Today

Review for Midterm.

Propositional logic.

A proposition is a statement that is true or false.

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$?

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$?

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ?

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$?

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Predicate?

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Predicate?

$n > 3$?

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Predicate?

$n > 3$? Predicate: $P(n)$!

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$?

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

$x + y$?

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

$x + y$? No.

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

$x + y$? No. An expression, not a proposition.

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

$x + y$? No. An expression, not a proposition.

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x,y)$!

$x + y$? No. An expression, not a proposition.

Quantifiers:

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

$x + y$? No. An expression, not a proposition.

Quantifiers:

$(\forall x)\ P(x)$. For every x, $P(x)$ is true.

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x,y)$!

$x + y$? No. An expression, not a proposition.

Quantifiers:

$(\forall x)\ P(x)$. For every x, $P(x)$ is true.

$(\exists x)\ P(x)$. There exists an x, where $P(x)$ is true.

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

$x + y$? No. An expression, not a proposition.

Quantifiers:

$(\forall x)\ P(x)$. For every x, $P(x)$ is true.

$(\exists x)\ P(x)$. There exists an x, where $P(x)$ is true.

When all variables are quantified, the statement turns into a proposition.

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!

$3 = 5$? Proposition!

3 ? Not a proposition!

$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a proposition.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

$x + y$? No. An expression, not a proposition.

Quantifiers:

$(\forall x)\ P(x)$. For every x, $P(x)$ is true.

$(\exists x)\ P(x)$. There exists an x, where $P(x)$ is true.

When all variables are quantified, the statement turns into a proposition.

$(\forall n \in N), n^2 \geq n$. $(\forall x \in R)(\exists y \in R) y > x$.

Propositional logic.

A proposition is a statement that is true or false.

Propositions?

$3 = 4 - 1$? Proposition!
$3 = 5$? Proposition!
3 ? Not a proposition!
$n = 3$? Not a proposition...but a predicate.

Predicate: Statement with free variable(s).
Example: $x = 3$ Given a value for x, becomes a proposition.
Predicate?
$n > 3$? Predicate: $P(n)$!
$x = y$? Predicate: $P(x, y)$!
$x + y$? No. An expression, not a proposition.

Quantifiers:
$(\forall x)\ P(x)$. For every x, $P(x)$ is true.
$(\exists x)\ P(x)$. There exists an x, where $P(x)$ is true.
When all variables are quantified, the statement turns into a proposition.

$(\forall n \in N), n^2 \geq n$. $(\forall x \in R)(\exists y \in R) y > x$.

Connecting Propositions with Boolean Operators

$A \wedge B$, $A \vee B$, $\neg A$, $A \implies B$.

Connecting Propositions with Boolean Operators

$A \wedge B$, $A \vee B$, $\neg A$, $A \implies B$.

Propositional Expressions and Logical Equivalence

Connecting Propositions with Boolean Operators

$A \wedge B$, $A \vee B$, $\neg A$, $A \implies B$.

Propositional Expressions and Logical Equivalence

$(A \implies B) \equiv (\neg A \vee B)$

Connecting Propositions with Boolean Operators

$A \wedge B$, $A \vee B$, $\neg A$, $A \implies B$.

Propositional Expressions and Logical Equivalence

$(A \implies B) \equiv (\neg A \vee B)$
$\neg(A \vee B) \equiv (\neg A \wedge \neg B)$

Connecting Propositions with Boolean Operators

$A \wedge B$, $A \vee B$, $\neg A$, $A \implies B$.

Propositional Expressions and Logical Equivalence

$(A \implies B) \equiv (\neg A \vee B)$

$\neg(A \vee B) \equiv (\neg A \wedge \neg B)$

Connecting Propositions with Boolean Operators

$A \wedge B$, $A \vee B$, $\neg A$, $A \implies B$.

Propositional Expressions and Logical Equivalence

$(A \implies B) \equiv (\neg A \vee B)$
$\neg(A \vee B) \equiv (\neg A \wedge \neg B)$

Proofs: truth table or manipulation of known formulas.

Connecting Propositions with Boolean Operators

$A \wedge B$, $A \vee B$, $\neg A$, $A \implies B$.

Propositional Expressions and Logical Equivalence

$(A \implies B) \equiv (\neg A \vee B)$
$\neg(A \vee B) \equiv (\neg A \wedge \neg B)$

Proofs: truth table or manipulation of known formulas.

Boolean simplification rules - De Morgan's law, commutativity, associativity, etc.

Connecting Propositions with Boolean Operators

$A \land B$, $A \lor B$, $\neg A$, $A \implies B$.

Propositional Expressions and Logical Equivalence

$$(A \implies B) \equiv (\neg A \lor B)$$
$$\neg(A \lor B) \equiv (\neg A \land \neg B)$$

Proofs: truth table or manipulation of known formulas.

Boolean simplification rules - De Morgan's law, commutativity, associativity, etc.

$$(\forall x)(P(x) \land Q(x)) \equiv (\forall x)P(x) \land (\forall x)Q(x)$$

Proofs!

Direct: $P \implies Q$

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even?

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies$ a^2 is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$a^2 = 2(2k^2)$

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies$ a^2 is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies$ a^2 is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies$ a^2 is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies$ a^2 is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies$ a is odd.

Contrapositive: a is even $\implies$ a^2 is even.

Contradiction: P

$\neg P \implies$ **false**

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Useful for prove something does not exist:

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies$ a^2 is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies$ a is odd.

Contrapositive: a is even $\implies$ a^2 is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$ does not exist.

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$ does not exist.

Example: finite set of primes

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$ does not exist.

Example: finite set of primes does not exist.

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$ does not exist.

Example: finite set of primes does not exist. Example: rogue couple does not exist.

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$ does not exist.

Example: finite set of primes does not exist. Example: rogue couple does not exist.

Proofs!

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$ does not exist.

Example: finite set of primes does not exist. Example: rogue couple does not exist.

Induction.

$$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$$

Induction.

$$P(0) \land ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$$

Thm: For all $n \geq 1$, $8 | 3^{2n} - 1$.

Induction.

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8|3^{2n} - 1$.

Induction on n.

Induction.

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8|3^{2n} - 1$.

Induction on n.

Base: $8|3^2 - 1$.

Induction.

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8|3^{2n}-1$.

Induction on n.

Base: $8|3^2-1$.

Induction.

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\, P(n)$.

Thm: For all $n \geq 1$, $8 | 3^{2n} - 1$.

Induction on n.

Base: $8 | 3^2 - 1$.

Induction Hypothesis: True for some n.
$(3^{2n} - 1 = 8d)$

Induction.

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n)$.

Thm: For all $n \geq 1$, $8 | 3^{2n} - 1$.

Induction on n.

Base: $8 | 3^2 - 1$.

Induction Hypothesis: True for some n.
($3^{2n} - 1 = 8d$)

Induction Step:

Induction.

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8|3^{2n}-1$.

Induction on n.

Base: $8|3^2-1$.

Induction Hypothesis: True for some n.
($3^{2n}-1 = 8d$)

Induction Step:

$3^{2n+2}-1 =$

Induction.

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8|3^{2n} - 1$.

Induction on n.

Base: $8|3^2 - 1$.

Induction Hypothesis: True for some n.
$(3^{2n} - 1 = 8d)$

Induction Step:

$$3^{2n+2} - 1 = 9(3^{2n}) - 1$$

Induction.

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8|3^{2n} - 1$.

Induction on n.

Base: $8|3^2 - 1$.

Induction Hypothesis: True for some n.
$(3^{2n} - 1 = 8d)$

Induction Step:

$3^{2n+2} - 1 = 9(3^{2n}) - 1$ (by induction hypothesis)

Induction.

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8 | 3^{2n} - 1$.

Induction on n.

Base: $8 | 3^2 - 1$.

Induction Hypothesis: True for some n.
$(3^{2n} - 1 = 8d)$

Induction Step:

$$\begin{aligned} 3^{2n+2} - 1 &= 9(3^{2n}) - 1 \quad \text{(by induction hypothesis)} \\ &= 9(8d+1) - 1 \end{aligned}$$

Induction.

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8|3^{2n}-1$.

Induction on n.

Base: $8|3^2-1$.

Induction Hypothesis: True for some n.
$(3^{2n}-1 = 8d)$

Induction Step:

$$\begin{aligned} 3^{2n+2}-1 &= 9(3^{2n})-1 \quad \text{(by induction hypothesis)} \\ &= 9(8d+1)-1 \\ &= 72d+8 \end{aligned}$$

Induction.

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8|3^{2n}-1$.

Induction on n.

Base: $8|3^2-1$.

Induction Hypothesis: True for some n.
$(3^{2n}-1 = 8d)$

Induction Step:

$$\begin{aligned} 3^{2n+2}-1 &= 9(3^{2n})-1 \quad \text{(by induction hypothesis)} \\ &= 9(8d+1)-1 \\ &= 72d+8 \\ &= 8(9d+1) \end{aligned}$$

Induction.

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8|3^{2n}-1$.

Induction on n.

Base: $8|3^2-1$.

Induction Hypothesis: True for some n.
$(3^{2n}-1 = 8d)$

Induction Step:

$$\begin{aligned} 3^{2n+2}-1 &= 9(3^{2n})-1 \quad \text{(by induction hypothesis)} \\ &= 9(8d+1)-1 \\ &= 72d+8 \\ &= 8(9d+1) \end{aligned}$$

Divisible by 8.

Induction.

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n)$.

Thm: For all $n \geq 1$, $8 | 3^{2n} - 1$.

Induction on n.

Base: $8 | 3^2 - 1$.

Induction Hypothesis: True for some n.
$(3^{2n} - 1 = 8d)$

Induction Step:

$$\begin{aligned} 3^{2n+2} - 1 &= 9(3^{2n}) - 1 \quad \text{(by induction hypothesis)} \\ &= 9(8d+1) - 1 \\ &= 72d + 8 \\ &= 8(9d+1) \end{aligned}$$

Divisible by 8.

□

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Stable Marriage: a study in definitions and WOP.

n-men, *n*-women.

Each person has completely ordered preference list

Stable Marriage: a study in definitions and WOP.

n-men, *n*-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.

Stable Marriage: a study in definitions and WOP.

n-men, *n*-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs?

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Rogue Couple in a pairing.
A m_j and w_k who like each other more than their current partners

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Rogue Couple in a pairing.
A m_j and w_k who like each other more than their current partners

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Rogue Couple in a pairing.
A m_j and w_k who like each other more than their current partners

Stable Pairing.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Rogue Couple in a pairing.
A m_j and w_k who like each other more than their current partners

Stable Pairing.
Pairing with no rogue couples.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Rogue Couple in a pairing.
A m_j and w_k who like each other more than their current partners

Stable Pairing.
Pairing with no rogue couples.

Does stable pairing exist?

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Rogue Couple in a pairing.
A m_j and w_k who like each other more than their current partners

Stable Pairing.
Pairing with no rogue couples.

Does stable pairing exist?

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Rogue Couple in a pairing.
A m_j and w_k who like each other more than their current partners

Stable Pairing.
Pairing with no rogue couples.

Does stable pairing exist?

No, for roommates problem.

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:

Every man proposes to favorite woman who has not yet rejected him.

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:

Every man proposes to favorite woman who has not yet rejected him.

Every woman rejects all but best of the men who propose.

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:
Man **crosses off** woman who rejected him.

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject."

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women.

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:
Every day, if man on string for woman, any future man on string is better.

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:
Every day, if man on string for woman, any future man on string is better.

Stability:

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:
Every day, if man on string for woman, any future man on string is better.

Stability:
No rogue couple.

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:
Every day, if man on string for woman, any future man on string is better.

Stability:
No rogue couple.
suppose rogue couple (M,W) $\implies$ M proposed to W

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:
Every day, if man on string for woman, any future man on string is better.

Stability:
No rogue couple.
suppose rogue couple (M,W) $\implies$ M proposed to W
$\implies$ W ended up with someone she liked better than M.

Stable Marriage Algorithm (SMA).

(Also called Traditional Marriage Algorithm)

Each Day:
Every man proposes to favorite woman who has not yet rejected him.
Every woman rejects all but best of the men who propose.

Useful Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:
Every day, if man on string for woman, any future man on string is better.

Stability:
No rogue couple.
suppose rogue couple (M,W) $\implies$ M proposed to W
$\implies$ W ended up with someone she liked better than M.
Not rogue couple!

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
- Not necessarily first in list.
- Possibly no stable pairing with that partner.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
- Not necessarily first in list.
- Possibly no stable pairing with that partner.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
  Not necessarily first in list.
  Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
  Not necessarily first in list.
  Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: SMA produces male optimal pairing, S.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
  Not necessarily first in list.
  Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: SMA produces male optimal pairing, S.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: SMA produces male optimal pairing, S.

Man optimal $\implies$ Woman pessimal.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
  Not necessarily first in list.
  Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: SMA produces male optimal pairing, S.

  Man optimal $\implies$ Woman pessimal.
  Woman optimal $\implies$ Man pessimal.

Graph Theory!

$G = (V, E)$

Graph Theory!

$G = (V, E)$
V - set of vertices.

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.
Focus on simple graphs (at most one edge from a vertex to another)

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.
Focus on simple graphs (at most one edge from a vertex to another)

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.
Focus on simple graphs (at most one edge from a vertex to another)

Undirected: no ordering to edge.

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.
Focus on simple graphs (at most one edge from a vertex to another)

Undirected: no ordering to edge. Directed: ordered pair of vertices.

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.
Focus on simple graphs (at most one edge from a vertex to another)

Undirected: no ordering to edge. Directed: ordered pair of vertices.

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.
Focus on simple graphs (at most one edge from a vertex to another)

Undirected: no ordering to edge.   Directed: ordered pair of vertices.

Adjacent, Incident, Degree.

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.
Focus on simple graphs (at most one edge from a vertex to another)

Undirected: no ordering to edge. Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.
Focus on simple graphs (at most one edge from a vertex to another)

Undirected: no ordering to edge.   Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.
Focus on simple graphs (at most one edge from a vertex to another)

Undirected: no ordering to edge.   Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.

Pair of Vertices are Connected:

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.
Focus on simple graphs (at most one edge from a vertex to another)

Undirected: no ordering to edge.   Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.

Pair of Vertices are Connected:
If there is a (simple) path between them.

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.
Focus on simple graphs (at most one edge from a vertex to another)

Undirected: no ordering to edge. Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.

Pair of Vertices are Connected:
If there is a (simple) path between them.
Related notions: cycle, walk, tour

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.
Focus on simple graphs (at most one edge from a vertex to another)

Undirected: no ordering to edge. Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.

Pair of Vertices are Connected:
If there is a (simple) path between them.
Related notions: cycle, walk, tour

Connected Component: maximal set of connected vertices.

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.
Focus on simple graphs (at most one edge from a vertex to another)

Undirected: no ordering to edge. Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.

Pair of Vertices are Connected:
If there is a (simple) path between them.
Related notions: cycle, walk, tour

Connected Component: maximal set of connected vertices.

Graph Theory!

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.
Focus on simple graphs (at most one edge from a vertex to another)

Undirected: no ordering to edge. Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.

Pair of Vertices are Connected:
If there is a (simple) path between them.
Related notions: cycle, walk, tour

Connected Component: maximal set of connected vertices.

Connected Graph: one connected component.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:
  Take a walk.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:
 Take a walk.
  Property: return to starting point.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Take a walk.

Property: return to starting point.

Proof Idea: Even degree.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Take a walk.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Take a walk.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Put together.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Take a walk.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Put together.

Property: walk visits every component.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Take a walk.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Put together.

Property: walk visits every component.

Proof Idea: Original graph connected.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Take a walk.

Property: return to starting point.

Proof Idea: Even degree.

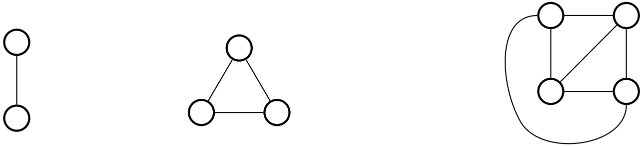
Recurse on connected components.

Put together.

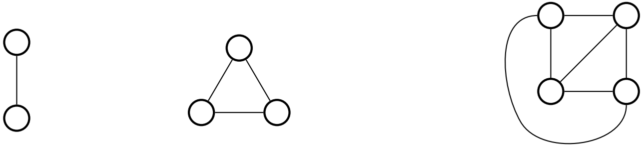
Property: walk visits every component.

Proof Idea: Original graph connected.

Graph Types: Complete Graph.

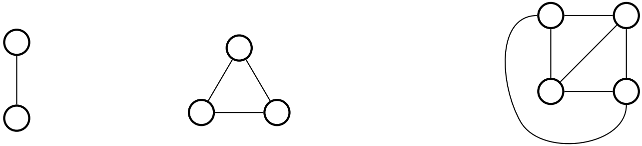


Graph Types: Complete Graph.



K_n, $|V| = n$

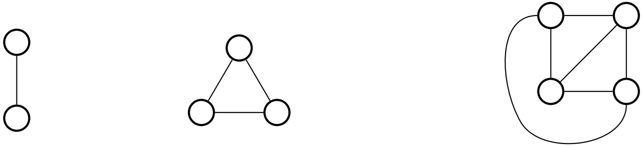
Graph Types: Complete Graph.



K_n, $|V| = n$

every edge present.

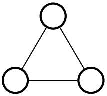
Graph Types: Complete Graph.

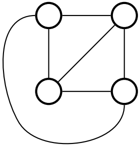


K_n, $|V| = n$

every edge present.
degree of vertex?

Graph Types: Complete Graph.

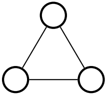


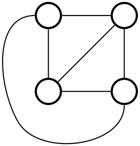


K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

Graph Types: Complete Graph.



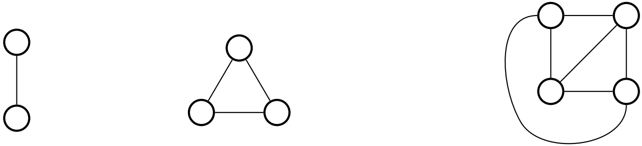


K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

Very connected.

Graph Types: Complete Graph.

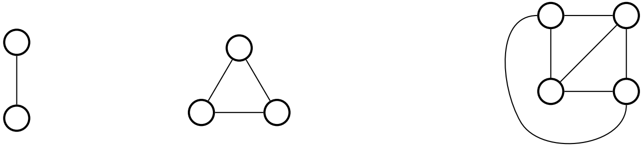


K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

Very connected.
Lots of edges:

Graph Types: Complete Graph.

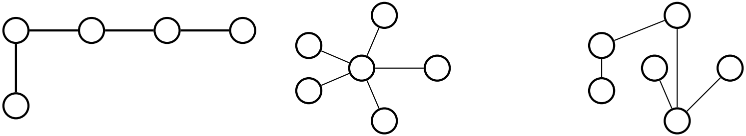


K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

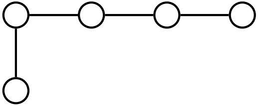
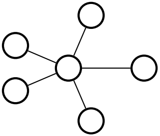
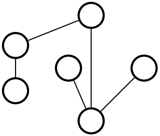
Very connected.
Lots of edges: $n(n-1)/2$.

Trees.



Definitions:

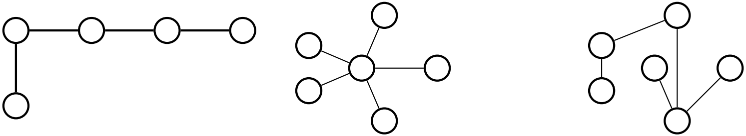
Trees.

Definitions:

A connected graph without a cycle.

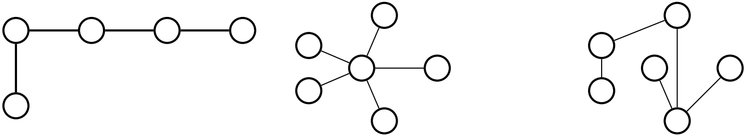
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.

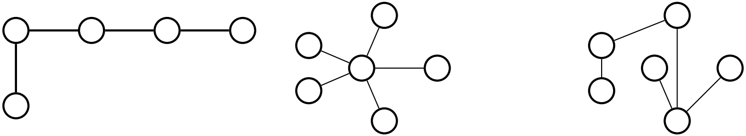
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.

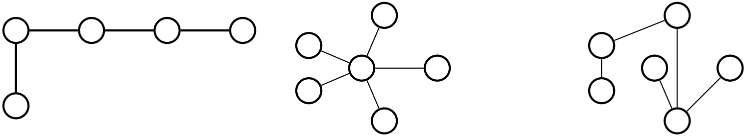
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V|-1$ edges.
A connected graph where any edge removal disconnects it.
A connected acyclic graph where any edge addition creates a cycle.

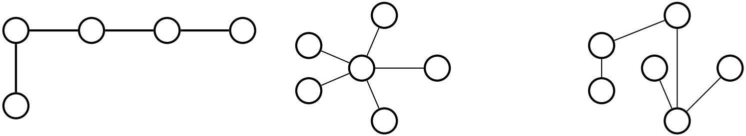
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V|-1$ edges.
A connected graph where any edge removal disconnects it.
A connected acyclic graph where any edge addition creates a cycle.

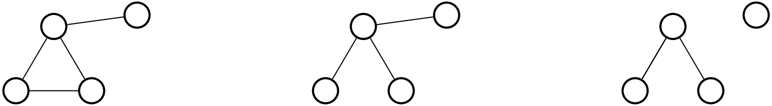
Trees.



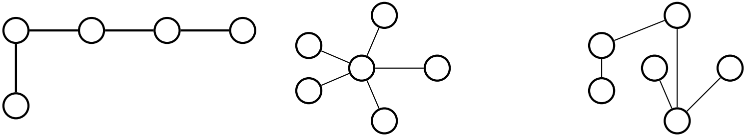
Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
A connected acyclic graph where any edge addition creates a cycle.

To tree or not to tree!



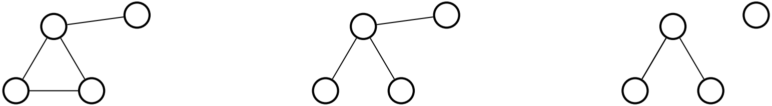
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
A connected acyclic graph where any edge addition creates a cycle.

To tree or not to tree!



Minimally connected, minimum number of edges to connect.

Hypercube

Hypercubes.

Hypercube

Hypercubes. Really connected.

Hypercube

Hypercubes. Really connected. $|V| \log |V|$ edges!

Hypercube

Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

Hypercube

Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

Hypercube

Hypercubes. Really connected. $|V|\log|V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$

Hypercube

Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$
$|V| = \{0, 1\}^n$,

Hypercube

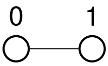
Hypercubes. Really connected. $|V|\log|V|$ edges!
Also represents bit-strings nicely.

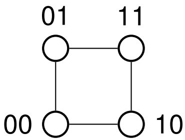
$G = (V, E)$
$|V| = \{0, 1\}^n$,
$|E| = \{(x, y) | x \text{ and } y \text{ differ in one bit position.}\}$

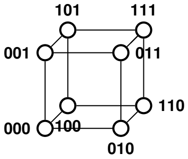
Hypercube

Hypercubes. Really connected. $|V|\log|V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$
$|V| = \{0,1\}^n$,
$|E| = \{(x,y)|x$ and y differ in one bit position.$\}$







Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

Recursive Definition.

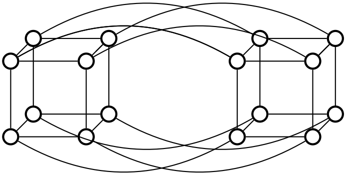
A 0-dimensional hypercube is a node labelled with the empty string of bits.

An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.

Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

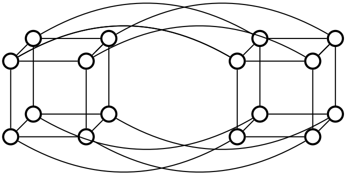
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

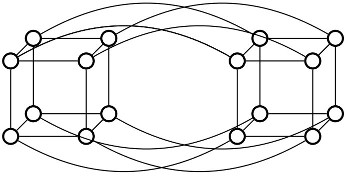
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ $(1x)$ with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

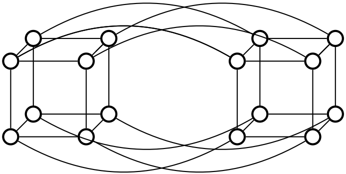
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

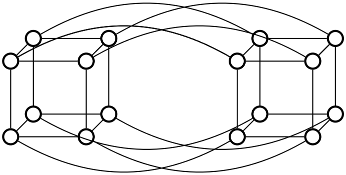
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

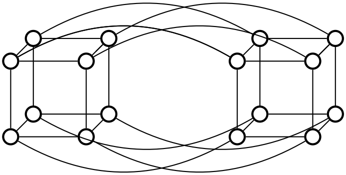
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Hypercube:properties

Hamiltonian (Rudrata) Cycle: cycle that visits every node.

Hypercube:properties

Hamiltonian (Rudrata) Cycle: cycle that visits every node.
  Eulerian?

Hypercube:properties

Hamiltonian (Rudrata) Cycle: cycle that visits every node.
Eulerian? If n is even.

Hypercube:properties

Hamiltonian (Rudrata) Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.

Hypercube:properties

Hamiltonian (Rudrata) Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
"Best" cut?

Hypercube:properties

Hamiltonian (Rudrata) Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
"Best" cut? Cut apart subcubes:

Hypercube:properties

Hamiltonian (Rudrata) Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
"Best" cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

...Modular Arithmetic...

Arithmetic modulo m.
  Elements of equivalence classes of integers.

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0, \ldots, m-1\}$

...Modular Arithmetic...

Arithmetic modulo m.
  Elements of equivalence classes of integers.
    $\{0,\ldots,m-1\}$
    and integer $i \equiv a \pmod{m}$

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0,\ldots,m-1\}$

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer k.

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0, \ldots, m-1\}$

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer k.

or if the remainder of i divided by m is a.

...Modular Arithmetic...

Arithmetic modulo m.
  Elements of equivalence classes of integers.
    $\{0,\ldots,m-1\}$
    and integer $i \equiv a \pmod{m}$
      if $i = a + km$ for integer k.
       or if the remainder of i divided by m is a.

...Modular Arithmetic...

Arithmetic modulo m.
  Elements of equivalence classes of integers.
    $\{0,\ldots,m-1\}$
    and integer $i \equiv a \pmod{m}$
      if $i = a + km$ for integer k.
       or if the remainder of i divided by m is a.

Can do calculations by taking remainders
  at the beginning,

...Modular Arithmetic...

Arithmetic modulo m.
  Elements of equivalence classes of integers.
    $\{0, \ldots, m-1\}$
    and integer $i \equiv a \pmod{m}$
      if $i = a + km$ for integer k.
        or if the remainder of i divided by m is a.

Can do calculations by taking remainders
  at the beginning,
    in the middle

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58+32=90=6 \pmod 7$
$58+32=2+4=6 \pmod 7$
$58+32=2+-3=-1=6 \pmod 7$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58+32=90=6 \pmod{7}$
$58+32=2+4=6 \pmod{7}$
$58+32=2+-3=-1=6 \pmod{7}$

Negative numbers work the way you are used to.

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

Negative numbers work the way you are used to.
$-3 = 0 - 3 = 7 - 3 = 4 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

Negative numbers work the way you are used to.
$-3 = 0 - 3 = 7 - 3 = 4 \pmod{7}$

Midterm format

Time: 120 minutes.

Midterm format

Time: 120 minutes.

Many short answers.

Midterm format

Time: 120 minutes.

Many short answers.
Get at ideas that we study.

Midterm format

Time: 120 minutes.

Many short answers.
  Get at ideas that we study.
   Know material well:

Midterm format

Time: 120 minutes.

Many short answers.
  Get at ideas that we study.
   Know material well: fast,

Midterm format

Time: 120 minutes.

Many short answers.
Get at ideas that we study.
Know material well: fast, correct.

Midterm format

Time: 120 minutes.

Many short answers.
  Get at ideas that we study.
   Know material well: fast, correct.
   Know material medium:

Midterm format

Time: 120 minutes.

Many short answers.
Get at ideas that we study.
Know material well: fast, correct.
Know material medium: slower,

Midterm format

Time: 120 minutes.

Many short answers.
Get at ideas that we study.
Know material well: fast, correct.
Know material medium: slower, less correct.

Midterm format

Time: 120 minutes.

Many short answers.

Get at ideas that we study.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	

Midterm format

Time: 120 minutes.

Many short answers.
Get at ideas that we study.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	Uh oh.

Midterm format

Time: 120 minutes.

Many short answers.
  Get at ideas that we study.
   Know material well: fast, correct.
   Know material medium: slower, less correct.
   Know material not so well: Uh oh.

Some longer questions.

Midterm format

Time: 120 minutes.

Many short answers.
  Get at ideas that we study.
   Know material well: fast, correct.
   Know material medium: slower, less correct.
   Know material not so well: Uh oh.

Some longer questions.
  Proofs,

Midterm format

Time: 120 minutes.

Many short answers.
Get at ideas that we study.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	Uh oh.

Some longer questions.
Proofs, think about algorithms, properties, etc.

Midterm format

Time: 120 minutes.

Many short answers.
- Get at ideas that we study.
  - Know material well: fast, correct.
  - Know material medium: slower, less correct.
  - Know material not so well: Uh oh.

Some longer questions.
- Proofs, think about algorithms, properties, etc.

Midterm format

Time: 120 minutes.

Many short answers.
 Get at ideas that we study.
  Know material well: fast, correct.
  Know material medium: slower, less correct.
  Know material not so well: Uh oh.

Some longer questions.
 Proofs, think about algorithms, properties, etc.

Not so much calculation.

Midterm format

Time: 120 minutes.

Many short answers.
Get at ideas that we study.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	Uh oh.

Some longer questions.
Proofs, think about algorithms, properties, etc.

Not so much calculation.

Good Luck!!!