Math 55: Discrete Math

G.S.I. Loren Looger

September 24, 1997

Homework Assignment #6

Section	*Problems*
§3.2	6, 7, 8, 10, 22, 32
§3.3	4, 6, 7, 10, 15, 16, 35
§3.4	3, 4, 6, 10, 11, 12, 13

(§3.2, #6) Find a formula for

$$\sum_{i=1}^{n} \frac{1}{i(i+1)}$$

by examining the values of the expression for small values of n. Use mathematical induction to prove your answer is correct.

We examine the first four values of the expression:

$$\sum_{i=1}^{1} \frac{1}{i(i+1)} = \frac{1}{2}$$

$$\sum_{i=1}^{2} \frac{1}{i(i+1)} = \frac{2}{3}$$

$$\sum_{i=1}^{3} \frac{1}{i(i+1)} = \frac{3}{4}$$

$$\sum_{i=1}^{4} \frac{1}{i(i+1)} = \frac{4}{5}$$

We conjecture that:

$$\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}$$

Let us say that P(n) is true if the above statement holds for a certain n. We have already shown that this holds true for n=1. This is the base case. All that remains to prove is the inductive step. Thus assume that P(n) is true. We will prove that P(n+1) is true. Namely:

$$\sum_{i=1}^{n+1} \frac{1}{i(i+1)} = \sum_{i=1}^{n} \frac{1}{i(i+1)} + \frac{1}{(n+1)(n+2)}$$

By the assumption, this equals

$$\frac{n}{n+1} + \frac{1}{(n+1)(n+2)} = \frac{n(n+2)+1}{(n+1)(n+2)}$$

But this equals

$$\frac{(n+1)(n+1)}{(n+1)(n+2)} = \frac{n+1}{n+2}$$

But this says that P(n+1) is true. So P(n) implies P(n+1). So the statement is true for all n.

(§3.2, #7) Prove that

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

for all positive integers n. Let P(n) be the statement that the equality holds for a certain n.

First, we check the base case, n=1. We see that

$$\sum_{i=1}^{1} i^2 = 1 = \frac{1(1+1)(2*1+1)}{6}$$

So P(1). So we must only show the inductive step. So assume P(n). Then

$$\sum_{i=1}^{n+1} i^2 = \sum_{i=1}^{n} i^2 + (n+1)^2$$

By the assumption, this equals

$$\frac{n(n+1)(2n+1)}{6} + (n+1)^2 = \frac{2n^3+3n^2+n}{6} + \frac{6n^2+12n+6}{6}$$

This is

$$\frac{2n^3+9n^2+13n+6}{6} = \frac{(n+1)(n+2)(2n+3)}{6} = \frac{(n+1)((n+1)+1)(2*(n+1)+1)}{6}$$

So P(n+1). Thus we have verified the inductive step. So the statement holds for all positive integers n.

(§3.2, #8) Prove that

$$\sum_{i=1}^{n} i^3 = (\frac{n(n+1)}{2})^2$$

for all positive integers n. Let P(n) be the statement that the equality holds for a certain n.

First, we check the base case, n=1. We see that

$$\sum_{i=1}^{1} i^3 = 1 = (\frac{1(1+1)}{2})^2$$

So P(1). So we must only show the inductive step. So assume P(n). Then

$$\sum_{i=1}^{n+1} i^3 = \sum_{i=1}^{n} i^3 + (n+1)^3$$

By the assumption, this equals

$$(\frac{n(n+1)}{2})^2 + (n+1)^3 = \frac{n^2(n+1)^2}{4} + \frac{4(n+1)^3}{4}$$

This is

$$\frac{n^4+2n^3+n^2}{4}+\frac{4n^3+12n^2+12n+4}{4}=\frac{n^4+6n^3+13n^2+12n+4}{4}$$

But this is precisely

$$\frac{(n+1)^2(n+2)^2}{4}=(\frac{(n+1)((n+1)+1)}{2})^2$$

So P(n+1). Thus we have verified the inductive step. So the statement holds for all positive integers n.

(§3.2, #10) Prove that

$$\sum_{i=1}^{n} i*i! = (n+1)!-1$$

for all positive integers n. Let P(n) be the statement that the equality holds for a certain n.

First, we check the base case, n=1. We see that

$$\sum_{i=1}^{1} i*i! = 1*1! = 1 = 2-1 = (1+1)!-1$$

So P(1). So we must only show the inductive step. So assume P(n). Then

$$\sum_{i=1}^{n+1} i*i! = \sum_{i=1}^{n} i*i! + (n+1)*(n+1)!$$

By the assumption, this equals

$$(n+1)!-1+(n+1)*(n+1)! = ((n+1)!)*((n+1)+1)-1 = ((n+1)!)*(n+2)-1 = (n+2)!-1 = ((n+1)+1)!-1$$

So P(n+1). We have proven the inductive step. So the statement is true for all positive integers n.

(§3.2, #22) Use mathematical induction to show that 6 divides n^3-n whenever n is a nonnegative integer. Let P(n) be the statement that the equality holds for a certain n.

First, we check the base case, n=0. We see that 6 divides $0^3-0=0$. So P(0). So we must show the inductive step. Assume P(n). Then

$$(n+1)^3-(n+1) = n^3+3n^2+2n = (n^3-n)+3n(n+1).$$

Now one of n or $(n+1)$ is even, so $3n(n+1)$ is divisible by 2. Obviously, $3n(n+1)$ is divisible by 3. So $3n(n+1)$ is divisible by 6. By the assumption, n^3-n is divisible by 6. The sum of two numbers each divisible by 6 is also divisible by 6. So $6|((n+1)^3-(n+1))$. So P(n+1). We have proven the inductive step. So the statement holds for all nonnegative integers n.

(§3.2, #32) Which amounts of money can be formed using just dimes and quarters? Prove your answer using a form of mathematical induction.

The answer is that we may form all multiples of five cents greater than or equal to 20 cents, as well as 10 cents. 5 cents and 15 cents are not able to be formed. Let P(n) be the statement that we may form $5n$ cents, where $n \geq 4$. Then P(4), as 4*5 = 20 cents requires just two dimes. So assume P(n), where $n \geq 4$. Either a quarter was used to form $5n$ cents, or it wasn't. If a quarter was used, then we may removed a quarter from our combination, and add three

dimes in its stead, thus increasing the amount of money by five cents. In this case, P(n+1). If a quarter was not used, then as the amount of money is at least 20 cents, at least two dimes were used. Thus we may remove two dimes from our combination, and add a quarter in their place, thus increasing the amount of money by five cents. In this case as well, P(n+1). So in either case, P(n) implies P(n+1).

By induction, then, we may form any amount of money that is $5n$ **cents, where** $n \geq 4$**. We may also form 10 cents.**

(§3.3, #4) Find $f(2), f(3), f(4), f(5)$ **if f is defined recursively by** $f(0) = 1, f(1) = 1$**, and for** $n = 1, 2...$**:**

$$a) f(n+1) = f(n) - f(n-1).$$

In this case

$$f(2) = f(1) - f(0) = 1 - 1 = 0.$$
$$f(3) = f(2) - f(1) = 0 - 1 = -1.$$
$$f(4) = f(3) - f(2) = -1 - 0 = -1.$$
$$f(5) = f(4) - f(3) = -1 - (-1) = 0.$$

$$b) f(n+1) = f(n) * f(n-1).$$

In this case

$$f(2) = f(1) * f(0) = 1 * 1 = 1.$$
$$f(3) = f(2) * f(1) = 1 * 1 = 1.$$
$$f(4) = f(3) * f(2) = 1 * 1 = 1.$$
$$f(5) = f(4) * f(3) = 1 * 1 = 1.$$

$$c) f(n+1) = f(n)^2 + f(n-1)^3.$$

In this case

$$f(2) = f(1)^2 + f(0)^3 = 1^2 + 1^3 = 2.$$
$$f(3) = f(2)^2 + f(1)^3 = 2^2 + 1^3 = 5.$$
$$f(4) = f(3)^2 + f(2)^3 = 5^2 + 2^3 = 33.$$
$$f(5) = f(4)^2 + f(3)^3 = 33^2 + 5^3 = 1214.$$

$$d) f(n+1) = f(n)/f(n-1).$$

In this case

$$f(2) = f(1)/f(0) = 1/1 = 1.$$
$$f(3) = f(2)/f(1) = 1/1 = 1.$$
$$f(4) = f(3)/f(2) = 1/1 = 1.$$
$$f(5) = f(4)/f(3) = 1/1 = 1.$$

(§3.3, #6) Give a recursive definition of the sequence $\{a_n\}, n = 1, 2...$ **if:**

$$a) a_n = 4n - 2.$$

We see that $a_{n+1} = 4(n+1) - 2 = 4n+2 = a_n + 4$. Also, $a_1 = 4*1-2 = 2$. This defines the sequence.

$$b) a_n = 1 + (-1)^n.$$

We see that $a_{n+1} = 1 + (-1)^{n+1} = 1 - (-1)^n = 2 - a_n$. Also, $a_1 = 1 + (-1)^1 = 0$. This defines the sequence.

$$c) a_n = n(n+1).$$

We see that $a_{n+1} = (n+1)(n+2) = n^2 + 3n + 2 = (n^2+n) + (2n+2) = a_n + 2(n+1)$. Also, $a_1 = 1*2 = 2$. This defines the sequence.

$$d) a_n = n^2.$$

We see that $a_{n+1} = (n+1)^2 = n^2 + 2n + 1 = n^2 + (2n+1) = a_n + (2n+1)$. Also, $a_1 = 1^2 = 1$. This defines the sequence.

(§3.3, #7) Let $F(n) = \sum_{i=1}^{n} i$. Find a recursive definition for F(n).

We see that $F(n+1) = \sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + (n+1) = F(n) + (n+1)$. Also, $F(1) = 1$. This is a recursive definition of $F(n)$.

(§3.3, #10) Prove that $\sum_{i=1}^{n} f_i^2 = f_n f_{n+1}$, where f_n is the nth Fibonacci number.

We prove by induction. Let P(n) be the statement that the equality holds for a certain n. First, we check the base case, n=1. We see that

$$\sum_{i=1}^{1} f_i^2 = f_1^2 = 1^2 = 1 = 1*1 = f_1 f_2.$$

So P(1). So we must only show the inductive step. So assume P(n). Then

$$\sum_{i=1}^{n+1} f_i^2 = \sum_{i=1}^{n} f_i^2 + f_{n+1}^2$$

By the assumption, this equals

$$f_n f_{n+1} + f_{n+1}^2 = (f_{n+1})(f_n + f_{n+1}) = f_{n+1} f_{n+2}.$$

So P(n+1). Thus we have verified the inductive step. So the statement holds for all positive integers n.

(§3.3, #15) Determine the number of divisions used by the Euclidean algorithm to find the greatest common divisor of the Fibonacci numbers f_n and f_{n+1}, where n is a nonnegative integer. Prove your answer using mathematical induction.

The number of divisions used by the Euclidean algorithm to find $gcd(f_{n+1}, f_n)$ is 0 for n = 0, 1 for n = 1, and n-1 for $n \geq 2$. We prove the latter result using induction.

Let P(n), where $n \geq 2$, we the statement that n-1 divisions are required to compute $gcd(f_{n+1}, f_n)$.

First, we check the base case, n = 2. One division shows that $gcd(f_3, f_2) = gcd(2,1) = gcd(1,0) = 0$. Thus P(2). So we must show the inductive step. So assume P(n). Then to find $gcd(f_{n+2}, f_{n+1})$, one division shows that $gcd(f_{n+2}, f_{n+1}) = gcd(f_{n+1}, f_n)$, as $f_{n+2} = 1 * f_{n+1} + f_n$. But by the inductive hypothesis, $n-1$ divisions will compute the latter gcd. Thus, a total of

$1+(n-1)=n=(n+1)-1$ divisions were used to compute $gcd(f_{n+2}, f_{n+1})$. So P(n+1). We have proven the inductive step. So P(n) holds for all $n \geq 2$.

For n=0, it takes 0 divisions to find that $gcd(f_1, f_0) = gcd(1,0) = 1$, as $gcd(n,0) = n$, by definition.

For n=1, it takes 1 division to find that $gcd(f_2, f_1) = gcd(1,1) = gcd(1,0) = 1$, as $1 = 1*1+0$.

So we have found the required number of divisions for all $n \geq 0$.

(§3.3, #16) Let

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

Show that

$$A^n = \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix}.$$

for n a positive integer.

We prove by induction on n. First we check the base case, n = 1. Then we see that

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} f_2 & f_1 \\ f_1 & f_0 \end{bmatrix}.$$

So P(1). Now we must check the inductive step. So assume P(n). Then

$$A^{n+1} = A^n * A = \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} f_{n+1}+f_n & f_{n+1} \\ f_n + f_{n-1} & f_n \end{bmatrix} = \begin{bmatrix} f_{n+2} & f_{n+1} \\ f_{n+1} & f_n \end{bmatrix}.$$

So P(n+1). Thus we have proven the inductive step. So the statement holds for all positive integers n.

(§3.3, #35) A partition of a positive integer n is a way to write n as the sum of positive integers. Let P_m be the number of ways to partition m, and let $P_{m,n}$ be the number of ways to partition m using numbers not exceeding n.

a) Show that $P_{m,m} = P_m$.

Obviously no partitions of m may use any integers greater than m. So $P_{m,m} = P_m$.

b) Verify the following recursive definition:

b1) If $m = 1$, then the only partition is $1 = 1$. So $P_{m,n} = 1$.

b2) If $n = 1$, then the only partition is $m = 1+1+...1$. So $P_{m,n} = 1$.

b3) If $m < n$, then as above, there are no partitions of m using integers greater than m, so $P_{m,n} = P_{m,m} = P_m$.

b4) If $m = n > 1$, then There is only one partition of m using the integer m, namely $m = m$. All other partitions use only the integers $1, 2...m-1$. So $P_{m,n} = 1 + P_{m,m-1}$.

b5) If $m > n > 1$, then either a partition of m uses n or it does not. If the partition does, then the remaining amount that we must sum to reach m is $m-n$. We may still use all integers $1, 2...n$. So the number of such partitions is $P_{m-n,n}$. If a partition does not use n, then it only uses the integers $1, 2...n-1$. So the number of such partitions is $P_{m,n-1}$. So the total number of partitions is $P_{m,n} = P_{m,n-1} + P_{m-n,n}$.

(§3.4, #3) Find a recursive algorithm for finding the sum of the first n odd positive integers.

We give pseudocode:

procedure sum-of-odds (n: positive integer)
if n $= 1$ then return(1);
else return(sum-of-odds$(n-1)$ + $(2n-1)$)

This is because $(2n-1)$ is the n^{th} odd positive integer.

(§3.4, #4) Give a recursive algorithm for finding the maximum of a finite set of integers.

We give pseudocode:

procedure max($a_1, a_2 ... a_n$: integers)
if $n = 1$ then max$(a_1, a_2 ... a_n)$: $= a_1$
else
begin
$m :=$ max$(a_1, a_2 ... a_{n-1})$
if $m > a_n$ then max$(a_1, a_2 ... a_n)$: $= m$
else max$(a_1, a_2 ... a_n)$: $= a_n$
end

We first find the maximum of the first $n-1$ numbers, and then compare this maximum with the n^{th} number.

(§3.4, #6) Find a recursive algorithm for finding x^n mod m where x, n, m are positive integers.

We give pseudocode:

procedure power (x, n, m: positive integers)
if n $= 1$ then power$(x, n, m) := x$ mod m
else power$(x, n, m) := x * power(x, n-1, m)$ mod m

So we just find the $n-1^{st}$ power of x mod m, then multiply this by x mod m.

(§3.4, #10) Devise a recursive algorithm to compute a^{2^n} where a is a real number, and n is a positive integer.

We give pseudocode:

procedure twopower (n: positive integer, a: real number)
if n $= 1$ then twopower$(n, a) := a^2$;
else twopower(n, a) := (twopower$(n-1, a))^2$

We just compute $a^{2^{n-1}}$, and then square that.

(§3.4, #11) How does the number of multiplications used in Exercise 10 compare with the number of multiplications used in Algorithm 1?

Exercise 10 employs just n multiplications to compute a^{2^n}, whereas Algorithm 1 uses 2^n multiplications.

(§3.4, #12) Use Exercise 10 to compute a^n when n is any nonnegative integer.

This is just the fast modular exponentiation that we have been using all semester. We give pseudocode:

procedure fastexponent (n: positive integer, a: real number)
if n $= 1$ then fastexponent$(n, a) := a$;
else if n is even then fastexponent(n, a) :=fastexponent$(n/2, a)^2$;
else fastexponent$(n, a) := a*$ fastexponent$((n-1)/2, a)^2$

This is just the logical way of working backwards through fast modular exponentiation.

(§3.4, #13) How does the number of multiplications used in Exercise 12 compare with the number used by Algorithm 1 to compute a^n?

Algorithm one obviously uses n multiplications to compute a^n. The algorithm in Exercise 12 uses $\leq 2 * m$, where m is the largest integer such that $2^m \leq n$. This is because we compute m powers $a^{2^1} ... a^{2^m}$, and then require at most m more multiplications to obtain a^n. Thus the algorithm works in $O(\log_2 n) = O(\log n)$ time.