Chris Hanusa Math 381 – Homework 4 **Answer Key** Nov 14, 2003

Question 1 – Consider a rectangular grid, with vertices (5,5), (-5,5), (-5,-5), (5,-5). Inside the grid there is a column, situated at (1,1), (0,1), (0,-1), (-1,-1). We play car-race on this drawing. We start at (5,5), we have to go around the column (points on the side of the large rectangle and the 'column' can be used, but we can not go across the column) and finish, where we've started, at (5,5). Our goal is to finish in the least possible steps, without hitting the 'walls'. The steps are defined recursively: v_0 is the 0-vector, and P_0 is (5,5). For $i > 0$, $P_i = P_(i-1) + v_i$, where $v_i = v_{i-1} + u_i$, where u_i is any of the 8 vectors whose every coordinate is +1,-1,0 (but u_i can not be the zero vector). (a) Describe a possible route from start to finish. (b) Give a lower estimate and an upper estimate on the length of possible routes. (c) Write up a mathematical model for this game, and describe the algorithm which will provide an optimal strategy for this game.

Answer – (a) The shortest possible path is of length 9. One of these shortest such paths is from (5,5) → (4,5) → (2,4) → (0,2) → (-1,0) → (-1,-1) → (0,-1) → (2,0) → (4,2) → (5,5).

(b) If you could change speed instantaneously, you would need to turn at least twice to get around the column. This means a lower bound on the number of moves is 3. An upper bound on a reasonable number of moves for this game is if you go one step at a time, speeding up in one direction, and slowing down to come to a stop in the next move. If you do this, you may take upwards of $2(6 \times 6) = 72$ moves. Any reasonable explanation for a minimum or maximum number of moves was accepted, as long as 9 was in your range.

(c) This game can be modeled either by a decision tree or a graph. The graph is not exactly the same graph as the graph provided because at each vertex on the graph you need to be able to take into account the velocity of the car at that point. So the easiest model for me to explain is a decision tree.

We model the game as a decision tree as follows: Start at a root of the tree, and the first branch of the tree is to choose one of the three points on the starting line. Then from each of those points, branch out 8 times depending on the choice of u_1. Continue in this fashion, branching out 8 times at each step depending on the choice of u_i.

To take into account the constraints of the game, we discard any branch of the tree which at any point leaves the boundaries of the graph. We also discard any branch that goes the wrong way across the finish line, to avoid any cheaters.

The optimal solution is the path that takes the shortest number of steps to reach the finish line, so an optimal strategy would be to search the first level of the tree to see if you have reached or passed the finish line. If not, search the second level of the tree. Continue on in this fashion and you are assured to stop when you reach a selection of $\{u_i\}$'s that allow you to arrive at the finish line in the smallest number of moves, giving you an optimal solution. (This is the idea behind breadth-first search.)

If you are interested in hearing how to model this as a graph, I would be happy to explain it to you.

Question 2 – Consider the following cities: Atlanta, Boston, Chicago, Denver, Houston, New York, Minneapolis, Salt Lake City, San Francisco, Seattle. Write up the modified adjacency matrix of these cities: the entries in the matrix are the distances of the cities. How many TSP tours are in this graph, starting and ending in Seattle? Find a minimum cost(=length) spanning tree in this graph with your favorite algorithm. Use this spanning tree for finding a not too bad (not too long) tour for the TSP, explain how good is your result, how far it is from the optimum tour.

Answer [10 points] – The adjacency matrix is a 10×10 matrix with the (actual) distances between the cities as its entries.

Since there are 10 cities, there are $(10-1)! = 9!$ ways TSP tours starting and ending in Seattle.

Using [your favorite] Algorithm, the spanning tree can be most easily represented as a graph: from Seattle to San Francisco to Salt Lake City to Denver to Minneapolis to Chicago to New York to Boston, with a second branch hanging from Chicago to Atlanta to Houston. This tree is 5956 miles long.

From this tree, we arrive at a route that traces the tree starting at Seattle and visiting each city more than once. Then we can get a "not too bad" spanning tree by taking shortcuts *since this is a Euclidean TSP*. For example, this path could be SEA $\to$ SLC $\to$ DEN $\to$ MSP $\to$ CHI $\to$ BOS $\to$ NYC $\to$ ATL $\to$ HOU $\to$ SFC $\to$ SEA.

Since the path is shorter than two times the minimal spanning tree, it must be less than two times the optimal path, which is very good.

Question 3 – Let G be a bipartite graph, and every degree is equal to d. Show that the edges of G can be split up into d disjoint perfect matchings.

Answer [10 points] – We will use induction to prove this claim.

Base Case Given the bipartite graph G, if $d = 1$, then we have each vertex attached to only one other vertex. This is already a perfect matching.

Induction Step Given our bipartite graph $G = (V, E)$ where every vertex is of degree $d + 1$, we want to apply the **Marriage Theorem**. So we must check if its assumptions are satisfied. [`Check these yourself. If you want help, ask me`] Therefore there exists a matching on G with edge set $M \subset E$. Consider the graph $G' = (V, E \setminus M)$. It is a bipartite graph and every vertex is of degree d. By the induction hypothesis, G' can be split up into d disjoint matchings $M_1, \ldots, M_d$. Adding in our matching M, we see that M is disjoint from all other matchings, and therefore G can be split up into $d + 1$ disjoint matchings $M, M_1, \ldots, M_d$. This proves the claim. •