Grundlagen der Wissensverarbeitung
Blatt 4

Daniel Speck, Lena Niermeyer

07.11.2015

Exercise 1.2 (heuristic search)

1.

We chose to take

$$h(node) = \text{abs}(\text{node}_x - \text{goal}_x) + \text{abs}(\text{node}_y - \text{goal}_y)$$

as our heuristic function. Basically this returns the sum of the delta of rows plus the delta of columns between the current node and the goal node. This always equals the minimal amount of edges on a 2D chess-like environment with the only movement options of "up", "down", "left", "right". Therefore this heuristic is *admissible.*

2.

Yes, we can ensure termination. Our algorithm simply expands and visits every single node which is reachable in the case that the goal node is not reachable. After expanding every reachable node, the open set is empty and no more nodes are available for expanding so the algorithm ends its while-loop. This behavior ensures termination even when the goal node is not reachable.

3.

We have implemented portals in our file parser and our algorithm recognizes them. This behavior can be seen e.g. in "test_env_7.txt". Our implementation simply adds an edge with 0 costs between the portal nodes.
However, we have not implemented a new heuristic function and therefore in "blatt4_environment_b.txt" the algorithm does not find the optimal solution. For doing so, the heuristic would have to update not only the heuristic costs of the portal nodes but the heuristic costs of every node between the start and portal 2. Otherwise the portal node 2 in the upper left would never be visited, because the nodes in between would be too expansive.

4.

Since the complexity of an A* search depends on the heuristic function we are not really sure what our complexity is. Considering that the calculation of the delta in the equation never changes and only calculates a sum of two

subtractions, the equation itself should have the time complexity $O(1)$, because the input size of the problem is irrelevant. Calculating this for every node in the frontier should end up in $O(n)$, because we never expand a node twice.
As we used the library *networkx* this time, we are not sure about the space complexity, either (depending on networkx's implementation). Our own Graph class from exercise 3 has something like $O(2 * (n + e)) = O(n + e)$ since we are storing each node and edge in a dictionary and additionally in a representation array containing the label of each node and edge.