cmput 204 asn 1 solutions fall 2015

1. Your number k depends only on the value $n' = n \pmod{89}$, where n is your student number. The following chart shows the values k, depending on the values n', starting from 0. E.g. the first row is for values 0,1,2, . . . , the next row for values 10,11,12, . . . , etc.

```
11 12 15 81 27 60 24 46 75 16
29 68 63 34 62 59 89 47 31 93
83 58 61 71 41 98 14 94 37 44
25 57 56 85 66 35 91 88 72 19
32 69 21 38 33 78 73 90 42 79
92 39 23 20 76 45 26 55 54 86
67 74 17 97 13 70 40 50 53 28
18 80 64 22 52 49 77 48 43 82
95 36 65 87 51 84 30 96 99
```

2. The answer depends on k. E.g., for $k = 87$, $3787.09 = 61.5^2 + 4.84$

```
                     6   1 . 5
                   ------------
                 \/ 3 7 8 7.0 9
6                   3 6
                    ---
                      1 8 7.
1 2 1                 1 2 1.
                      -----
                        6 6.0 9
1 2 2 5                 6 1.2 5
                        -------
                          4.8 4
```

3. (i) This code outputs the conversion.

```
def display(n):
  if n>1: display(n/2)
  print n,
def db(n):
  if n>1: db(n/2)
  print n%2,
n = 587
display(n)
print ''
db(n)
print ''
```

```
1 2 4 9 18 36 73 146 293 587
1 0 0 1 0 0 1 0 1 1
```

(ii) E.g., for $k = 87, 587 = 24^2 + 11$.

```
                   1   1   0   0   0
                 --------------------
               \/ 1 0 0 1 0 0 1 0 1 1
1                  1
                 ----
                   1 0 1
1 0 1              1 0 1
                   -----
                         0 0
1 1 0 0                    0
                         ---
                           0 1 0
1 1 0 0 0                      0
                           -----
                             1 0 1 1
1 1 0 0 0 0                        0
                             -------
                             1 0 1 1
```

4.
```
def fib(n):
  if n<2: return n
  return fib(n-1) + fib(n-2)
```

(i) Here, `fib(t-2)` and `fib(t-1)` are defined, so $t \geq 2$, so `fib(t)` returns `fib(t-1) + fib(t-2)`, so

$$\begin{aligned}
\text{fib(t)} &= & \text{fib(t-1)} + \text{fib(t-2)} &\text{ since } t \geq 2\\
&\geq & 1.6^{t-3} + 1.6^{t-4} &\text{ assumption}\\
&= & (1.6)1.6^{t-4} + 1.6^{t-4} &\text{ arithmetic}\\
&= & (1.6+1)1.6^{t-4} &\text{ arithmetic}\\
&= & (2.6)1.6^{t-4} &\text{ arithmetic}\\
&> & (2.56)1.6^{t-4} &\text{ arithmetic}\\
&= & (1.6^2)1.6^{t-4} &\text{ arithmetic}\\
&= & 1.6^{t-2} &\text{ arithmetic}
\end{aligned}$$

(ii) $\text{fib}(0) = 0 < 1.6^{-2}$. $\text{fib}(1) = 1 \geq 1.6^{-1}$. $\text{fib}(2) = 1 = 1.6^0$. By (i) and induction, $\text{fib(k)} \geq 1.6^{k-2}$ for all $k \geq 3$. So Q is the set of all integers greater than or equal to 1.

(iii)

• No, this does not imply $\text{fib(n)} \in O(1.6^n)$.

Assume by way of contradiction that $\text{fib(n)} \in O(1.6^n)$. Then there is a positive constant c such that $\text{fib(n)} \leq c1.6^n$ for all positive integers n. But we saw in class that there is a positive constant c' such that $\text{fib(n)} \geq c'1.618^n$ for all positive integers n. So we would have

$$\begin{aligned}
c'1.618^n \leq\ \text{fib(n)}\ &\leq c1.6^n && \text{so}\\
c'1.618^n &\leq c1.6^n && \text{so}\\
1 &\leq \frac{c1.6^n}{c'1.618^n} && \text{so}\\
1 &\leq \left(\frac{c}{c'}\right)\left(\frac{1.6}{1.618}\right)^n && \text{but}
\end{aligned}$$

this is not possible: c and c' are positive constants, so c'/c is a positive constant, but $(1.6/1.618)^n$ gets arbitrarily close to 0 as n gets large, so for some n $(1.6/1.618)^n < c'/c$ and $(c/c')(1.6/1.618)^n$ is less than 1, contradiction.

• Yes, this implies that $\text{fib(n)} \in \Omega(1.6^n)$, since from (ii) $\text{fib(n)} \geq c1.6^n$ for all positive integers n, where $c = 1.6^{-2} = 1/2.56$ is a positive constant. Thus, from the definition of Ω, $\text{fib(n)} \in \Omega(n)$.

• No. A function $f(n)$ is in $\Theta(1.6^n)$ if and only if it is in both $O(1.6^n)$ and $\Omega(1.6^n)$.

5. The following code will print the recursion tree **upside down**.

```
def rm(x,y):
  if y==0:
    print '(',x,y,0,')\n       |'
    return 0
  elif 0==y%2:
    t = 2*rm(x,y/2)
    print '(',x,y,t,')\n       |'
    return t
  else:
    t = x+2*rm(x,y/2)
    print '(',x,y,t,')\n       |'
    return t
```

To the right is the tree for the call `(191,902)`:

```
( 191 902 172282 )
       |
( 191 451 86141 )
       |
( 191 225 42975 )
       |
( 191 112 21392 )
       |
( 191 56 10696 )
       |
( 191 28 5348 )
       |
( 191 14 2674 )
       |
( 191 7 1337 )
       |
( 191 3 573 )
       |
( 191 1 191 )
       |
( 191 0 0 )
```

6. 9803 is odd, so `rmult(x,9803)` returns `x + 2*rmult(x,9803/2)` `= x + 2*rmult(x,4901)`.

   4901 is less than 9802, so by our assumption `rmult(x,4901)` returns `x*4901`.

   So `x + 2*rmult(x,4901) =` `x + 2*(x*4901) =` `x + 9802*x =` `9803*x`.

7. (i) $\Theta(n)$. The algorithm adds numbers bit by bit, starting from the least significant bit. There is at most one carry bit from each add. So the total number of adds will be either n or $n+1$, so $\Theta(n)$. Each add of two bits plus a carry bit takes constant time. So $\Theta(n) \times \Theta(1) = \Theta(n)$ time.

(ii)

- We expect the ratio to be $\dfrac{c(2k)^2}{ck^2} = \dfrac{c4k^2}{ck^2} = 4.$

- We expect the ratio to be $\dfrac{c(2k)^3}{ck^3} = \dfrac{c8k^3}{ck^3} = 8.$

(iii) From the experimental timing example in the lecture notes, the runtime ratio $t(2n)/t(n)$ is around 4, so we expect that the runtime is in $\Theta(n^2)$.

The loop iterates exactly n times.

Define $f(t)$ as the t'th Fibonacci number. After t iterations, the sum is $f(t)$.

Consider any fixed integer t. From the seminar, the number of bits in $f(t)$ is in $\Theta(t)$. From the lecture notes, the time to add two numbers whose sum has k bits is in $\Theta(k)$, so to add two numbers whose sum has $\Theta(t)$ bits is in $\Theta(t)$.

So the runtime of this algorithm is given by the sum

$$\sum_{t=1}^{n} \Theta(t) \; = \; \Theta(\sum_{t=1}^{n} t) \; = \Theta(n^2)\,.$$

So the runtime is in $\Theta(n^2)$, as expected.