\{1, 2, \ldots, n\}$

- n choices for first element.
  - Independent of first choice, we have $n-1$ choices for second number
    - Independent of first and 2nd choices, we have $n-2$ choices for third number
      - Given the first $i-1$ elements, we always have exactly $n-(i-1) = n-i+1$ choices for the ith element.

General formula to compute the number of
k-element permutations of the set $\{1, 2, \ldots, n\}$

- n choices for first element.
  - Independent of first choice, we have
    $n - 1$ choices for second number
    - Independent of first and 2nd choices, we have
      $n - 2$ choices for third number
      - Given the first $i-1$ elements, we always have exactly
        $n - (i - 1) = n - i + 1$ choices for the ith element.
        - $\cdots$

General formula to compute the number of
k-element permutations of the set $\{1, 2, \ldots, n\}$

- n choices for first element.
  - Independent of first choice, we have $n-1$ choices for second number
    - Independent of first and 2nd choices, we have $n-2$ choices for third number
      - Given the first $i-1$ elements, we always have exactly $n-(i-1) = n-i+1$ choices for the ith element.
        - $\cdots$

Thus, by version 2 of product principle, there are $n(n-1)\cdots(n-k+1)$ ways to choose a k-element permutation.

We just saw that there are

$$n(n-1)\cdots(n-k+1) = \prod_{i=0}^{k-1}(n-i)$$

ways to choose a k-element permutation.

In the special case of $k = n$ (a permutation) this reduces to

$$n(n-1)\cdots 3 \cdot 2 \cdot 1 = n!$$

So there are $n!$ different permutations of a set of size n.

Handy notation suggested by Donald E. Knuth, is $n^{\underline{k}}$, the k**th falling factorial power of** n, defined as

$$n^{\underline{k}} = n(n-1)\cdots(n-k+1) = \prod_{i=0}^{k-1}(n-i).$$

Handy notation suggested by Donald E. Knuth, is $n^{\underline{k}}$, the k**th falling factorial power of** n, defined as

$$n^{\underline{k}} = n(n-1)\cdots(n-k+1) = \prod_{i=0}^{k-1}(n-i).$$

Theorem 1.1

The #of k-element permutations of an n-element set is

$$n^{\underline{k}} = \frac{n!}{(n-k)!}.$$

Handy notation suggested by Donald E. Knuth, is $n^{\underline{k}}$, the k**th falling factorial power of** n, defined as

$$n^{\underline{k}} = n(n-1)\cdots(n-k+1) = \prod_{i=0}^{k-1}(n-i).$$

Donald Ervin Knuth (born 1938) is *Professor Emeritus of the Art of Computer Programming* at Stanford University. He is most famous for his 3+-volume set, *The Art of Computer Programming* but he's done many other things, including designing the system, TeX, with which these notes were typeset.

- Have seen that # of k-element *permutations* is $n^{\underline{k}}$.

- Have seen that # of k-element *permutations* is $n^{\underline{k}}$.
- What does this say about # of k-element *subsets*?

- Have seen that # of k-element *permutations* is $n^{\underline{k}}$.

- What does this say about # of k-element *subsets*?

- Let $\binom{n}{k}$ denote the # of k-element subsets
  The set of all k-element permutations of $\{1, 2, \ldots, n\}$
  can be *partitioned* into $\binom{n}{k}$ disjoint blocks,
  Each block comprises all k-element permutations
  of the same k-element subset.

- Have seen that # of k-element *permutations* is $n^{\underline{k}}$.

- What does this say about # of k-element *subsets*?

- Let $\binom{n}{k}$ denote the # of k-element subsets
  The set of all k-element permutations of $\{1, 2, \ldots, n\}$
  can be *partitioned* into $\binom{n}{k}$ disjoint blocks,
  Each block comprises all k-element permutations
  of the same k-element subset.

- However, the number of k-element permutations of a k-element set is $k!$ (Theorem 1.1 with $n = k$).

- Have seen that # of k-element *permutations* is $n^{\underline{k}}$.
- What does this say about # of k-element *subsets*?
- Let $\binom{n}{k}$ denote the # of k-element subsets
  The set of all k-element permutations of $\{1, 2, \ldots, n\}$
  can be *partitioned* into $\binom{n}{k}$ disjoint blocks,
  Each block comprises all k-element permutations
  of the same k-element subset.
- However, the number of k-element permutations of a k-element set is $k!$ (Theorem 1.1 with $n = k$).
- Thus, by sum principle,

$$n^{\underline{k}} = (\#\text{ of blocks}) \times (\#\text{ in each block}) = \binom{n}{k} k!.$$

Theorem 1.2

For integers n and k with $0 \leq k \leq n$, the number of k-element subsets of an n-element set is

$$\binom{n}{k} = \frac{n^{\underline{k}}}{k!} = \frac{n!}{k!(n-k)!}.$$

Theorem 1.2

For integers n and k with $0 \le k \le n$, the number of k-element subsets of an n-element set is

$$\binom{n}{k} = \frac{n^{\underline{k}}}{k!} = \frac{n!}{k!(n-k)!}.$$

Proof:

Theorem 1.2
For integers n and k with $0 \leq k \leq n$, the number of k-element subsets of an n-element set is

$$\binom{n}{k} = \frac{n^{\underline{k}}}{k!} = \frac{n!}{k!(n-k)!}.$$

Proof:

We just proved this except for case $k = 0$.

Theorem 1.2

For integers n and k with $0 \leq k \leq n$, the number of k-element subsets of an n-element set is

$$\binom{n}{k} = \frac{n^{\underline{k}}}{k!} = \frac{n!}{k!(n-k)!}.$$

Proof:

We just proved this except for case $k = 0$.

The only subset of an n-element set of size zero is the empty set, so we have exactly one such subset and should have $\binom{n}{0} = 1$

Theorem 1.2

For integers n and k with $0 \le k \le n$, the number of k-element subsets of an n-element set is

$$\binom{n}{k} = \frac{n^{\underline{k}}}{k!} = \frac{n!}{k!(n-k)!}.$$

Proof:

We just proved this except for case $k = 0$.

The only subset of an n-element set of size zero is the empty set, so we have exactly one such subset and should have $\binom{n}{0} = 1$

This will work if we define $0! = 1$.

Note: Both cases $k = 0$ and $k = n$ use fact that $0! = 1$.

Binomial Coefficients

The number of ways of choosing a k-element subset from a set of size n is

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

This term is called a binomial coefficient.
Be aware that there are other, alternative, notations for the same thing, occasionally used, e.g., $C(n,k)$ or nCk