Assignment 3 A*
By Brandon Young and Ruicheng Wu

c. Optimizations

To optimize the search algorithms, we used a Python dictionary as a heap, rather than a Python list (array), to implement the closed list more efficiently. Since we just needed to insert or check if a pair of coordinates existed inside the close list, a heap would speed up the check and may help the insertion. Overall, checking if a element exists takes O(n) for an array and O(1) for a heap.

For the heap, we initially used the sum of coordinate values to use as the key. However, this meant for every (x, y) there is a matching pair (y, x) with the same key. In addition, as the sums increased, there were a greater number of pairs that summed to the same value. As a result, we decided to distinguish (x, y) by multiplying x with a hash code value. With some testing, higher hash codes lowered the average size of a list for each key so we ended up using 91 as the hash code.

We also optimized the way the heuristic was applied to the grid. At first we applied the heuristic over the entire graph before running the search algorithm. However, this wasted resources when only a portion of the overall grid was searched. Instead, we compute the heuristic during the search, to ensure only relevant cells in the grid have the heuristic applied to them.

---

d. Heuristics
1. Diagonal Distance Heuristic

The formula for cell $(x1, y1)$ and goal $(x2, y2)$ is:

$$h((x1, y1)) = dx + dy(\sqrt{2} - 2) * min(dx, dy)$$

where $dx = |x2 - x1|$ and $dy = |y2 - y1|$.

The best admissible/consistent heuristic we used was the diagonal distance heuristic. The diagonal distance heuristic is similar to the Manhattan distance heuristic, except it every cell in the grid is an unblocked cell, then computes the shortest path using diagonal, horizontal and vertical steps it takes to move from the start to the goal.

Diagonal distance was preferred over Euclidean distance because the movements it allows is close to the movements allowed on the actual grid.

Therefore, the heuristic value will come closer to the actual optimal distance than the value from Euclidean distance.

2. Euclidean Distance

Euclidean distance is also a admissible/consistent heuristic as it works on a subproblem of the grid problem where any degree of movement is allowed.

The formula for cell $(x1, y1)$ and goal $(x2, y2)$ is:

$$h((x1, y1)) = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

3. Manhattan Distance

This heuristic assumes every cell in the grid is an unblocked cell, then computes the shortest path using only horizontal and vertical steps.

The formula for cell $(x1, y1)$ and goal $(x2, y2)$ is:

$$h((x1, y1)) = dx + dy$$

where $dx = |x2 - x1|$ and $dy = |y2 - y1|$.

The heuristic is inadmissible for the grid problem because diagonal movements are allowed. As a result, Manhattan distance tends to overestimate the actual cost to the goal. Nevertheless, this metric can come in handy for sequential A* search because it overestimates.

4. Euclidean Squared