**CS 577: Introduction to Algorithms** 9/28/15

# Homework 6 (Rubric)

Instructor: Dieter van Melkebeek TA: Bryce Sandlund

## Problem 3 [10 points]

- [1 point] dynamic programming
- [4 points] correctness
  - [2 points] correct states (-1 for not stating what each DP state denotes, -2 if incorrect)
  - [2 points] correct recurrence
- [2 points] $O(n^2)$ algorithm (runtime points only given for solutions that were close to correct)
- [2 points] runtime argument
- [1 point] correctness argument

Of the solutions that were not correct, some of them either did not use dynamic programming at all, or had different states and recurrence than the intended solution. It doesn't really work to put a non-integer value as part of the state space, and other formulations were unnecessarily complicated and difficult to verify correct or incorrect. As with greedy (I guess this can be said in general), if you find yourself adding complications to the states and extra values to fix issues, you're probably going about it the wrong way. Try to think of a simpler idea and experiment with different ways to break the problem into subproblems and compute recurrences between the subproblems. It takes a bit of practice to get the knack for finding the correct state breakdown and recurrence for DP problems.

Of the solutions that were correct, I took off a point if it was not explicitly stated what the calculation at each state meant. It's a really good idea to state this, as it can get much more complicated when you work on more advanced DP algorithms, and it clarifies to the reader exactly what is going on. I also took off a point if it was not explicitly mentioned how to evaluate the recurrence in $O(n)$ time, since computing $\max(a_1, a_2, \ldots, a_k)$ in each iteration naively leads to $O(n^2)$ work to compute a single value in the DP table.

Finally, I'd like to explicitly note that for DP algorithms, you really want to write a formal recurrence. It is the most concise way to state the algorithm. For proofs of correctness, I was pretty lenient on this homework, but in general you should make some kind of argument as to why every possible best solution is evaluated in your algorithm. For this homework, a good proof would mention that for the computation of $OPT(i)$, where $OPT(i)$ denotes the lowest height of placing books $1..i$ on the bookshelf, every configuration of the last shelf in the bookshelf is being considered. Take a look at the model solution outside of Dieter's office for more details.