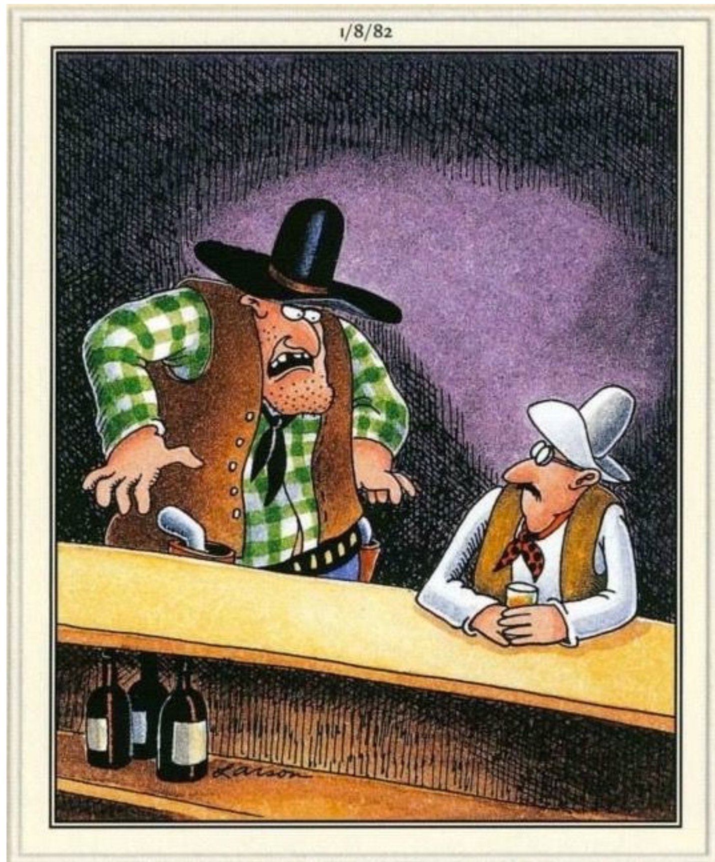
CSE 311: Foundations of Computing

Lecture 12: Modular Exponentiation, Set Theory



"I *asked* you a question, buddy. ... What's the square root of 5,248?"

Last class: Euclid's Algorithm for GCD

Repeatedly use $\gcd(a, b) = \gcd(b, a \bmod b)$ to reduce numbers until you get $\gcd(a, 0) = a$.

Equations with recursive calls:

gcd(660,126) = gcd(126, 660 mod 126) = gcd(126, 30)
= gcd(30, 126 mod 30) = gcd(30, 6)
= gcd(6, 30 mod 6) = gcd(6, 0)
= 6

Tableau form (which is much easier to work with and will be more useful):

660 = 5 * 126 + 30
126 = 4 * 30 + 6
30 = 5 * 6 + 0

Each line computes both quotient and remainder of the shifted numbers

Last class: Extended Euclidean algorithm

- Can use Euclid's Algorithm to find s, t such that

$$\gcd(a, b) = sa + tb$$

Example: $a = 35, b = 27$

Compute $\gcd(35, 27)$:

$35 = 1 * 27 + 8$	$8 = 35 - 1 * 27$
$27 = 3 * 8 \quad + 3$	$3 = 27 - 3 * 8$
$8 = 2 * 3 \quad + 2$	$2 = 8 - 2 * 3$
$3 = 1 * 2 \quad + 1$	$1 = 3 - 1 * 2$
$2 = 2 * 1 \quad + 0$	

Last class: Extended Euclidean algorithm

- Can use Euclid's Algorithm to find s, t such that

$$\gcd(a, b) = sa + tb$$

Example: $a = 35$, $b = 27$

Use equations to substitute back

$$8 = 35 - 1 * 27$$
$$3 = 27 - 3 * 8$$
$$2 = 8 - 2 * 3$$
$$1 = 3 - 1 * 2$$

$$\begin{aligned}
1 &= 3 - 1 * 2 \\
&= 3 - 1 * (8 - 2 * 3) \\
&= 3 - 8 + 2 * 3 \\
&= (-1) * 8 + 3 * 3 \\
&= (-1) * 8 + 3 * (27 - 3 * 8) \\
&= (-1) * 8 + 3 * 27 + (-9) * 8 \\
&= 3 * 27 + (-10) * 8 \\
&= 3 * 27 + (-10) * (35 - 1 * 27) \\
&= 3 * 27 + (-10) * 35 + 10 * 27 \\
&= (-10) * 35 + 13 * 27
\end{aligned}$$

Optional Check:

$(-10) * 35 = -350$

$13 * 27 = 351$

Last class: Multiplicative inverse $(\text{mod } m)$

Let $0 \leq a, b < m$. Then, b is the *multiplicative inverse of a (modulo m)* iff $ab \equiv 1 \pmod{m}$.

x	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	0	0	0	0	0	0
1	0	**1**	2	3	4	5	6
2	0	2	4	6	**1**	3	5
3	0	3	6	2	5	**1**	4
4	0	4	**1**	5	2	6	3
5	0	5	3	**1**	6	4	2
6	0	6	5	4	3	2	**1**

mod 7

x	0	**1**	2	**3**	4	5	6	**7**	8	**9**
0	0	0	0	0	0	0	0	0	0	0
1	0	**1**	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	**1**	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	**1**	8	5	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	**1**

mod 10

Last class: Multiplicative inverse $(\mathrm{mod}\ m)$

Let $0 \leq a, b < m$. Then, b is the *multiplicative inverse of* a *(modulo* m*)* iff $ab \equiv 1 \pmod{m}$.

This can't exist if a and m have a common factor >1.

Idea: b is like $a^{-1} \pmod{m}$ so multiplying by b is equivalent to dividing by a.

x	0	**1**	2	**3**	4	5	6	**7**	8	**9**
0	0	0	0	0	0	0	0	0	0	0
1	0	**1**	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	**1**	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	**1**	8	5	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	**1**

mod 10

Finding multiplicative inverse $\bmod m$

Suppose that $\gcd(a, m) = 1$.

Using Extended Euclidean Algorithm

find integers s and t such that $sa + tm = 1$.

Therefore $sa \equiv 1 \pmod{m}$.

The multiplicative inverse b of a modulo m must also satisfy $0 \leq b < m$ so we set $b = s \bmod m$.

It works since $ba \equiv sa \equiv 1 \pmod{m}$

Example

Solve: $7x \equiv 1 \pmod{26}$

Example

Solve: $7x \equiv 1 \pmod{26}$

First compute and check that $\gcd(26, 7) = 1$

$$26 = 3 * 7 + 5$$
$$7 = 1 * 5 + 2$$
$$5 = 2 * 2 + 1$$
$$2 = 2 * 1 + 0$$

Example

Solve: $7x \equiv 1 \pmod{26}$

Then rewrite equations in form for substitution

$26 = 3 * 7 + 5$	$5 = 26 - 3 * 7$
$7 = 1 * 5 + 2$	$2 = 7 - 1 * 5$
$5 = 2 * 2 + 1$	$1 = 5 - 2 * 2$
$2 = 2 * 1 + 0$	

Example

Solve: $7x \equiv 1 \pmod{26}$

Apply substitutions from bottom to top.

$$26 = 3 * 7 + 5 \qquad 5 = 26 - 3 * 7$$
$$7 = 1 * 5 + 2 \qquad 2 = 7 - 1 * 5$$
$$5 = 2 * 2 + 1 \qquad 1 = 5 - 2 * 2$$
$$2 = 2 * 1 + 0$$

$$\begin{aligned}
1 &= 5 - 2 * 2 \\
&= 5 - 2 * (7 - 1 * 5) \\
&= (-2) * 7 + 3 * 5 \\
&= (-2) * 7 + 3 * (26 - 3 * 7) \\
&= (-11) * 7 + 3 * 26
\end{aligned}$$

Example

Solve: $7x \equiv 1 \pmod{26}$

Read off coefficient and reduce modulo 26.

$$26 = 3 * 7 + 5 \qquad 5 = 26 - 3 * 7$$
$$7 = 1 * 5 + 2 \qquad 2 = 7 - 1 * 5$$
$$5 = 2 * 2 + 1 \qquad 1 = 5 - 2 * 2$$
$$2 = 2 * 1 + 0$$

$$\begin{aligned} 1 &= 5 - 2 * 2 \\ &= 5 - 2 * (7 - 1 * 5) \\ &= (-2) * 7 + 3 * 5 \\ &= (-2) * 7 + 3 * (26 - 3 * 7) \\ &= (-11) * 7 + 3 * 26 \end{aligned}$$

Multiplicative inverse of 7 modulo 26

Now $(-11) \bmod 26 = 15$**.** **So,** $x = 15 + 26k$ **for integer** k.

Example of a more general equation

Now solve: $7y \equiv 3 \pmod{26}$

We already computed that 15 is the multiplicative inverse of 7 modulo 26. That is, $7 \cdot 15 \equiv 1 \pmod{26}$

If y is a solution, then multiplying by 15 we have

$$15 \cdot 7 \cdot y \equiv 15 \cdot 3 \pmod{26}$$

Substituting $15 \cdot 7 \equiv 1 \pmod{26}$ on the left gives

$$y = 1 \cdot y \equiv 15 \cdot 3 \equiv 19 \pmod{26}$$

This shows that <u>every</u> solution y is congruent to 19.

Example of a more general equation

Now solve: $7y \equiv 3 \pmod{26}$

Multiplying both sides of $y \equiv 19 \pmod{26}$ by 7 gives

$$7y \equiv 7 \cdot 19 \equiv 3 \pmod{26}$$

So, <u>any</u> $y \equiv 19 \pmod{26}$ is a solution.

Thus, the set of numbers of the form $y = 19 + 26k$, for any integer k, are <u>exactly</u> solutions of this equation.

Math mod a prime is especially nice

$\gcd(a, m) = 1$ **if** m **is prime and** $0 < a < m$ **so can always solve these equations mod a prime.**

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

x	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	0	0	0	0	0	0
1	0	**1**	2	3	4	5	6
2	0	2	4	6	**1**	3	5
3	0	3	6	2	5	**1**	4
4	0	4	**1**	5	2	6	3
5	0	5	3	**1**	6	4	2
6	0	6	5	4	3	2	**1**

mod 7

Attack on RSA security with GCD

- RSA *public key* includes m that is the product of two large *randomly chosen* primes p, q
  - Everyone can see all the public keys (millions)
  - Security depends on keeping p and q secret
  - OK since factoring m seems very hard

- In 2012 a new attack using GCD broke a huge number of RSA public keys!
  - Weak keys: Algorithms/devices cut corners: Skimped on random bits or size of primes

Attack on RSA security with GCD

Weak keys: few random bits

- Few enough that some public keys m_1 and m_2 happen to share just one of their two factors:

$$m_1 = pq \text{ and } m_2 = pr$$

- Then can break both since $p = \gcd(m_1, m_2)$

2012: 11 million RSA keys, 23,500 broken

2016: 1024-bit RSA keys available from Internet

- 26 million keys, 63,500 broken

2019: 750 million RSA keys, 250,000 broken

- IoT (Internet of Things) devices often the culprit

RSA Relies on Modular Exponentiation

x	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

a	a^1	a^2	a^3	a^4	a^5	a^6
1	1	1	1	1	1	1
2	2	4	1	2	4	1
3	3	2	6	4	5	1
4	4	2	1	4	2	1
5	5	4	6	2	3	1
6	6	1	6	1	6	1

mod 7

Modular Exponentiation: (Essential for RSA)

- **Compute** 78365^{80429}

- **Compute** $78365^{80429} \bmod 104729$

- **Output is small**
  - need to keep intermediate results small

Small Multiplications

By the multiplicative property modulo $\boldsymbol{m}$, if you want to compute $\boldsymbol{ab} \bmod \boldsymbol{m}$ then you can do the following:

1. Reduce $\boldsymbol{a}$ and $\boldsymbol{b}$ modulo $\boldsymbol{m}$ to get $\boldsymbol{a} \bmod \boldsymbol{m}$ and $\boldsymbol{b} \bmod \boldsymbol{m}$
2. Multiply to produce $\boldsymbol{c} = (\boldsymbol{a} \bmod \boldsymbol{m})(\boldsymbol{b} \bmod \boldsymbol{m})$
3. Output $\boldsymbol{c} \bmod \boldsymbol{m}$

$a = qm + r$

Claim: $\boldsymbol{c} \bmod \boldsymbol{m} = \boldsymbol{ab} \bmod \boldsymbol{m}$

Proof: Just need to show that $\boldsymbol{c} \equiv \boldsymbol{ab} \pmod{\boldsymbol{m}}$.

That follows from $(\boldsymbol{a} \bmod \boldsymbol{m}) \equiv \boldsymbol{a} \pmod{\boldsymbol{m}}$

$(\boldsymbol{b} \bmod \boldsymbol{m}) \equiv \boldsymbol{b} \pmod{\boldsymbol{m}}$

and the multiplicative property since $\boldsymbol{c}$ is the product of the left sides and $\boldsymbol{ab}$ is the product of the right sides. ■

Repeated Squaring – small and fast

Then we have $\boldsymbol{ab} \bmod \boldsymbol{m} = \big((\boldsymbol{a} \bmod \boldsymbol{m})(\boldsymbol{b} \bmod \boldsymbol{m})\big) \bmod \boldsymbol{m}$

So $\boldsymbol{a}^2 \bmod \boldsymbol{m} = (\boldsymbol{a} \bmod \boldsymbol{m})^2 \bmod \boldsymbol{m}$

and $\boldsymbol{a}^4 \bmod \boldsymbol{m} = (\boldsymbol{a}^2 \bmod \boldsymbol{m})^2 \bmod \boldsymbol{m}$

and $\boldsymbol{a}^8 \bmod \boldsymbol{m} = (\boldsymbol{a}^4 \bmod \boldsymbol{m})^2 \bmod \boldsymbol{m}$

and $\boldsymbol{a}^{16} \bmod \boldsymbol{m} = (\boldsymbol{a}^8 \bmod \boldsymbol{m})^2 \bmod \boldsymbol{m}$

and $\boldsymbol{a}^{32} \bmod \boldsymbol{m} = (\boldsymbol{a}^{16} \bmod \boldsymbol{m})^2 \bmod \boldsymbol{m}$

Can compute $\boldsymbol{a}^{\boldsymbol{k}} \bmod \boldsymbol{m}$ for $\boldsymbol{k} = \boldsymbol{2}^{\boldsymbol{i}}$ in only $\boldsymbol{i}$ steps

What if $\boldsymbol{k}$ is not a power of $\boldsymbol{2}$?

Fast Modular Exponentiation

Simple Example:

To compute $\boldsymbol{a}^{10} \bmod \boldsymbol{m}$:

Compute $\boldsymbol{a}^2 \bmod \boldsymbol{m} = (\boldsymbol{a} \bmod \boldsymbol{m})^2 \bmod \boldsymbol{m}$

$\boldsymbol{a}^4 \bmod \boldsymbol{m} = (\boldsymbol{a}^2 \bmod \boldsymbol{m})^2 \bmod \boldsymbol{m}$

$\boldsymbol{a}^8 \bmod \boldsymbol{m} = (\boldsymbol{a}^4 \bmod \boldsymbol{m})^2 \bmod \boldsymbol{m}$

Then $\boldsymbol{a}^{10} \bmod \boldsymbol{m} = ((\boldsymbol{a}^8 \bmod \boldsymbol{m})(\boldsymbol{a}^2 \bmod \boldsymbol{m})) \bmod \boldsymbol{m}$

Also $\boldsymbol{a}^{11} \bmod \boldsymbol{m} = \big((\boldsymbol{a}^{10} \bmod \boldsymbol{m})(\boldsymbol{a} \bmod \boldsymbol{m})\big) \bmod \boldsymbol{m}$

Fast Exponentiation Algorithm

80429 in binary is 10011101000101101

$80429 = 2^{16} + 2^{13} + 2^{12} + 2^{11} + 2^9 + 2^5 + 2^3 + 2^2 + 2^0$

$$a^{80429} = a^{2^{16}+2^{13}+2^{12}+2^{11}+2^9+2^5+2^3+2^2+2^0}$$
$$= a^{2^{16}} \cdot a^{2^{13}} \cdot a^{2^{12}} \cdot a^{2^{11}} \cdot a^{2^9} \cdot a^{2^5} \cdot a^{2^3} \cdot a^{2^2} \cdot a^{2^0}$$

$$a^{80429} \bmod m = (a^{2^{16}} \cdot a^{2^{13}} \cdot a^{2^{12}} \cdot a^{2^{11}} \cdot a^{2^9} \cdot a^{2^5} \cdot a^{2^3} \cdot a^{2^2} \cdot a^{2^0}) \bmod m$$
$$= (\ldots(((((a^{2^{16}} \bmod m \cdot$$
$$a^{2^{13}} \bmod m\ ) \bmod m \cdot$$
$$a^{2^{12}} \bmod m) \bmod m \cdot$$
$$a^{2^{11}} \bmod m) \bmod m \cdot$$
$$a^{2^{9}} \bmod m) \bmod m \cdot$$
$$a^{2^{5}} \bmod m) \bmod m \cdot$$
$$a^{2^{3}} \bmod m) \bmod m \cdot$$
$$a^{2^{2}} \bmod m) \bmod m \cdot$$
$$a^{2^{0}} \bmod m)\ \bmod m$$

Uses only 16 + 8 = 24 multiplications

The fast exponentiation algorithm computes $a^k \bmod m$ using $\leq 2\log k$ multiplications $\bmod m$

Fast Exponentiation: $a^k \bmod m$ for all k

Another way....

$$a^{2j} \bmod m = \left(a^j \bmod m\right)^2 \bmod m$$

$$a^{2j+1} \bmod m = \left((a \bmod m) \cdot (a^{2j} \bmod m)\right) \bmod m$$

Recursive Fast Exponentiation

```
public static int FastModExp(int a, int k, int modulus) {

        if (k == 0) {
            return 1;

        } else if ((k % 2) == 0) {
            long temp = FastModExp(a,k/2,modulus);
            return (temp * temp) % modulus;

        } else {
            long temp = FastModExp(a,k-1,modulus);
            return (a * temp) % modulus;
        }
}
```

$$a^{2j} \bmod m = \left(a^{j} \bmod m\right)^{2} \bmod m$$

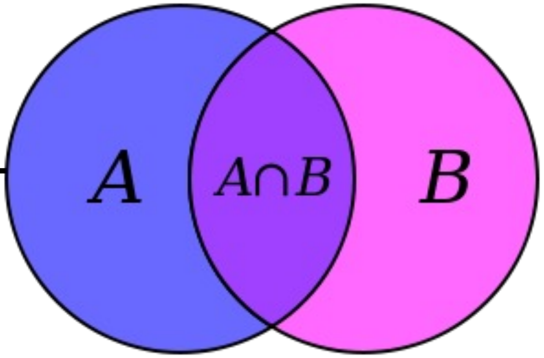
$$a^{2j+1} \bmod m = \left((a \bmod m) \cdot (a^{2j} \bmod m)\right) \bmod m$$

Using Fast Modular Exponentiation

- Your e-commerce web transactions use SSL (Secure Socket Layer) based on RSA encryption
- RSA ...as of 2023
  - Vendor chooses random 1024-bit or 2048-bit primes p, q and 1024/2048-bit exponent e. Computes $m = p \cdot q$
  - Vendor broadcasts (m, e)
  - To send a to vendor, you compute $C = a^e \bmod m$ using *fast modular exponentiation* and send C to the vendor.
  - Using secret p, q the vendor computes d that is the *multiplicative inverse* of e mod $(p-1)(q-1)$.
  - Vendor computes $C^d \bmod m$ using *fast modular exponentiation*.
  - **Fact:** $a = C^d \bmod m$ for $0 < a < m$ unless $p|a$ or $q|a$

Sets

Sets



Sets are collections of objects called elements.

Write $a \in B$ **to say that** a **is an element of set** B**, and** $a \notin B$ **to say that it is not.**

Some simple examples

A = {1}

B = {1, 3, 2}

C = {□, 1}

D = {{17}, 17}

E = {1, 2, 7, cat, dog, ∅, α}

Some Common Sets

$\mathbb{N}$ is the set of **Natural Numbers;** $\mathbb{N}$ = {0, 1, 2, …}

$\mathbb{Z}$ is the set of **Integers**; $\mathbb{Z}$ = {…, -2, -1, 0, 1, 2, …}

$\mathbb{Q}$ is the set of **Rational Numbers**; e.g. ½, -17, 32/48

$\mathbb{R}$ is the set of **Real Numbers**; e.g. 1, -17, 32/48, π, $\sqrt{2}$

[n] is the set **{1, 2, …, n}** when **n** is a natural number

$\varnothing$ = **{}** is the **empty set**; the *only* set with no elements

Sets can be elements of other sets

For example

$A = \{\{1\},\{2\},\{1,2\},\emptyset\}$

$B = \{1,2\}$

Then $B \in A$.

Definitions

- A and B are *equal* if they have the same elements

$$A = B \ := \ \forall x\, (x \in A \leftrightarrow x \in B)$$

- A is a *subset* of B if every element of A is also in B

$$A \subseteq B \ := \ \forall x\, (x \in A \rightarrow x \in B)$$

- Notes:

$$(A = B) \equiv (A \subseteq B) \wedge (B \subseteq A)$$

$A \supseteq B$ means $B \subseteq A$

$A \subset B$ means $A \subseteq B$ but $A \neq B$

Definition: Equality

A and B are *equal* if they have the same elements

$$A = B \;:=\; \forall x\,(x \in A \leftrightarrow x \in B)$$

A = {1, 2, 3}
B = {3, 4, 5}
C = {3, 4}
D = {4, 3, 3}
E = {3, 4, 3}
F = {4, {3}}

Which sets are equal to each other?

Definition: Subset

A is a *subset* of B if every element of A is also in B

$$A \subseteq B \ := \ \forall x\,(x \in A \rightarrow x \in B)$$

A = {1, 2, 3}
B = {3, 4, 5}
C = {3, 4}

<u>QUESTIONS</u>

$\varnothing \subseteq A$?

$A \subseteq B$?

$C \subseteq B$?

Definition: Subset

A is a *subset* of B if every element of A is also in B

$$A \subseteq B \ := \ \forall x\, (x \in A \rightarrow x \in B)$$

Note the domain restriction.

We will use a shorthand restriction to a set

$$\forall x{\in}A,\, P(x) \ := \ \forall x\, (x \in A \rightarrow P(x))$$

Restricting all quantified variables improves *clarity*

Sets & Logic

Building Sets from Predicates

Every set S defines a predicate "$x \in S$".

We can also define a set from a predicate P:

$$S := \{x : P(x)\}$$

S = the set of all x (in some universe U) for which $P(x)$ is true

In other words... $x \in S \leftrightarrow P(x)$

Proofs About Sets

$A := \{x : P(x)\}$ $B := \{x : Q(x)\}$

Suppose we want to prove A ⊆ B.

This is a predicate:

$A \subseteq B := \forall x\,(x \in A \rightarrow x \in B)$

Typically: use direct proof of the implication

Proofs About Sets

$A \subseteq B := \forall x\,(x \in A \rightarrow x \in B)$

$A := \{x : P(x)\}$ $B := \{x : Q(x)\}$

Prove that $A \subseteq B$ for P(x):= "x>2" **and** Q(x):="x^2>3"

Proof: Let x be an arbitrary object (in the universe).

Suppose that $x \in A$. By definition, this means $P(x)$.

… Therefore $x > 2$ so $x^2 > 4$ which implies $x^2 > 3$.

Thus, we have $Q(x)$. By definition, this means $x \in B$.

Since x was arbitrary, we have shown, by definition, that $A \subseteq B$. ∎