HW14

Daniel Ginsburg

This algorithm is based on the unbounded backpack algorithm with some variations.

Definitions:

The following is a list of hotel values to be used as examples throughout the algorithm.

Hotel	a_1	a_2	a_3	a_4	a_5
Distance	180	220	400	540	900

Let a[i] represent an array of size n where a[i] contains the distance of a_i from the origin.
Ex. a[3] = 400.
Let K[d] represent the lowest penalty (and therefore the optimal value) of getting to hotel a_d.
Ex. K[1] = $(200 - 180)^2$ = 400.
Let dif = the difference between a[d] (the current hotel) and a[i].
Let previous[d] contain the number corresponding to the last hotel used to get to hotel d.
Ex. previous[1] = 0 (the start).

Defining subproblem/algorithm:

The problem asks for the minimum penalty to travel to hotel n, given that one may only stop at other hotels. The *subproblem* is finding the minimum penalty to travel to every hotel along the way. (Side Note: The problem states that the a_i's are in order. We start at hotel 1 and expand until hotel n. There is only one option at hotel one which is to travel directly to the hotel. Thus K[d] is 400 and previous[1] = 0. For each of the subsequent hotels, we set K[d] to infinity and then write a recursive definition

$$K[d] = \min_i \{K[d-i] + (200 - \text{dif})^2, K[d]\} \text{ for all } i = 0 \text{ to } d-1$$

Whenever K[d] is updated we set previous[d] = i. After running the algorithm on all n hotels, we run a simple algorithm to back track all the hotels. We simply add p = previous[n] to an array and then call previous[p]… until we get to 0. (We also add n itself to the list).
Ex. Thus our final trail for a_5 after running the algorithm would look like this [0,2,3,4,5]

PROOF OF CORRECTNESS:

We use a combination of Strong induction and proof by contradiction.

Base Case:

The base case is n = 1. There is only one hotel and therefore the only possible route is to go directly to a_1. This must also be the optimal route.

Inductive Step:

Assuming we have all the minimum penalties up to hotel a_d, we prove that the algorithm will find the minimum at a_d. The algorithm looks at every potential previous hotel. Because our algorithm uses the min function when comparing the various options, by definition it will choose the lowest available option amongst all the choices. The only way it would chose a suboptimal solution is if there is a better route to get to one of the previous hotels, which by definition is impossible based on the assumption of the inductive step.

Thus we prove that this algorithm is optimal for all $a_n : n > 0$.

RUN TIME:

The run time is $O(n^2)$. For each of the n hotels we run a loop over all the previous K[i], so it comes out to around $O(\frac{1}{2}n^2) = O(n^2)$. The backtracking is O(n) because we kept track along the way. In the worst case, we use every hotel in our solution and therefore have to call previous[i] n times.