CSCE 222 (Carlisle), Discrete Structures for Computing
Spring 2022
Homework 8

Type your name below the pledge to sign
On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work.
HUY QUANG LAI

Instructions:

- The exercises are from the textbook. You are encouraged to work extra problems to aid in your learning; remember, the solutions to the odd-numbered problems are in the back of the book.
- Grading will be based on correctness, clarity, and whether your solution is of the appropriate length.
- Always justify your answers.
- Don't forget to acknowledge all sources of assistance in the section below, and write up your solutions on your own.
- *Turn in .pdf file to Gradescope by the start of class on Monday, March 21, 2022.* It is simpler to put each problem on its own page using the LaTeX clearpage command.

Help Received:

- Rosen, Kenneth H. *Discrete Mathematics and Its Applications*. McGraw-Hill, 2019.

Exercises for Section 5.3:

6(a,c,d): (3 points). Do NOT do proofs. For 6d it should read $n \geq 2$
Determine whether each of these proposed definitions is a valid recursive definition of a function f from the set of non-negative integers to the set of integers. If f is well defined, find a formula for $f(n)$ when n is a non-negative integer and prove that your formula is valid.

$f(0) = 1, f(n) = -f(n-1)$ for $n \geq 1$
Valid recursive definition.
$f(n) = (-1)^n$

$f(0) = 0, f(1) = 1, f(n) = 2f(n+1)$ for $n \geq 2$
Invalid recursive definition.

$f(0) = 0, f(1) = 1, f(n) = 2f(n-1)$ for $n \geq 2$
Valid recursive definition.
$f(2) = 2f(1) = 2, f(3) = 2f(2) = 4, \cdots$
$f(n) = 2^{\lfloor \frac{n}{2} \rfloor}$

12: (2 points).

f_n is the nth Fibonacci number.

Prove that $f_1^2 + f_2^2 + \cdots + f_n^2 = f_n f_{n+1}$ when n is a positive integer.

Base Case:
$f_1^2 = 1^2 = 1, f_1 \cdot f_2 = 1 \cdot 1 = 1$
True for $n = 1$

$f_1^2 + f_2^2 = 1^2 + 1^2 = 2, f_2 \cdot f_3 = 1 \cdot 2 = 2$
True for $n = 2$

Inductive Hypothesis:
Assume $\displaystyle\sum_{i=1}^{n} f_i^2 = f_n \cdot f_{n+1}$

Inductive Step:
$\displaystyle\sum_{i=1}^{n+1} f_i^2$
$= \displaystyle\sum_{i=1}^{n} f_i^2 + f_{n+1}^2$
$= f_n \cdot f_{n+1} + f_{n+1}^2$
$= f_{n+1}(f_n + f_{n+1})$
$= f_{n+1} f_{n+2}$

18: (3 points).

Let
$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$

Show that
$A^n = \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix}$

when n is a positive integer.

Base Case:
$A^1 = \begin{bmatrix} f_2 & f_1 \\ f_1 & f_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$

Inductive Hypothesis:
Assume: $A^n = \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix}$ for all $n \geq 2$

Inductive Step:
$A^{n+1} = A^n \times A^1 = \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} =$
$\begin{bmatrix} f_{n+1} + f_n & f_n + f_{n-1} \\ f_{n+1} & f_n \end{bmatrix} = \begin{bmatrix} f_{n+1} & f_{n+1} \\ f_{n+1} & f_n \end{bmatrix}$

46: (2 points).
Use structural induction to show that $l(T)$, the number of leaves of a full binary tree T, is 1 more than $i(T)$, the number of internal vertices of T.

The full binary tree consisting of a single vertex, denoted $\bullet$, clearly has 1 leaf and 0 internal vertices: $l(\bullet) = i(\bullet) + 1$.

Let T_L be the full binary tree left of $\bullet$ and T_R be the full binary tree right of $\bullet$.
Because T_L and T_R are connected with an internal vertex, we can define $i(T)$ as follows: $i(T) = i(T_L) + i(T_R) + 1$.
Similarly, $l(T)$ can be defined as: $l(T) = l(T_L) + l(T_2)$. Combined with the base case, $l(T) = (i(T_L) + 1) + (i(T_R) + 1)$.
$l(T) = (i(T_L) + 1) + (i(T_R) + 1)$
$= i(T_L) + i(T_R) + 1 + 1$
$= i(T) + 1$

With this, the total length of a full binary tree is 1 more than the number of internal vertices.

Exercises for Section 5.4:

In the following problems, assume that the following functions are defined:

- def isNull(T : Tree) $\rightarrow$ bool:
- def value(T : Tree) $\rightarrow$ int:# may not be called on Null tree
- def left(T : Tree) $\rightarrow$ Tree: # may not be called on Null tree
- def right(T : Tree) $\rightarrow$ Tree: # may not be called on Null tree

Also, assume a binary search tree. That is, given Tree x, all values in the left subtree are $\leq value(x)$ and all values in the right subtree are $> value(x)$.

Custom 1: (3 points) Write a recursive method that finds the smallest item in the tree.

```
def findMin(T: Tree)
    if (isNull(T)):
        raise TypeError

    if isNull(left(T)):
        return value(T)

    return findMin(left(T))
```

Custom 2: (3 points) Write a recursive method that finds the sum of all the values in the tree.

```
def sum(T: Tree):
    if isNull(T):
        return 0

    return sum(left(T)) + value(T) + sum(right(T));
```

Custom 3: (2 points) Write a recursive method that returns $True$ if its integer parameter, x, is in the tree and $False$ otherwise.

```
def contains(x: value, T: Tree):
    if (isNull(T)):
        return False

    if value(T) == x:
        return True

    return contains(x, left(T)) or contains(x, right(T))
```

50: (2 points)

Sort $3, 5, 7, 8, 1, 9, 2, 4, 6$ using the quick sort.

```
protected static int[] quicksort(int[] list) {
    if (list.length == 0)
        return [];
    if (list.length == 1)
        return list;

    int pivot = median(list);
    int[] left = [], right = [];
    for (int value : list) {
        if (value < pivot)
            left = left.append(value);
        if (value > pivot)
            right = right.append(value);
    }
    return quicksort(left).append(pivot).append(quicksort(right));
}
```

1. Pivot = 5
   - Left = $\{3, 1, 2, 4\}$
   - Right = $\{7, 8, 9, 6\}$
2. Quicksort Left
   (a) Pivot = 2
      - Left = $\{1\}$
      - Right = $\{3, 4\}$

   (b) Quicksort Left
   List is length 1, do nothing

   (c) Quicksort Right
      i. Pivot = 3
         - Left = $\{\}$
         - Right = $\{4\}$

      ii. Quicksort Left
      List is empty do, nothing.

   iii. Quicksort Right
      List is length 1, do nothing
   iv. return Left, Pivot, Right
      $\{3, 4\}$
  (d) return Left, Pivot, Right
    $\{1, 2, 3, 4\}$

3. Quicksort Right

  (a) Pivot = 7
    - Left = $\{6\}$
    - Right = $\{8, 9\}$
  (b) Quicksort Left
    List is length 1, do nothing
  (c) Quicksort Right
   i. Pivot = 8
      - Left = $\{\}$
      - Right = $\{9\}$
   ii. Quicksort Left
      List is empty do, nothing.
   iii. Quicksort Right
      List is length 1, do nothing
   iv. return Left, Pivot, Right
      $\{8, 9\}$
  (d) return Left, Pivot, Right
    $\{6, 7, 8, 9\}$

4. return Left, Pivot, Right
  $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$