Homework 4

Josh Bowden

CS 330 - Discrete Structures - Professor Sasaki

February 27, 2017

Problem 1. Let $P(n) = 1^3 + 2^3 + \cdots + n^3 = \left(\frac{n(n+1)}{2}\right)^2$. Prove by induction that $\forall n \in \mathbb{N}\,(P(n))$.

Theorem. For every postive integer n,

$$\sum_{j=1}^{k} j^3 = \left(\frac{n(n+1)}{2}\right)^2$$

□

Proof. By induction on n,

Base case: $n = 1$

$$\sum_{j=1}^{1} j^3 = \left(\frac{1(1+1)}{2}\right)^2 \iff 1^3 = \left(\frac{2}{2}\right)^2 \iff 1 = 1$$

$\therefore \sum_{j=1}^{1} j^3 = \left(\frac{1(1+1)}{2}\right)^2$

Inductive step: Suppose that for positive integer k, $\sum_{j=1}^{k} j^3 = \left(\frac{n(n+1)}{2}\right)^2$, we will show that

$$\sum_{j=1}^{k+1} j^3 = \left(\frac{(k+1)(k+2)}{2}\right)^2$$

Starting with the left side of the equation:

$$\begin{aligned}
\sum_{j=1}^{k+1} j^3 &= \left(\sum_{j=1}^{k} j^3\right) + (k+1)^3 && \text{by separating out the last term} \\
&= \left(\frac{k(k+1)}{2}\right)^2 + (k+1)^3 && \text{by the inductive hypothesis} \\
&= \frac{k^2(k+1)^2}{4} + \frac{4(k+1)^3}{4} \\
&= \frac{k^2(k+1)^2 + 4(k+1)^3}{4} \\
&= \frac{(k+1)^2\left(k^2 + 4(k+1)\right)}{4} \\
&= \frac{(k+1)^2\left(k^2 + 4k + 4\right)}{4} \\
&= \frac{(k+1)^2(k+2)^2}{4} \\
&= \left(\frac{(k+1)(k+2)}{2}\right)^2 && \text{by algebra}
\end{aligned}$$

$\therefore \sum_{j=1}^{k+1} j^3 = \left(\frac{(k+1)(k+2)}{2}\right)^2$ □

Problem 2. Determine whether each of the proposed definitions is a valid recusive definition of a function f from the set of nonnegative integers to the set of integers. If f is well defiend, find a formula for $f(n)$ when n is a nonnegative integer and prove that your formula is valid.

(a) $f(0) = 1, f(n) = -f(n-1)$ for $n \geq 1$

This is a valid recursive definition of a function f.

A valid formula for $f(n)$ is

$$f(n) = (-1)^n$$

Proof. By induction on n,

Base case: $n = 0$

$$\begin{aligned} f(0) &= (-1)^0 \\ 1 &= 1 \end{aligned}$$

$\therefore f(0) = (-1)^0 = 1$

Inductive step: Suppose $f(n) = -f(n-1) = (-1)^n$ for $n \geq 1$, we will show that:

$$f(n) = -f(n-1) = (-1)^n$$

Since,

$$\begin{aligned} f(n) &= -f(n-1) \\ &= -(-1)^{(n-1)} && \text{by the inductive hypothesis} \\ &= -\frac{(-1)^n}{(-1)^1} \\ &= (-1)^n \end{aligned}$$

$\therefore f(n) = (-1)^n$ for $n \geq 0$. $\square$

(b) $f(0) = 1, f(1) = 0, f(2) = 2, f(n) = 2f(n-3)$ for $n \geq 3$

This is a valid recursive definition of a function f.

A valid formula for $f(n)$ is

$$g(n) = \begin{cases} 1 & n = 0 \\ 0 & (n \bmod 3) = 1 \\ 2^{\lfloor (n-1)/3 \rfloor + 1} & (n > 1) \text{ and } ((n \bmod 3) \neq 1) \end{cases}$$

Proof. By induction on n,

Base case: $n = 0$

$$f(0) = g(0) \iff 1 = 1 \qquad g(0) \implies (n = 0) \equiv (0 = 0) \equiv \mathrm{T}$$

$\therefore f(0) = 1 = g(0)$

Base case: $n = 1$

$$f(1) = g(1)$$
$$0 = 0 \qquad g(1) \implies ((n \bmod 3) = 1) \equiv ((1 \bmod 3) = 1) \equiv (1 = 1) \equiv \mathrm{T}$$

$\therefore\ f(1) = 0 = g(1)$

Base case: $n = 2$. Show that $f(2) = 2 = g(n)$.

$$\begin{aligned} f(2) &= g(2) \\ &= 2^{\lfloor (n-1)/3 \rfloor + 1} && \text{since } g(2) \implies ((n > 1) \wedge ((n \bmod 3) \neq 1)) \\ &&& \equiv ((2 > 1) \wedge ((2 \bmod 3) \neq 1)) \\ &&& \equiv (\mathrm{T} \wedge (2 \neq 1)) \\ &&& \equiv (\mathrm{T} \wedge \mathrm{T}) \\ &&& \equiv \mathrm{T} \\ &= 2^{\lfloor (2-1)/3 \rfloor + 1} \\ &= 2^{\lfloor 1/3 \rfloor + 1} \\ &= 2^{0+1} \\ &= 2 \end{aligned}$$

$\therefore\ f(1) = 0 = g(1)$

Inductive step: Suppose $f(n) = g(n)$ for $n > 1$ and $(n \bmod 3) \neq 1$, we will show that:

$$\begin{aligned} f(n) &= g(n) && \text{for } n > 1 \text{ and } (n \bmod 3) \neq 1 \\ 2f(n-3) &= 2^{\lfloor (n-1)/3 \rfloor + 1} && \text{for } n > 1 \text{ and } (n \bmod 3) \neq 1 \end{aligned}$$

Since,

$$2f(n-3) = 2^{\lfloor (n-1)/3 \rfloor + 1}$$

$\therefore\ f(n) = g(n)$ for $n > 1$ and $(n \bmod 3) \neq 1$. □

(c) $f(0) = 0, f(1) = 1, f(n) = 2f(n+1)$ for $n \geq 2$

This is not a valid recursive definition since $f(n)$ refers to $f(n+1)$ which will then refer to $f(n+2)$ and so on.

(d) $f(0) = 0, f(1) = 1, f(n) = 2f(n-1)$ for $n \geq 1$

This is a valid recursive definition of a function f.

A valid formula for $f(n)$ is

$$g(n) = \begin{cases} 0 & n = 0 \\ 2^{n-1} & n \geq 1 \end{cases}$$

Proof. By induction on n,

Base case: $n = 0$

$$f(0) = g(0)$$
$$0 = 0 \qquad g(0) \implies (n = 0) \equiv (0 = 0) \equiv \mathrm{T}$$

$\therefore\ f(0) = g(0) = 0$

Base case: $n = 1$

$$\begin{aligned} f(1) &= g(1) \\ 1 &= 2^{n-1} && g(1) \implies (n = 0) \equiv (0 = 0) \equiv \mathrm{T} \\ 1 &= 2^{1-1} \\ 1 &= 2^0 \\ 1 &= 1 \end{aligned}$$

$\therefore\ f(1) = g(1) = 1$

Inductive step: Suppose $f(n) = g(n)$ for $n \geq 1$, we will show that:

$$f(k+1) = g(k+1) = 2^k$$

Since,

$$\begin{aligned} f(k+1) &= g(k+1) && \text{by the inductive hypothesis} \\ &= 2^{(k+1)-1} && g(k+1) \implies (k+1 \geq 1) \equiv (k \geq 0) \equiv \mathrm{T} \\ &= 2^k \end{aligned}$$

$\therefore\ f(n) = g(n)$ for $n \geq 1$. $\square$

Problem 3. Give a recursive definition of the sequence $\{a_n\}, n = 1, 2, 3, \ldots$ if

(a) $a_n = 4n - 2$

$$f(n) = 4n - 6$$

(b) $a_n = 1 + (-1)^n$

$$f(n) = 1 + (-1)^n$$

(c) $a_n = n(n+1)$

$$f(n) = n^2 - n$$

(d) $a_n = n^2$

$$f(n) = n^2 - 2n + 1$$

Problem 4. Give a recursive definition

(a) the set of odd positive integers.

$$f(1) = 1, f(n) = f(n-1) + 2$$

Problem 5. Give a recursive algorithm for finding the maximum of a finite set of intergers, making use of the fact that the maxiumum of n is the larger of the last integer in the list and the maxiumum is the larger of last integer in the list and the maximum of the first $n-1$ integer in the list.

```
def max(xs: List<Int>, cur: Int) where xs.length >= 1:
    def f(xs: List<Int>, cur: Int) where xs.length >= 1:
       if xs.length == 0:
           return cur
       else:
          let x = xs[xs.length - 1]
          let rest = xs[0..(xs.length - 2)]
          if x > cur:
             return max(rest, x)
          else
             return max(rest, cur)
     return f(xs, xs[0])
```

Problem 6. Devise a recursive algorithm to find a^{2^n}, where a is a real number and n is a positive integer.

Let $f(a, n)$ where $a \in \mathbb{R}$ and $n \in \mathbb{N}$ represent the expression a^{2^n}.
Let $\exp(a, n)$ where $a \in \mathbb{R}$ and $n \in \mathbb{N}$ represent the expression a^n.

$$\exp(a, 1) = a$$
$$\exp(a, n) = a * \exp(a, n-1)$$
$$f(a, n) = \exp(a, \exp(2, n))$$

Problem 7. Give a recursive algorithm for computing $n * a$ ("n times a") using addition, where n is a positive integer and a is a real number.

Let $f(n, a)$ where $n \in \mathbb{N}$ and $a \in \mathbb{R}$ represent the expression $n * a$.

$$f(1, a) = a$$
$$f(n, a) = a + f(n-1, a)$$

Problem 8. Write the algorithm and the loop invariant (for the outer loop) for the iterative version of Bubble Sort and prove all three cases (Initialization, Maintenance, Termination).

```
def bubble_sort(xs: List):
    let n = xs.length
    let didSwap = true
    while didSwap:
        for x in {0..n-1}:
            if xs[i] > xs[i + 1]:
                let temp = xs[i]
                xs[i] = xs[i+1]
                xs[i+1] = temp
                didSwap = true
```