CSC236 tutorial exercises, Week #8
best before Thursday evening

These exercises are intended to give you some practice proving bounds on recurrences, and proving correctness of recursive programs.

1. Examine the recurrence $R(n)$ below.

$$R(n) = \begin{cases} 0 & \text{if } n = 1 \\ n + 3R(\lceil n/3 \rceil) & \text{if } n > 1 \end{cases}$$

Assume that for all $k \in \mathbb{N}, R(3^k) = k3^k$.

sample solution: Prove that $R \in \mathcal{O}(n \lg n)$. Define $n^* = 3^{\lceil \log_3 n \rceil}$. Then we have:

$$\lceil \log_3 n \rceil - 1 < \log_3 n \leq \lceil \log_3 n \rceil \Rightarrow n^*/3 < n \leq n^*$$

I will also use the assumption (proved last week) that R is nondecreasing.
Let $d = 6$. Then $d \in \mathbb{R}^+$. Let $B = 3$. Then $B \in \mathbb{N}^+$. Let n be an arbitrary natural number no smaller than B. Then

$$\begin{aligned}
R(n) &\leq R(n^*) \qquad \# \text{ since } R \text{ nondecreasing and } n^* \geq n \\
&= n^* \log_3 n^* \qquad \# \text{ by assumption} \\
&\leq 3n \log_3(3n) \qquad \# \; n > n^*/3 \Rightarrow 3n > n^* \\
&= 3n(\log_3(n) + 1) \leq 3n(\log_3 n + (1)\log_3 n) \qquad \# \; n \geq B \geq 3 \Rightarrow \log_3 n \geq 1 \\
&= 6n \log_3 n \leq dn \log_3 n \qquad \# \; d = 6
\end{aligned}$$

So $R \in \mathcal{O}(n \lg n)$, since $\log_3 n$ differs from $\lg n$ by a constant factor. ■

sample solution: Prove that $R \in \Omega(n \lg n)$. Define $n^* = 3^{\lceil \log_3 n \rceil}$. Then we have:

$$\lceil \log_3 n \rceil - 1 < \log_3 n \leq \lceil \log_3 n \rceil \Rightarrow n^*/3 < n \leq n^*$$

I will also use the assumption (proved last week) that R is nondecreasing.
Let $d = 1/6$. Then $d \in \mathbb{R}^+$. Let $B = 9$. Then $B \in \mathbb{N}$.
Let n be an arbitrary natural number no smaller than B. Then

$$\begin{aligned}
R(n) &\geq R(n^*/3) \qquad \# \text{ since } R \text{ nondecreasing and } n > n^*/3 \\
&= n^*/3 \log_3 n^*/3 \qquad \# \text{ by assumption} \\
&\geq n/3 \log_3(n/3) \qquad \# \; n^* \geq n \Rightarrow n^*/3 > n/3 \\
&= n/3(\log_3(n) - 1) = n/3 \log_3 n - n/3 = n/6 \log_3 n + n/6 \log_3 n - n/3 \\
&\geq n/6 \log_3 n \qquad \# \; n \geq 9 \geq B \Rightarrow n/6 \log_3 n \geq n/3 \\
&= dn \log_3 n \qquad \# \; d = 1/6
\end{aligned}$$

So $R \in \Omega(n \lg n)$, since $\log_3 n$ differs from $\lg n$ by a constant. ■

2. Read over the code for decimal_to_binary below:

```
def decimal_to_binary(n: int) -> str:
    """
    Return binary string representing n.

    precondition: n is a natural number.

    >>> decimal_to_binary(0)
    '0'
    >>> decimal_to_binary(5)
    '101'

    postcondition: returns binary string representing
    n with no leading zeros (except if n == 0).
    """
    if n < 2:
        return str(n)
    else:
        return decimal_to_binary(n // 2) + decimal_to_binary(n % 2)
```

Use the technique from week 7 notes to prove that the precondition implies termination and the postcondition, **or** find a counter-example.

sample solution: Let $n \in \mathbb{N}$ and bits $b_0, \ldots, b_k \in \{0, 1\}$ be such that $n = \sum_{i=0}^{i=k} 2^i b_i$. I will use the identities:

$$\lfloor n/2 \rfloor = \sum_{i=1}^{i=k} 2^{i-1} b_i \qquad \text{and} \qquad n \equiv b_0 \bmod 2$$

Define $P(n)$: "If n is a natural number, then decimal_to_binary(n) terminates and returns the binary string representing n with no leading zeros, except if n is 0."

I will use complete induction to prove $\forall n \in \mathbb{N}, P(n)$.

inductive step: Let $n \in \mathbb{N}$. Assume $\bigwedge_{j=0}^{j=n-1} P(j)$. I will show that $P(n)$ follows.

case $n < 2$: If $n < 2$ the "if" branch executes, and str(n) is returned: "0" if $n = 0$ and "1" if $n = 1$, which are the binary strings representing 0, and 1, respectively, which can be verified by evaluating the sum $0 = \sum_{j=0}^{j=0} 0$ and $1 = \sum_{j=0}^{j=0} 1$. So $P(n)$ holds in this case.

case $n \geq 2$: We have $0 \leq \text{n\%2} \leq \text{n//2} < n$, so we know $P(\text{n\%2}) \wedge P(\text{n//2})$ by the IH. Let bits $b_0, \ldots, b_k \in \{0, 1\}$ be such that $n = \sum_{i=0}^{i=k} 2^i b_i$, then (by the identity above):

$$2(\lfloor n/2 \rfloor) + b_0 = 2 \sum_{i=1}^{i=k} 2^{i-1} b_i + b_0 = \sum_{i=0}^{i=k} 2^i b_i = n$$

so the binary string representing n is the concatenation of the strings for $b_1, \ldots, b_k$ (representing n//2, so returned by decimal_to_binary(n//2) by the IH) with the string for b_0 (representing n%2, so returned by decimal_to_binaryy(n%2) by the IH), which is what is returned by the "else" branch. ■