MATH182 HOMEWORK #6
DUE July 31, 2020

Write a brief reflective essay discussing your personal journey and growth in the subject of algorithms these past six weeks. Some questions to consider:

- What new points of view do you have?
- What new ways do you think about things?
- Are there any topics we covered which resonated with you?
- In the future, which things from this class do you imagine using?

Despite having previously encountered many of the topics covered in this class because of a long-standing Project Euler hobby, studying such concepts formally and rigorously has been profoundly illuminating. There are, for me, few feelings in the world comparable to solving a problem that once felt unapproachable, and I've enjoyed solving problems of exactly that nature this summer.

Some of my favourite problems from when I used to work towards a Project Euler problem every other day were those that required dynamic programming solutions. From my first DP solution (Problem 18, Maximum Path Sum I), I've loved the "cleverness" of DP algorithms. Much of high school and undergraduate coursework is about plugging and chugging — learning all sorts of formulae for solving classes of problems — and as such, most of my admittedly limited experience with mathematics so far has involved "brute-forcing" through everything. Maybe it's for that reason dynamic programming (and, similarly, greedy algorithms) feel so wondrously novel.

In either case, when I'd just started with Algorithms, I thought I understood DP reasonably well only to later realise, through discussions after class (seriously, thanks so much for staying back to take my doubts every other day) and in office hours, how limited my understanding of optimal substructure was. Broadening that understanding unlocked a whole class of problems I previously had no idea how to approach, such as Project Euler Problem 83 (Path Sum: Four Ways), a problem I hadn't been able to solve despite taking multiple stabs at it over the years. I'd been trying to solve it with a DP algorithm similar to that I used for Problems 81 and 82, where I would solve each subproblem exactly once and store the solution, but that approach just wasn't working for Problem 83 because of the extra dimension of each sub-problem. Convinced the problem didn't have optimal substructure (and now a little more familiar with greedy algorithms), I implemented Djikstra's algorithm and got an efficient, albeit a little unsatisfying, solution.

In all my reading on Djikstra's algorithm, it was presented as a greedy algorithm, and it certainly seemed to work like one (and nothing like a DP algorithm). In a discussion after lecture, Professor Gehret pointed out how if we define the sub-problem slightly differently, Djikstra's algorithm can in fact be thought of as a DP algorithm.[1] This was one of my favourite realisations of the quarter, right up there with Julian's solutions to the homework-completion problem and the knapsack problem, and the extremely elegant solution to the Totient ratio homework problem.

I feel a lot more confident not only in my ability to find solutions that work, but also rigorously prove the correctness of those solutions. Earlier, I would spend a lot more time stumbling in the dark, trying different things and occasionally happening upon a solution. Now, I'm able to step

[1] There is an interesting paper on the dynamic programming view of Djikstra's algorithm that involves a redefinition of the subproblem in a similar way.

back and really analyse a problem before I write a single line of code, and gaining that ability alone is more than I would get out of most classes.

I'm now back on the Project Euler bandwagon and hope to reach the 200 problem mark by the end of the year. I really like working on the kind of problems we have been, and I'm excited to apply and build upon every tool I've picked up in this class. Thanks so much, Professor Gehret and Julian, for a wonderful summer. I wish we had more time to go into more concepts and solve problems, but I'm happy with what I'm walking away with.