CSC236 tutorial exercises, Week #4
sample solutions

1. Consider the following code implementing binary search (you should be familiar with this algorithm from CSC148):

```
def binsearch(A, x):
    """Return i such that A[i] = x.
    PRECONDITION: A is a non-empty sorted list, and x is an element of A.
    """
    if len(A) == 1:
        return 0
    mid = len(A) // 2
    if A[mid] > x:
        return binsearch(A[:mid], x)
    else:
        return binsearch(A[mid:], x) + mid
```

   (a) Devise a recurrence $T(n)$ that describes the worst-case number of steps taken by `binsearch` on input of size n (i.e. having `len(A) = n`). As usual, you may assume that n is a 'nice' size so that you can avoid floors and ceilings when dividing the input up into sublists (i.e., in this case, $n = 2^k$ for some $k \in \mathbb{N}$).

   Solution:

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 1 + T(n/2) & \text{if } n > 1 \end{cases}$$

   Remark: For simplicity, I've chosen to count any constant amount of work as '1'. If I wanted to be more fastidious, I might have written the $n > 1$ case as something like $7 + T(n/2)$ to try to account for each individual operation (i.e. each comparison, arithmetic operation, slice, etc.) but this isn't necessary. I could also have simply used a variable such as c to represent the constant. Any of these choices are fine - they will have some effect on the closed form you find in part (b), though the big-Θ complexity will be the same regardless.

   (b) Use the technique of unwinding (AKA repeated substitution) to find a closed form for $T(n)$. You are welcome to use either of the techniques shown in lecture - either repeatedly expanding an algebraic expression, or drawing out a tree of recursive calls, and reasoning about the total height, and number of steps taken at each level. Verify your closed form by testing it on a small value of n. (You don't need to prove it.)

Solution:

$$\begin{aligned} T(n) &= 1 + T(n/2) \\ &= 1 + (1 + T(n/4)) \\ &= 1 + (1 + (1 + T(n/8))) \\ &\dots \\ &= 1 + (1 + (1 + \dots T(n/n))) \end{aligned}$$

I observe that the inputs to T follow the progression $n, n/2, n/4, n/8 ... n/n$, and that every time we expand the call to T, we add a 1 to the sum. Based on this pattern, I claim that the final sum will have $\log n + 1$ terms, making $T(n) = 1 \cdot (\log n + 1) = \log n + 1$. This closed form gives $T(2) = 1 + 1 = 2$, which agrees with the recursive definition of $T(2) = 1 + T(1) = 1 + 1$.

Alternative: If you instead drew a call tree for `binsearch`, it should have looked like the one we drew in lecture for `fact` - i.e. just a straight line with a single node at each level. Each node would have a value of 1, since the non-recursive work is constant. Observing that the nodes follow a progression of dealing with input sizes $n, n/2, n/4 \dots n/n$, I surmise its height is $\log n$, meaning it has $\log n + 1$ levels. Giving me a closed form of $T(n) = \log n + 1$.

2. Consider the following recurrence defined over $\mathbb{N}^+$ (the positive naturals):

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ n + 4T(n/4) & \text{if } n > 1 \end{cases}$$

Use induction to prove the closed form $T(n) = n \log_4 n + n$ holds for all powers of 4.

(I suggest using complete induction with the predicate $P(n) : n$ is a power of 4 $\implies T(n) = n \log_4 n + n$, but it can also be done using simple induction, if you do the induction on a different variable.)

Solution:

Complete induction proof using the predicate $P(n)$ defined above.

Let $n \in \mathbb{N}^+$. Assume $\forall k \in \mathbb{N}, k < n \implies P(k)$

Case 0: n is not a power of 4 Then $P(n)$ is vacuously true.

Case 1: $n = 1$ Then, by definition, $T(n) = 1 = 0 + 1 = \log_4 1 + 1$. So $P(n)$ holds.

Case 2: n is a power of 4 and $n > 1$ Then $n/4$ is also a power of 4 and is less than n, so by $P(n/4)$ we can write:

$$\begin{aligned} T(n) &= n + 4T(n/4) \\ &= n + 4(\frac{n}{4} \log_4 \frac{n}{4} + \frac{n}{4}) \\ &= n + n \log_4 \frac{n}{4} + n \\ &= n + n(\log_4 n - 1) + n && \text{\# By log identity} \\ &= n \log_4 n + n \end{aligned}$$

As required. Thus $P(n)$.

Alternative: Using a change of variable, this could also be proven using simple induction with the predicate $P(j) : T(4^j) = 4^j \cdot j + 4^j$