CS440 - Artificial Intelligence
Assignment 1 (3 Credit Hours)

Jon Reynolds (jdrynld2)
Matthew Krikorian (krikorn2)
Patrick McMahon (pfmcmah2)

October 2, 2017

Implementation Described

State Representation

The states in our project are determined by the following attributes:

- x, y position in maze
- number of dots collected thus far
- which dots were collected

In order to optimize performance, we represent which dots have been collected as an unsigned 32-bit integer, where the i-th bit represents whether the i-th dot has been collected or not. We can use bitwise operators to create these numbers quickly.

In order to allow fast lookup of neighbors, we store all maze states visited in a 3D array of hashmaps. The first argument to the array is x-coordinate, the second is y-coordinate, the third is the number of dots collected, and the key to the hashmap is the unsigned integer representing which dots have been collected thus far. This has the benefit of storing **only** the combinations of dots that have been collected thus far, instead of allocating space to account for all $2^{numberOfDots}$ combinations for every possible position and number of dots collected.

Frontier Representation

Our frontier is a doubly linked list implemented as a deque, or *double-ended queue*, and synchronized with a min-priority queue. Double-ended means push and pop operations work on both the front and back of the frontier, allowing it to act as a stack or a queue. Moreover, since the frontier is backed by a min-priority queue, the frontier node with the lowest cost can be found and removed in $O(\lg n)$ time.

It follows that we can use the following functionalities of the frontier for the four different search methods:

- DFS - uses frontier as a stack
- BFS - uses frontier as a queue
- Greedy - uses frontier as a min-priority queue on heuristic cost
- A* - uses frontier as a min-priority queue on heuristic cost + path cost

Section 1.1, Basic Pathfinding

Medium Maze Search Outputs

Below are the outputs we generated for the big maze using the four different search algorithms.

DFS

```
Starting DFS search on ./mazes/1-1-medium-maze.txt
Goal size: 1
443 nodes explored during search.
272 steps to reach goal.
```

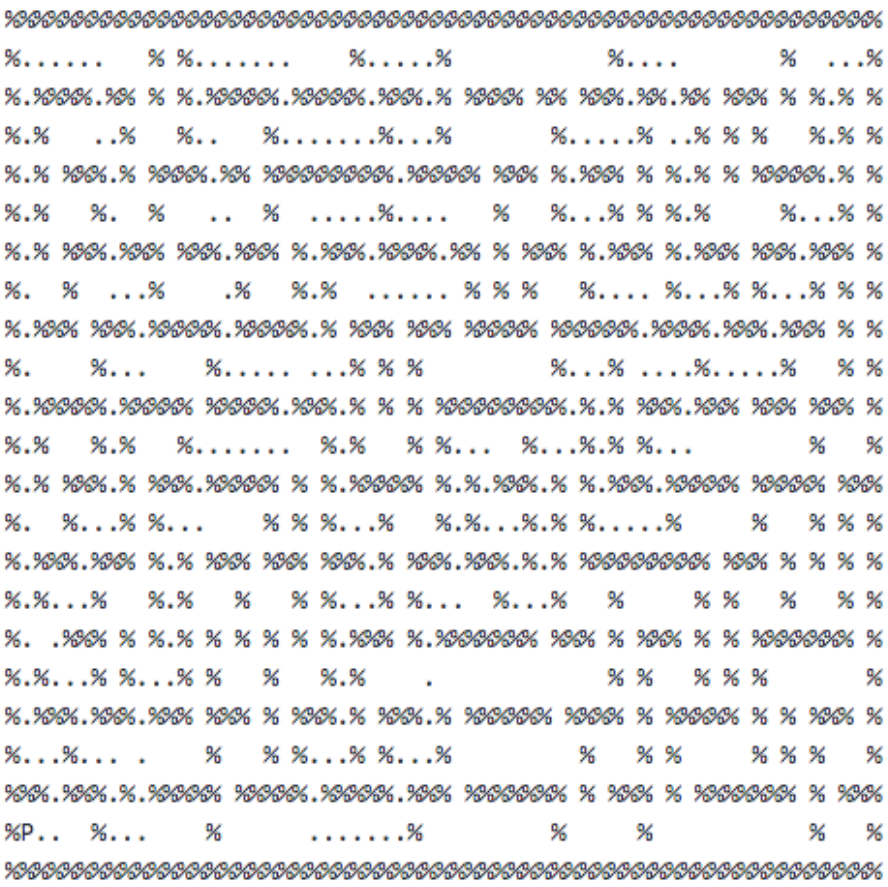

Figure 1: Depth-first search on the medium maze

BFS

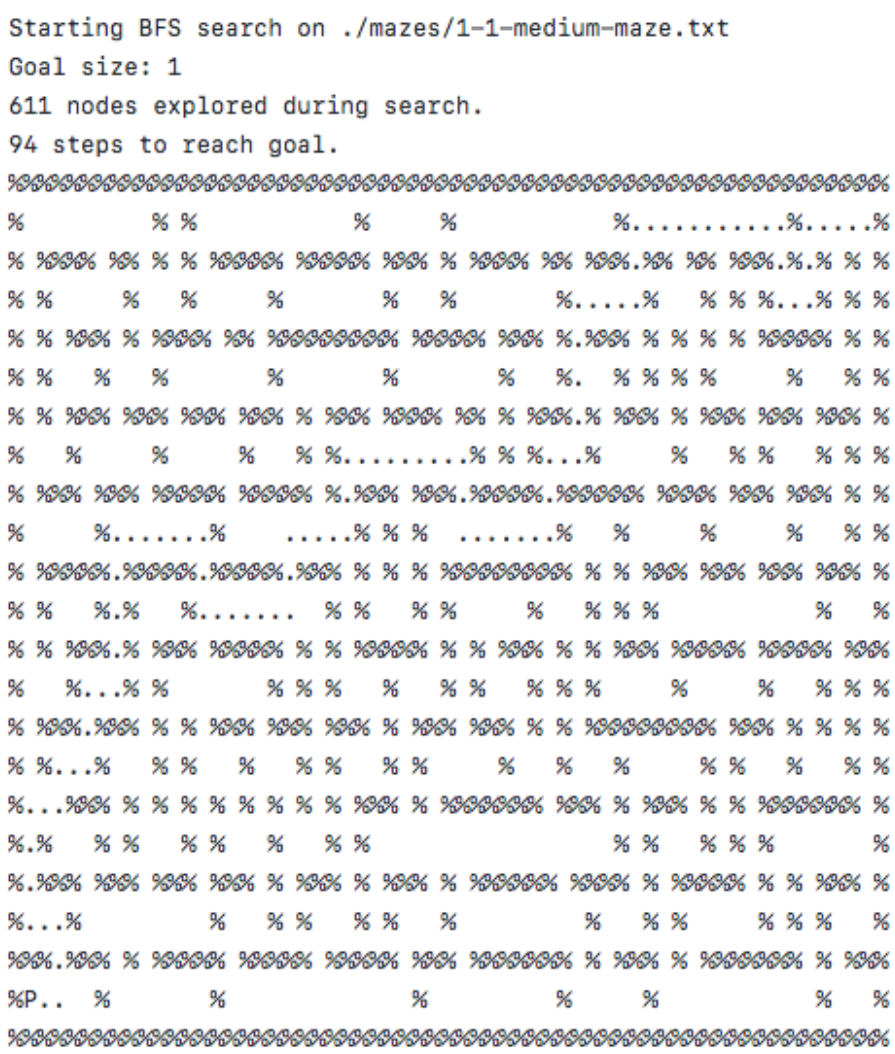

Figure 2: Breadth-first search on the medium maze

Greedy

```
Starting greedy search on ./mazes/1-1-medium-maze.txt
Goal size: 1
103 nodes explored during search.
94 steps to reach goal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          % %           %     %      ....  %...........%.....%
% %%%% %% % % %%%%% %%%%% %%% % %%%%.%%.%%%.%% %% %%%.%.% % %
% %      %   %     %        %   %  .... %.....%   % % %...% % %
% % %%% % %%%% %% %%%%%%%%% %%%%%.%%% % %%% % % % % %%%%% % %
% %   %   %       %        %    ....%   %   % % % %     %   % %
% % %%% %%% %%% %%% % %%% %%%%.%% % %%% % %%% % %%% %%% %%% %
%   %     %     %   % %........ % % %   %     %   % %   % % %
% %%% %%% %%%%% %%%%% %.%%% %%% %%%%% %%%%%% %%%% %%% %%% % %
%     %.......%     .....% % %          %   %     %     %   % %
% %%%%%.%%%%%.%%%%%.%%% % % % %%%%%%%%%% % % %%% %%% %%% %%% %
% %   %.%   %.......  % %   % %     %   % % %           %   %
% % %%%.% %%% %%%%% % % %%%%% % % %%% % % %%% %%%%% %%%%% %%%
%   %...% %       % % %   %   % %   % % %     %     %   % % %
% %%%.%%% % % %%% %%% %%% % %%% %%% % % %%%%%%%%% %%% % % % %
% %...%   % %   %   % %   % %     %   %   %     % %   %   % %
%...%%% % % % % % % % % % %%% % %%%%%%% %%% % %%% % % %%%%%%% %
%.%   % %   % %   %   % %                 % %   % % %       %
%.%%% %%% %%% %%% % %%% % %%% % %%%%%% %%%% % %%%%% % % %%% %
%...%         %   % %   % %   %         %   % %     % % %   %
%%%.%%% % %%%%% %%%%% %%%%% %%% %%%%%%% % %%% % %%%%%%% % %%%
%P..  %       %             %         %     %           %   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 3: Greedy search on the medium maze

A*

```
Starting A* search on ./mazes/1-1-medium-maze.txt
Goal size: 1
335 nodes explored during search.
94 steps to reach goal.
```

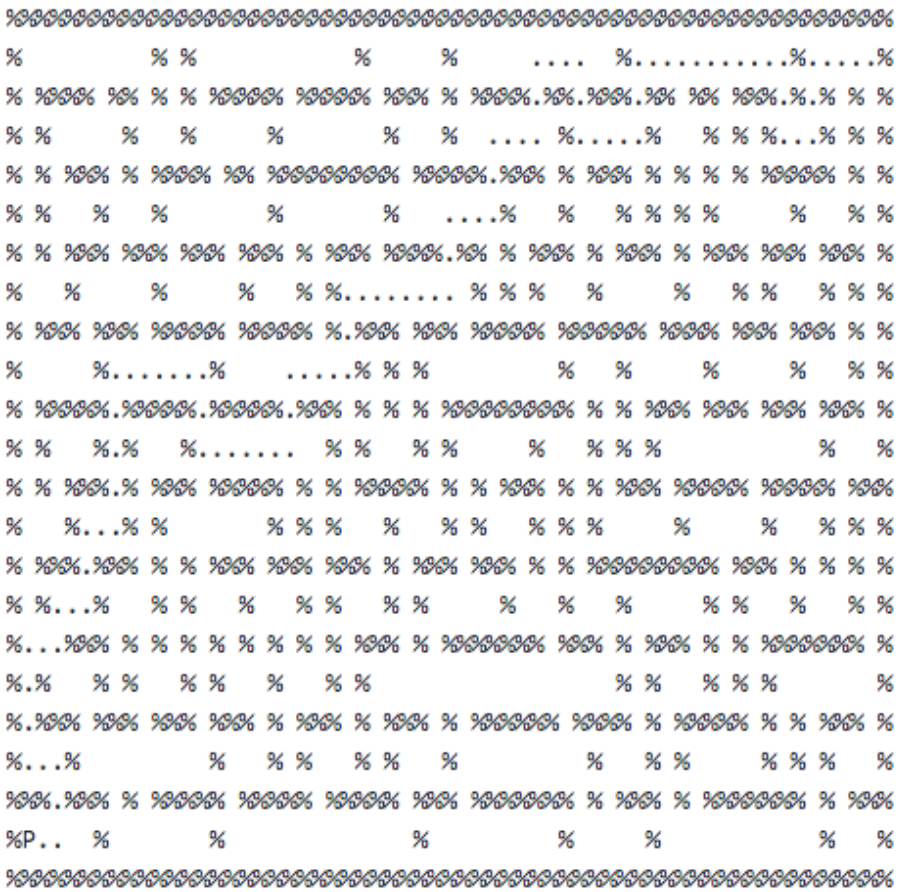

Figure 4: A* search on the medium maze

Big Maze Search Outputs

Below are the outputs we generated for the big maze using the four different search algorithms.

DFS

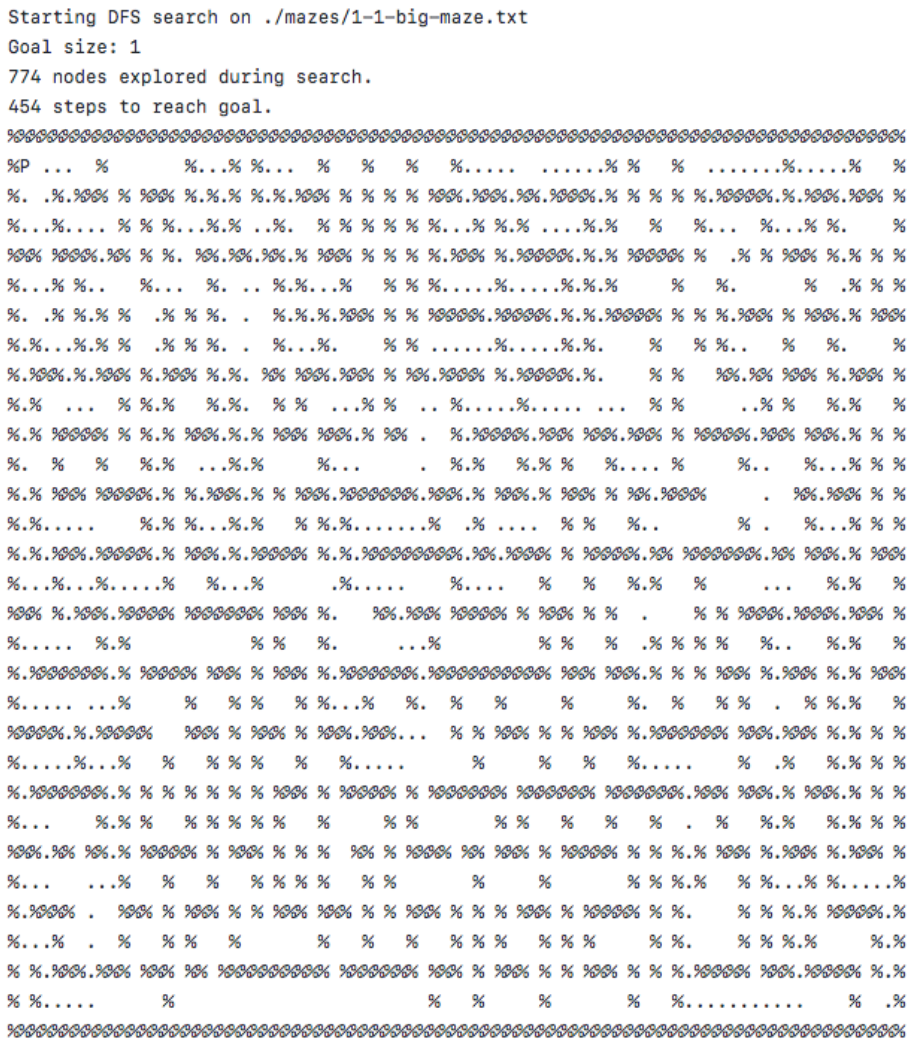

Figure 5: Depth-first search on the big maze

BFS

```
Starting BFS search on ./mazes/1-1-big-maze.txt
Goal size: 1
1259 nodes explored during search.
148 steps to reach goal.
```

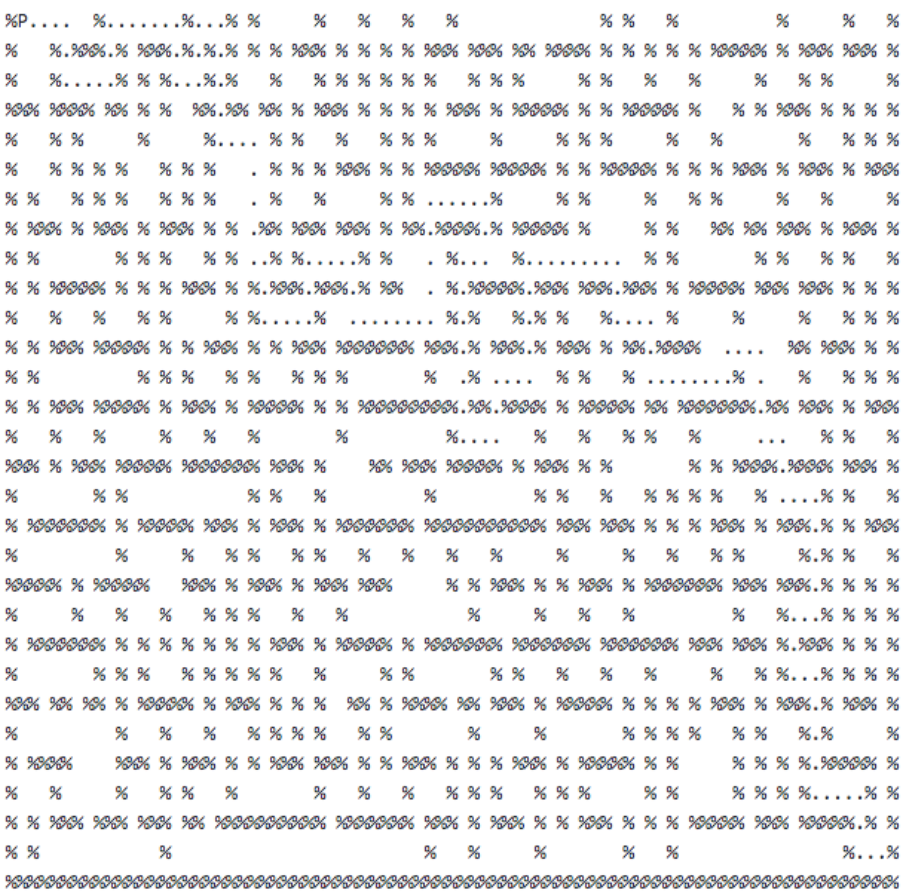

Figure 6: Breadth-first search on the big maze

Greedy

```
Starting greedy search on ./mazes/1-1-big-maze.txt
Goal size: 1
286 nodes explored during search.
222 steps to reach goal.
```

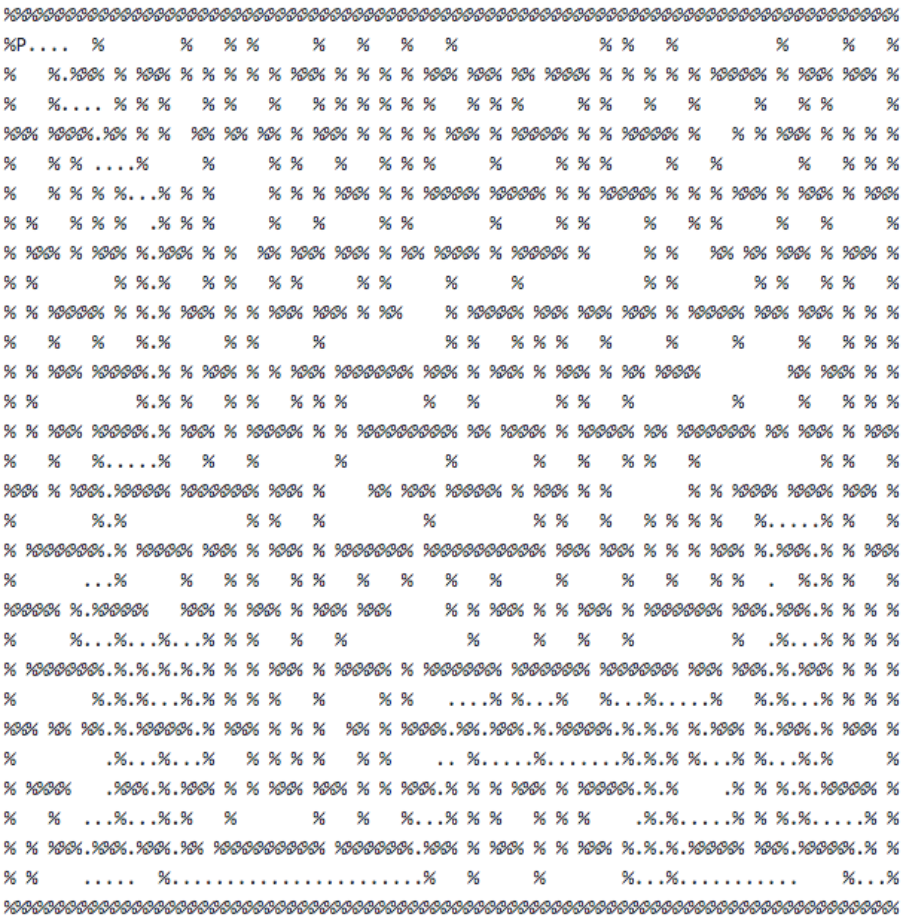

Figure 7: Greedy search on the big maze

A*

```
Starting A* search on ./mazes/1-1-big-maze.txt
Goal size: 1
1113 nodes explored during search.
148 steps to reach goal.
```

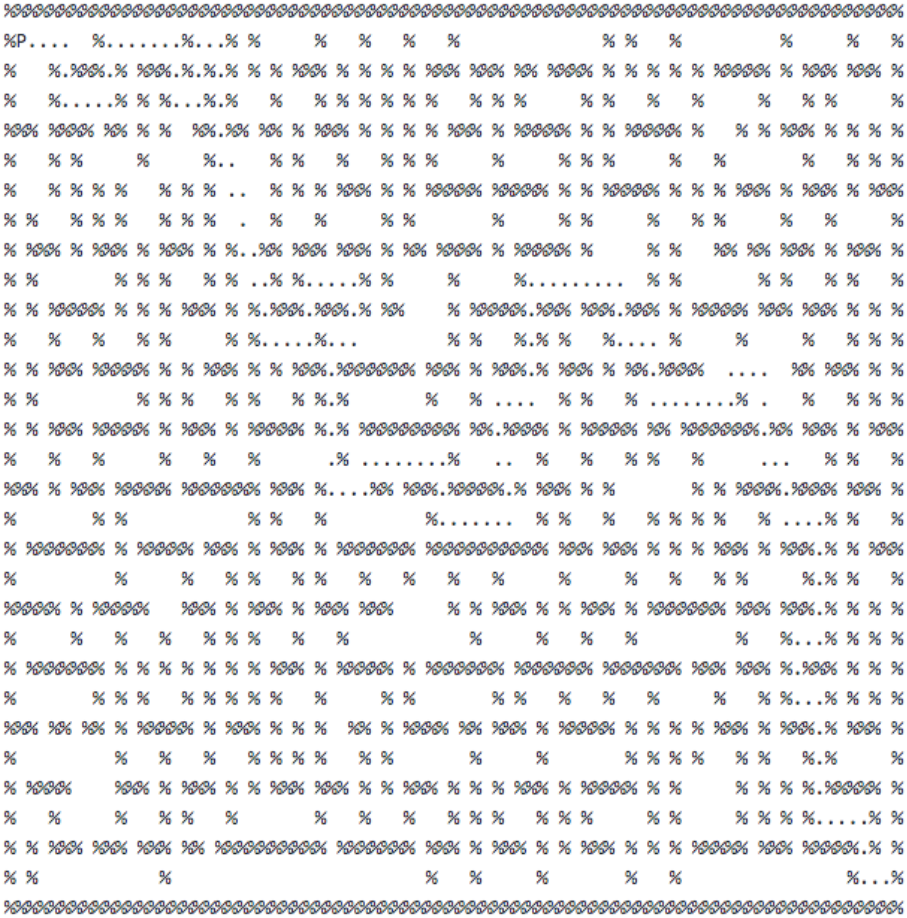

Figure 8: A* search on the big maze

Open Maze Search Outputs

Below are the outputs we generated for the open maze using the four different search algorithms.

DFS

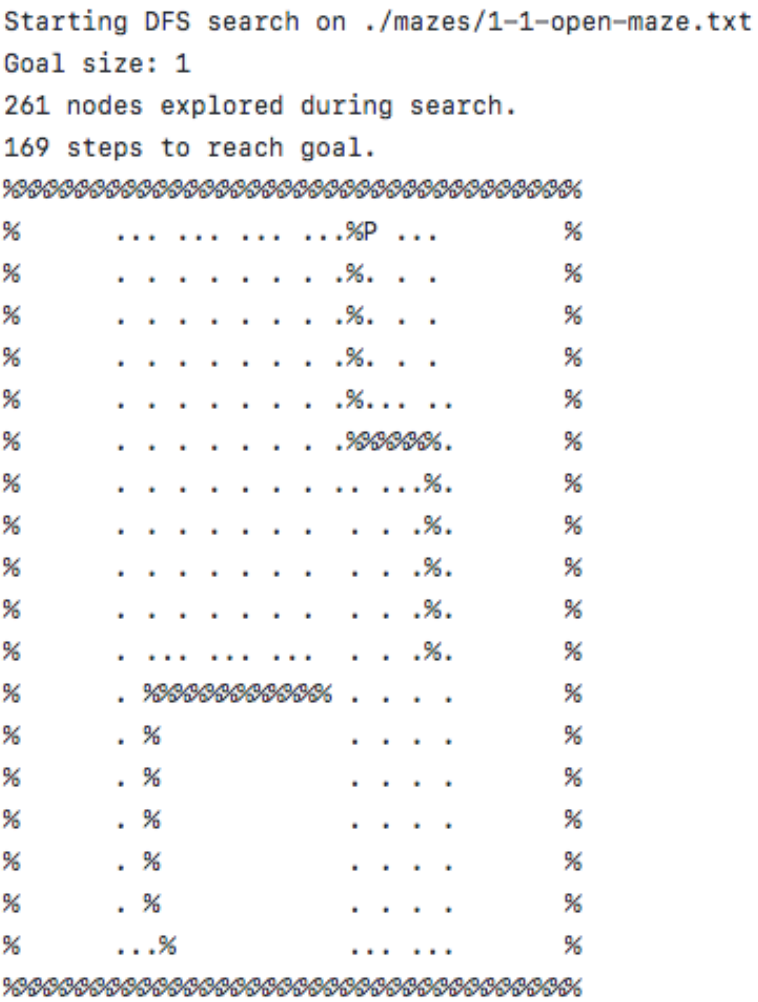

Figure 9: Depth-first search on the open maze

BFS

Figure 10: Breadth-first search on the open maze

Greedy

```
Starting greedy search on ./mazes/1-1-open-maze.txt
Goal size: 1
151 nodes explored during search.
45 steps to reach goal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     %P            %
%                     %.            %
%                     %.            %
%                     %.            %
%                     %......       %
%                     %%%%%%.       %
%                          %.       %
%                          %.       %
%                          %.       %
%                          %.       %
%       ..............     %.       %
%       .%%%%%%%%%%%%........       %
%       .%                          %
%       .%                          %
%       .%                          %
%       .%                          %
%       .%                          %
%       ..%                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 11: Greedy search on the open maze

A*

```
Starting A* search on ./mazes/1-1-open-maze.txt
Goal size: 1
238 nodes explored during search.
45 steps to reach goal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        %P              %
%                        %.              %
%                        %.              %
%                        %.              %
%                        %......        %
%                        %%%%%%.        %
%                             %.        %
%                             %.        %
%                             %.        %
%                             %.        %
%          ..............     %.        %
%          .%%%%%%%%%%%%%.....          %
%          .%                            %
%          .%                            %
%          .%                            %
%          .%                            %
%          .%                            %
%          ..%                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 12: A* search on the open maze

Section 1.2, Search With Multiple Dots

Tiny Maze Search Outputs

Below are the outputs we generated for the tiny maze using the BFS and A* search algorithms.

BFS

```
Starting BFS search on ./mazes/1-2-tiny-search.txt
Goal size: 12
56859 nodes explored during search.
36 steps to reach goal.
%%%%%%%%%%
%8  % 5  %
% %7% %% %
% %   6%4%
% 9%P%   %
%a  1  2 %
% %%%% % %
%b c   %3%
%%%%%%%%%%
```

Figure 13: Breadth-first search on the tiny maze

A*

```
Starting A* search on ./mazes/1-2-tiny-search.txt
Goal size: 12
4538 nodes explored during search.
36 steps to reach goal.
%%%%%%%%%%
%8  % 5  %
% %7% %% %
% %   6%4%
% 9%P%   %
%a  1  2 %
% %%%% % %
%b c   %3%
%%%%%%%%%%
```

Figure 14: A* search on the tiny maze

Small Maze Search Outputs

Below are the outputs we generated for the small maze using the BFS and A* search algorithms.

BFS

```
Starting BFS search on ./mazes/1-2-small-search.txt
Goal size: 15
4713546 nodes explored during search.
143 steps to reach goal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       P       1%       5  6    %
%   %%%%% %%%%%% %  % % %%   %
%   %d %      %   %  % %  %  7%
%c     %      2   %4   %  %%%%%
%%%%% %%%% %%%  %%%%%%%      8%
%e                3     % %%%  %
%% %%%%%%%% %%%%%%%%%% %    %
%        %                % %%%%
%        %%%%%      %a          %
%f% %%%     %  %    %% %% %%%%%%
%           % b%               9%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 15: Breadth-first search on the small maze

A*

```
Starting A* search on ./mazes/1-2-small-search.txt
Goal size: 15
2116049 nodes explored during search.
143 steps to reach goal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       P       1%       5  6    %
%    %%%%% %%%%%% %  % % %%    %
%    %c %      %    %  % %   %  7%
%d      %      2    %4    %   %%%%%
%%%%% %%%% %%%   %%%%%%%       8%
%e                  3     % %%%   %
%% %%%%%%%% %%%%%%%%%%% %     %
%         %                  % %%%%
%         %%%%%      %a           %
%f% %%%     %   %    %% %% %%%%%%
%            % b%                9%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 16: A* search on the small maze

Medium Maze Search Outputs

Below are the outputs we generated for the medium maze using the BFS and A* search algorithms.

A*

```
Starting A* search on ./mazes/1-2-medium-search.txt
Goal size: 20
91143862 nodes explored during search.
209 steps to reach goal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% g  %         d%                %  8%      % %    %6 %
%    %%%% %%%%%% %  % % %%          %%%%  %  %%% %
%    %e%    % c% %  % %  %   %   %           %   %
%   f             %b     %%%% %   %% 7%%% %       %
% %%%%% %%%%%%  %%%% %            9%     %5 %%%%%%%
%h       %   k%        a% %%%  %%%%%%  % %  %   % %
%%% %%  % %%%%%%% %%%%% %    %  1 %% % %    %3% %
%    %  %       %  %       P% %%        %   %%%% % %
%i       %   %  %  %      %            %       2%   %
% % %%%    % %    %% %% %%% %% %  % %%%%%%%%%% %
%   %j   %    %                %    %         4    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 17: A* search on the medium maze

Our Heuristic, Explained

For our A* search, we developed a heuristic based on the total weight of a manhattan distance minimum spanning tree that spans the current state position and the dots remaining.

First realize that in graph-theoretic terms, the optimal walk, W, of the maze graph through all remaining dots covers some subgraph S of the maze containing the current state position and each of the remaining dots. It follows that this optimal walk will cost at least as much as the cost of the subgraph it covers, more when edges are traversed multiple times. Letting $|\ |$ denote the cost of its argument, we have

$$|S| \leq |W|.$$

By definition of a minimum spanning tree, any minimum spanning tree is a minimum cost subgraph of the nodes it contains. Also, notice that by using the manhattan distance function, we are actually underestimating the cost of connecting the dots and the current position in a minimum spanning tree since we don't account the true distance between dots around obstacles. Since S contains at least the same nodes as the minimum spanning tree described, and the minimum spanning tree described (call it MST) is an underestimate of the minimum cost subgraph of these nodes, we can conclude that

$$|MST| \leq |S| \leq |W|.$$

Therefore, our heuristic is admissible.

Minimum Spanning Tree Implementation, Explained

The MSTs are created in $O(V^2 \lg V)$, where V is the number of dots remaining in the maze + 1 for the current state position. Since there are often few dots in the mazes, this heuristic can run very quickly. We used disjoint sets in conjunction with Kruskal's algorithm to form the MST.

Team Involvement

Jon Reynolds was responsible for implementing the maze class, and the node class as well as implementing the minimum spanning tree in section 1.2, as well as documenting his respective work in the report. Matthew Krikorian was responsible for writing the maze class and the main file which processed the mazes, as well as documenting the respective work in the report. Patrick McMahon was responsible for coming up with the heuristic for section 1.2, as well as implementing the disjoint sets class and the frontier class, as well as documenting his respective work in the report.