Lab 3 Report

Anthony Weems

November 22, 2015

1 Problem Definition

The overall problem is to maximize the grade of a student by planning their time allotments for projects. Given a set of non-decreasing functions, we must maximize the student's total grade by determining an hour allotment for each function.

2 Optimal Substructure

We will show that the problem above has optimal substructure by defining its subproblems. Calculating the maximum grade for a particular problem can be done by considering the subproblem with one fewer project. We must also consider all hour allotments using the current project and compare them to the subproblem with one fewer project and H hours, where H is the difference between the total hours and the considered hour allotment. Maximizing the grade of these comparisons will return the ultimate highest grade possible for the student.

3 Recursive Definition

Our recursive function G[H,j] represents the maximum grade given H hours using the first j projects. We also define $G[H,j]$ to be equal to zero for all j equal to zero.

$$G[H,j] = \max\{\max_{1 \le i \le H}\{G[H-i, j-1] + f_j(i)\}, G[H][j-1]\} \tag{1}$$

During the runtime of the algorithm, we track a second matrix of "interesting" values that allow us to reconstruct the solution after finding the maximum score. This matrix represents the hours used by a particular project at the point $G[H,j]$.

$$M[H,j] = \begin{cases} -1 & \text{if } G[H][j] = G[H][j-1] \\ i \text{ that maximized } G[H][j] & \text{otherwise} \end{cases} \quad (2)$$

4 Iterative Algorithm

```
function solve(hours)
  initialize G[H,j]
  if we have solved this problem, return the solution
  for all classes j
    for all hours H
      find an i that maximizes G[h-i][j-1]+f_j(i)
      G[H][j] = the max of the previous maximum and G[H][j-1]

  calculate solution by finding the best values in G for each j
  save this solution
  return the solution
```

5 Testing Methodology

All testing was manually verified by hand. Each test included random non-decreasing function to attempt to find corner-cases. The `MyGradeFunction` class implements the random function generator used for testing.