Phuc Le
Phuoc Le
Matthew Maksim
9/16/2022

Homework 1 Writeup

For our code we have implemented the manhattan distance and the 2D euclidean distance as the 2 extra heuristic calculations.

For our experiment, our team will compare the two new heuristic calculations.

For each heuristic function, we will show our code implementation and the solution path that each function outputs for three sample maps.

Manhattan Distance Heuristic experiments:

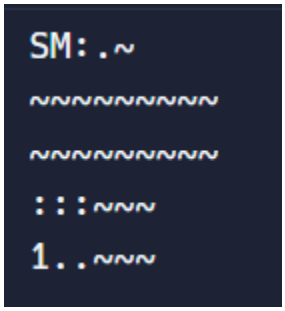
The manhattan distance is calculated by using the change in x, y and z coordinates of the current and goal node. The calculation for this would look like this:
Manhattan distance = $(X - x) + (Y - y) + (Z - z)$

This is our implementation of it in code:

```
elif sys.argv[2] == "1": # the 3D manhattan distance
            return abs((explorerPosition[0] - summitPosition[0])
                + (explorerPosition[1] - summitPosition[1])
                + (int(explorerPosition[2]) - summitPosition[2])
            )
```

We expect this heuristic calculation to negatively affect the time it takes to find the goal. We think this is so because it is less precise than the 3D euclidean distance. This might make a difference on the path the explorer takes to get to the summit. We will test this heuristic with the following text files:

Test 1: text.txt (initial state)

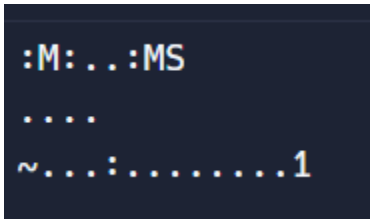


Runtime Output text.txt:

```
M_0
SM:.~
~~~~~~~~~
~~~~~~~~~
:::~~~
1..~~~
```

For this test with map text.txt, our code failed to find a path using the manhattan distance formula. It outputs the initial map, but then gets stuck in an infinite loop.

Test 2: text2.txt (initial state)



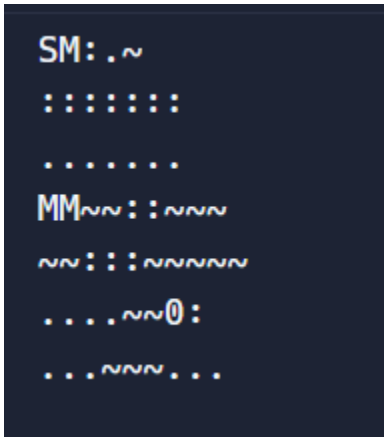
Runtime Output text2.txt:

```
M_0
:M:..:MS
....
~...:.........1

M_1
:M:..:MS
....
~...:........1.

M_2
:M:..:MS
....
~...:.......1..

M_3
:M:..:MS
....
~...:......1...

M_4
:M:..:MS
....
~...:.....1....

M_5
:M:..:MS
....
~...:....1.....
```

```
M_6
:M:..:MS
....
~...:..1......

M_7
:M:..:MS
....
~....:.1.......

M_8
:M:..:MS
....
~...:1........

M_9
:M:..:MS
....
~...2.........

M_10
:M:..:MS
...1
~...:.........

M_11
:M:.1:MS
....
~...:.........

M_12
:M:..2MS
....
~...:.........

M_13
:M:..:3S
....
~...:.........

M_14
:M:..:M4
....
```

~...:.....

Test 3: text3.txt (initial state)



Runtime Output text3.txt:

```
M_0
SM:.~
~~~~~~~~~
~~~~~~~~~
:::~~~
1..~~~
```

The manhattan heuristic was again unsuccessful in reaching the goal state for text3.txt. Our code failed to find a path using the manhattan distance formula. It outputs the initial map, but then gets stuck in an infinite loop.

Euclidean 2D Distance Heuristic experiments:

Euclidean 2D distance = $\sqrt{(X - x)^2 + (Y - y)^2}$

```
elif sys.argv[2] == "2": # the 2D (x and y only) euclidean distance
            return math.sqrt(
                (explorerPosition[0] - summitPosition[0]) ** 2
                + (explorerPosition[1] - summitPosition[1]) ** 2
            )
```

The Euclidean distance we expect to behave similarly as the 3D Euclidean Distance behaved in the first heuristic. The square root of the change in the x and y coordinates of the current node and the goal node is calculated to determine the optimal route through the map. We tested this heuristic with the following text file:

Test 1: text.txt (initial state)

```
SM:.~
~~~~~~~~~~
~~~~~~~~~~
:::~~~
1..~~~
```

Runtime Output of text.txt:

```
M_0
SM:.~
~~~~~~~~~
~~~~~~~~~
:::~~~
1..~~~
```

```
M_1
SM:.~
~~~~~~~~~
~~~~~~~~~
:2:~~~
...~~~

M_2
SM:.~
~~~~~~~~~
~~~~~~~~~
:::~~~
..1~~~

M_3
SM:.~
~~~~~~~~~
~~~~~~~~~
:::0~~
...~~~

M_4
SM:.~
~~~~~~~~~
~~0~~~~~~
:::~~~
...~~~

M_5
SM:.~
~~0~~~~~~
~~~~~~~~~
:::~~~
...~~~

M_6
SM:1~
~~~~~~~~~
~~~~~~~~~
:::~~~
...~~~
```

```
M_7
SM2.~
~~~~~~~~~
~~~~~~~~~
:::~~~
...~~~

M_8
S3:.~
~~~~~~~~~
~~~~~~~~~
:::~~~
...~~~

M_9
4M:.~
~~~~~~~~~
~~~~~~~~~
:::~~~
...~~~
```

Test 2: text2.txt(initial state)

```
:M:..:MS
....
~...:........1
```

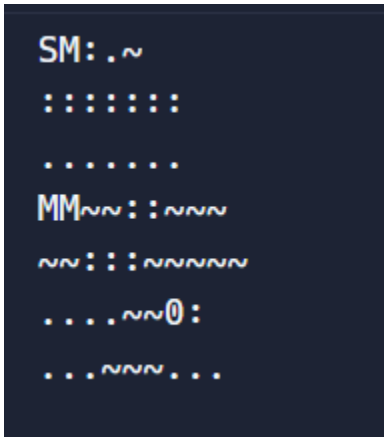
Runtime Output of text2.txt:

```
M_0
:M:..:MS
....
~...:........1
```

```
M_1
:M:..:MS
....
~...:.....1.

M_2
:M:..:MS
....
~...:.....1..

M_3
:M:..:MS
....
~...:.....1...

M_4
:M:..:MS
....
~...:.....1....

M_5
:M:..:MS
....
~...:....1.....

M_6
:M:..:MS
....
~...:..1......

M_7
:M:..:MS
....
~...:.1.......

M_8
:M:..:MS
....
~...:1........

M_9
:M:..:MS
....
~...2.........

M_10
:M:..:MS
...1
~...:.........
```

```
M_11
:M:.1:MS
....
~...:.....

M_12
:M:..2MS
....
~...:.....

M_13
:M:..:3S
....
~...:.....

M_14
:M:..:M4
....
~...:.....
```

Test 3: text3.txt(initial state)



Runtime Output text3.txt:

```
M_0
SM:.~
:::::::
.......
MM~~::~~~
```

```
~~:::~~~~~
....~~0:
...~~~...

M_1
SM:.~
:::::::
.......
MM~~::~~~
~~:::0~~~~
....~~~:
...~~~...

M_2
SM:.~
:::::::
.......
MM~~::0~~
~~:::~~~~~
....~~~:
...~~~...

M_3
SM:.~
:::::::
.....1.
MM~~::~~~
~~:::~~~~~
....~~~:
...~~~...

M_4
SM:.~
::::2::
.......
MM~~::~~~
~~:::~~~~~
....~~~:
...~~~...

M_5
SM:1~
:::::::
```

```
.......
MM~~::~~~
~~:::~~~~~
....~~~:
...~~~...

M_6
SM2.~
:::::::
.......
MM~~::~~~
~~:::~~~~~
....~~~:
...~~~...

M_7
S3:.~
:::::::
.......
MM~~::~~~
~~:::~~~~~
....~~~:
...~~~...

M_8
4M:.~
:::::::
.......
MM~~::~~~
~~:::~~~~~
....~~~:
...~~~...
```

Test 3 was successful at reaching the goal state. The heuristic works as expected and selected a similar path as the euclidean 3D distance heuristic.

Result and conclusion:

In the experiment with the Manhattan heuristic function, we have found that it did not behave as we had expected it to. For the experiment we had expected the Manhattan function to be slower than the euclidean function because it is less precise. For map text.txt and text3.txt, we have found that the program was not able to find a path from the start to the goal state. We don't suspect our implementation of the distance formula to be causing the error, but it might have been the way we implemented the priority queue of the frontier list to break ties in the distance. We think that since the Manhattan distance will give a smaller integer distance, and being that we had used the path distance of the state as the priority order of the frontier list, the program can have a hard time breaking the tie in distance of 2 states.

For the experiment with the 2D euclidean function, it performed as we had expected it to perform. It took a similar path as when the 3D euclidean function was used with only 1 or 2 steps being different. We think this is due to the fact that the 2D euclidean function did not include the height value. This caused the calculated heuristic to have a slight difference so it might move up instead of up and to the left.