# CS5050 Assignment Three

Dustin E. Homan

January 31, 2016

## Mathematical Analysis

### Recursive Algorithm

To begin our analysis of the recursive algorithm we used to solve our Nim problem, let us look at the following, simple pseudo-code:

```
function WIN_R(a, b, c)
    BASE CASE
    for i to a do Win_R(a − i, b, c)
    end for
    for i to b do Win_R(a, b − i, c)
    end for
    for i to c do Win_R(a, b, c − i)
    end for
end function
```

Let $f(n) :=$ the number of base cases generated and $f(1) = 1$. From the above algorithm we can deduce that $f(n) = \sum_{i=0}^{a} f(n-i) + \sum_{i=0}^{b} f(n-i) + \sum_{i=0}^{c} f(n-i) = f(n-1) + f(n-2) + \cdots + f(n-3a)$

Therefore $f(n)$ is a linear-homogeneous recurrence relation and can be solved with the method of characteristic roots. To simply the analysis we will say that $f(n) = O(C^n)$.

### Memoizing Algorithm

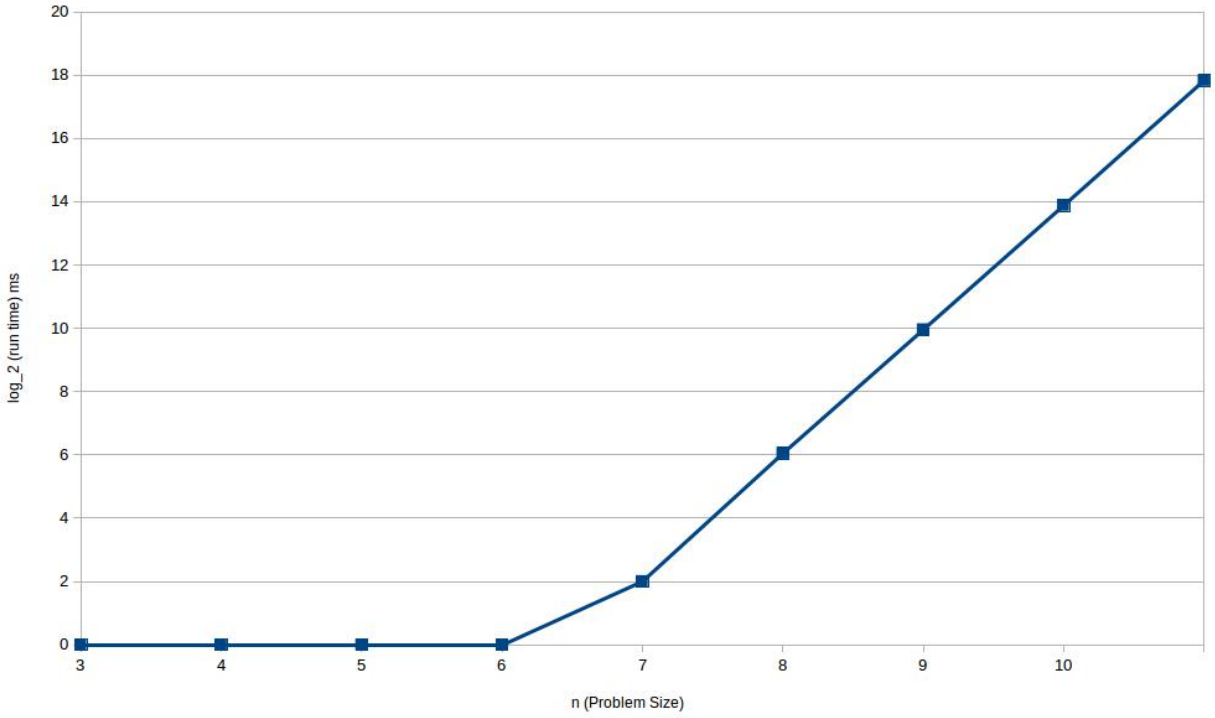
Let $f(n) :=$ the number of base cases generated and $f(1) = 1$. The memoizing algorithm has a logic short-circuit. This works because we start at the solution and build it from going back. Therefore the first time we encounter an instance where we can win, we just return the solution and stop. However in the worst-case we will have to build the entire solution. Therefore $f(n)$ will equal the number of solutions in the problem space. With three piles $f(n) = O(n^3)$ in the worst-case.

### Dynamic Programming Algorithm

Let $f(n) :=$ the number of assignments to the cache and $f(1) = 1$. Like the memoizing algorithm the dynamic programming algorithm builds the solution. Unlike the memoizing the DP algorithm will always find every solution in the problem space. Therefore $f(n) = O(n^3)$.

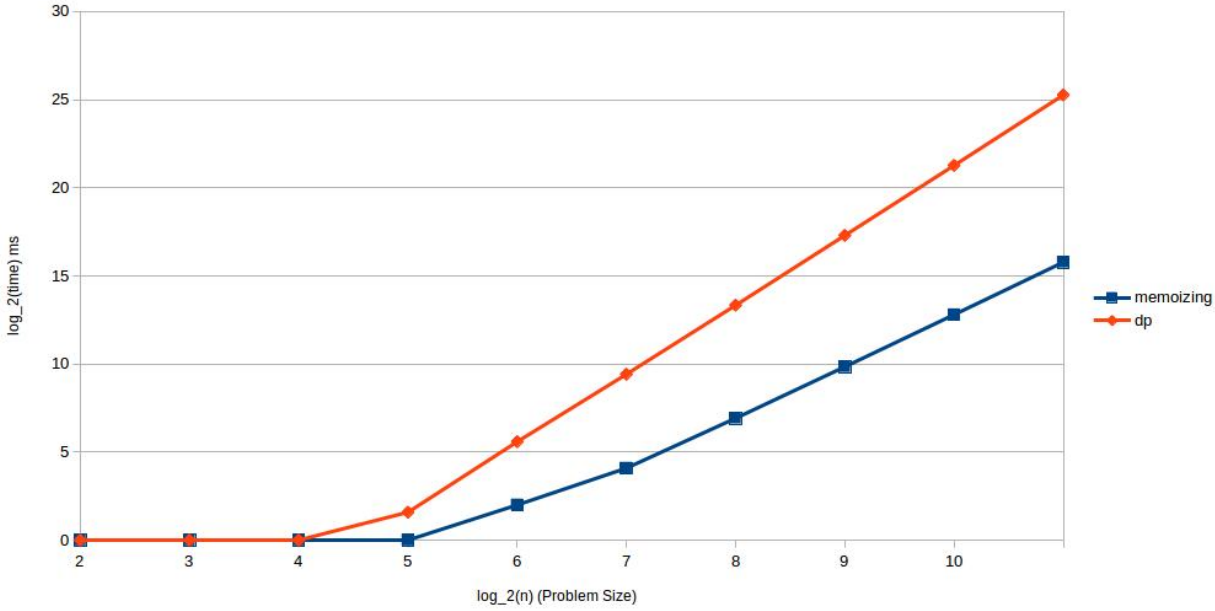
# Empirical Study

## Recursive Algorithm




Slope: 4 $log_2(ms)$ / 1unit increase in problem size

## Memoizing & Dynamic Programming Algorithms




Memoizing Slope: 3 $log_2(ms)/log_2$(unit increase)
DP Slope: 4 $log_2(ms)$ / $log_2$(unit increase)

## Conclusions

In conclusion the data from the empirical studies match the estimated results from the mathematical analysis. The recursive algorithm was estimated to have $O(C^n)$ complexity. The data shows that for 1 unit increase in problem size the log base 2 of the run time increases by a factor of 4. Therefore the constant, $C$, is approximately 2.

The mathematical analysis of both the memoizing and dynamic programming algorithms were predicted to have $O(n^3)$ complexity. The data for the DP algorithm shows that as the problem size doubles the run time increases by a factor of 8. This verifies our prediction of $O(n^3)$ complexity. The data for the memoizing algorithm shows that the complexity is less than cubic. This is due to the short circuit mentioned above. The difference in the empirical study and prediction shows that the algorithm short-circuits more than it goes through the whole problem space.