Project 1: Solving Mazes with A*

Paul Beggs

September 15, 2025

This heuristic calculates the Manhattan distance between the current location of the `MazeExplorer` and the end location of the maze. Since n_1 and n_2 are the coordinates of the current location and e_1 and e_2 are the coordinates of the end location, the Manhattan distance is calculated as $|n_1 - e_1| + |n_2 - e_2|$. This
This is the start to my paper, and here is the link to the repository.

```
1 public class Manhattan implements ToIntFunction<MazeExplorer> {
2     @Override
3     public int applyAsInt(MazeExplorer value) {
4         return value.getLocation().getManhattanDist(value.getM().getEnd());
5     }
6 }
```

Figure 1: Manhattan-Only Heuristic

```java
public class CombinedDist implements ToIntFunction<MazeExplorer> {
    @Override
    public int applyAsInt(MazeExplorer value) {
        Pos currentLocation = value.getLocation();
        Pos endLocation = value.getM().getEnd();

        Set<Pos> allTreasures = value.getAllTreasureFromMaze();
        Set<Pos> foundTreasures = value.getAllTreasureFound();

        Set<Pos> remainingTreasures = new HashSet<>(allTreasures);
        remainingTreasures.removeAll(foundTreasures);

        if (remainingTreasures.isEmpty()) {
            return currentLocation.getManhattanDist(endLocation);
        }

        int minDistFromCurrent = Integer.MAX_VALUE;
        int minDistToEnd = Integer.MAX_VALUE;

        for (Pos treasure : remainingTreasures) {
            minDistFromCurrent = Math.min(minDistFromCurrent,
            ↪  currentLocation.getManhattanDist(treasure));
            minDistToEnd = Math.min(minDistToEnd,
            ↪  treasure.getManhattanDist(endLocation));
        }
        return minDistFromCurrent + minDistToEnd;
    }
}
```

Figure 2: Combined Distance Heuristic

```java
public class MaxDist implements ToIntFunction<MazeExplorer> {
    @Override
    public int applyAsInt(MazeExplorer value) {
        Pos currentLocation = value.getLocation();

        Set<Pos> allTreasures = value.getAllTreasureFromMaze();
        Set<Pos> foundTreasures = value.getAllTreasureFound();

        Set<Pos> remainingTreasures = new HashSet<>(allTreasures);
        remainingTreasures.removeAll(foundTreasures);

        int maxDist =
        ↪  currentLocation.getManhattanDist(value.getM().getEnd());

        for (Pos treasure : remainingTreasures) {
            int distToTreasure = currentLocation.getManhattanDist(treasure);
            if (distToTreasure > maxDist) {
                maxDist = distToTreasure;
            }
        }
        return maxDist;
    }
}
```

Figure 3: Max Distance Heuristic

```java
public class OverlyConfident implements ToIntFunction<MazeExplorer> {
    @Override
    public int applyAsInt(MazeExplorer value) {
        int manhattanDistance =
        ↪  value.getLocation().getManhattanDist(value.getM().getEnd());
        return manhattanDistance * 3;
    }
}

```

Figure 4: Overly Confident Heuristic