ECS 170 Project 1 Part 1

Hardy Jones
999397426
Professor Davidson
Winter 2014

1. For the cost function

```
// New height divided by old height
public double getCost(final Point p1, final Point p2)
{
    return (getTile(p2) / (getTile(p1) + 1));
}
```

We create an admissible heuristic by relaxing the rule that you can only move to an adjacent tile.

This is admissible, because it is akin to having every tile exactly next to the goal state. The value generated by this heuristic will always be less than or equal to the actual cost to get to the goal state because any actual path to the goal state might include additional nodes which add to the actual cost of reaching the goal state.

2. For the cost function

```
// Exponential of the height difference
public double getCost(final Point p1, final Point p2)
{
    return Math.exp(getTile(p2) - getTile(p1));
}
```

We create an admissible heuristic by looking at three cases.

Case 1 The current node is at the same height as the goal node.

In this case, we just default to the Chebyshev distance, as this is the least cost we could have, meaning, any path from p1 to the goal that isn't a flat path at the same height will have a greater cost than the Chebyshev distance.

Case 2 The current node is above the goal node.

In this case, we drop immediately to the same height as the goal node for one space, and continue with the Chebyshev distance to the node. This is admissible because any other path that doesn't maximize the drop from the current height to the goal node height will have a greater actual cost to the goal node.

Case 2 The current node is below the goal node.

In this case, we gradually make our way up to the goal node height by averaging out each increment over the Chebyshev distance to the goal. This is admissible because any other path would take a greater cost at some step along the way, which overall contributes to a greater actual path cost.