CS70 Midterm 1 HKN Review

Eric Atkinson
Riyaz Faizullabhoy
Jyo Shim

Review of Proofs

Direct Proof: Say we want to show $P \Rightarrow Q$

Step 1: Assume P

Step 2: Show that Q logically follows.

Proof by Contraposition:

Instead of showing $P \Rightarrow Q$, show $\neg Q \Rightarrow \neg P$

Review of Proofs

Proof by Contradiction:

Say we want to show that P is true.

If we can show that $\neg P \Rightarrow \text{false}$ then P must be true

Example: Prove that there is no largest integer.

Assume there is a largest integer N. N+1 is also an integer, and N+1 is larger than N. Therefore, N is not the largest integer. This contradicts our assumption that N is the largest integer. Therefore, there is no largest integer.

Induction

Problem: prove $\forall n \in \mathbb{N}, P(n)$

Step 1: Prove a base case for a small value of n: $P(1)$

Step 2: Prove that $P(n) \Rightarrow P(n+1)$

Conclusion: $\forall n \geq 1, P(n)$

Induction

Example: Consider real numbers c and k, where $c > 1, k > \frac{1}{c-1}$

Prove that for every natural number n,

$$\sum_{i=0}^{n-1} c^i < kc^n$$

Induction

Solution:

Base case:

Induction

Solution:

Base case: at n=1, we must show that

$$1 < kc$$

Induction

Solution:

Base case: at n=1, we must show that

$$1 < kc$$

We know that $k > \dfrac{1}{c-1}$

So $kc > \dfrac{c}{c-1} > 1$

Induction

Solution:

Inductive Step: Assume $\sum_{i=0}^{n-1} c^i < kc^n$

Induction

Solution:

Inductive Step: Assume $\sum_{i=0}^{n-1} c^i < kc^n$

$$\Rightarrow \sum_{i=0}^{n-1} \frac{c^i}{k} < c^n \Rightarrow \sum_{i=0}^{n} \frac{c^i}{k} < c^n + \frac{c^n}{k} \Rightarrow \sum_{i=0}^{n} \frac{c^i}{k} < c^n(1 + \frac{1}{k})$$

Induction

Solution:

Inductive Step: Assume $\sum_{i=0}^{n-1} c^i < kc^n$

$$\Rightarrow \sum_{i=0}^{n-1} \frac{c^i}{k} < c^n \Rightarrow \sum_{i=0}^{n} \frac{c^i}{k} < c^n + \frac{c^n}{k} \Rightarrow \sum_{i=0}^{n} \frac{c^i}{k} < c^n(1 + \frac{1}{k})$$

$$k > \frac{1}{c-1} \Rightarrow \frac{1}{k} + 1 < c \Rightarrow c^n(\frac{1}{k} + 1) < c^{n+1}$$

Induction

Solution:

$$\sum_{i=0}^{n} \frac{c^i}{k} < c^n(1 + \frac{1}{k}) < c^{n+1}$$

$$\Rightarrow \sum_{i=0}^{n} c^i < kc^{n+1}$$

So $\sum_{i=0}^{n-1} c^i < kc^n \Rightarrow \sum_{i=0}^{n} c^i < kc^{n+1}$

Strong Induction

In the inductive step, instead of proving that

$$P(n) \Rightarrow P(n+1)$$

it is sufficient to prove that

$$(\forall k \leq n, P(k)) \Rightarrow P(n+1)$$

Strong Induction

Example (Vazirani Fa12):

If I have an `m x n` chocolate bar, and I want to break it into `1 x 1` squares, show that it takes at least **`mn - 1`** steps to do so.

Strong Induction

Solution:

Base case: m = 1, n = 1. No breaks are needed, and $mn - 1 = 0$

Strong Induction

Inductive Step: Let the steps to break up a chocolate bar of size mn be $f(mn)$

A chocolate bar of size $mn+1$can be broken up into smaller bars of size p and q, where $p \leq q \leq mn$ and $p+q=mn+1$

Assume $f(p)=p-1$ and $f(q)=q-1$

$$f(mn+1)=f(p)+f(q)+1=p+q-1=(mn+1)-1$$

Stable Marriage

The stable marriage problem is a question of how to pair up N men and N women such that the pairing is **stable**.

A pairing is stable if there are no **rogue couples**, which are pairs of men and women who would be happier with each other than the people we paired them with.

Stable Marriage

The **propose-and-reject** algorithm is a solution to the stable marriage problem. Each unengaged man proposes to the woman he prefers the most and has not yet been rejected by.

Each woman considers all of her proposals, gets engaged to the man she prefers the most, and rejects all others.

This repeats until all men and women are engaged, at which point they get married.

Stable Marriage

The propose-and-reject algorithm is **male-optimal**, meaning every man marries his most preferred woman such that the pairing is stable.

Stable Marriage

True or False? (assume the propose-and-reject algorithm is being used)

Every man can marry the woman he prefers the most, even if the woman places him at the bottom of her list.

If a man is second on every woman's list, can he wind up married to his least favorite woman?

What if he is first on every woman's list?

Stable Marriage

True or False? (assume the propose-and-reject algorithm is being used)

Every man can marry the woman he prefers the most, even if the woman places him at the bottom of her list. TRUE

If a man is second on every woman's list, can he wind up married to his least favorite woman?

What if he is first on every woman's list?

Stable Marriage

True or False? (assume the propose-and-reject algorithm is being used)

Every man can marry the woman he prefers the most, even if the woman places him at the bottom of her list. TRUE

If a man is second on every woman's list, can he wind up married to his least favorite woman? TRUE

What if he is first on every woman's list?

Stable Marriage

True or False? (assume the propose-and-reject algorithm is being used)

Every man can marry the woman he prefers the most, even if the woman places him at the bottom of her list. TRUE

If a man is second on every woman's list, can he wind up married to his least favorite woman? TRUE

What if he is first on every woman's list? FALSE

Stable Marriage

A **worst-case couple** occurs when a couple winds up married after placing each other on the bottom of their respective lists. Does the propose-and-reject algorithm allow worst-case couples?

How many worst-case couples are possible with any stable marriage algorithm?

Stable Marriage

A **worst-case couple** occurs when a couple winds up married after placing each other on the bottom of their respective lists. Does the propose-and-reject algorithm allow worst-case couples?

YES

How many worst-case couples are possible with any stable marriage algorithm?

Stable Marriage

A **worst-case couple** occurs when a couple winds up married after placing each other on the bottom of their respective lists. Does the propose-and-reject algorithm allow worst-case couples?

YES

How many worst-case couples are possible with any stable marriage algorithm?

1, since if there were more than one, we would have a rogue couple

The Well-ordering Principle

Modular Arithmetic: inverses

Evaluate:

```
extended-gcd(37,10)
```

Show all recursive steps and return values. Use this information to provide a solution, if any to:

```
10x = 1 mod 37
```

```
algorithm extended-gcd(x,y):
   if y = 0:
     return(x, 1, 0)
   else:
     (d, a, b) := extended-gcd(y, x mod y)
     return((d, b, a - (x div y) * b))
```

Modular Arithmetic: inverses

The recursion calls:

```
extended–gcd (37, 10)
extended–gcd (10, 7)
extended–gcd (7, 3)
extended–gcd (3, 1)
extended–gcd (1, 0)
```

Modular Arithmetic: inverses

The recursion calls:

```
extended–gcd (37, 10)
extended–gcd (10, 7)
extended–gcd (7, 3)
extended–gcd (3, 1)
extended–gcd (1, 0) → returns (1, 1, 0)
```

the base case:

```
if y = 0:
    return(x, 1, 0)
```

Modular Arithmetic: inverses

The recursion calls:

```
extended–gcd (37, 10)
extended–gcd (10, 7)
extended–gcd (7, 3)
extended–gcd (3, 1) → returns (1, 0, 1)
extended–gcd (1, 0) → returns (1, 1, 0)
```

the recursion case:

```
return((d, b, a - (x div y) * b))
    where d = 1, b = 0, a = 1
```

Modular Arithmetic: inverses

The recursion calls:

```
extended–gcd (37, 10)
extended–gcd (10, 7)
extended–gcd (7, 3) → returns (1, 1, -2)
extended–gcd (3, 1) → returns (1, 0, 1)
extended–gcd (1, 0) → returns (1, 1, 0)
```

the recursion case:

```
return((d, b, a - (x div y) * b))
    where d = 1, b = 1, a = 0
```

Modular Arithmetic: inverses

The recursion calls:

```
extended–gcd (37, 10)
extended–gcd (10, 7) → returns (1, -2, 3)
extended–gcd (7, 3) → returns (1, 1, -2)
extended–gcd (3, 1) → returns (1, 0, 1)
extended–gcd (1, 0) → returns (1, 1, 0)
```

the recursion case:

```
return((d, b, a - (x div y) * b))
    where d = 1, b = -2, a = 1
```

Modular Arithmetic: inverses

The recursion calls:

```
extended–gcd (37, 10) → returns (1, 3, -11)
extended–gcd (10, 7) → returns (1, -2, 3)
extended–gcd (7, 3) → returns (1, 1, -2)
extended–gcd (3, 1) → returns (1, 0, 1)
extended–gcd (1, 0) → returns (1, 1, 0)
```

the recursion case:

```
return((d, b, a - (x div y) * b))
    where d = 1, b = 3, a = -2
```

Modular Arithmetic: inverses

The recursion calls:

```
extended–gcd (37, 10) → returns (1, 3, -11)
extended–gcd (10, 7) → returns (1, -2, 3)
extended–gcd (7, 3) → returns (1, 1, -2)
extended–gcd (3, 1) → returns (1, 0, 1)
extended–gcd (1, 0) → returns (1, 1, 0)
```

the recursion case:

```
return((d, b, a - (x div y) * b))
    where d = 1, b = 3, a = -2
```

Now that we finished, what's our answer?

Modular Arithmetic: inverses

```
d = a*x + b*y
1 = 37*3 + -11*10
```

```
1 = -11*10 mod 37
```

Our inverse is -11!

```
extended-gcd (37, 10) → returns (1, 3, -11)
extended-gcd (10, 7) → returns (1, -2, 3)
extended-gcd (7, 3) → returns (1, 1, -2)
extended-gcd (3, 1) → returns (1, 0, 1)
extended-gcd (1, 0) → returns (1, 1, 0)
```

Modular Arithmetic: FLT

Fermat's Little Theorem:

$a^{p-1} = 1 \bmod p$, given p is prime

Modular Arithmetic: FLT

Fermat's Little Theorem:

$a^{p-1} = 1 \bmod p$, given p is prime

Evaluate $2^{125} \bmod 127$

(hint: 127 is prime)

Modular Arithmetic: FLT

$2^{125} \bmod 127$

$= 2^{-1} * 2^{126} \bmod 127$

$= 2^{-1} \bmod 127$ (by FLT)

...now what?

Modular Arithmetic: FLT

$2^{125} \bmod 127$

$= 2^{-1} * 2^{126} \bmod 127$

$= 2^{-1} \bmod 127$ (by FLT)

Find the inverse! Extended GCD!

Modular Arithmetic: FLT

$2^{125} \bmod 127$

$= 2^{-1} * 2^{126} \bmod 127$

$= 2^{-1} \bmod 127$ (by FLT)

egcd(1, 0) → returns (1, 1, 0)

egcd(2, 1) → returns (1, 0, 1)

egcd(127, 2) → returns (1, 1, **-63**)

$= -63 \bmod 127$

$= 64 \bmod 127$

RSA (Sahai Sp13)

Rather than doing traditional RSA based on two prime numbers, suppose that your friend suggests using **three** prime numbers.

She decides to use `N = 105 = 3*5*7` and selects `e = 5` so that the public key is:

`(N, e) = (105, 5)`

1. Encrypt the message 2 using this public key.
2. Encrypt the message 3 using this public key.
3. What property should `d` satisfy? Calculate `d`.

RSA

Does this work? We prove that `D(E(x)) = x mod N`

Solution: As seen in lecture, we need to show that $(x^e)^d = x \mod N$. We first consider the exponent ed. By definition of d, we know that $ed = 1 \mod (p-1)(q-1)(r-1)$; hence we can write $ed = 1 + k(p-1)(q-1)(r-1)$ for some integer k, and therefore $x^{ed} - x = x^{1+k(p-1)(q-1)(r-1)} - x = x(x^{k(p-1)(q-1)(r-1)} - 1)$

Our goal is to show that this last expression is equal to $0 \mod N$ for every x. To do so, we claim that it is divisible by p, of which there are two cases:

Case 1: *x is not a multiple of p.* In this case, $x \neq 0 \mod p$, we can use Fermat's Little Theorem to deduce that $x^{p-1} = 1 \mod p$, and hence $x^{k(p-1)(q-1)(r-1)} - 1 = 0 \mod p$, as required.

Case 2: *x is a multiple of p.* In that case, the expression $x(x^{k(p-1)(q-1)(r-1)} - 1)$ clearly has a factor of x, so it is divisible by p.

By an entirely symmetrical argument, $x(x^{k(p-1)(q-1)(r-1)} - 1)$ is also divisible by q and r. Therefore, it is divisible by all three numbers, all of which are prime; thus, it must be divisible by their product, $pqr = N$. But this implies that the expression is equal to $0 \mod N$, which is exactly what we want to prove.

RSA

1. Encrypt the message 2 using this public key.

We just do the same thing as regular RSA!

$E(x) = x^e \bmod N$

$= 2^5 \bmod 105$

$= 32$

RSA

2. Encrypt the message 3 using this public key.

We just do the same thing as regular RSA!

$E(x) = x^e \bmod N$

$= 3^5 \bmod 105$

$= 243 \bmod 105$

$= 33$

RSA

3. What property should d satisfy? Calculate d

We want to calculate the number d such that d is the inverse of:

$$e \bmod (p-1)(q-1)(r-1) = 5 \bmod 48$$

To solve this, we can use extended-gcd(48, 5), and we will get: **d = 29**

Polynomials