Intro to Algorithms, COMP-160, Homework #10

Benjamin Tanen, 04/15/2016

2. **Let G be an $n \times n$ grid. You start at a particular position, with zero velocity. At every time step you can modify your horizontal velocity by 1, or keep it the same. The same holds independently for vertical velocity. So if at a particular time you are at position x, y and already have velocity v_x, v_y, then at the next time step you will jump and land at position $x + v_x$, $y + v_y$, after which each component of your velocity may change by ± 1 if you wish.**

   At several (known) grid positions there are pits full of dragons that you want to avoid landing on. It is ok to jump over a dragon pit. You also can't jump off the grid. The goal is to arrive at a specified target position, with zero velocity, in as little time as possible. Show how fast you can compute an optimal trajectory or decide that there is no way to reach the target.

   In order to solve this problem in an optimal way, we can create a graph that represents our movement through our $n \times n$ grid. Each vertex in this graph can represent a possible state as we move through our grid (associated with a particular x, y, v_x, and v_y).

   To more efficiently build this graph, we want to set a reasonable upper bound on our v_x and v_y. We know that we will only ever traverse a distance of n in the horizontal position or the vertical position. Let us consider our horizontal velocity v_x. Since we will only every travel a distance of n in our horizontal direction, we know that the sum of all our horizontal velocities v_x must sum to n. However, since we will start at $v_x = 0$ and we know that we must stop with $v_x = 0$, over the course of traversing our grid, our velocity will increment up from 0 to our max velocity and then decrement back down to $v_x = 0$. Thus, we can form the following:

$$\sum_{i=0}^{v_{max}} i + \sum_{i=v_{max}-1}^{0} i = \sum_{i=0}^{v_{max}} i + \sum_{i=0}^{v_{max}-1} i = O(v_{max}^2) = n \Rightarrow v_{max} = O(\sqrt{n})$$

   Therefore, we can see that, at a maximum, we can at most travel with a horizontal (or vertical) velocity of $O(\sqrt{n})$.

   Now that we have a limit for the number of velocities that we can have and thus a better limit on the number of possible states (vertices in our graph), we can begin to form our graph. We can do this by traveling to each of the n^2 positions in our grid. At each position, we can consider all the velocities ($2O(\sqrt{n}) \cdot 2O(\sqrt{n}) = O(n)$) that we can have at this position.

   Say we are considering position x and y with $v_x = i$ and $v_y = j$ for $-v_{max} \leq i, j \leq v_{max}$. If the position $(x + i, y + j)$ is another valid position (not a dragon and a position with in our $n \times n$ grid), we draw an edge from the current vertex to the the associated vertex for this next state (if this next state vertex doesn't already exist, add it to our graph with an edge from the current vertex).

Since we will have to consider at most $O(\sqrt{n}\sqrt{n}) = O(n)$ velocities for each position, we will add $O(n)$ vertices / connections for each of our $O(n^2)$ positions, resulting in $O(n^3)$ states / vertices in our graph.

From here, we want to find the ideal path from our starting position to our ending position. We can do this using BFS (which would result in the optimal path from start to end), taking $O(V+E)$ time. We already know that $V = n^3$ since we know there are n^3 states (see above). Since there are a constant number of transitions in our velocities (and thus positions) (± 1 for the vertical or horizontal velocities), we therefore know that there will be a constant number of edges out of any particular vertex. Thus, we can describe $E = \Theta(1)V = V$ and alternatively describe the cost of our BFS as $O(V+E) = O(V+V) = O(V) = O(n^3)$ time to get the optimal path.

This gives us a total computation time of $O(n^3)+O(n^3) = O(n^3)$ to solve this problem.