Brian DeFlaminio HW2 Intro to AI 11/8/23

1. Chapter 3, Exercise 3.3 Suppose two friends live in different cities on a map, such as the Romania map shown in Figure 3.2. On every turn, we can simultaneously move each friend to a neighboring city on the map. The amount of time needed to move from city i to neighbor j is equal to the road distance d(i, j) between the cities, but on each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin. We want the two friends to meet as quickly as possible.
2. **Write a detailed formulation for this search problem. (You will find it helpful to define some formal notation here.)**

*State Space: Every city-neighbor pair (i, j) as defined above.*

*Successor function: All pairs (i1, j1) that are adjacent to (istart, jstart). We will call the function Adjacent(i1, istart) and Adjacent(j1, jstart). In cases where there are 3 directions to go let’s assume we drop the direction we came from or the largest d(i, j) one if just starting. In the case of one neighbor just go that way.*

*Goal: For both friends to be at the same coordinate (iend, jend)*

*Step cost function: distance from (istart, jstart) to (iend, jend) is measured by:*

*max[d(i1, istart), d(j1, jstart)] + max[d(i2, i1), d(j2, j1)] + … + max[d(iend, iend-1), d(jend, jend-1)]*

**b. Let D(i, j) be the straight-line distance between cities i and j. Which of the following heuristic functions are admissible?**

**(i) D(i, j);** *Admissible, it’s optimistic and is less than or equal to the true cost.*

**(ii) 2 · D(i, j);** *Not this one, it overestimates the distance by a factor of 2*

**(iii) D(i, j)/2** *Admissible, it’s optimistic and is less than the true cost by a factor of 0.5*

**c. Are there completely connected maps for which no solution exists?**

*There’s at least one, the map of two locations with one line between them. It’s got no solution because the two people would never meet at the same station because they both have to move every turn and there’s one place to go.*

**d. Are there maps in which all solutions require one friend to visit the same city twice?**

*If you had a map with 2 nodes with one route but one of the nodes has a loop that costs less than the one that changes stations the one person takes the route that brings them back to their starting station and the other can only take the one route to the other station then one friend is required to revisit the city and it has a solution.*

*Kind of like this* A drawing of a person with a arrow pointing to the side

Description automatically generated

1. Chapter 3, Exercise 3.16 A basic wooden railway set contains the pieces shown in Figure 3.32. The task is to connect these pieces into a railway that has no overlapping tracks and no loose ends where a train could run off onto the floor.
2. **Suppose that the pieces fit together exactly with no slack. Give a precise formulation of the task as a search problem.**

*Initial State: Any track piece*

*Successor Function: Attach a subsequent track piece from remaining options, track piece can be placed in any orientation/direction to facilitate the complete loop desired.*

*Goal Test: All track pieces used, complete loop, no tracks that lead nowhere.*

*Step Cost: -1 piece from the total remaining(?)*

1. **Identify a suitable uninformed search algorithm for this task and explain your choice.**

*Since we have to use every piece of track in order to judge the completeness of a solution prioritizing depth is the best route. So a depth first search is a viable choice for our uninformed search algorithm.*

1. **Explain why removing any one of the “fork” pieces makes the problem unsolvable.**

*In order to fork back into one track we need an even number of “fork” pieces. With just the one we use it somewhere and now have 3 tracks to split into even groups of 2 for connection purposes.*

1. **Give an upper bound on the total size of the state space defined by your formulation. (Hint: think about the maximum branching factor for the construction process and the maximum depth, ignoring the problem of overlapping pieces and loose ends. Begin by pretending that every piece is unique.)**

*32 pieces in total, all unique and in any order. 32!/(12! \* 16! \* 2! \* 2!) = 6.5637979e+12*

1. Chapter 3, Exercise 3.26 Consider the unbounded version of the regular 2D grid shown in Figure 3.9. The start state is at the origin, (0,0), and the goal state is at (x, y).
2. **What is the branching factor b in this state space?**

*4*

1. **How many distinct states are there at depth k (for k > 0)?**

*4k for k>0. Since there are 4 states branching from every state*

1. **What is the maximum number of nodes expanded by breadth-first tree search?**

*In order to find the number of nodes expanded in the x direction we use*

*Same thing for y direction, we add the two and simplify =…=*

1. **What is the maximum number of nodes expanded by breadth-first graph search?**

*2i nodes to be expanded each level of grid with dimensions x \* y at level i. (x direction plus y direction i+i). To get total nodes at each level from 1 to x+y we use a summation which can also be represented by the formula . Substituting our max i value with n and factoring out the 2 we get = total nodes expanded*

1. **Is h = |u − x| + |v − y| an admissible heuristic for a state at (u, v)? Explain**.

*Yes. This heuristic represents the Manhattan distance between current state (u, v) and goal state (x, y). This Manhattan heuristic is considered a valid heuristic because it calculates the minimum number of moves required to reach the goal state, satisfying the optimistic estimate requirement as well as consistency with its successors.*

1. **How many nodes are expanded by A∗ graph search using h?**

*x+y*

1. **Does h remain admissible if some links are removed?**

*Yes, removing links generally makes navigation harder which means that the heuristic will underestimate which is acceptable.*

1. **Does h remain admissible if some links are added between nonadjacent states?**

*No, the opposite happens here, navigation would become easier, and our previous heuristic starts to overestimate which is not acceptable.*

1. Chapter 4, Exercise 4.3

4.3 In this exercise, we explore the use of local search methods to solve TSPs of the type

defined in Exercise 3.30. (The MST cost of a set of cities is the smallest sum of the link

costs of any tree that connects all the cities.)

1. **Implement and test a hill-climbing method to solve TSPs.**

*(see attached file for python code)*

*Tests yielded expected results, given specific inputs there are instances where the MST weaknesses can be exploited, stuck at local maxima, not helpful on specific inputs of things like linear cities where the heuristic just selects the same edges every time, limiting routes discovered.*

**b. Repeat part (a) using a genetic algorithm instead of hill climbing. You may want to consult Larra ̃naga et al. (1999) for some suggestions for representations.**

*(also in the python code file)*

*Given some inputs we see that the genetic algorithm actually chooses a different path than our MST cost method but arrives at the same cost since I did purposefully choose this input matrix to demonstrate differences in decisions.*

*OUTPUT:*

*Genetic Algorithm - Optimal Tour: [1, 3, 0, 2]*

*Genetic Algorithm - Optimal Cost: 14*

*Hill Climbing - Optimal Tour: [0, 1, 2, 3]*

*Hill Climbing - Optimal Cost: 14*

*INPUT:*

*distances = [*

*[0, 1, 2, 3],*

*[1, 0, 4, 5],*

*[2, 4, 0, 6],*

*[3, 5, 6, 0]*

*]*

*-The distances between cities 1 and 2, 1 and 3, 2 and 3, and 1 and 4 are all equal, creating multiple optimal paths, as seen by both optimal paths being 14*

*-The distances between cities 2 and 4, and 3 and 4 are also equal, adding complexity to the problem.*

*-The layout is somewhat symmetrical, making it difficult for algorithms to distinguish between equivalent solutions.*