Avi Hirsch

HW #15: Dynamic Programming

1. If we define OPT(i) as the minimum penalty to get from the start (a_0) to a hotel i, then this can be solved by realizing that the sum of OPT(j)—for each hotel j that comes before i—and the penalty incurred for travelling from j to i—or $(200 - (a_j - a_i))^2$—will give the minimum penalty from j to i. To determine which hotel j one should stay in before staying in i, one need only select the minimum of that calculation over all hotels j before i. Simply put, OPT(i) = min(OPT(j) + (200 – (a_j – a_i))), for all j from j = 0 until j = i – 1. The base case, OPT(0), is set at 0.

   This recursive algorithm will work because in this case, optimizing the subproblems will lead to an optimal "super-problem." This is true because of a proof by contradiction: given that some function OPT(i) gives the most optimal solution for problem i. If, for a given subproblem j within i, a more optimal solution exists than OPT(j), then this would necessitate that OPT(i) is not optimal, thus contradicting our premise.

   This algorithm has an $O(n^2)$ running time, because each subproblem i takes approximately O(i) time and there are n subproblems.