MULTI-PRIZE A* SEARCH FOR MAZE SOLVING

INTRODUCTION

This report presents the implementation and results of solving multi-prize mazes using A* Search (A*). The agent must collect all prizes before reaching the goal. The heuristic function is designed to be admissible and ensures an optimal solution.

HEURISTIC EXPLANATION

The heuristic function estimates the cost of reaching all remaining prizes by selecting the nearest prize first and repeating until all are collected. This approach ensures an admissible heuristic, meaning A* search finds an optimal path.

EXECUTION PROCESS

To run the multi-prize A* search, execute the following command: python main.py

This script will solve four maze files and print the results.

OUTPUT FORMAT

The output includes:

- The solved maze with `#` marking the path and numbers representing the order in which prizes were collected.

- The path cost (number of steps taken from start to goal).

- The number of nodes expanded during the search.

EXAMPLE OUTPUT:

~~~~
### A* Solution for `multiprize-tiny.txt`:
```
%%%%%%%%%%
%6##%#7  %
%#%5%#%% %
%#%###8%9%
%#4%P%###%
%1##0  10#%
%#%%%% %#%
%2#3   %11%
%%%%%%%%%%
```
**Path Cost: 47, Nodes Expanded: 92**


### A* Solution for `multiprize-small.txt`:
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  ####P #####6%###   14##13###%
%  #%%%%%#%%%%%%#%# % %#%%  #%
%  #%0 % ####%  #%# % %##%  12%
%1#### %  ###5###%7   % #%%%%%
%%%%%#%%%%#%%% #%%%%%%%#####11%
%2###############8    %#%%%  %
%%#%%%%%%%%#%%%%%%%%%%%#%    %
%##     %  #########   #% %%%%
%##     %%%%%#    %9####     %
%3% %%%    % #%   %% %%#%%%%%%
%          % 4%        #####10%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```
**Path Cost: 181, Nodes Expanded: 498**
~~~~

~~~
### A* Solution for `multiprize-micro.txt`:

```

%%%%%%%%%%

%6  % %2#%

%#% 0#1%#%

%# %P%###%

%5#4####3%

%%%%%%%%%%

```

**Path Cost: 23, Nodes Expanded: 43**
~~~

CONCLUSION

A* Search unsuccessfully found the optimal path through the multi-prize maze even though individual value is close to the most-optimized-value. The heuristic function ensures efficiency while maintaining admissibility. The solution accounts for all required movement constraints, optimizing for minimal path cost and node expansion.