Assignment 1 Question 1

Dylan M Ang

Modified January 24, 2022

1 Exponential

```python
def h(self, map_, start):
    # get goal
    goal = map_.goal
    # get distances
    y_dist = abs(goal.y - start.y)
    x_dist = abs(goal.x - start.x)
    # get height difference
    goal_height = map_.getTile(goal.x, goal.y)
    start_height = map_.getTile(start.x, start.y)
    delta_h = goal_height - start_height
    # min steps
    s = min(x_dist, y_dist) + abs(y_dist - x_dist)
    if goal_height > start_height: # going up
        # h = l * s
        # 2^l * h/l = total cost
        m_diff = 1/math.log(2)
        m_cost = 2 ** m_diff
        return (delta_h / m_diff) * m_cost
    elif goal_height < start_height: # going down
        return s * 2**((delta_h)/s) if s != 0 else 0
    else: # same level
        return s
```

There are three cases,

1.1 The goal state is lower than the start.

- $h = \begin{cases} s * 2^{\Delta_h/s} & s \neq 0 \\ 0 & s = 0 \end{cases}$

- Where s is the minimum number of steps to reach the goal state and Δ_h is the difference in height of the goal state and the start state.
- The ideal path to any lower goal state is one that is consistenly decreasing since each step costs between 0 and 1, but going flat or up at any time adds a cost greater than or equal to 1.
- To get a path that is always decreasing, we take the total difference in height and divide by the minimum number of steps to get the average decrease in height, Δ_h for an optimal path.
- Then, we take the exponential cost of a Δ_h change in height, and multiply by s for the total cost.
- The general idea is to return the cost of an ideal path, which will be the lowest possible path cost. It isn't possible to get a lower path cost, therefore this heuristic is admissible for this case.

1.2 The goal state is on the same level than the start.

- $h = s$ where s is the minimum number of steps required to reach the goal.
- The ideal path from a start state to a goal state at the same tile height is a flat path with no valleys or hills.
- Even if we find a path downwards (which costs less than going flat), it isn't worth it to take that path because you will eventually need to go back up and that will cost more than it saves. For example, descending one unit will cost 0.5, saving 0.5, but going back up one unit will cost an extra 1 unit.

1.3 The goal state is higher than the start.

- $h = \frac{\Delta_h}{\delta_h} * 2^{\delta_h}$ where $\delta_h = \frac{1}{ln(2)}$
- δ_h is the average amount to go up by each step and Δ_h is the total difference in height.
- This heuristic estimate uses a similar method to 1.1. The optimal path for going upwards is to go up by a consistent amount each step. However, unlike 1.1, the minimum number of steps doesn't necessarily produce the optimal path.

- $\Delta_h = \delta_h * s$.
- Total Cost $= 2^{\delta_h} * \frac{\Delta_h}{\delta_h}$ There must exist a value in which it becomes worth it to take extra steps to avoid a higher upward path.
- The number of steps in a perfect path, s, depends on the amount by which you go up each step, δ_h. But, δ_h depends on s.
- However, we want the minimum value of δ_h, which we can get by taking the derivative of the Total cost function and finding the roots.
- Doing this allows us to find that $\delta_h = \frac{1}{ln(2)}$.
- In other words, the most efficient cost path is the one which traverses up by $\frac{1}{ln(2)}$ units of height every step.
- This is admissible since it estimates the cost for an perfectly optimal path. No path which ends above the start height can follow a more efficient path, therefore the heuristic is admissible.

2 Division

```
def h(self, map_, start):
    # get goal
    goal = map_.goal
    # get distances
    y_dist = abs(goal.y - start.y)
    x_dist = abs(goal.x - start.x)
    # min steps
    s = min(x_dist, y_dist) + abs(y_dist - x_dist)
    return s
```

The heuristic I use for division is not admissible, because it just counts 1 for every step, no matter the change in elevation. I included it because I simply couldn't find a single admissible heuristic for the division cost function and this one at least came up with the correct answer when used with one of the seed maps.

Unlike with the exponential cost function, the division cost function doesn't increase consistently. We can't say that what a perfectly optimal path would look like.