Frederik Lohner
Oct 7 2015
Discrete Structures

Challenge Problem 2

1. Graphing the output of our program for several hundred inputs tells us that the equation for the pile splitting problem is that the sum will always be: $\frac{n(n-1)}{2}$. We setup, a base case of 2, and split it into 1 and 1. 1x1 is 1 and $\frac{2(2-1)}{2}$ = 1, so we prove our base case. We assume that it is true for 1…k. So now we will look to prove it for k+1. We assume that if we split k into x and y that both x and y will be less than or equal to k. Which is part of our initial assumption therefore we can simplify x to $\frac{x(x-1)}{2}$ and y to $\frac{y(y-1)}{2}$. We also include the product of x and y as xy then look to simplify.

$$k+1= xy + \frac{x(x-1)}{2} + \frac{y(y-1)}{2}$$
$$= \frac{(x+y)(x+y-1)}{2}$$
$$= \frac{n(n-1)}{2}$$

2. Similar to the first question we utilized a program in order to get the solution to the problem, which ends up being that the input will always equal the output. Or n = x. We setup a base case of 2, and split it into 1 and 1. 1/1 + 1/1 is 2. So we prove our base case. We assume it is true for 1…k and then split into x and y where we know they are both less than or equal to k.

```
import UIKit
/*
The aim of our code was to recursively call upon itself in order
to be able to quickly solve and test for very large numbers.
Additionally, we included a way to quickly calculate values for a
large set of inputs in order to graph the input vs output
relationship and easily find relationships.

We begin by creating an arrayOfProducts of input type int, as
well as creating a structure called solutions with two variables
of type int for input and output.  We then initialize the array and
begin declaring our function.
*/
var arrayOfProducts : Array = [Int]()
struct solutions {
   var input : Int?
   var output : Int?
}
var arrayOfSolutions = [solutions]()
/*
We begin by declaring our function with the input of the number we
are looking to split (as an Int), followed by checking if it is an
even number.  If it is even, we split it equally into two values and
```

```
then append the product to our array.  Then we check if the split
values are equal to 1 and if not then we recursively call the
function on the value again.  If the value is not even, then we split
in half (since the value will round down) and then add 1 to the
second value.  We then check if they are equal to 1 and if not call
the function recursively on the value again.
*/
func pileSplitter(numberToSplit: Int)  {
   if(numberToSplit%2 == 0) {
      let firstNumber = numberToSplit/2
      let secondNumber = firstNumber
      let product = firstNumber * secondNumber
      arrayOfProducts.append(product)
      if(firstNumber != 1) {
         print(firstNumber)
         pileSplitter(firstNumber)
      }
      if(secondNumber != 1) {
         print(secondNumber)
         pileSplitter(secondNumber)
      }
   } else {
      let firstNumber = numberToSplit/2
      let secondNumber = (numberToSplit/2)+1
      let product = firstNumber * secondNumber
      arrayOfProducts.append(product)
      if(firstNumber != 1) {
         print(firstNumber)
         pileSplitter(firstNumber)
      }
      if(secondNumber != 1) {
         print(secondNumber)
         pileSplitter(secondNumber)
      }
   }
}
/*
The purpose of this second function is to allow us to input a large amount of
values into our pileSplitter function in order to get a large amount of outputs
and be able to use this data to determine what our equation is.  We begin by
passing in the range that we wish to check to and then execute a for loop up
to the value.
*/
func findTheFormula(range: Int) {
   for index in 2...range {
      arrayOfProducts.removeAll()
      pileSplitter(index)
      var sum = 0
      for items in arrayOfProducts {
         sum += items
      }
      let currentSolution = solutions(input: index, output: sum)
      arrayOfSolutions.append(currentSolution)
   }
   for index in arrayOfSolutions {
      print(index.output!)
   }
}

/*================================================================*/
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
/*================================================================*/

var arrayOfReciprocalSums : Array = [Double]()
struct solutionsReciprocal {
   var input : Int?
   var output : Double?
}
var arrayOfReciprocalSolutions = [solutionsReciprocal]()
/*
```

```
Our second function is very similar to our previous function but due
to the reciprocals we have to convert some of our values to doubles in
order to be able to calculate fractions.
*/
func pileSplitterReciprocal(numberToSplit: Int){
    if(numberToSplit%2 == 0) {
        let firstNumber = numberToSplit/2
        print(firstNumber)
        let secondNumber = firstNumber
        print(secondNumber)
        let firstNumberDouble : Double = Double(firstNumber)
        let secondNumberDouble : Double = Double(secondNumber)
        let reciprocalSum = (1.0/firstNumberDouble) + (1.0/secondNumberDouble)
        print(reciprocalSum)
        arrayOfReciprocalSums.append(reciprocalSum)
        if(firstNumber != 1) {
            print(firstNumber)
            pileSplitterReciprocal(firstNumber)
        }
        if(secondNumber != 1) {
            print(secondNumber)
            pileSplitterReciprocal(secondNumber)
        }
    } else {
        let firstNumber = numberToSplit/2
        print(firstNumber)
        let secondNumber = (numberToSplit/2)+1
        print(secondNumber)
        let firstNumberDouble : Double = Double(firstNumber)
        let secondNumberDouble : Double = Double(secondNumber)
        let reciprocalSum = (1.0/firstNumberDouble) + (1.0/secondNumberDouble)
        print(reciprocalSum)

        arrayOfReciprocalSums.append(reciprocalSum)
        if(firstNumber != 1) {
            print(firstNumber)
            pileSplitterReciprocal(firstNumber)
        }
        if(secondNumber != 1) {
            print(secondNumber)
            pileSplitterReciprocal(secondNumber)
        }
    }
}
func findTheFormulaReciprocal(range: Int){
    for index in 2...range {
        arrayOfReciprocalSums.removeAll()
        pileSplitterReciprocal(index)
        var product = 1.0
        for items in arrayOfReciprocalSums {
            product = product * items
        }
        let currentSolution = solutionsReciprocal(input: index, output: product)
        arrayOfReciprocalSolutions.append(currentSolution)
    }
    for index in arrayOfReciprocalSolutions {
        print(index.output!)
    }
}

findTheFormula(10)
findTheFormulaReciprocal(10)
```