CS 317

Homework 1

Nolan Anderson

Section 1.1 #6

a) 1 gcd(31415, 14142)

31415 mod 14142 = 3131; m ← 14142, n ← 3131

2 gcd(14142, 3131)

14142 mod 3131 = 1618; m ← 3131, n ← 1618

3 gcd(3131, 1618)

3131 mod 1618 = 1513; m ← 1618, n ← 1513

4 gcd(1618, 1513)

1618 mod 1513 = 105; m ← 1513, n ← 105

5 gcd(1513, 105)

1513 mod 105 = 43; m ← 105, n ← 43

6 gcd(105, 43)

105 mod 43 = 19; m ← 43, n ← 19

7 gcd(43, 19)

43 mod 19 = 5; m ← 19, n ← 5

8 gcd(19, 5)

19 mod 5 = 4; m ← 5, n ← 4

9 gcd(5, 4)

5 mod 4 = 1; m ← 4, n ← 1

10 gcd(4, 1)

4 mod 1 = 0 m ← 1, n ← 0

11 gcd(1, 0) -> m = 1 / gcd(31415, 14142) = 1

b Since the gcd of 31415 & 14142 is 1, and you subtract by one every time the remainer is not zero, it would take 14142 iterations and 28284 divisions. Euclid's was 11 iterations.

28284/11 = 1285.6; # of times faster Euclid's is.

or 2571.3 times less divisions

Section 1.2 #5b $2^0=1$ $2^1=2$ $2^2=4$ $2^3=8$ $2^4=16$

Decimal -> Binary

13 -> binary -> $1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 =$ 1101

```
n ← positive decimal integer // Variable for integer
while n ←0 do                  // while n !=0
    r ← n mod 2                // r gets remainder of n/2
    n ← n/2                    // n gets n/2 to update value
    print (r)                  // print remainder, no end line so output is the same
end
```

Section 1.2 #8

For example... computing the perimeter of a rectangle is length + length + width + width. A simpler algorithm would be 2 x (length + width). Neither is more efficient though. For both it only takes one step.

factorial: iterative:

```
if(n==0)do
   return 1;
else
   while (n!=0) do
       r ← n(n-1)      // result
       f= f+r          // final result
       n = n - 1       // update n
   end while
return f
```

recursive

```
factorial(n)
   if(n==0)
       return 1
   return n x factorial(n-1)
```

As you can see, the recursive algorithm for factorial is much simpler and efficient than an iterative approach. The iterative approach has significantly more operations.

Section 1.3

#1 a

0	0	0	0	0	0
3	0	1	1	0	0
	1	2	2	0	1
		4	3	0	1
			5	0	1
				0	2

3 1 4 5 0 2 → 14 | 35 | 47 | 60 | 81 | 98

c) I would say it is not in place b/c it uses a seperate array to sort the data, which takes up memory.

Problem A:

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

$$\sum_{i=0}^{k+1} \frac{1}{(2i-1)(2i+1)} = \text{left side}$$

$\frac{1}{(2(0)+1)(2(0)-1)} = \frac{1}{3}$ ✓ $\frac{1}{2(1)+1} = \frac{1}{3}$ ✓

$$\sum_{i=0}^{k} \frac{1}{(2i-1)(2i+1)} + \frac{1}{(2(k+1)-1)(2(k+1)+1)} = \text{Show } \frac{k}{2k+1} + \frac{1}{(2k+1)(2k+3)} = \frac{k+1}{2k+3}$$

$$\frac{k(2k+3)}{(2k+1)(2k+3)} + \frac{1}{(2k+1)(2k+3)} = \frac{2k^2+3k+1}{(2k+1)(2k+3)} \quad \frac{(2k+1)(k+1)}{(2k+1)(2k+3)} = \frac{k+1}{2k+3}$$

$$\frac{k+1}{2k+3} = \frac{k+1}{2k+3}$$ ∎

Prob B: Solve the following to find the closed form sol'n:

$S(1) = 1$

$S(n) = S(n-1) + (2n-1)$

$S(1) = 1$
$S(2) = S(1) + 3 = 4$
$S(3) = S(2) + 5 = 9$
$S(4) = S(3) + 7 = 16$

General Sol'n formula: $S(n) = c^{n-1} S(1) + \sum_{i=2}^{n} c^{n-i} g(i)$

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

$= 1 + \sum_{i=2}^{n} 2i - 1 = 1 + (3 + 4 + \dots + (2n-1)) = \boxed{n^2}$