Today.

Finish up undecidability.

Today.

Finish up undecidability.

Counting.

Programs and Diagonalization.

Write me a program checker!

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P,I)$
  P - program

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
 P - program
 I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine!

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P,I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P,I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.
Text string can be an input to a program.

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P,I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.
Text string can be an input to a program.
Program can be an input to a program.

Programs and Diagonalization.

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P,I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.
Text string can be an input to a program.
Program can be an input to a program.

Implementing HALT.

Implementing HALT.

$HALT(P, I)$

Implementing HALT.

$HALT(P, I)$
  P - program

Implementing HALT.

$HALT(P, I)$
P - program
I - input.

Implementing HALT.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Implementing HALT.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Run P on I and check!

Implementing HALT.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Run P on I and check!

How long do you wait?

Implementing HALT.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Run P on I and check!

How long do you wait?

Something about infinity here, maybe?

Halt does not exist.

Halt does not exist.

$HALT(P, I)$

Halt does not exist.

$HALT(P, I)$

P - program

Halt does not exist.

$HALT(P, I)$

P - program

I - input.

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes!

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No!

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes!

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No!

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No!

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes!

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No!

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes!

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! ..

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

What is he talking about?

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

What is he talking about?
(A) He is confused.

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

What is he talking about?
(A) He is confused.
(B) Fermat's Theorem.

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

What is he talking about?
(A) He is confused.
(B) Fermat's Theorem.
(C) Diagonalization.

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

What is he talking about?
(A) He is confused.
(B) Fermat's Theorem.
(C) Diagonalization.

(C).

Halt and Turing.

Proof:

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Halt and Turing.

Proof: Assume there is a program $\mathit{HALT}(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) =“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that “is” the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Contradiction.

Halt and Turing.

Proof: Assume there is a program *HALT*$(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) =“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that “is” the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Contradiction. Program HALT does not exist!

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Contradiction. Program HALT does not exist! $\square$

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Contradiction. Program HALT does not exist! $\square$
Questions?

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Contradiction. Program HALT does not exist! $\square$
Questions?

Another view of proof: diagonalization.

Any program is a fixed length string.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Halt - diagonal.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Halt - diagonal.
Turing - is not Halt.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

$$
\begin{array}{c|cccc}
 & P_1 & P_2 & P_3 & \cdots \\
\hline
 & & & & \\
P_1 & \color{blue}{H} & H & L & \cdots \\
P_2 & L & \color{blue}{L} & H & \cdots \\
P_3 & L & H & \color{blue}{H} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
$$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list. Turing is not a program.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list. Turing is not a program.
Turing can be constructed from Halt.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list. Turing is not a program.
Turing can be constructed from Halt.
Halt does not exist!

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

$$\begin{array}{c|cccc} & P_1 & P_2 & P_3 & \cdots \\ \hline \\ P_1 & \text{H} & \text{H} & \text{L} & \cdots \\ P_2 & \text{L} & \text{L} & \text{H} & \cdots \\ P_3 & \text{L} & \text{H} & \text{H} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array}$$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list. Turing is not a program.
Turing can be constructed from Halt.
Halt does not exist! $\square$

Discussion of Proof.

Undecidability:

Discussion of Proof.

Undecidability:
HALT(P) - does not exist.

Discussion of Proof.

Undecidability:
HALT(P) - does not exist.

Discussion of Proof.

Undecidability:
  HALT(P) - does not exist.

Why not?

Discussion of Proof.

Undecidability:
  HALT(P) - does not exist.

Why not?

Discussion of Proof.

Undecidability:
  HALT(P) - does not exist.

Why not?

Programs are text.

Discussion of Proof.

Undecidability:
 HALT(P) - does not exist.

Why not?

Programs are text.

Discussion of Proof.

Undecidability:
  HALT(P) - does not exist.

Why not?

Programs are text.

List programs,

Discussion of Proof.

Undecidability:
HALT(P) - does not exist.

Why not?

Programs are text.

List programs, Turing not in list of programs!

Discussion of Proof.

Undecidability:
  HALT(P) - does not exist.

Why not?

Programs are text.

List programs, Turing not in list of programs!

Argue directly by saying Turing(Turing) neither halts nor runs forever.

Discussion of Proof.

Undecidability:
  HALT(P) - does not exist.

Why not?

Programs are text.

List programs, Turing not in list of programs!

Argue directly by saying Turing(Turing) neither halts nor runs forever.

Really Same: Says there is no text which can be Turing.

Notes.

Fairly quick. Right?

Notes.

Fairly quick. Right?

No computers in early 20th century.

Notes.

Fairly quick. Right?

No computers in early 20th century.

Turings proof came up with the notion of computer.

Notes.

Fairly quick. Right?

No computers in early 20th century.

Turings proof came up with the notion of computer.

Also, the incompleteness theorems also designed a computer using arithmetic.

Notes.

Fairly quick. Right?

No computers in early 20th century.

Turings proof came up with the notion of computer.

Also, the incompleteness theorems also designed a computer using arithmetic.

CS 61C + one slide

Notes.

Fairly quick. Right?

No computers in early 20th century.

Turings proof came up with the notion of computer.

Also, the incompleteness theorems also designed a computer using arithmetic.

CS 61C + one slide $\implies$ undecidability of halting.

Undecidable questions about Programs.

Does a program, P, print "Hello World"?

Undecidable questions about Programs.

Does a program, P, print "Hello World"?
How?

Undecidable questions about Programs.

Does a program, P, print "Hello World"?
How? What is P?

Undecidable questions about Programs.

Does a program, P, print “Hello World”?
How? What is P? Text!!!!!!

Undecidable questions about Programs.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Undecidable questions about Programs.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Solve HALT(P,I):

Undecidable questions about Programs.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Solve HALT(P,I):
  Make P' as follows:

Undecidable questions about Programs.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Solve HALT(P,I):
  Make P' as follows:
    Remove all print statements.

Undecidable questions about Programs.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Solve HALT(P,I):
  Make P' as follows:
    Remove all print statements.
    Find exit points add statement: **Print** "Hello World."

Undecidable questions about Programs.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Solve HALT(P,I):
  Make P' as follows:
    Remove all print statements.
    Find exit points add statement: **Print** "Hello World."
Call PrintsHelloWorld(P',I)

Undecidable questions about Programs.

Does a program, P, print “Hello World”?
How? What is P? Text!!!!!!

Solve HALT(P,I):
  Make P' as follows:
    Remove all print statements.
    Find exit points add statement: **Print** “Hello World.”
Call PrintsHelloWorld(P',I)

P halts if and only if P' prints hello world.

Undecidable questions about Programs.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Solve HALT(P,I):
  Make P' as follows:
    Remove all print statements.
    Find exit points add statement: **Print** "Hello World."
Call PrintsHelloWorld(P',I)

P halts if and only if P' prints hello world.

Many things one can ask about programs that are undecidable.

Undecidable questions about Programs.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Solve HALT(P,I):
  Make P' as follows:
    Remove all print statements.
    Find exit points add statement: **Print** "Hello World."
Call PrintsHelloWorld(P',I)

P halts if and only if P' prints hello world.

Many things one can ask about programs that are undecidable.

Because programs are text.

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no.

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no. This "problem" is not undecidable.

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no. This "problem" is not undecidable.

Undecidability for Diophantine set of equations

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no. This "problem" is not undecidable.

Undecidability for Diophantine set of equations
$\Longrightarrow$ no program can take any set of integer equations and

Other example undecidable problems.

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no. This "problem" is not undecidable.

Undecidability for Diophantine set of equations
$\implies$ no program can take any set of integer equations and
always corectly output whether it has an integer solution.

Comments: undecidability.

Computer Programs are an interesting thing.

Comments: undecidability.

Computer Programs are an interesting thing.
Like Math.

Comments: undecidability.

Computer Programs are an interesting thing.
Like Math.
Formal Systems.

Comments: undecidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Comments: undecidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Comments: undecidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Example: no computer program can tell if any other computer program HALTS.

Comments: undecidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea:

Comments: undecidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.

Comments: undecidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
  Program: Turing (or DIAGONAL) takes P.

Comments: undecidability.

Computer Programs are an interesting thing.
 Like Math.
 Formal Systems.

Computer Programs cannot completely "understand" computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
 Program: Turing (or DIAGONAL) takes P.
 Assume there is HALT.

Comments: undecidability.

Computer Programs are an interesting thing.
Like Math.
Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
Program: Turing (or DIAGONAL) takes P.
Assume there is HALT.
DIAGONAL flips answer:Loops if P halts, halts if P loops.

Comments: undecidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
  Program: Turing (or DIAGONAL) takes P.
  Assume there is HALT.
   DIAGONAL flips answer:Loops if P halts, halts if P loops.
  What does Turing do on turing?

Comments: undecidability.

Computer Programs are an interesting thing.
Like Math.
Formal Systems.

Computer Programs cannot completely "understand" computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
Program: Turing (or DIAGONAL) takes P.
Assume there is HALT.
DIAGONAL flips answer:Loops if P halts, halts if P loops.
What does Turing do on turing? Doesn't loop or HALT.

Comments: undecidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely "understand" computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
  Program: Turing (or DIAGONAL) takes P.
  Assume there is HALT.
   DIAGONAL flips answer:Loops if P halts, halts if P loops.
  What does Turing do on turing? Doesn't loop or HALT.
    HALT does not exist!

Comments: undecidability.

Computer Programs are an interesting thing.
 Like Math.
 Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Example: no computer program can tell if any other computer program HALTS.

Proof Idea: Diagonalization.
 Program: Turing (or DIAGONAL) takes P.
 Assume there is HALT.
  DIAGONAL flips answer:Loops if P halts, halts if P loops.
 What does Turing do on turing? Doesn’t loop or HALT.
  HALT does not exist! □

Computation as a lens

Computation is a lens for other action in the world.

Computation as a lens

Computation is a lens for other action in the world.

E.g. Turing's work on linear systems (condition number), chemical networks (embryo.)

Computation as a lens

Computation is a lens for other action in the world.

E.g. Turing's work on linear systems (condition number), chemical networks (embryo.)

Today: Quantum computing, evolution models, models of the brain, complexity of Nash equilibria, ...

Your future (in this course).

What's to come?

Your future (in this course).

What's to come? Probability.

Your future (in this course).

What's to come? Probability.

A bag contains:

Your future (in this course).

What's to come? Probability.

A bag contains:

Your future (in this course).

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Your future (in this course).

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue.

Your future (in this course).

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total.

Your future (in this course).

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

Your future (in this course).

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

Today:

Your future (in this course).

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

Today: Counting!

Your future (in this course).

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

Today: Counting!

Later: Probability.

Your future (in this course).

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

Today: Counting!

Later: Probability. Professor Ayazifar.

Your future (in this course).

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

Today: Counting!

Later: Probability. Professor Ayazifar. Babak.

Your future (in this course).

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

Today: Counting!

Later: Probability. Professor Ayazifar. Babak.

Babak

Your future (in this course).

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

Today: Counting!

Later: Probability. Professor Ayazifar. Babak.

Babak $\equiv$ "Bob" Back.

Outline: basics

1. Counting.
2. Tree
3. Rules of Counting
4. Sample with/without replacement where order does/doesn't matter.

Probability is soon..but first let’s count.

Count?

How many outcomes possible for k coin tosses?
How many poker hands?
How many handshakes for n people?
How many diagonals in a convex polygon?
How many 10 digit numbers?
How many 10 digit numbers without repetition?

Using a tree..

How many 3-bit strings?

Using a tree..

How many 3-bit strings?

How many different sequences of three bits from $\{0, 1\}$?

Using a tree..

How many 3-bit strings?

How many different sequences of three bits from $\{0, 1\}$?

How would you make one sequence?

Using a tree..

How many 3-bit strings?

How many different sequences of three bits from $\{0, 1\}$?

How would you make one sequence?

How many different ways to do that making?

Using a tree..

How many 3-bit strings?

How many different sequences of three bits from $\{0, 1\}$?

How would you make one sequence?

How many different ways to do that making?

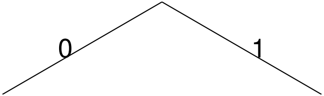
Using a tree..

How many 3-bit strings?

How many different sequences of three bits from $\{0, 1\}$?

How would you make one sequence?

How many different ways to do that making?



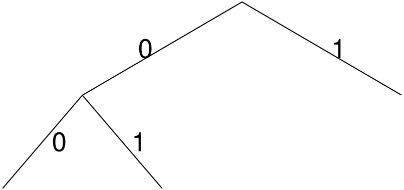
Using a tree..

How many 3-bit strings?

How many different sequences of three bits from $\{0, 1\}$?

How would you make one sequence?

How many different ways to do that making?



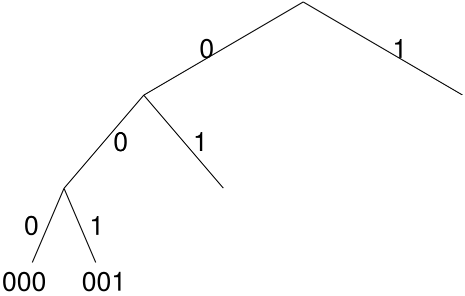
Using a tree..

How many 3-bit strings?

How many different sequences of three bits from $\{0, 1\}$?

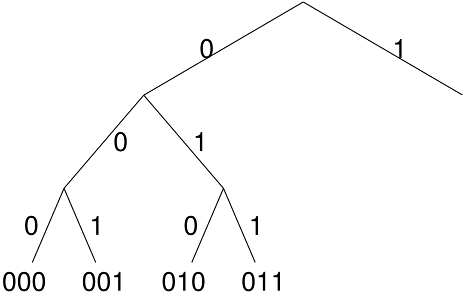
How would you make one sequence?

How many different ways to do that making?



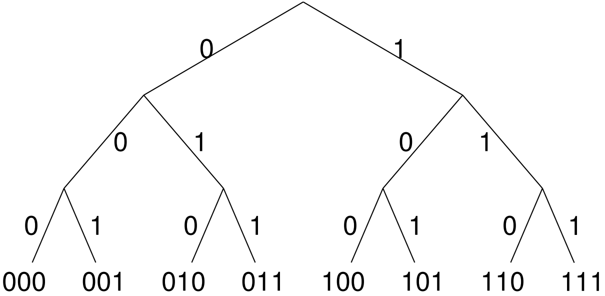
Using a tree..

How many 3-bit strings?
  How many different sequences of three bits from $\{0, 1\}$?
  How would you make one sequence?
  How many different ways to do that making?



Using a tree..

How many 3-bit strings?
How many different sequences of three bits from $\{0, 1\}$?
How would you make one sequence?
How many different ways to do that making?



8 leaves which is $2 \times 2 \times 2$.

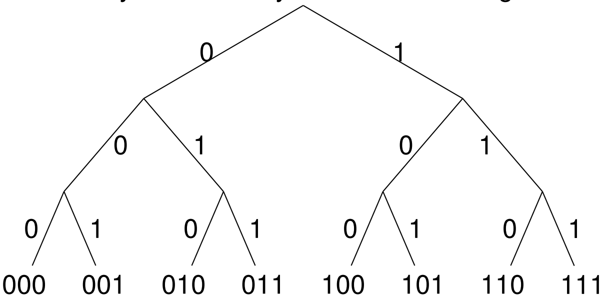
Using a tree..

How many 3-bit strings?

How many different sequences of three bits from $\{0, 1\}$?

How would you make one sequence?

How many different ways to do that making?



8 leaves which is $2 \times 2 \times 2$. One leaf for each string.

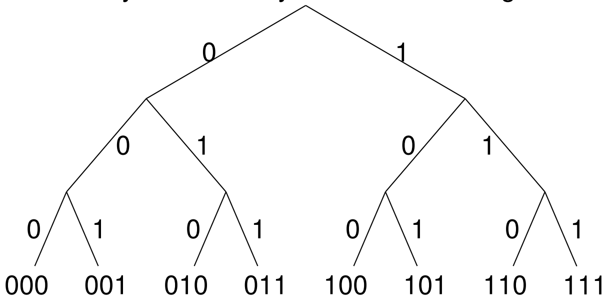
Using a tree..

How many 3-bit strings?

How many different sequences of three bits from $\{0, 1\}$?

How would you make one sequence?

How many different ways to do that making?



8 leaves which is $2 \times 2 \times 2$.   One leaf for each string.

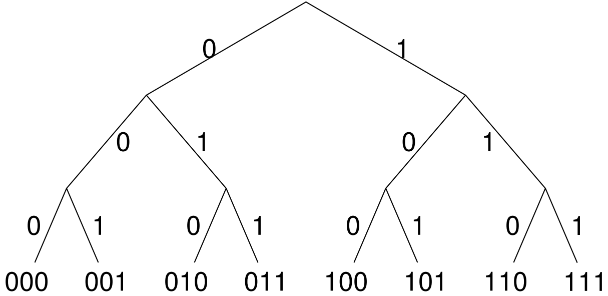
Using a tree..

How many 3-bit strings?

How many different sequences of three bits from $\{0,1\}$?

How would you make one sequence?

How many different ways to do that making?



8 leaves which is $2 \times 2 \times 2$. One leaf for each string.

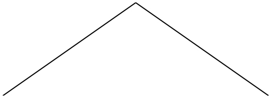
8 3-bit srings!

First Rule of Counting: Product Rule

Objects made by choosing from n_1, then n_2, ..., then n_k
the number of objects is $n_1 \times n_2 \cdots \times n_k$.

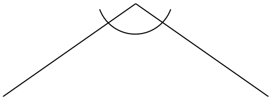
First Rule of Counting: Product Rule

Objects made by choosing from n_1, then n_2, …, then n_k
the number of objects is $n_1 \times n_2 \cdots \times n_k$.



First Rule of Counting: Product Rule

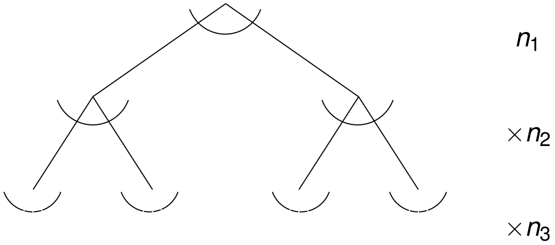
Objects made by choosing from n_1, then n_2, …, then n_k
the number of objects is $n_1 \times n_2 \cdots \times n_k$.



n_1

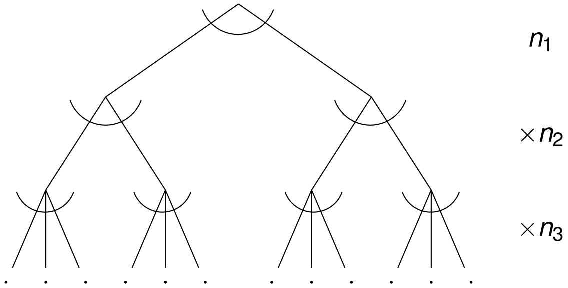
First Rule of Counting: Product Rule

Objects made by choosing from n_1, then n_2, …, then n_k
the number of objects is $n_1 \times n_2 \cdots \times n_k$.



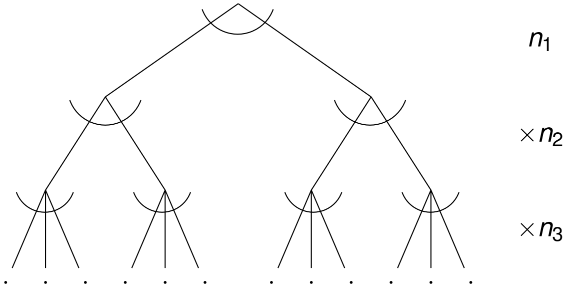
First Rule of Counting: Product Rule

Objects made by choosing from n_1, then n_2, ..., then n_k
the number of objects is $n_1 \times n_2 \cdots \times n_k$.



First Rule of Counting: Product Rule

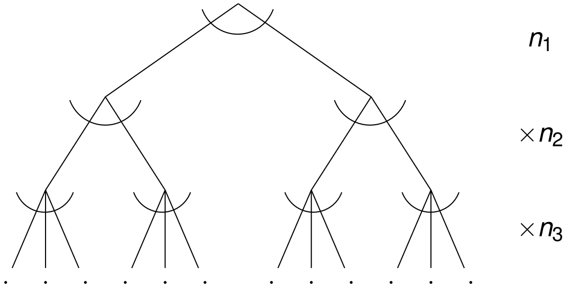
Objects made by choosing from n_1, then n_2, …, then n_k
the number of objects is $n_1 \times n_2 \cdots \times n_k$.



In picture, $2 \times 2 \times 3 = 12$!

First Rule of Counting: Product Rule

Objects made by choosing from n_1, then n_2, ..., then n_k
the number of objects is $n_1 \times n_2 \cdots \times n_k$.



In picture, $2 \times 2 \times 3 = 12$!

Using the first rule..

How many outcomes possible for k coin tosses?

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice,

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
2

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
2×2

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots$

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2$

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

How many 10 digit numbers?

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

How many 10 digit numbers?

10 ways for first choice,

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

How many 10 digit numbers?

10 ways for first choice, 10 ways for second choice, ...

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

How many 10 digit numbers?

10 ways for first choice, 10 ways for second choice, ...
10

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

How many 10 digit numbers?

10 ways for first choice, 10 ways for second choice, ...
$10 \times$

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

How many 10 digit numbers?

10 ways for first choice, 10 ways for second choice, ...
$10 \times 10 \cdots$

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

How many 10 digit numbers?

10 ways for first choice, 10 ways for second choice, ...
$10 \times 10 \cdots \times 10$

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

How many 10 digit numbers?

10 ways for first choice, 10 ways for second choice, ...
$10 \times 10 \cdots \times 10 = 10^k$

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

How many 10 digit numbers?

10 ways for first choice, 10 ways for second choice, ...
$10 \times 10 \cdots \times 10 = 10^k$

How many n digit base m numbers?

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

How many 10 digit numbers?

10 ways for first choice, 10 ways for second choice, ...
$10 \times 10 \cdots \times 10 = 10^k$

How many n digit base m numbers?

m ways for first,

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

How many 10 digit numbers?

10 ways for first choice, 10 ways for second choice, ...
$10 \times 10 \cdots \times 10 = 10^k$

How many n digit base m numbers?

m ways for first, m ways for second, ...

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

How many 10 digit numbers?

10 ways for first choice, 10 ways for second choice, ...
$10 \times 10 \cdots \times 10 = 10^k$

How many n digit base m numbers?

m ways for first, m ways for second, ...
m^n

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

How many 10 digit numbers?

10 ways for first choice, 10 ways for second choice, ...
$10 \times 10 \cdots \times 10 = 10^k$

How many n digit base m numbers?

m ways for first, m ways for second, ...
m^n

(Is 09, a two digit number?)

Using the first rule..

How many outcomes possible for k coin tosses?

2 ways for first choice, 2 ways for second choice, ...
$2 \times 2 \cdots \times 2 = 2^k$

How many 10 digit numbers?

10 ways for first choice, 10 ways for second choice, ...
$10 \times 10 \cdots \times 10 = 10^k$

How many n digit base m numbers?

m ways for first, m ways for second, ...
m^n

(Is 09, a two digit number?)

If no. Then $(m-1)m^{n-1}$.

Functions, polynomials.

How many functions f mapping S to T?

Functions, polynomials.

How many functions f mapping S to T?

$|T|$ ways to choose for $f(s_1)$,

Functions, polynomials.

How many functions f mapping S to T?

$|T|$ ways to choose for $f(s_1)$, $|T|$ ways to choose for $f(s_2)$, ...

Functions, polynomials.

How many functions f mapping S to T?

$|T|$ ways to choose for $f(s_1)$, $|T|$ ways to choose for $f(s_2)$, ...

....$|T|^{|S|}$

Functions, polynomials.

How many functions f mapping S to T?

$|T|$ ways to choose for $f(s_1)$, $|T|$ ways to choose for $f(s_2)$, ...

....$|T|^{|S|}$

How many polynomials of degree d modulo p?

Functions, polynomials.

How many functions f mapping S to T?

$|T|$ ways to choose for $f(s_1)$, $|T|$ ways to choose for $f(s_2)$, ...

....$|T|^{|S|}$

How many polynomials of degree d modulo p?

p ways to choose for first coefficient,

Functions, polynomials.

How many functions f mapping S to T?

$|T|$ ways to choose for $f(s_1)$, $|T|$ ways to choose for $f(s_2)$, ...

....$|T|^{|S|}$

How many polynomials of degree d modulo p?

p ways to choose for first coefficient, p ways for second, ...

Functions, polynomials.

How many functions f mapping S to T?

$|T|$ ways to choose for $f(s_1)$, $|T|$ ways to choose for $f(s_2)$, ...

....$|T|^{|S|}$

How many polynomials of degree d modulo p?

p ways to choose for first coefficient, p ways for second, ...
...p^{d+1}

Functions, polynomials.

How many functions f mapping S to T?

$|T|$ ways to choose for $f(s_1)$, $|T|$ ways to choose for $f(s_2)$, ...

....$|T|^{|S|}$

How many polynomials of degree d modulo p?

p ways to choose for first coefficient, p ways for second, ...
...p^{d+1}

p values for first point,

Functions, polynomials.

How many functions f mapping S to T?

$|T|$ ways to choose for $f(s_1)$, $|T|$ ways to choose for $f(s_2)$, ...

....$|T|^{|S|}$

How many polynomials of degree d modulo p?

p ways to choose for first coefficient, p ways for second, ...
...p^{d+1}

p values for first point, p values for second, ...

Functions, polynomials.

How many functions f mapping S to T?

$|T|$ ways to choose for $f(s_1)$, $|T|$ ways to choose for $f(s_2)$, ...

....$|T|^{|S|}$

How many polynomials of degree d modulo p?

p ways to choose for first coefficient, p ways for second, ...
...p^{d+1}

p values for first point, p values for second, ...
...p^{d+1}

Functions, polynomials.

How many functions f mapping S to T?

$|T|$ ways to choose for $f(s_1)$, $|T|$ ways to choose for $f(s_2)$, ...

....$|T|^{|S|}$

How many polynomials of degree d modulo p?

p ways to choose for first coefficient, p ways for second, ...
...p^{d+1}

p values for first point, p values for second, ...
...p^{d+1}

Questions?

Permutations.

[1] By definition: $0! = 1$.

Permutations.

How many 10 digit numbers **without repeating a digit**?

[1] By definition: $0! = 1$.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first,

[1] By definition: $0! = 1$.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second,

---

[1] By definition: $0! = 1$.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third,

[1] By definition: $0! = 1$.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third, ...

---

[1] By definition: $0! = 1$.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third, ...

... $10 * 9 * 8 \cdots * 1 = 10!$.[1]

[1] By definition: 0! = 1.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third, ...

... $10 * 9 * 8 \cdots * 1 = 10!$.[1]

How many different samples of size k from n numbers **without replacement.**

[1] By definition: 0! = 1.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third, ...

... $10 * 9 * 8 \cdots * 1 = 10!$.[1]

How many different samples of size k from n numbers **without replacement.**

n ways for first choice,

[1] By definition: 0! = 1.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third, ...

... $10 * 9 * 8 \cdots * 1 = 10!$.[1]

How many different samples of size k from n numbers **without replacement.**

n ways for first choice, $n-1$ ways for second,

[1] By definition: 0! = 1.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third, ...

... $10 * 9 * 8 \cdots * 1 = 10!$.[1]

How many different samples of size k from n numbers **without replacement.**

n ways for first choice, $n-1$ ways for second,
$n-2$ choices for third,

[1] By definition: 0! = 1.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third, ...

... $10 * 9 * 8 \cdots * 1 = 10!$.[1]

How many different samples of size k from n numbers **without replacement.**

n ways for first choice, $n-1$ ways for second,
$n-2$ choices for third, ...

[1] By definition: 0! = 1.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third, ...

... $10 * 9 * 8 \cdots * 1 = 10!$.[1]

How many different samples of size k from n numbers **without replacement.**

n ways for first choice, $n-1$ ways for second,
$n-2$ choices for third, ...

... $n * (n-1) * (n-2) \cdot * (n-k+1) = \frac{n!}{(n-k)!}$.

[1] By definition: 0! = 1.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third, ...

... $10 * 9 * 8 \cdots * 1 = 10!$.[1]

How many different samples of size k from n numbers **without replacement.**

n ways for first choice, $n-1$ ways for second,
$n-2$ choices for third, ...

... $n * (n-1) * (n-2) \cdot * (n-k+1) = \frac{n!}{(n-k)!}$.

How many orderings of n objects are there?
Permutations of n objects.

[1] By definition: 0! = 1.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third, ...

... $10 * 9 * 8 \cdots * 1 = 10!$.[1]

How many different samples of size k from n numbers **without replacement.**

n ways for first choice, $n-1$ ways for second,
$n-2$ choices for third, ...

... $n*(n-1)*(n-2)\cdot*(n-k+1) = \frac{n!}{(n-k)!}$.

How many orderings of n objects are there?
Permutations of n objects.

n ways for first,

[1] By definition: 0! = 1.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third, ...

... $10 * 9 * 8 \cdots * 1 = 10!$.[1]

How many different samples of size k from n numbers **without replacement.**

n ways for first choice, $n-1$ ways for second,
$n-2$ choices for third, ...

... $n * (n-1) * (n-2) \cdot * (n-k+1) = \frac{n!}{(n-k)!}$.

How many orderings of n objects are there?
Permutations of n objects.

n ways for first, $n-1$ ways for second,

[1] By definition: 0! = 1.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third, ...

... $10 * 9 * 8 \cdots * 1 = 10!$.[1]

How many different samples of size k from n numbers **without replacement.**

n ways for first choice, $n-1$ ways for second,
$n-2$ choices for third, ...

... $n * (n-1) * (n-2) \cdot * (n-k+1) = \frac{n!}{(n-k)!}$.

How many orderings of n objects are there?
Permutations of n objects.

n ways for first, $n-1$ ways for second,
$n-2$ ways for third,

[1] By definition: 0! = 1.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third, ...

... $10 * 9 * 8 \cdots * 1 = 10!$.[1]

How many different samples of size k from n numbers **without replacement.**

n ways for first choice, $n-1$ ways for second,
$n-2$ choices for third, ...

... $n * (n-1) * (n-2) \cdot * (n-k+1) = \frac{n!}{(n-k)!}$.

How many orderings of n objects are there?
Permutations of n objects.

n ways for first, $n-1$ ways for second,
$n-2$ ways for third, ...

[1] By definition: 0! = 1.

Permutations.

How many 10 digit numbers **without repeating a digit**?

10 ways for first, 9 ways for second, 8 ways for third, ...

... $10 * 9 * 8 \cdots * 1 = 10!$.[1]

How many different samples of size k from n numbers **without replacement.**

n ways for first choice, $n-1$ ways for second,
$n-2$ choices for third, ...

... $n*(n-1)*(n-2)\cdot*(n-k+1) = \frac{n!}{(n-k)!}$.

How many orderings of n objects are there?
Permutations of n objects.

n ways for first, $n-1$ ways for second,
$n-2$ ways for third, ...

... $n*(n-1)*(n-2)\cdot*1 = n!$.

[1] By definition: $0! = 1$.

One-to-One Functions.

One-to-One Functions.

How many one-to-one functions from $|S|$ to $|S|$.

One-to-One Functions.

How many one-to-one functions from $|S|$ to $|S|$.

$|S|$ choices for $f(s_1)$,

One-to-One Functions.

How many one-to-one functions from $|S|$ to $|S|$.

$|S|$ choices for $f(s_1)$, $|S|-1$ choices for $f(s_2)$, ...

One-to-One Functions.

How many one-to-one functions from $|S|$ to $|S|$.

$|S|$ choices for $f(s_1)$, $|S|-1$ choices for $f(s_2)$, ...

One-to-One Functions.

How many one-to-one functions from $|S|$ to $|S|$.

$|S|$ choices for $f(s_1)$, $|S|-1$ choices for $f(s_2)$, ...

So total number is $|S| \times |S|-1 \cdots 1 = |S|!$

One-to-One Functions.

How many one-to-one functions from $|S|$ to $|S|$.

$|S|$ choices for $f(s_1)$, $|S|-1$ choices for $f(s_2)$, ...

So total number is $|S| \times |S|-1 \cdots 1 = |S|!$
A one-to-one function is a permutation!

Counting sets..when order doesn't matter.

How many poker hands?

[2]When each unordered object corresponds equal numbers of ordered objects.

Counting sets..when order doesn't matter.

How many poker hands?

$52 \times 51 \times 50 \times 49 \times 48$

[2]When each unordered object corresponds equal numbers of ordered objects.

Counting sets..when order doesn't matter.

How many poker hands?

$52 \times 51 \times 50 \times 49 \times 48$???

[2]When each unordered object corresponds equal numbers of ordered objects.

Counting sets..when order doesn't matter.

How many poker hands?

$52 \times 51 \times 50 \times 49 \times 48$???

Are $A, K, Q, 10, J$ of spades
and $10, J, Q, K, A$ of spades the same?

[2] When each unordered object corresponds equal numbers of ordered objects.

Counting sets..when order doesn't matter.

How many poker hands?

$52 \times 51 \times 50 \times 49 \times 48$???

Are $A, K, Q, 10, J$ of spades
and $10, J, Q, K, A$ of spades the same?
Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.[2]

[2] When each unordered object corresponds equal numbers of ordered objects.

Counting sets..when order doesn't matter.

How many poker hands?

$52 \times 51 \times 50 \times 49 \times 48$???

Are $A, K, Q, 10, J$ of spades
and $10, J, Q, K, A$ of spades the same?
Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.[2]

Number of orderings for a poker hand: "5!"

[2]When each unordered object corresponds equal numbers of ordered objects.

Counting sets..when order doesn't matter.

How many poker hands?

$52 \times 51 \times 50 \times 49 \times 48$???

Are $A, K, Q, 10, J$ of spades
and $10, J, Q, K, A$ of spades the same?
Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.[2]

Number of orderings for a poker hand: "5!"
(The "!" means factorial, not Exclamation.)

[2] When each unordered object corresponds equal numbers of ordered objects.

Counting sets..when order doesn't matter.

How many poker hands?

$52 \times 51 \times 50 \times 49 \times 48$???

Are $A, K, Q, 10, J$ of spades
and $10, J, Q, K, A$ of spades the same?
Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.[2]

Number of orderings for a poker hand: "5!"

$$\frac{52 \times 51 \times 50 \times 49 \times 48}{5!}$$

[2] When each unordered object corresponds equal numbers of ordered objects.

Counting sets..when order doesn't matter.

How many poker hands?

$52 \times 51 \times 50 \times 49 \times 48$???

Are $A, K, Q, 10, J$ of spades
and $10, J, Q, K, A$ of spades the same?
Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.[2]

Number of orderings for a poker hand: "5!"

Can write as...
$$\frac{52 \times 51 \times 50 \times 49 \times 48}{5!}$$
$$\frac{52!}{5! \times 47!}$$

[2]When each unordered object corresponds equal numbers of ordered objects.

Counting sets..when order doesn't matter.

How many poker hands?

$52 \times 51 \times 50 \times 49 \times 48$???

Are $A, K, Q, 10, J$ of spades
and $10, J, Q, K, A$ of spades the same?
Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.[2]

Number of orderings for a poker hand: "5!"

Can write as...

$$\frac{52 \times 51 \times 50 \times 49 \times 48}{5!}$$

$$\frac{52!}{5! \times 47!}$$

Generic: ways to choose 5 out of 52 possibilities.

---

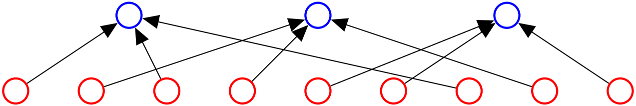
[2]When each unordered object corresponds equal numbers of ordered objects.

Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.

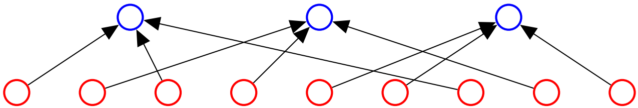
Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.



Ordered to unordered.

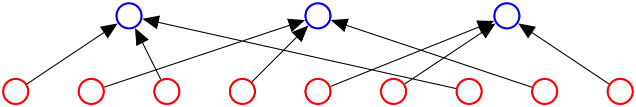
Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.



How many red nodes (ordered objects)?

Ordered to unordered.

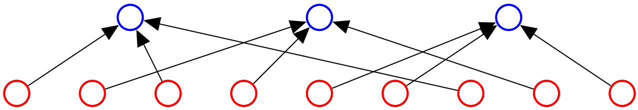
Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.



How many red nodes (ordered objects)? 9.

Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.

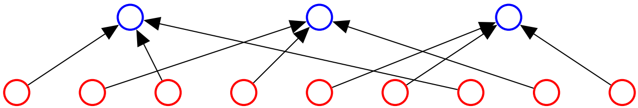


How many red nodes (ordered objects)? 9.

How many red nodes mapped to one blue node?

Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.

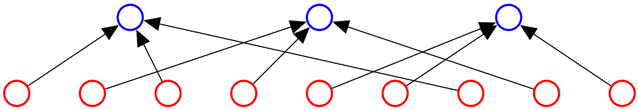


How many red nodes (ordered objects)? 9.

How many red nodes mapped to one blue node? 3.

Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.



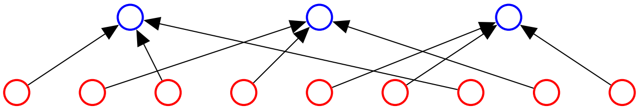
How many red nodes (ordered objects)? 9.

How many red nodes mapped to one blue node? 3.

How many blue nodes (unordered objects)?

Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.



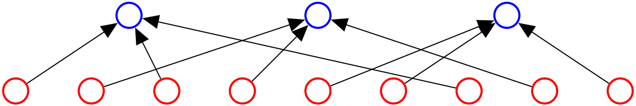
How many red nodes (ordered objects)? 9.

How many red nodes mapped to one blue node? 3.

How many blue nodes (unordered objects)? $\frac{9}{3}$

Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.



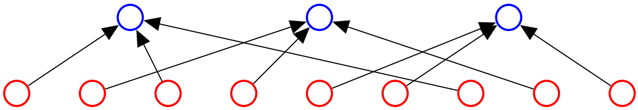
How many red nodes (ordered objects)? 9.

How many red nodes mapped to one blue node? 3.

How many blue nodes (unordered objects)? $\frac{9}{3} = 3$.

Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.



How many red nodes (ordered objects)? 9.

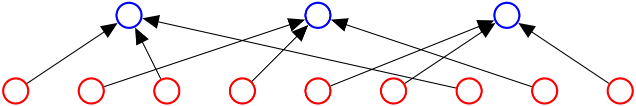
How many red nodes mapped to one blue node? 3.

How many blue nodes (unordered objects)? $\frac{9}{3} = 3$.

How many poker deals?

Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.



How many red nodes (ordered objects)? 9.

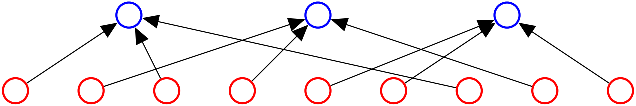
How many red nodes mapped to one blue node? 3.

How many blue nodes (unordered objects)? $\frac{9}{3} = 3$.

How many poker deals? $52 \cdot 51 \cdot 50 \cdot 49 \cdot 48$.

Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.



How many red nodes (ordered objects)? 9.

How many red nodes mapped to one blue node? 3.

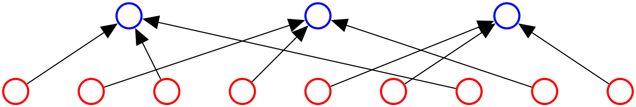
How many blue nodes (unordered objects)? $\frac{9}{3} = 3$.

How many poker deals? $52 \cdot 51 \cdot 50 \cdot 49 \cdot 48$.

How many poker deals per hand?

Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.



How many red nodes (ordered objects)? 9.

How many red nodes mapped to one blue node? 3.

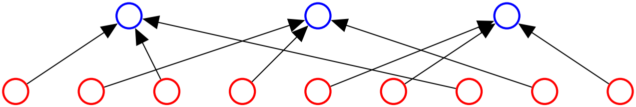
How many blue nodes (unordered objects)? $\frac{9}{3} = 3$.

How many poker deals? $52 \cdot 51 \cdot 50 \cdot 49 \cdot 48$.

How many poker deals per hand?
  Map each deal to ordered deal:

Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.



How many red nodes (ordered objects)? 9.

How many red nodes mapped to one blue node? 3.

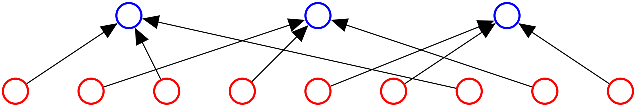
How many blue nodes (unordered objects)? $\frac{9}{3} = 3$.

How many poker deals? $52 \cdot 51 \cdot 50 \cdot 49 \cdot 48$.

How many poker deals per hand?
  Map each deal to ordered deal: $5!$

Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.



How many red nodes (ordered objects)? 9.

How many red nodes mapped to one blue node? 3.

How many blue nodes (unordered objects)? $\frac{9}{3} = 3$.

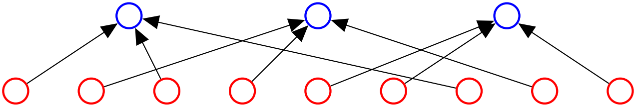
How many poker deals? $52 \cdot 51 \cdot 50 \cdot 49 \cdot 48$.

How many poker deals per hand?
Map each deal to ordered deal: $5!$

How many poker hands?

Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.



How many red nodes (ordered objects)? 9.

How many red nodes mapped to one blue node? 3.

How many blue nodes (unordered objects)? $\frac{9}{3} = 3$.

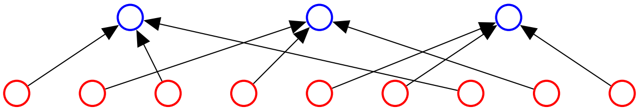
How many poker deals? $52 \cdot 51 \cdot 50 \cdot 49 \cdot 48$.

How many poker deals per hand?
  Map each deal to ordered deal: $5!$

How many poker hands? $\frac{52 \cdot 51 \cdot 50 \cdot 49 \cdot 48}{5!}$

Ordered to unordered.

Second Rule of Counting: If order doesn't matter count ordered objects and then divide by number of orderings.



How many red nodes (ordered objects)? 9.

How many red nodes mapped to one blue node? 3.

How many blue nodes (unordered objects)? $\frac{9}{3} = 3$.

How many poker deals? $52 \cdot 51 \cdot 50 \cdot 49 \cdot 48$.

How many poker deals per hand?
Map each deal to ordered deal: $5!$

How many poker hands? $\frac{52 \cdot 51 \cdot 50 \cdot 49 \cdot 48}{5!}$

Questions?

..order doesn’t matter.

..order doesn’t matter.

Choose 2 out of n?

..order doesn’t matter.

Choose 2 out of n?

$$\frac{n \times (n-1)}{}$$

..order doesn't matter.

Choose 2 out of n?

$$\frac{n \times (n-1)}{2}$$

..order doesn't matter.

Choose 2 out of n?

$$\frac{n \times (n-1)}{2} = \frac{n!}{(n-2)! \times 2}$$

..order doesn't matter.

Choose 2 out of n?

$$\frac{n \times (n-1)}{2} = \frac{n!}{(n-2)! \times 2}$$

Choose 3 out of n?

..order doesn't matter.

Choose 2 out of n?

$$\frac{n \times (n-1)}{2} = \frac{n!}{(n-2)! \times 2}$$

Choose 3 out of n?

$$\frac{n \times (n-1) \times (n-2)}{}$$

..order doesn't matter.

Choose 2 out of n?

$$\frac{n \times (n-1)}{2} = \frac{n!}{(n-2)! \times 2}$$

Choose 3 out of n?

$$\frac{n \times (n-1) \times (n-2)}{3!}$$

..order doesn't matter.

Choose 2 out of n?

$$\frac{n \times (n-1)}{2} = \frac{n!}{(n-2)! \times 2}$$

Choose 3 out of n?

$$\frac{n \times (n-1) \times (n-2)}{3!} = \frac{n!}{(n-3)! \times 3!}$$

..order doesn't matter.

Choose 2 out of n?

$$\frac{n \times (n-1)}{2} = \frac{n!}{(n-2)! \times 2}$$

Choose 3 out of n?

$$\frac{n \times (n-1) \times (n-2)}{3!} = \frac{n!}{(n-3)! \times 3!}$$

Choose k **out of** n?

$$\frac{n!}{(n-k)!\qquad\quad}$$

..order doesn't matter.

Choose 2 out of n?

$$\frac{n \times (n-1)}{2} = \frac{n!}{(n-2)! \times 2}$$

Choose 3 out of n?

$$\frac{n \times (n-1) \times (n-2)}{3!} = \frac{n!}{(n-3)! \times 3!}$$

Choose k **out of** n?

$$\frac{n!}{(n-k)!}$$

..order doesn't matter.

Choose 2 out of n?

$$\frac{n \times (n-1)}{2} = \frac{n!}{(n-2)! \times 2}$$

Choose 3 out of n?

$$\frac{n \times (n-1) \times (n-2)}{3!} = \frac{n!}{(n-3)! \times 3!}$$

Choose k **out of** n?

$$\frac{n!}{(n-k)! \times k!}$$

..order doesn't matter.

Choose 2 out of n?

$$\frac{n \times (n-1)}{2} = \frac{n!}{(n-2)! \times 2}$$

Choose 3 out of n?

$$\frac{n \times (n-1) \times (n-2)}{3!} = \frac{n!}{(n-3)! \times 3!}$$

Choose k **out of** n?

$$\frac{n!}{(n-k)! \times k!}$$

Notation: $\binom{n}{k}$ **and pronounced "**n **choose** k**."**

..order doesn't matter.

Choose 2 out of n?

$$\frac{n \times (n-1)}{2} = \frac{n!}{(n-2)! \times 2}$$

Choose 3 out of n?

$$\frac{n \times (n-1) \times (n-2)}{3!} = \frac{n!}{(n-3)! \times 3!}$$

Choose k **out of** n?

$$\frac{n!}{(n-k)! \times k!}$$

Notation: $\binom{n}{k}$ **and pronounced "n choose k."**

Familiar?

..order doesn't matter.

Choose 2 out of n?

$$\frac{n \times (n-1)}{2} = \frac{n!}{(n-2)! \times 2}$$

Choose 3 out of n?

$$\frac{n \times (n-1) \times (n-2)}{3!} = \frac{n!}{(n-3)! \times 3!}$$

Choose k **out of** n?

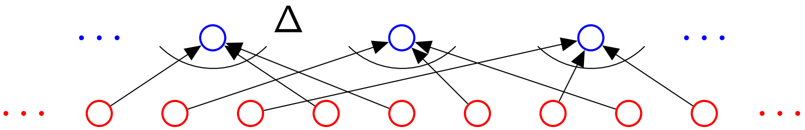
$$\frac{n!}{(n-k)! \times k!}$$

Notation: $\binom{n}{k}$ **and pronounced "**n **choose** k**."**

Familiar? Questions?

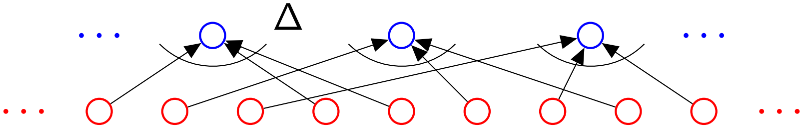
Example: Visualize the proof..

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



Example: Visualize the proof..

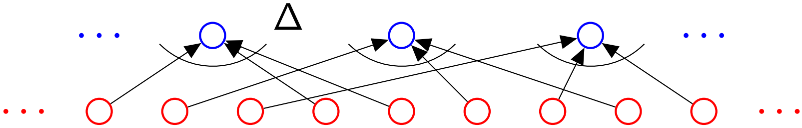
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: 52

Example: Visualize the proof..

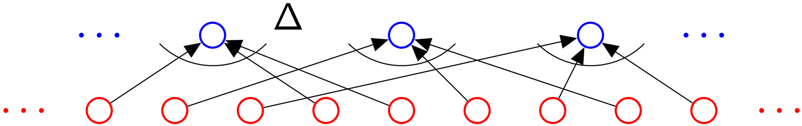
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: 52×51

Example: Visualize the proof..

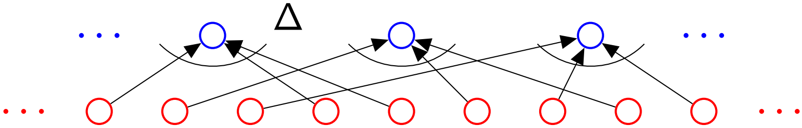
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50$

Example: Visualize the proof..

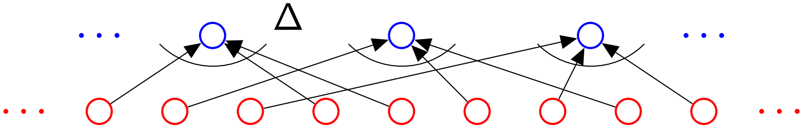
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$.

Example: Visualize the proof..

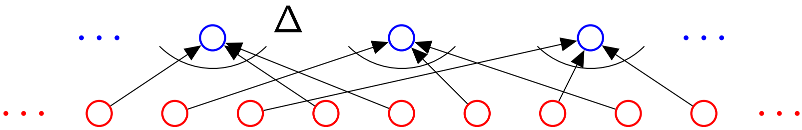
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.

Example: Visualize the proof..

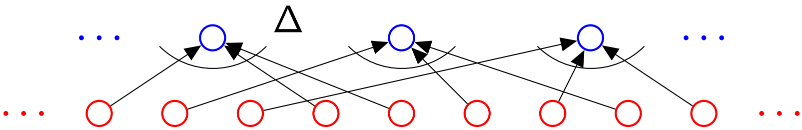
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?

Example: Visualize the proof..

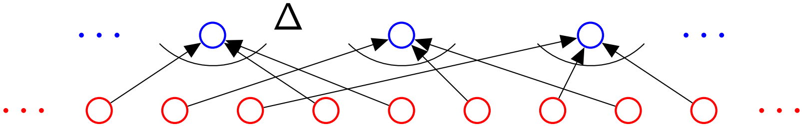
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
  Hand: Q, K, A.

Example: Visualize the proof..

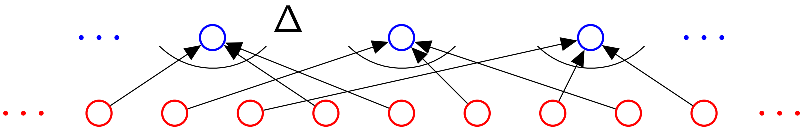
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
  Hand: Q, K, A.
  Deals: Q, K, A :

Example: Visualize the proof..

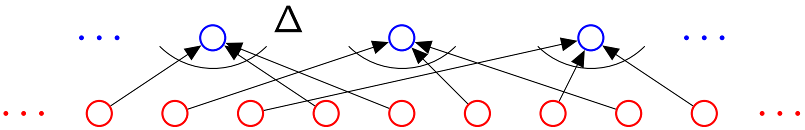
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q, K, A : Q, A, K :

Example: Visualize the proof..

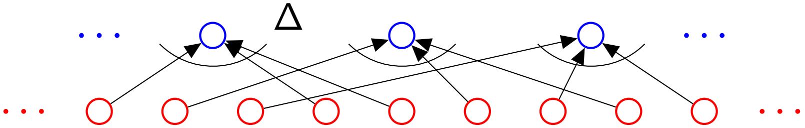
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
  Hand: Q, K, A.
  Deals: $Q, K, A : Q, A, K : K, A, Q : K, A, Q : A, K, Q : A, Q, K$.

Example: Visualize the proof..

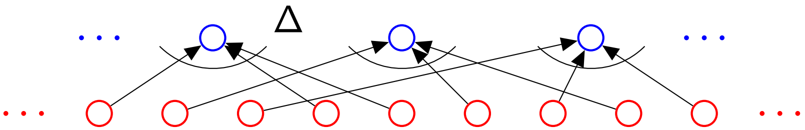
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: $Q, K, A : Q, A, K : K, A, Q : K, A, Q : A, K, Q : A, Q, K$.
$\Delta = 3 \times 2 \times 1$

Example: Visualize the proof..

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?

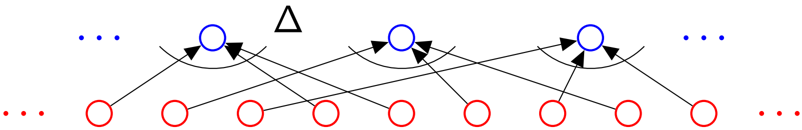
Hand: Q, K, A.

Deals: $Q, K, A : Q, A, K : K, A, Q : K, A, Q : A, K, Q : A, Q, K$.

$\Delta = 3 \times 2 \times 1$ First rule again.

Example: Visualize the proof..

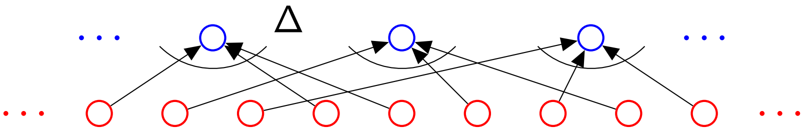
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: $Q, K, A : Q, A, K : K, A, Q : K, A, Q : A, K, Q : A, Q, K$.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total:

Example: Visualize the proof..

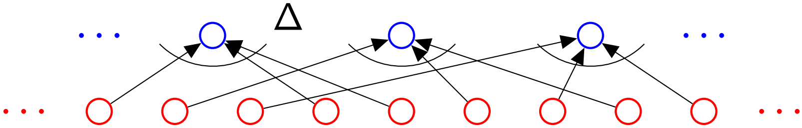
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: $Q, K, A : Q, A, K : K, A, Q : K, A, Q : A, K, Q : A, Q, K$.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$

Example: Visualize the proof..

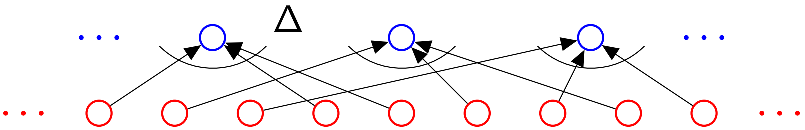
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: $Q, K, A : Q, A, K : K, A, Q : K, A, Q : A, K, Q : A, Q, K$.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Example: Visualize the proof..

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...

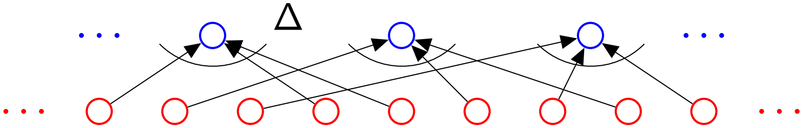


3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
 Hand: Q, K, A.
 Deals: $Q, K, A : Q, A, K : K, A, Q : K, A, Q : A, K, Q : A, Q, K$.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Choose k out of n.

Example: Visualize the proof..

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...

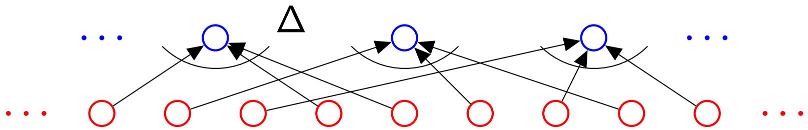


3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: $Q, K, A : Q, A, K : K, A, Q : K, A, Q : A, K, Q : A, Q, K$.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Choose k out of n.
Ordered set: $\frac{n!}{(n-k)!}$

Example: Visualize the proof..

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...

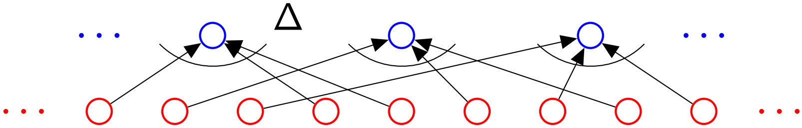


3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: $Q, K, A : Q, A, K : K, A, Q : K, A, Q : A, K, Q : A, Q, K$.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Choose k out of n.
Ordered set: $\frac{n!}{(n-k)!}$ Orderings of one hand?

Example: Visualize the proof..

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...

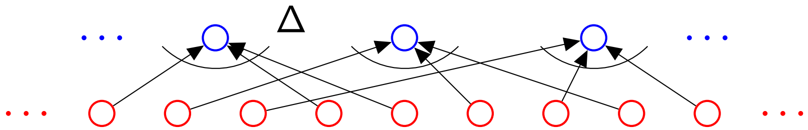


3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
  Hand: Q, K, A.
  Deals: Q, K, A : Q, A, K : K, A, Q : K, A, Q : A, K, Q : A, Q, K.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Choose k out of n.
  Ordered set: $\frac{n!}{(n-k)!}$ Orderings of one hand? $k!$

Example: Visualize the proof..

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?

Hand: Q, K, A.

Deals: $Q, K, A : Q, A, K : K, A, Q : K, A, Q : A, K, Q : A, Q, K$.

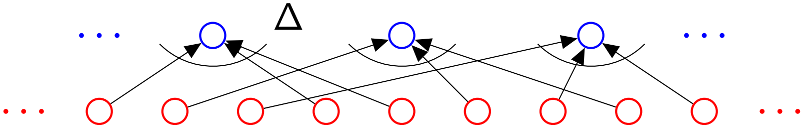
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Choose k out of n.

Ordered set: $\frac{n!}{(n-k)!}$ Orderings of one hand? $k!$ (By first rule!)

Example: Visualize the proof..

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?

Hand: Q, K, A.

Deals: $Q, K, A : Q, A, K : K, A, Q : K, A, Q : A, K, Q : A, Q, K$.

$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

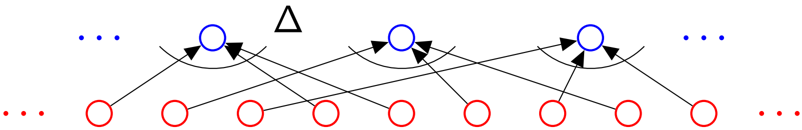
Choose k out of n.

Ordered set: $\frac{n!}{(n-k)!}$ Orderings of one hand? $k!$ (By first rule!)

$\implies$ Total: $\frac{n!}{(n-k)!k!}$

Example: Visualize the proof..

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?

Hand: Q, K, A.

Deals: $Q, K, A : Q, A, K : K, A, Q : K, A, Q : A, K, Q : A, Q, K$.

$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

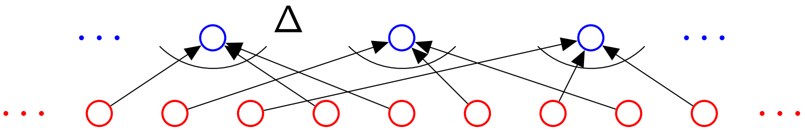
Choose k out of n.

Ordered set: $\frac{n!}{(n-k)!}$ Orderings of one hand? $k!$ (By first rule!)

$\implies$ Total: $\frac{n!}{(n-k)!k!}$ Second rule.

Example: Visualize the proof..

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...

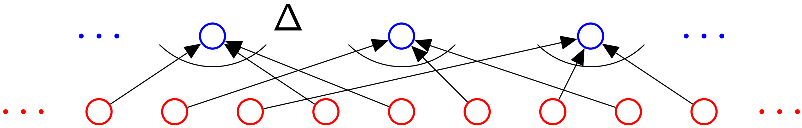


3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q, K, A : Q, A, K : K, A, Q : K, A, Q : A, K, Q : A, Q, K.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Choose k out of n.
Ordered set: $\frac{n!}{(n-k)!}$ Orderings of one hand? $k!$ (By first rule!)
$\Longrightarrow$ Total: $\frac{n!}{(n-k)!k!}$ Second rule.

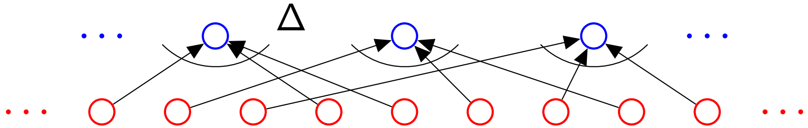
Example: Anagram

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



Example: Anagram

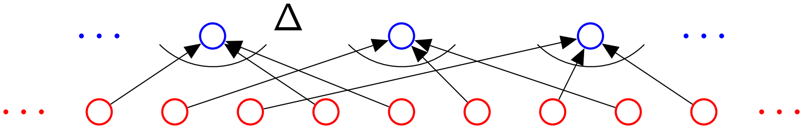
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



Orderings of ANAGRAM?

Example: Anagram

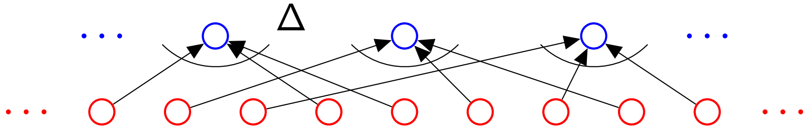
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



Orderings of ANAGRAM?
Ordered Set: 7!

Example: Anagram

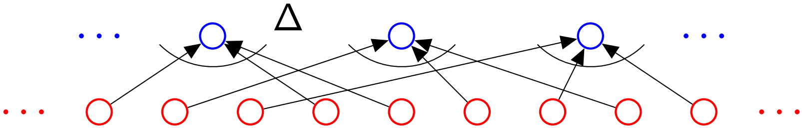
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



Orderings of ANAGRAM?
Ordered Set: 7! First rule.

Example: Anagram

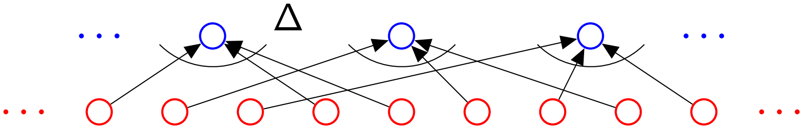
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



Orderings of ANAGRAM?
  Ordered Set: 7! First rule.
  A's are the same!

Example: Anagram

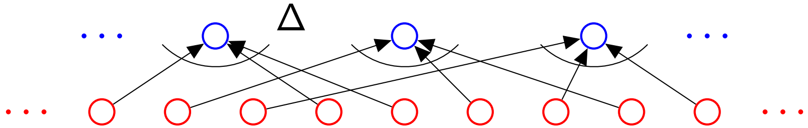
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



Orderings of ANAGRAM?
Ordered Set: 7! First rule.
A's are the same!
What is Δ?

Example: Anagram

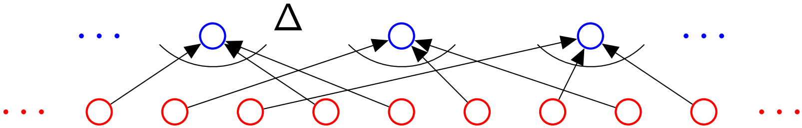
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



Orderings of ANAGRAM?
Ordered Set: 7! First rule.
A's are the same!
What is Δ?
ANAGRAM

Example: Anagram

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



Orderings of ANAGRAM?

Ordered Set: 7! First rule.

A's are the same!

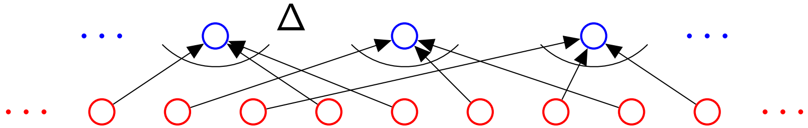
What is Δ?

ANAGRAM

$A_1NA_2GRA_3M$,

Example: Anagram

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



Orderings of ANAGRAM?

Ordered Set: 7! First rule.

A's are the same!

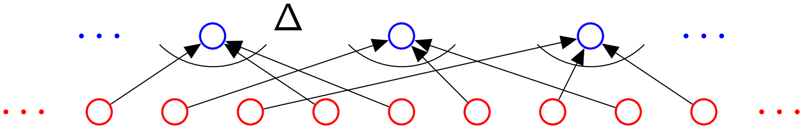
What is Δ?

ANAGRAM

$A_1NA_2GRA_3M$, $A_2NA_1GRA_3M$,

Example: Anagram

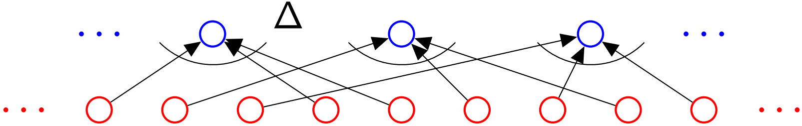
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



Orderings of ANAGRAM?
Ordered Set: 7! First rule.
A's are the same!
What is Δ?
ANAGRAM
$A_1NA_2GRA_3M$, $A_2NA_1GRA_3M$, ...

Example: Anagram

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



Orderings of ANAGRAM?

Ordered Set: 7! First rule.

A's are the same!

What is Δ?

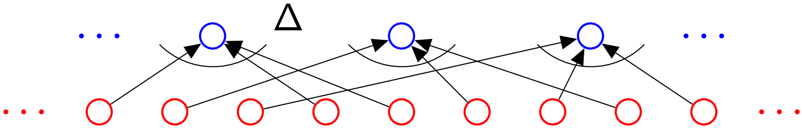
ANAGRAM

$A_1NA_2GRA_3M$, $A_2NA_1GRA_3M$, ...

$\Delta = 3 \times 2 \times 1$

Example: Anagram

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



Orderings of ANAGRAM?

Ordered Set: 7! First rule.

A's are the same!

What is Δ?

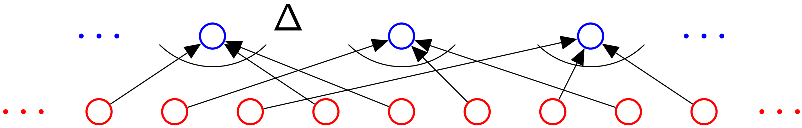
ANAGRAM

$A_1NA_2GRA_3M$, $A_2NA_1GRA_3M$, ...

$\Delta = 3 \times 2 \times 1 = 3!$

Example: Anagram

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide...



Orderings of ANAGRAM?

Ordered Set: 7! First rule.

A's are the same!

What is Δ?

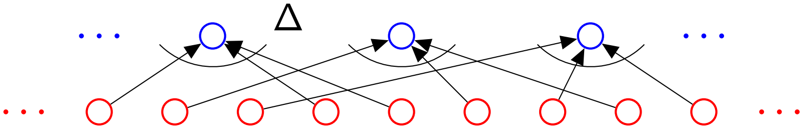
ANAGRAM

$A_1NA_2GRA_3M$, $A_2NA_1GRA_3M$, ...

$\Delta = 3 \times 2 \times 1 = 3!$ First rule!

Example: Anagram

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



Orderings of ANAGRAM?

Ordered Set: 7! First rule.

A's are the same!

What is Δ?

ANAGRAM

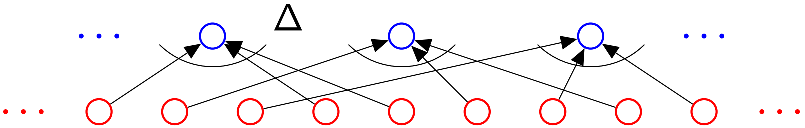
$A_1NA_2GRA_3M$, $A_2NA_1GRA_3M$, ...

$\Delta = 3 \times 2 \times 1 = 3!$ First rule!

$\implies \frac{7!}{3!}$

Example: Anagram

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide...



Orderings of ANAGRAM?

Ordered Set: 7! First rule.

A's are the same!

What is Δ?

ANAGRAM

$A_1NA_2GRA_3M$, $A_2NA_1GRA_3M$, ...

$\Delta = 3 \times 2 \times 1 = 3!$ First rule!

$\Longrightarrow \frac{7!}{3!}$ Second rule!

Some Practice.

How many orderings of letters of CAT?

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1$

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Ordered,

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Ordered, except for A!

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Ordered, except for A!

total orderings of 7 letters.

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Ordered, except for A!

total orderings of 7 letters. 7!

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Ordered, except for A!

total orderings of 7 letters. 7!
total "extra counts" or orderings of three A's?

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Ordered, except for A!

total orderings of 7 letters. 7!
total "extra counts" or orderings of three A's? 3!

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Ordered, except for A!

total orderings of 7 letters. 7!
total “extra counts” or orderings of three A’s? 3!

Total orderings?

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Ordered, except for A!

total orderings of 7 letters. 7!
total "extra counts" or orderings of three A's? 3!

Total orderings? $\frac{7!}{3!}$

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Ordered, except for A!

total orderings of 7 letters. 7!
total "extra counts" or orderings of three A's? 3!

Total orderings? $\frac{7!}{3!}$

How many orderings of MISSISSIPPI?

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Ordered, except for A!

total orderings of 7 letters. 7!
total “extra counts” or orderings of three A’s? 3!

Total orderings? $\frac{7!}{3!}$

How many orderings of MISSISSIPPI?

4 S’s, 4 I’s, 2 P’s.

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Ordered, except for A!

total orderings of 7 letters. 7!
total "extra counts" or orderings of three A's? 3!

Total orderings? $\frac{7!}{3!}$

How many orderings of MISSISSIPPI?

4 S's, 4 I's, 2 P's.
11 letters total.

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Ordered, except for A!

total orderings of 7 letters. $7!$
total "extra counts" or orderings of three A's? $3!$

Total orderings? $\frac{7!}{3!}$

How many orderings of MISSISSIPPI?

4 S's, 4 I's, 2 P's.
11 letters total.
$11!$ ordered objects.

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Ordered, except for A!

total orderings of 7 letters. 7!
total "extra counts" or orderings of three A's? 3!

Total orderings? $\frac{7!}{3!}$

How many orderings of MISSISSIPPI?

4 S's, 4 I's, 2 P's.
11 letters total.
11! ordered objects.
$4! \times 4! \times 2!$ ordered objects per "unordered object"

Some Practice.

How many orderings of letters of CAT?

3 ways to choose first letter, 2 ways for second, 1 for last.

$\implies 3 \times 2 \times 1 = 3!$ orderings

How many orderings of the letters in ANAGRAM?

Ordered, except for A!

total orderings of 7 letters. 7!
total "extra counts" or orderings of three A's? 3!

Total orderings? $\frac{7!}{3!}$

How many orderings of MISSISSIPPI?

4 S's, 4 I's, 2 P's.
11 letters total.
$11!$ ordered objects.
$4! \times 4! \times 2!$ ordered objects per "unordered object"

$\implies \frac{11!}{4!4!2!}$.

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers

$$\binom{52}{5}$$

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker

$$\binom{52}{5} + \binom{52}{4}$$

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands?

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands?

$$\binom{52}{5} +$$

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

$$\binom{52}{5} + 2 * \binom{52}{4} +$$

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

$$\binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$$

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

$$\binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$$

Wait a minute!

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

$$\binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$$

Wait a minute! Same as

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

$$\binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$$

Wait a minute! Same as choosing 5 cards from 54

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

$$\binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$$

Wait a minute! Same as choosing 5 cards from 54 or

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

$$\binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$$

Wait a minute! Same as choosing 5 cards from 54 or

$$\binom{54}{5}$$

Theorem: $\binom{54}{5}$

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers “exclusive” or One Joker “exclusive” or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

$$\binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$$

Wait a minute! Same as choosing 5 cards from 54 or

$$\binom{54}{5}$$

Theorem: $\binom{54}{5} = \binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$.

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

$$\binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$$

Wait a minute! Same as choosing 5 cards from 54 or

$$\binom{54}{5}$$

Theorem: $\binom{54}{5} = \binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$.

Algebraic Proof:

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

$$\binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$$

Wait a minute! Same as choosing 5 cards from 54 or

$$\binom{54}{5}$$

Theorem: $\binom{54}{5} = \binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$.

Algebraic Proof: Why?

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5}+\binom{52}{4}+\binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

$$\binom{52}{5}+2*\binom{52}{4}+\binom{52}{3}$$

Wait a minute! Same as choosing 5 cards from 54 or

$$\binom{54}{5}$$

Theorem: $\binom{54}{5}=\binom{52}{5}+2*\binom{52}{4}+\binom{52}{3}$.

Algebraic Proof: Why? Just why?

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

$$\binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$$

Wait a minute! Same as choosing 5 cards from 54 or

$$\binom{54}{5}$$

Theorem: $\binom{54}{5} = \binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$.

Algebraic Proof: Why? Just why? Especially on Thursday!

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers “exclusive” or One Joker “exclusive” or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

$$\binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$$

Wait a minute! Same as choosing 5 cards from 54 or

$$\binom{54}{5}$$

Theorem: $\binom{54}{5} = \binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$.

Algebraic Proof: Why? Just why? Especially on Thursday!
Already have a **combinatorial proof.**

Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers "exclusive" or One Joker "exclusive" or Two Jokers

$$\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.$$

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

$$\binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$$

Wait a minute! Same as choosing 5 cards from 54 or

$$\binom{54}{5}$$

Theorem: $\binom{54}{5} = \binom{52}{5} + 2 * \binom{52}{4} + \binom{52}{3}$.

Algebraic Proof: Why? Just why? Especially on Thursday!
Already have a **combinatorial proof.** $\square$

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S **and** T**,** $|S \cup T| = |S| + |T|$

Simple Inclusion/Exclusion

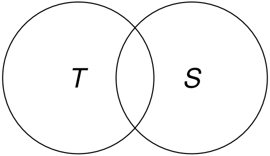
Sum Rule: For disjoint sets S **and** T**,** $|S \cup T| = |S| + |T|$

Inclusion/Exclusion Rule:
For any S **and** T**,** $|S \cup T| = |S| + |T| - |S \cap T|$.

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S **and** T**,** $|S \cup T| = |S| + |T|$

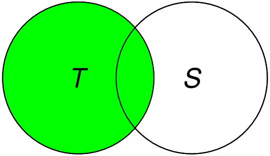
Inclusion/Exclusion Rule:
For any S **and** T**,** $|S \cup T| = |S| + |T| - |S \cap T|$.



Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S **and** T**,** $|S \cup T| = |S| + |T|$

Inclusion/Exclusion Rule:
For any S **and** T**,** $|S \cup T| = |S| + |T| - |S \cap T|$.

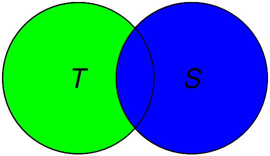


In T. $\implies$ $|T|$

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S **and** T**,** $|S \cup T| = |S| + |T|$

Inclusion/Exclusion Rule:
For any S **and** T**,** $|S \cup T| = |S| + |T| - |S \cap T|$.

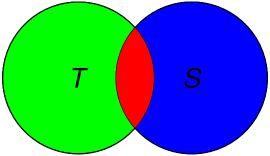


In T. $\implies |T|$
In S. $\implies + \quad |S|$

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S and T, $|S \cup T| = |S| + |T|$

Inclusion/Exclusion Rule:
For any S and T, $|S \cup T| = |S| + |T| - |S \cap T|$.



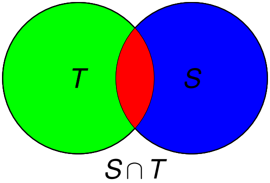
$S \cap T$

In T. $\implies$ $|T|$
In S. $\implies$ $+$ $|S|$
Elements in $S \cap T$ are counted twice.

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S **and** T**,** $|S \cup T| = |S| + |T|$

Inclusion/Exclusion Rule:
For any S **and** T**,** $|S \cup T| = |S| + |T| - |S \cap T|$.



In T. $\implies |T|$

In S. $\implies + \ |S|$

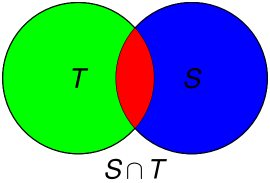
Elements in $S \cap T$ are counted twice.

Subtract. $\implies -|S \cap T|$

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S **and** T**,** $|S \cup T| = |S| + |T|$

Inclusion/Exclusion Rule:
For any S **and** T**,** $|S \cup T| = |S| + |T| - |S \cap T|$.



In T. $\implies |T|$

In S. $\implies + \ |S|$

Elements in $S \cap T$ are counted twice.

Subtract. $\implies -|S \cap T|$

$$|S \cup T| = |S| + |T| - |S \cap T|$$

Using Inclusion/Exclusion.

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Using Inclusion/Exclusion.

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

Using Inclusion/Exclusion.

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

S = phone numbers with 7 as first digit.

Using Inclusion/Exclusion.

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

S = phone numbers with 7 as first digit. $|S| = 10^9$

Using Inclusion/Exclusion.

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

S = phone numbers with 7 as first digit.$|S| = 10^9$

T = phone numbers with 7 as second digit.

Using Inclusion/Exclusion.

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

S = phone numbers with 7 as first digit.$|S| = 10^9$

T = phone numbers with 7 as second digit. $|T| = 10^9$.

Using Inclusion/Exclusion.

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

S = phone numbers with 7 as first digit.$|S| = 10^9$

T = phone numbers with 7 as second digit. $|T| = 10^9$.

$S \cap T$ = phone numbers with 7 as first and second digit.

Using Inclusion/Exclusion.

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

S = phone numbers with 7 as first digit.$|S| = 10^9$

T = phone numbers with 7 as second digit. $|T| = 10^9$.

$S \cap T$ = phone numbers with 7 as first and second digit. $|S \cap T| = 10^8$.

Using Inclusion/Exclusion.

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

S = phone numbers with 7 as first digit.$|S| = 10^9$

T = phone numbers with 7 as second digit. $|T| = 10^9$.

$S \cap T$ = phone numbers with 7 as first and second digit. $|S \cap T| = 10^8$.

Answer: $|S| + |T| - |S \cap T| = 10^9 + 10^9 - 10^8$.

Counting.

First Rule:

Make object out of sequence of choices.

n_i - number of choices for choice i.

Number of objects:

Counting.

First Rule:

Make object out of sequence of choices.

n_i - number of choices for choice i.

Number of objects: $\prod_i n_i$.

Counting.

First Rule:

Make object out of sequence of choices.

n_i - number of choices for choice i.

Number of objects: $\prod_i n_i$.

Second Rule:

When order doesn't matter, divide out orderings.

Counting.

First Rule:
  Make object out of sequence of choices.
    n_i - number of choices for choice i.
  Number of objects: $\prod_i n_i$.

Second Rule:
  When order doesn't matter, divide out orderings.

Sum Rule:
  Size of union of disjoint sets is sum of sizes.

Counting.

First Rule:
  Make object out of sequence of choices.
    n_i - number of choices for choice i.
  Number of objects: $\prod_i n_i$.

Second Rule:
  When order doesn't matter, divide out orderings.

Sum Rule:
  Size of union of disjoint sets is sum of sizes.

Inclusion/Exclusion:
  Size of union of sets is sum of sizes
    minus the size of intersection.