Math 15 - Spring 2017 - Homework 5.6 Solutions

1. (5.6 # 14) Consider the merge subroutine of Algorithm 5.11. The goal of this exercise is to prove that this algorithm is correct.

   (a) State an appropriate loop invariant. (Hint: It will be similar to [but not the same as] the invariant used to verify the selection sort in Example 5.27.)
   ANS: The constant element here is that $\{x_1, x_2, \ldots, x_{k-1}\}$ contains both $\{y_1, y_2, \ldots, y_{i-1}\}$ and $\{z_1, z_2, \ldots, z_{j-1}\}$, and $x_1 \leq x_2 \leq \cdots \leq x_{k-1}$, where $i + j = k + 1$, and the remaining x's and y's are larger than x_k.

   (b) Prove that your proposed statement is a loop invariant. (Hint: Your proof can [and should] make use of the preconditions of the algorithm.)
   ANS: If $x_1 \leq x_2 \leq \cdots \leq x_{k-1}$ and $\{x_1, x_2, \ldots, x_{k-1}\}$ includes $\{y_1, y_2, \ldots, y_{i-1}\}$ and $\{z_1, z_2, \ldots, z_{j-1}\}$, and $i+j = k+1$, and the remaining y's and z's are greater than or equal to x_{k-1} as a precondition to an iteration of the loop, then what happens on the next loop? For sure, $k = k+1$. Consider the following cases:

   - If $i > l$, then all the remaining y's are in $\{x_1, \ldots, x_{k-1}\}$, so $x_k = z_j$, the smallest remaining z, and $j = j + 1$. So $\{x_1, \ldots, x_{k-1}\}$ is still in order, but now contains z_{j-1} too.
   - If $j > m$, then all the remaining z's are in $\{x_1, \ldots, x_{k-1}\}$, so $x_k = y_i$, the smallest of the remaining y's, and $i = i + 1$. So $\{x_1, \ldots, x_{k-1}\}$ is still ordered, but now contains y_{i-1} too.
   - Otherwise, y_i is the smallest of the remaining y's and z_j is the smallest of the remaining z's. If $y_i \leq z_j$, then $x_k = y_i$, the smallest remaining element, and $i = i + 1$. So $\{x_1, \ldots, x_{k-1}\}$ is still in order, but now contains y_{i-1} too. Likewise, if $y_i > z_j$, then $x_k = z_j$, the smallest remaining element, and $j = j + 1$. So $\{x_1, \ldots, x_{k-1}\}$ is still in order, and now contains z_{j-1} too.

   In every case, either $i = i + 1$ or $j = j + 1$, but not both, so $i + j = k + 1$.

   (c) Prove that the Merge algorithm is correct.
   ANS: Initially, $k = i = j = 1$, so $i+j = k+1$, and the rest of the invariant is true. Thus the invariant stays true until termination, when $k = l + m + 1$. Therefore $i + j = l + m$, so $\{x_1, x_2, \ldots, x_n\}$ is in order and contains all the y's and z's.

   ANS: Proof (by induction on n, the size of X.)
   If $n = 0$, then $X = \emptyset$, which does not contain t, and so $\texttt{RSearch}(t, X)$ returns false.
   The inductive hypothesis assumes that $\texttt{RSearch}(t, X)$ returns true if and only if $t \in X$, for $|X| = k - 1$, for some $k > 0$.
   Let X be a set with k elements. Then $\texttt{RSearch}(t, X)$ returns $(t = x_k) \vee \texttt{RSearch}(t, X \setminus \{x_k\})$.
   By the inductive hypothesis, $\texttt{RSearch}(t, X \setminus \{x_k\})$ will be true if and only if $t \in X \setminus \{x_k\}$. Therefore, $t \in X \Leftrightarrow (t = x_k) \vee (t \in X \setminus \{x_k\}) \Leftrightarrow (t = x_k) \vee \texttt{RSearch}(t, X \setminus \{x_k\}) \Leftrightarrow \texttt{RSearch}(t, X)$, as desired.

2. (5.6 # 16) Use an invariant to show that Prim's algorithm (5.9) always produces a tree.
   Proof. Suppose $x|m$ and $x|(n \mod m)$. Then by definition there is some integer k such that $m = kx$ and there is some integer j such that $r = jx$, where $r = n \mod m$. By the definition of the mod operation, there is some integer q such that $n = qm + r$. Therefore $n = qkx + jx = (qk + j)x$, so $x|n$.

3. (5.5 # 22) Test Algorithm 5.19 for solving the Towers of Hanoi puzzle in the case of $n = 3$ disks using a top-down evaluation.
   ANS: The invariant is that T

$$\begin{aligned}\texttt{H}(3,1,3) &= \texttt{H}(2,1,2), (1,3), \texttt{H}(2,2,3) \\ &= \texttt{H}(1,1,3), (1,2), \texttt{H}(1,3,2), (1,3), \texttt{H}(1,2,1), (2,3), \texttt{H}(1,1,3) \\ &= (1,3), (1,2), (3,2), (1,3), (2,1), (2,3), (1,3)\end{aligned}$$

   Here's the Python code we produced in class:

```
def Hanoi(n,start,goal):
    if n==1:
        return (start,goal)
    else:
        temp = 6-(start+goal)
        return Hanoi(n-1,start,temp),(start,goal),Hanoi(n-1,temp,goal)
```

If you then `print(Hanoi(3,1,3))`, the output is

```
(((1, 3), (1, 2), (3, 2)), (1, 3), ((2, 1), (2, 3), (1, 3)))
```

The extra parentheses add some clarity: First move the top two discs from 1 to 2, then move the bottom disc from 1 to 3, then move the two discs from 2 to 3, using 1 as the temp.
If you `print(Hanoi(8,1,3))`, the output is longer, but has the same basic flavor:

```
(((((((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2))),
  (1, 3),
     (((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3)))),
  (1, 2),
    ((((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1))),
  (3, 2),
     (((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2)))),
  (1, 3),
   (((((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3))),
  (2, 1),
     (((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1)))),
  (2, 3),
    ((((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2))),
  (1, 3),
     (((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3))))),
  (1, 2),
  ((((((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1))),
  (3, 2),
     (((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2)))),
  (3, 1),
    ((((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3))),
  (2, 1),
     (((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1)))),
  (3, 2),
   (((((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2))),
  (1, 3),
     (((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3)))),
  (1, 2),
    ((((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1))),
  (3, 2),
     (((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2))))))),
  (1, 3),
 (((((((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3))),
  (2, 1),
     (((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1)))),
```

```
(2, 3),
  ((((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2))),
(1, 3),
   (((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3)))))),
(2, 1),
 (((((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1))),
(3, 2),
   (((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2)))),
(3, 1),
  ((((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3))),
(2, 1),
   (((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1))))))),
(2, 3),
((((((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2))),
(1, 3),
   (((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3)))),
(1, 2),
  ((((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1))),
(3, 2),
   (((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2))))),
(1, 3),
 (((((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3))),
(2, 1),
   (((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1)))),
(2, 3),
  ((((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2))),
(1, 3),
   (((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3))))))))
```

4. (5.5 # 24) Use a recurrence relation to show that Algorithm 5.19 (the Towers of Hanoi algorithm) will always return a sequence of $2^n - 1$ ordered pairs.
ANS: First we want a recurrence relation. Let $P(n) =$ the number of pairs returned by `Hanoi`(n, T_1, T_2). If there's only one disc, then `Hanoi` will return one pair `(start,goal)`, so `Hanoi(start,goal) = (start,goal)`. If $n > 1$, then we have the recursion $P(n) = P(n-1) + 1 + P(n1)$, given $2P(n-1) + 1$ pairs. The recurrence relation is,

$$P(n) = \begin{cases} 1 & : n = 1 \\ 2P(n-1) & : n > 1 \end{cases}$$

We are to prove that $P(n) = 2^n - 1$, which we do in the usual manner of induction on n. If $n = 1$, $P(1) = 1 = 2^1 - 1$. The inductive hypothesis has that $P(k-1) = 2^{k-1} - 1$, for $k > 1$.
Now $P(k) = 2P(k-1) + 1 = 2 \cdot (2^{k1} - 1) + 1 = 2^k - 1$, as desired.
Take a look at MIT OCW for a discussion of this.

As described there, a general homogeneous linear recurrence of order d has the form

$$P(n) = a_1 P(n-1) + a_2 P(n-2) + \cdots + a_d P(n-d)$$

We benefit centuries of hindsight. Linear recurrences tend to have exponential solutions. So we guess that

$$P(n) = x^n$$

where x is a parameter introduced to help make making a good guess. We about how to choose the value for x later. To further improve our odds, we neglect the boundary conditions, $P(0) = P_0$ and $P(1) = P_1$, for now. Plugging this into the recurrence relation leads to the *characteristic polynomial*,

$$x^n = a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_d x^{n-d}$$

The characteristic polynomial will have n (potentially distinct) zeros and any linear combination of these is also a solution to the recurrence relation.
Recall that this is how we found the closed-form solution to the Fibonaccie recurrence, $F(n) = F(n-1)+F(n-2)$ whose charactoristic polynomial equation is $x^2 = x+1$ with root $\alpha = \dfrac{1+\sqrt{5}}{2}$ and $\beta = \dfrac{1-\sqrt{5}}{2}$.
Now to find a linear combination that solves the boundary conditions $F(0) = 1$ and $F(1) = 1$ we write $F(0) = a \cdot \alpha^0 + b \cdot \beta^0 = a + b = 1$ and $F(1) = a \cdot \alpha^1 + b \cdot \beta^1 = a\alpha + b\beta = 1$ we solved this $s \times 2$ linear system to get $a = \dfrac{\alpha}{\sqrt{5}}$ and $b = -\dfrac{\beta}{\sqrt{5}}$ to get

$$F(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^{n+1}$$

The Tower of Hanoi recurrence is not homogeneous, so we need to be a little more ingenious. The general non-homogeneous recurrence relation has the form

$$P(n) = a_1 P(n-1) + a_2 P(n-2) + \cdots + a_d P(n-d) + g(n)$$

where $g(n)$ is the non-homogeneous term. Here are the steps for solving a non-homogeneous recurrence relation:

1. Replace $g(n)$ by 0, leaving a homogeneous recurrence.
2. Solve the homogeneous recurrence, but don't impose the boundary conditions yet. This is called the homogeneous part of the solution.
3. Now put back $g(n)$ and find a single solution to the recurrence, ignoring boundary conditions. This is called the *particular solution.*
4. Add the homogeneous and particular solutions to obtain a *general solution.*
5. Now impose the boundary conditions to determine constants by the usual method of generating and solving a sysmtem of equations.

The homogeneous equation for the Towers of Hanoi is $P(n) = 2P(n-1) \Leftrightarrow x^n = 2x^{n-1} \Rightarrow x = 2$ ($x = 0$ is just not gonna work.) Our "ansatz" for the non-homogeneous equation here is $P(n) = an + b$. Substituting this into the recurrence relation gives $an + b = 2(a(n-1) + b) + 1 \Leftrightarrow an + b - 2a + 1 = 0$ so $a = 0$ and $b = -1$ and the particular solution is $P(n) = -1$
Now we combine the homogeneous and particular solutions to get

$$P(n) = c2^n - 1$$

and impose the boundary condition, $P(1) = 1 \Rightarrow c2^1 - 1 = 1 \Leftrightarrow c = 1$ and the solution is, indeed, $P(n) = 2^n - 1$.

The last lecture of the MIT OCW course address the problem of gambler's ruin, as applied to Roulette, which I'll recap here.

A gambler goes to Atlantic City with \$n to gamble. Her plan is to make only \$1 bets shell play until she is broke or she has won \$m. In the latter case, she will go home with $w = n + m$, the money she

started with plus her winnings.

In roulette, the probability of winning a bet is $p = \frac{9}{19}$ and probability of losing is $1 - p = \frac{10}{19}$. We want a recurrence relation for the probability that she wins starting with n dollars:

$$P(n) = \begin{cases} 1 & : n = w \\ 0 & : n = 0 \\ pP(n+1) + (1-p)P(n-1) & : 0 < n < w \end{cases}$$

This is the homogeneous recurrence $pP(n+1) - P(n) + (1-p)P(n-1)$ with the characteristic equation

$$px^2 - x + 1 - p = 0 \Leftrightarrow \left(x - \frac{1}{2p}\right)^2 = \frac{p-1}{p} + \frac{1}{4p^2}$$

yielding $x = \frac{1}{2p} \pm \sqrt{\frac{4p^2 - 4p + 1}{4p^2}} = \frac{1 \pm (2p-1)}{2p} = 1$ or $\frac{1-p}{p}$ A linear combination of these gives the general solution, $P(n) = a \cdot 1^n + b \cdot \left(\frac{1-p}{p}\right)^n$ and imposing the boundary conditions yields the system

$$\begin{aligned} a + b &= 0 \\ a + \left(\frac{1-p}{p}\right)^w \cdot b &= 1 \end{aligned}$$

which has the solution $a = \frac{1}{\left(\frac{1-p}{p}\right)^w - 1}, \quad b = -\frac{1}{\left(\frac{1-p}{p}\right)^w - 1}$

Thus the solution is

$$\begin{aligned} P(n) &= \left(\frac{1}{\left(\frac{1-p}{p}\right)^w - 1}\right)\left(\frac{1-p}{p}\right)^n - \frac{1}{\left(\frac{1-p}{p}\right)^w - 1} \\ &= \frac{\left(\frac{1-p}{p}\right)^n - 1}{\left(\frac{1-p}{p}\right)^w - 1} \end{aligned}$$

So if our gambler is playing roulette, where $p = \frac{9}{19}$ so that $\frac{1-p}{p} = \frac{10}{19} \cdot \frac{19}{9} = \frac{10}{9}$, and she starts with $n = 1000$ and wants to gamble until she's broke or wins \$100, then we're computing $P(1000) = \frac{\left(\frac{10}{9}\right)^{1000} - 1}{\left(\frac{10}{9}\right)^{1100} - 1} \approx \left(\frac{9}{10}\right)^{100} \approx 2.7 \times 10^{-5}$ or 1 in 38000. Not likely!

How does this story change if

- It's a fair game, that is, $p - \frac{1}{2}$?
- You're rolling two dice and there's a payoff/penalty for each of the 11 different sums for the two dice?
- Space is tesselated by some dodecahedral structure and you have a probability distribution determining the likelihood of travelling to an adjacent polyhedron through one of the 12 faces. There are various consequences for reaching various boundaries of your space. Analyze.