Problem Set 6, Problem 1

Kevin Peng
Collaborators: Genghis Chau, Jodie Chen, Anubhav Jain, Martin Ma
Recitation: Vladislav Kontsevoi WF3

This problem set is due **Thursday, December 5** at **11:59PM**.

This solution template should be turned in through our submission site. [1]

For written questions, full credit will be given only to correct solutions that are described clearly *and concisely*.

Please fill in the TA and recitation section you attend. Otherwise you may not be able to get your problem sets back in section!

---

[1] Register an account, if you haven't done so. Then go to Homework, Problem Set 6, and upload your files.

Problem 6-1. [35 points] **Party Bus**

You're on one of the new Party+Tour Buses, which serves as both a party bus and a tour bus. The bus makes n stops, $s_1, s_2, \ldots, s_n$. You only have time to get out at k of the stops to see tourist attractions. (You need $n - k$ time to party!) You may decide to stop at fewer stops, however. If you get out at stop s_i, you get h_i happiness points. Additionally, each time you decide to party instead of getting out, if you have just partied for $j \geq 0$ consecutive stops, then you get j additional happiness points. (Thus your happiness grows roughly quadratically with long stretches of partying.) Design and analyze an efficient dynamic programming algorithm to compute the optimal $\leq k$ stops to get out at, in $O(n^2k)$ time.

We define a subproblem based on the number of times the bus has stopped. Thus, each problem corresponds to a value t where $0 \leq t \leq k$. Our solution to a subproblem will be the maximum happiness attained while making $\leq t$ stops. We have a recursive solution for our subproblems: the solution to a subproblem with t stops is equal to the max(max(solutions to a subproblem with $t - 1$ stops - cost of breaking up a chain of parties + happiness for going on a tour) for all possible tours to stop at, solution to subproblem with $t - 1$ stops). Since we are using a greedy solution, if we do not make an extra stop (that is, if the solution to the subproblem with t stops is the exact same as the solution to the subproblem with $t - 1$ stops), we can terminate with the correct answer. This algorithm will run in $O(n^2k)$ time since there are k subproblems, n stops to consider per subproblem, and $O(n)$ stops where the bus might be partying per subproblem. If we know the locations where the bus parties for each subproblem, we can calculate the cost of breaking up a chain of parties in $O(1)$ time using a mathematical formula. Thus, this algorithm runs in $O(n^2k)$.