CS 1510: Homework 13

John Hofrichter   Kyra F. Lee   Zach Sadler

jmh162@pitt.edu   kfl5@pitt.edu   zps6@pitt.edu

October 1, 2013

Dynamic Programming Problem 16

Explanation

The first thing to note in this problem is that our bounds for runtime are very large. For instance, if our input is $1, 2, 2, 3, 4, 5, 6, 7$ then we have 8 numbers total ($n = 8$) and L = $\max(792, 10080) = 10080$. So to solve this small input of simply 8 single-digit numbers we are required to be within polynomial of over $10,000$ operations, which is enormous compared to $2^8 = 256$, which is exponential runtime and typically considered to be way too slow for an acceptable solution.

So we understand that even an exponential solution may be a good one compared to our acceptable bounds, so what exactly will we have for our solution? We've already seen that 2^n can be a better runtime than the bounds we're given, and in fact our solution will run in about $O(2^n)$.

We will calculate the power set of our input and then calculate the cubic sum and total product of each subset to see if it is a solution. If it is, then we finish and return that subset as a valid solution. So consider the following algorithm:

Algorithm

```
PowerSet = null
for i = 1 to n do
    newPowerSet = null
```

```
    for subset 1 to PowerSet.size do
        newPowerSet.add(subset)
        newSubset = subset + i
        newPowerSet.add(newSubset)
    end for
    PowerSet = newPowerSet
end for
//So now we have PowerSet as our complete power set of the input.
for i = 1 to PowerSet.size do
    if cubicSum(PowerSet[i]) == totalProduct(PowerSet[i]) then
        return i
    end if
end for
return "no solution"
```