ECE 448

MP1: Search Algorithm

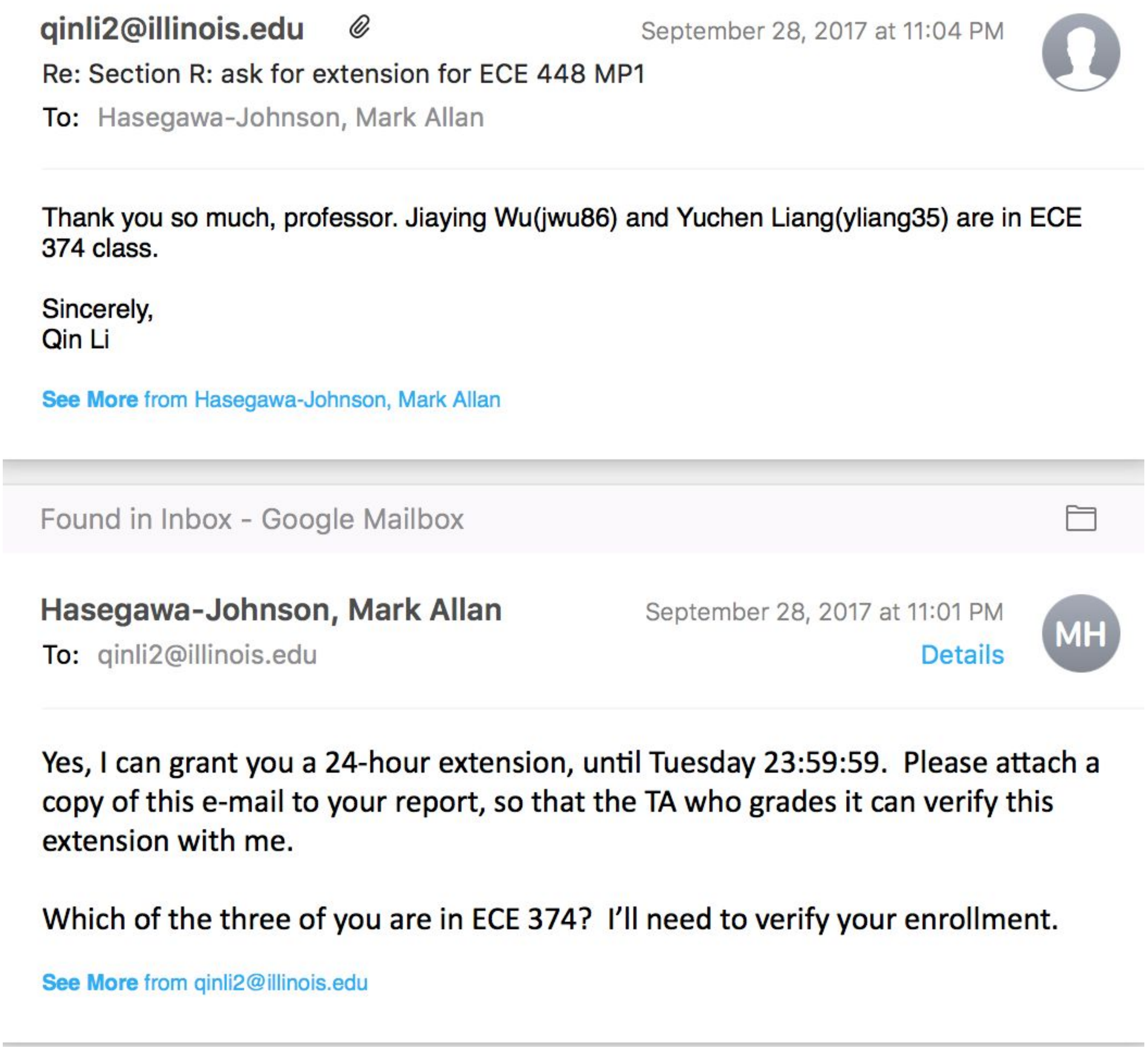
Date:2017/10/03

Credit: 3
Section: R

Team Member:

Jiaying Wu (jwu86)
Qin Li (qinli2)
Yuchen Liang (yliang35)

Professor Johnson has approved our one-day extension request for our group; the email is attached below



qinli2@illinois.edu September 28, 2017 at 11:04 PM

Re: Section R: ask for extension for ECE 448 MP1

To: Hasegawa-Johnson, Mark Allan

Thank you so much, professor. Jiaying Wu(jwu86) and Yuchen Liang(yliang35) are in ECE 374 class.

Sincerely,
Qin Li

See More from Hasegawa-Johnson, Mark Allan

Found in Inbox - Google Mailbox

Hasegawa-Johnson, Mark Allan September 28, 2017 at 11:01 PM

To: qinli2@illinois.edu

Details

Yes, I can grant you a 24-hour extension, until Tuesday 23:59:59. Please attach a copy of this e-mail to your report, so that the TA who grades it can verify this extension with me.

Which of the three of you are in ECE 374? I'll need to verify your enrollment.

See More from qinli2@illinois.edu

<u>Part 1</u>

Overall Structure

Before concerning about the searching algorithm of the maze, we wrote some helper functions and created some structures to help us complete the task. We built a class called maze, inside which we can call different searching algorithm. Firstly, we read the txt file to load the maze and converted the graph into 2-D array. Then, we searched for the start and goal points in the maze. After those preprocesses are finished, we can start our searching algorithm. To help with the searching algorithm, we built a helper function named *canTravel* to tell us whether a certain direction can be proceeded when we are in a specific point. Once we face a wall or outside the boundary of a maze, we cannot go to the direction. The whole searching process will be terminated after all of the points are found.

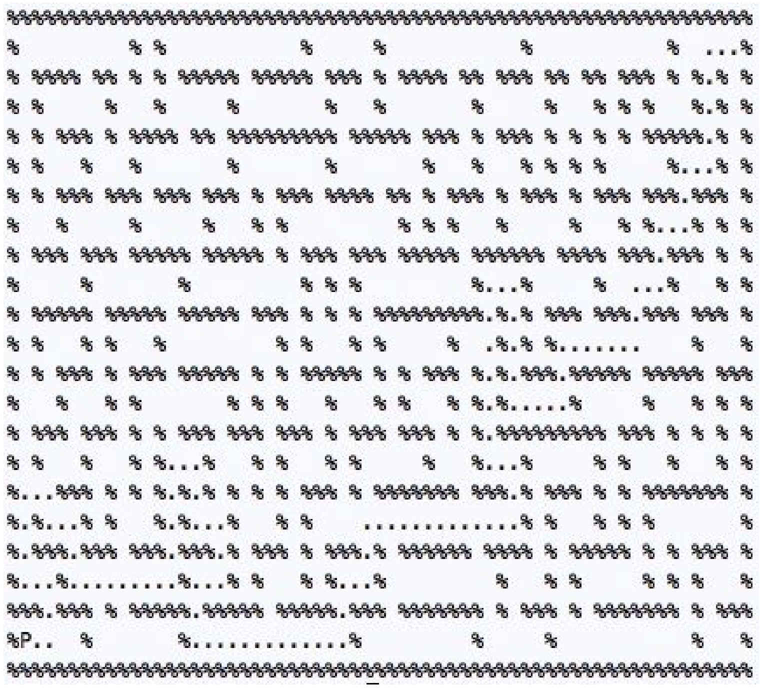
Part 1.1

DFS

Algorithm description:

In our DFS algorithm, we first build a stack to keep track the path of the maze, and a 2D array to help us to check if the maze is visited or not. Then, at each point, we search through the maze with direction order up, down, left and right. If there is a wall at a specific direction of a given point, then that direction is not accessible, and we move on to next position. If there is a point available to go for a direction, we will go to that point, and do our search with order of direction up, down, left and right again. For every step we have made, we will push it to the stack, so if we meet the dead end, we will able to pop the stack to backtrack the last available point where other directions are accessible. In addition, for every point we made, we will mark that point as a wall in the 2D array (which is the copy of the maze); this ensures that we will not go to the direction we have visited again. Once the target point is find, the stack will be returned. If there is no way to get the target point, a negative one error warning will be returned. In this search, it is not guaranteed to find the shortest path of the maze, but a viable path will be found.

Medium:



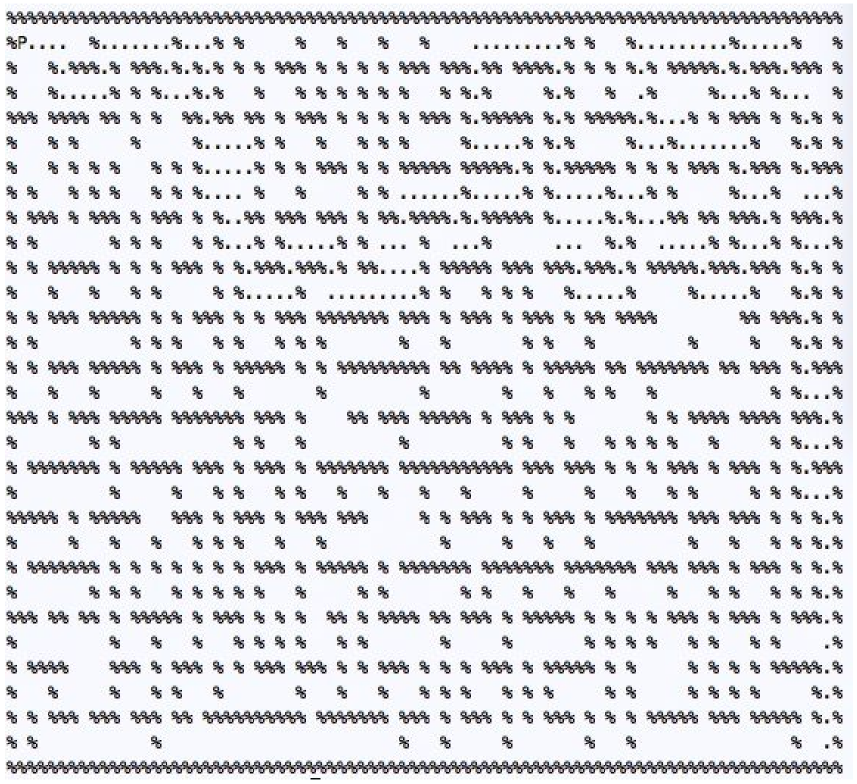
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%         % %           %     %           %           %  ...%
% %%%% %% % % %%%%% %%%%% %%% % %%%% %% %%% %% %% %%% % %.% %
% %     %   %     %       %   %       %     %   % % %   %.% %
% % %%% % %%%% %% %%%%%%%%% %%%%% %%% % %%% % % % % %%%%%.% %
% %   %   %       %       %       %   %   % % % %     %...% %
% % %%% %%% %%% %%% % %%% %%%% %% % %%% % %%% % %%% %%%.%%% %
%   %     %     %   % %         % % %   %     %   % %...% % %
% %%% %%% %%%%% %%%%% % %%% %%% %%%%% %%%%%% %%%% %%%.%%% % %
%     %       %         % % %         %...%     %  ...%   % %
% %%%%% %%%%% %%%%% %%% % % % %%%%%%%%%.%.% %%% %%%.%%% %%% %
% %   % %   %         % %   % %     %  .%.% %.......    %   %
% % %%% % %%% %%%%% % % %%%%% % % %%% %.%.%%%.%%%%% %%%%% %%%
%   %   % %       % % %   %   % %   % %.%.....%     %   % % %
% %%% %%% % % %%% %%% %%% % %%% %%% % %.%%%%%%%%% %%% % % % %
% %   %   % %...%   % %   % %     %   %...%     % %   %   % %
%...%%% % % %.%.% % % % %%% % %%%%%%% %%%.% %%% % % %%%%%%% %
%.%...% %   %.%...%   % %    .............% %   % % %       %
%.%%%.%%% %%%.%%%.% %%% % %%%.% %%%%%% %%%% % %%%%% % % %%% %
%...%.........%...% %   % %...%         %   % %     % % %   %
%%%.%%% % %%%%%.%%%%% %%%%%.%%% %%%%%%% % %%% % %%%%%%% % %%%
%P..  %       %.............%         %     %           %   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Computed path:See dots above

Solution cost: 126

Number of expanded nodes: 261

Big:



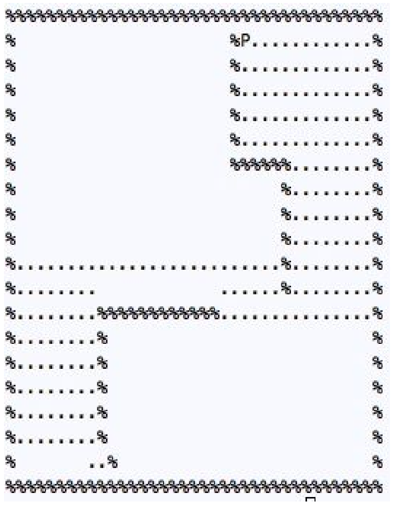
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%P....  %.......%...% %     %   %   %   %     .........% %   %.........%.....%   %
%   %.%%%.% %%%.%.%.% % % %%% % % % % %%% %%%.%% %%%%.% % % %.% %%%%%.%.%%%.%%% %
%   %.....% % %...%.%   %   % % % % % %   % %.%     %.%   %  .%     %...% %...   %
%%% %%%% %% % %  %%.%% %% % %%% % % % % %%% %.%%%%% %.% %%%%%.%...% % %%% % %.% %
%   % %     %     %.....% %   %   % % %     %.....% %.%     %...%.......%   %.% %
%   % % % %   % % %.....% % % %%% % % %%%%% %%%%%.% %.%%%%% % % % %%% %.%%% %.%%%
% %   % % %   % % %.... %   %     % % ......%.....% %.....%...% %     %...%  ...%
% %%% % %%% % %%% % %..%% %%% %%% % %%.%%%%.%.%%%%% %.....%.%...%% %% %%%.% %%%.%
% %       % % %   % %...% %.....% % ... %  ...%      ...  %.%  .....% %...% %...%
% % %%%%% % % % %%% % %.%%%.%%%.% %%....% %%%%% %%% %%%.%%%.% %%%%%.%%%.%%% %.% %
%   %   %   % %     % %.....%  .........% %   % % %   %.....%     %.....%   %.% %
% % %%% %%%%% % % %%% % % %%% %%%%%%% %%% % %%% % %%% % %% %%%%        %% %%%.% %
% %         % % %   % %   % % %       %   %       % %   %         %     %   %.% %
% % %%% %%%%% % %%% % %%%%% % % %%%%%%%%% %% %%%% % %%%%% %% %%%%%%% %% %%% %.%%%
%   %   %     %   %   %       %         %       %   %   % %   %           % %...%
%%% % %%% %%%%% %%%%%%% %%% %    %% %%% %%%%% % %%% % %       % % %%%% %%%% %%%.%
%       % %           % %   %         %         % %   %   % % % %   %     % %...%
% %%%%%%% % %%%%% %%% % %%% % %%%%%%% %%%%%%%%%%% %%% %%% % % % %%% % %%% % %.%%%
%         %     %   % %   % %   %   %   %   %     %     %   %   % %     % % %...%
%%%%% % %%%%%   %%% % %%% % %%% %%%     % % %%% % % %%% % %%%%%%% %%% %%% % % %.%
%     %   %   %   % % %   %   %           %     %   %   %         %   %   % % %.%
% %%%%%%% % % % % % % % %%% % %%%%% % %%%%%%% %%%%%%% %%%%%%% %%% %%% % %%% % %.%
%         % % %   % % % % %   %     % %       % %   %   %   %     %   % %   % % %.%
%%% %% %% % %%%%% % %%% % % %  %% % %%%% %% %%% % %%%%% % % % % %%% % %%% % %%%.%
%         %   %   %   % % % %   % %       %     %       % % % %   % %   % %    .%
% %%%%    %%% % %%% % % %%% %%% % % %%% % % % %%% % %%%%% % %     % % % % %%%%%.%
%   %     %   % %   %       %   %   %   % % %   % % %     % %     % % % %     %.%
% % %%% %%% %%% %% %%%%%%%%%% %%%%%%% %%% % %%% % % %%% % % % %%%%% %%% %%%%% %.%
% %         %                       %   %     %       %   %               %  .%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Computed path: See dots above

Solution cost: 240

Number of expanded nodes: 491

Open:



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     %P............%
%                     %.............%
%                     %.............%
%                     %.............%
%                     %.............%
%                     %%%%%%........%
%                          %........%
%                          %........%
%                          %........%
%..........................%........%
%........            ......%........%
%........%%%%%%%%%%%%...............%
%........%                          %
%........%                          %
%........%                          %
%........%                          %
%........%                          %
%       ..%                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Computed path:See dots above

Solution cost: 217

Number of expanded nodes: 944

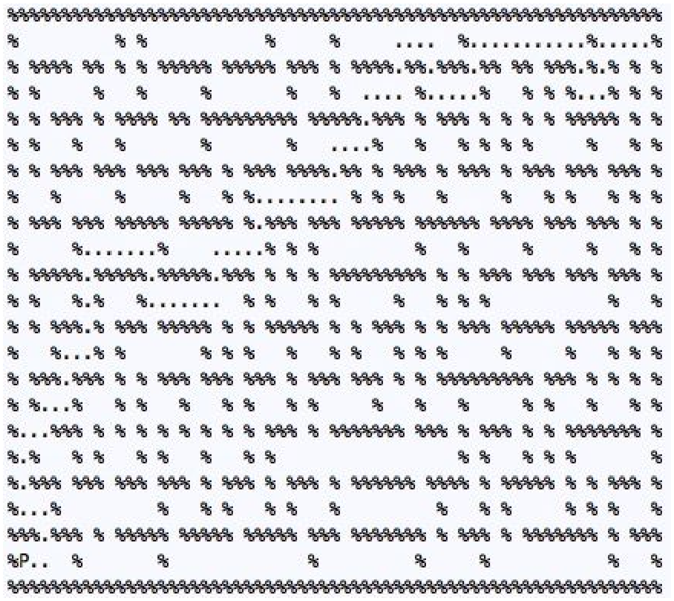
BFS

Algorithm description:

When using the BFS algorithm, we create a queue firstly, and push the start point into the queue. The a while loop is applied, the while loop continues unless there is no points in the queue (no solution) or the goal has been found. Inside the while loop, one point will be dequeued in each iteration. The point will be expanded to 4 directions. If the point expanded has not been explored or can be traveled to, the point will be enqueued.

A dictionary is applied. Once a parent point want to expand certain child point, the child point will be treated as key and the parent point will be its corresponding value. Therefore, once we reach the goal, we can trace back to find the optimized path. The dictionary helps us store the path during the searching process. Also, to prevent from repetition, a 2-D array named *explored* with the same dimensions of the graph will be applied to denote whether a certain position has been detected. The values in the array are initialized to zero Once we reach certain points, the value of that position in explored array will be turned to one. If the point are explored before, the point will not be enqueued.

Medium:



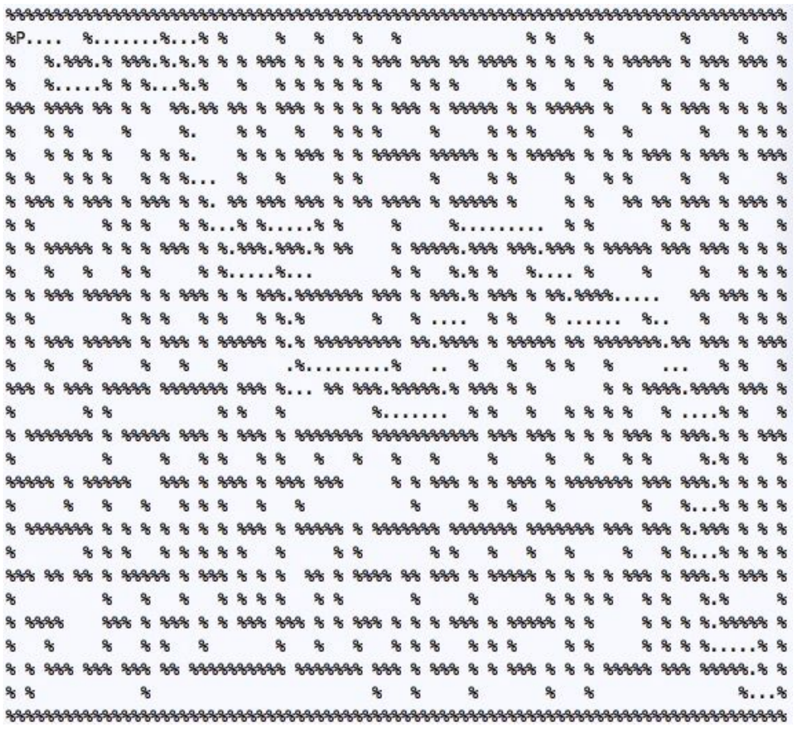
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%         % %           %     %     ....  %...........%.....%
% %%%% %% % % %%%%% %%%%% %%% % %%%%.%%.%%%.%% %% %%%.%.% % %
% %     %   %     %       %   %  .... %.....%   % % %...% % %
% % %%% % %%%% %% %%%%%%%%% %%%%%.%%% % %%% % % % % %%%%% % %
% %   %   %       %       %   ....%   %   % % % %     %   % %
% % %%% %%% %%% %%% % %%% %%%%.%% % %%% % %%% % %%% %%% %%% %
%   %     %     %   % %........ % % %   %     %   % %   % % %
% %%% %%% %%%%% %%%%% %.%%% %%% %%%%% %%%%%% %%%% %%% %%% % %
%     %.......%    .....% % %         %   %     %     %   % %
% %%%%%.%%%%%.%%%%%.%%% % % % %%%%%%%%% % % %%% %%% %%% %%% %
% %   %.%   %.......  % %   % %     %   % % %           %   %
% % %%%.% %%% %%%%% % % %%%%% % % %%% % % %%% %%%%% %%%%% %%%
%   %...% %       % % %   %   % %   % % %     %     %   % % %
% %%%.%%% % % %%% %%% %%% % %%% %%% % % %%%%%%%%% %%% % % % %
% %...%   % %   %   % %   % %     %   %   %     % %   %   % %
%...%%% % % % % % % % % %%% % %%%%%%% %%% % %%% % % %%%%%%% %
%.%   % %   % %   %   % %                 % %   % % %       %
%.%%% %%% %%% %%% % %%% % %%% % %%%%%% %%%% % %%%%% % % %%% %
%...%         %   % %   % %   %         %   % %     % % %   %
%%%.%%% % %%%%% %%%%% %%%%% %%% %%%%%%% % %%% % %%%%%%% % %%%
%P..  %       %             %         %     %           %   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Computed path:See dots above

Solution cost: 94

Number of expanded nodes: 609

Big:



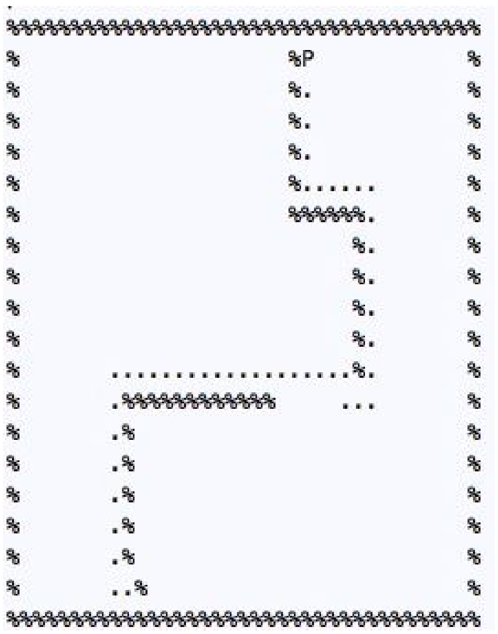
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%P....  %.......%...% %     %   %   %   %             % %   %         %     %   %
%   %.%%%.% %%%.%.%.% % % %%% % % % % %%% %%% %% %%%% % % % % % %%%%% % %%% %%% %
%   %.....% % %...%.%   %   % % % % % %   % % %     % %   %   %     %   % %     %
%%% %%%% %% % %  %%.%% %% % %%% % % % % %%% % %%%%% % % %%%%% %   % % %%% % % % %
%   % %     %     %.    % %   %   % % %     %     % % %     %   %       %   % % %
%   % % % %   % % %.    % % % %%% % % %%%%% %%%%% % % %%%%% % % % %%% % %%% % %%%
% %   % % %   % % %...  %   %     % %       %     % %     %   % %     %   %     %
% %%% % %%% % %%% % %. %% %%% %%% % %% %%%% % %%%%% %     % %   %% %% %%% % %%% %
% %       % % %   % %...% %.....% %     %     %.........  % %       % %   % %   %
% % %%%%% % % % %%% % %.%%%.%%%.% %%    % %%%%%.%%% %%%.%%% % %%%%% %%% %%% % % %
%   %   %   % %     % %.....%...        % %   %.% %   %.... %     %     %   % % %
% % %%% %%%%% % % %%% % % %%%.%%%%%%% %%% % %%%.% %%% % %%.%%%%.....   %% %%% % %
% %         % % %   % %   % %.%       %   % ....  % %   % ......  %..   %   % % %
% % %%% %%%%% % %%% % %%%%% %.% %%%%%%%%% %%.%%%% % %%%%% %% %%%%%%%.%% %%% % %%%
%   %   %     %   %   %      .%.........%   ..  %   %   % %   %     ...   % %   %
%%% % %%% %%%%% %%%%%%% %%% %... %% %%%.%%%%%.% %%% % %       % % %%%%.%%%% %%% %
%       % %           % %   %         %.......  % %   %   % % % %   % ....% %   %
% %%%%%%% % %%%%% %%% % %%% % %%%%%%% %%%%%%%%%%% %%% %%% % % % %%% % %%%.% % %%%
%         %     %   % %   % %   %   %   %   %     %     %   %   % %     %.% %   %
%%%%% % %%%%%   %%% % %%% % %%% %%%     % % %%% % % %%% % %%%%%%% %%% %%%.% % % %
%     %   %   %   % % %   %   %           %     %   %   %         %   %...% % % %
% %%%%%%% % % % % % % % %%% % %%%%% % %%%%%%% %%%%%%% %%%%%%% %%% %%% %.%%% % % %
%       % % %   % % % % %   %     % %       % %   %   %   %     %   % %...% % % %
%%% %% %% % %%%%% % %%% % % %  %% % %%%% %% %%% % %%%%% % % % % %%% % %%%.% %%% %
%         %   %   %   % % % %   % %       %     %       % % % %   % %   %.%     %
% %%%%    %%% % %%% % % %%% %%% % % %%% % % % %%% % %%%%% % %     % % % %.%%%%% %
%   %     %   % %   %       %   %   %   % % %   % % %     % %     % % % %.....% %
% % %%% %%% %%% %% %%%%%%%%%% %%%%%%% %%% % %%% % % %%% % % % %%%%% %%% %%%%%.% %
% %           %                       %   %     %       %   %               %...%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Computed path:

Solution cost: 148

Number of expanded nodes: 1256

Open:



```
.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     %P            %
%                     %.            %
%                     %.            %
%                     %.            %
%                     %......       %
%                     %%%%%%.       %
%                          %.       %
%                          %.       %
%                          %.       %
%                          %.       %
%       ...................%.       %
%       .%%%%%%%%%%%%     ...       %
%       .%                          %
%       .%                          %
%       .%                          %
%       .%                          %
%       .%                          %
%       ..%                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Computed path:See dots above

Solution cost: 45

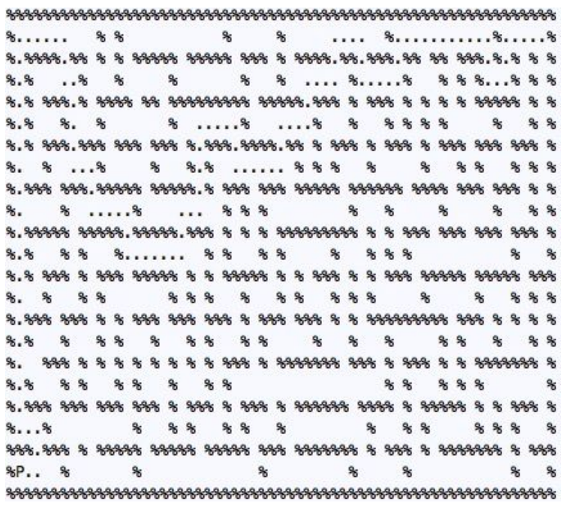
Number of expanded nodes: 524

Greedy

Algorithm description:

The basic structure of Greedy algorithm is same as the BFS. The queue used for BFS algorithm is replaced with a Priority Queue so that the value with highest priority (lowest key value) will be extracted during each dequeuing process. The value stored in the queue is tuple of key and the position. The value of key equals to the heuristic which is the manhattan distance between goal and current position.

Medium:



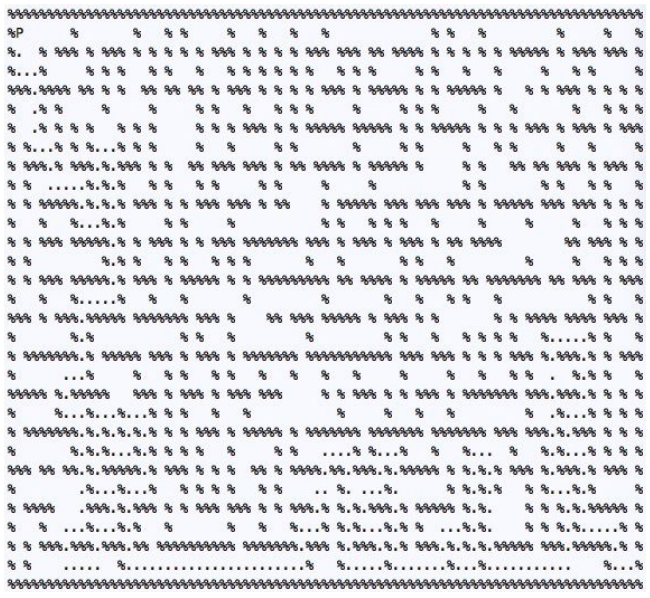
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%......   % %           %     %     ....  %...........%.....%
%.%%%%.%% % % %%%%% %%%%% %%% % %%%%.%%.%%%.%% %% %%%.%.% % %
%.%   ..%   %     %       %   %  .... %.....%   % % %...% % %
%.% %%%.% %%%% %% %%%%%%%%% %%%%%.%%% % %%% % % % % %%%%% % %
%.%   %.  %       %  .....%   ....%   %   % % % %     %   % %
%.% %%%.%%% %%% %%% %.%%%.%%%%.%% % %%% % %%% % %%% %%% %%% %
%.  %  ...%     %   %.%  ...... % % %   %     %   % %   % % %
%.%%% %%%.%%%%% %%%%%.% %%% %%% %%%%% %%%%%% %%%% %%% %%% % %
%.    %  .....%    ...  % % %         %   %     %     %   % %
%.%%%%% %%%%%.%%%%%.%%% % % % %%%%%%%%% % % %%% %%% %%% %%% %
%.%   % %   %.......  % %   % %     %   % % %           %   %
%.% %%% % %%% %%%%% % % %%%%% % % %%% % % %%% %%%%% %%%%% %%%
%.  %   % %       % % %   %   % %   % % %     %     %   % % %
%.%%% %%% % % %%% %%% %%% % %%% %%% % % %%%%%%%%% %%% % % % %
%.%   %   % %   %   % %   % %     %   %   %     % %   %   % %
%.  %%% % % % % % % % % %%% % %%%%%%% %%% % %%% % % %%%%%%% %
%.%   % %   % %   %   % %                 % %   % % %       %
%.%%% %%% %%% %%% % %%% % %%% % %%%%%% %%%% % %%%%% % % %%% %
%...%         %   % %   % %   %         %   % %     % % %   %
%%%.%%% % %%%%% %%%%% %%%%% %%% %%%%%%% % %%% % %%%%%%% % %%%
%P..  %       %             %         %     %           %   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Computed path:See dots above

Solution cost: 114

Number of expanded nodes: 134

Big:



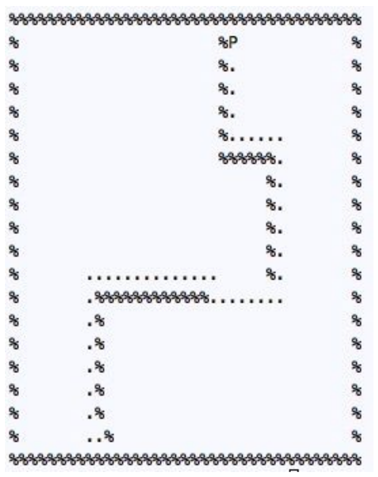
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%P      %       %   % %     %   %   %   %             % %   %         %     %   %
%.  % %%% % %%% % % % % % %%% % % % % %%% %%% %% %%%% % % % % % %%%%% % %%% %%% %
%...%     % % %   % %   %   % % % % % %   % % %     % %   %   %     %   % %     %
%%%.%%%% %% % %  %% %% %% % %%% % % % % %%% % %%%%% % % %%%%% %   % % %%% % % % %
%  .% %     %     %     % %   %   % % %     %     % % %     %   %       %   % % %
%  .% % % %   % % %     % % % %%% % % %%%%% %%%%% % % %%%%% % % % %%% % %%% % %%%
% %...% % %...% % %     %   %     % %       %     % %     %   % %     %   %     %
% %%%.% %%%.%.%%% % %  %% %%% %%% % %% %%%% % %%%%% %     % %   %% %% %%% % %%% %
% %  .....%.%.%   % %   % %     % %     %     %           % %       % %   % %   %
% % %%%%%.%.%.% %%% % % %%% %%% % %%    % %%%%% %%% %%% %%% % %%%%% %%% %%% % % %
%   %   %...%.%     % %     %           % %   % % %   %     %     %     %   % % %
% % %%% %%%%%.% % %%% % % %%% %%%%%%% %%% % %%% % %%% % %% %%%%        %% %%% % %
% %         %.% %   % %   % % %       %   %       % %   %         %     %   % % %
% % %%% %%%%%.% %%% % %%%%% % % %%%%%%%%% %% %%%% % %%%%% %% %%%%%%% %% %%% % %%%
%   %   %.....%   %   %       %         %       %   %   % %   %           % %   %
%%% % %%%.%%%%% %%%%%%% %%% %    %% %%% %%%%% % %%% % %       % % %%%% %%%% %%% %
%       %.%           % %   %         %         % %   %   % % % %   %.....% %   %
% %%%%%%%.% %%%%% %%% % %%% % %%%%%%% %%%%%%%%%%% %%% %%% % % % %%% %.%%%.% % %%%
%      ...%     %   % %   % %   %   %   %   %     %     %   %   % %  .  %.% %   %
%%%%% %.%%%%%   %%% % %%% % %%% %%%     % % %%% % % %%% % %%%%%%% %%%.%%%.% % % %
%     %...%...%...% % %   %   %           %     %   %   %         %  .%...% % % %
% %%%%%%%.%.%.%.%.% % % %%% % %%%%% % %%%%%%% %%%%%%% %%%%%%% %%% %%%.%.%%% % % %
%       %.%.%...%.% % % %   %     % %   ....% %...%   %   %...  %   %.%...% % % %
%%% %% %%.%.%%%%%.% %%% % % %  %% % %%%%.%%.%%%.%.%%%%% % %.%.% %%% %.%%%.% %%% %
%        .%...%...%   % % % %   % %    .. %. ...%.      % %.%.%   % %...%.%     %
% %%%%   .%%%.%.%%% % % %%% %%% % % %%%.% %.%.%%%.% %%%%% %.%.    % % %.%.%%%%% %
%   %  ...%...%.%   %       %   %   %...% %.%...%.% %  ...%.%.    % % %.%.....% %
% % %%%.%%%.%%%.%% %%%%%%%%%% %%%%%%%.%%% %.%%%.%.% %%%.%.%.%.%%%%% %%%.%%%%%.% %
% %    .....  %.......................%   %.....%.......%...%...........    %...%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Computed path:See dots above

Solution cost: 234

Number of expanded nodes: 293

Open:



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     %P            %
%                     %.            %
%                     %.            %
%                     %.            %
%                     %......       %
%                     %%%%%%.       %
%                          %.       %
%                          %.       %
%                          %.       %
%                          %.       %
%       ..............     %.       %
%       .%%%%%%%%%%%%........       %
%       .%                          %
%       .%                          %
%       .%                          %
%       .%                          %
%       .%                          %
%       ..%                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Computed path:See dots above

Solution cost: 45

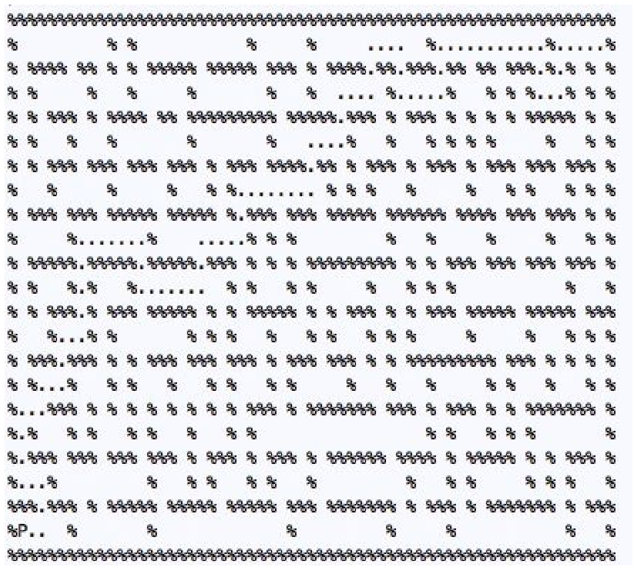
Number of expanded nodes: 148

*A**

Algorithm description:

The A* algorithm has the same structure as the greedy algorithm, the only difference is the content pushed into the Priority Queue, sorted by the cost plus heuristic to find position with the minimum of the sum. A tuple (key, position, cost) is pushed into Priority Queue during each enqueuing process. The value of key is calculated by adding the cost and the Manhattan distance between the current position and the goal point.

Medium:



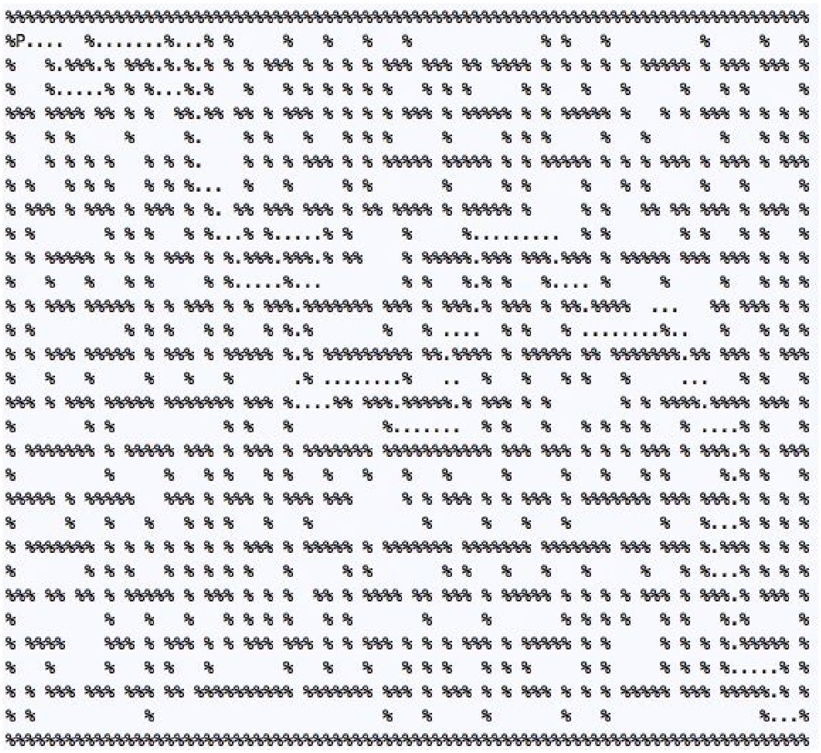
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%         % %           %     %     ....  %...........%.....%
% %%%% %% % % %%%%% %%%%% %%% % %%%%.%%.%%%.%% %% %%%.%.% % %
% %     %   %     %       %   %  .... %.....%   % % %...% % %
% % %%% % %%%% %% %%%%%%%%% %%%%%.%%% % %%% % % % % %%%%% % %
% %   %   %       %       %   ....%   %   % % % %     %   % %
% % %%% %%% %%% %%% % %%% %%%%.%% % %%% % %%% % %%% %%% %%% %
%   %     %     %   % %........ % % %   %     %   % %   % % %
% %%% %%% %%%%% %%%%% %.%%% %%% %%%%% %%%%%% %%%% %%% %%% % %
%     %.......%    .....% % %         %   %     %     %   % %
% %%%%%.%%%%%.%%%%%.%%% % % % %%%%%%%%% % % %%% %%% %%% %%% %
% %   %.%   %.......  % %   % %     %   % % %           %   %
% % %%%.% %%% %%%%% % % %%%%% % % %%% % % %%% %%%%% %%%%% %%%
%   %...% %       % % %   %   % %   % % %     %     %   % % %
% %%%.%%% % % %%% %%% %%% % %%% %%% % % %%%%%%%%% %%% % % % %
% %...%   % %   %   % %   % %     %   %   %     % %   %   % %
%...%%% % % % % % % % % %%% % %%%%%%% %%% % %%% % % %%%%%%% %
%.%   % %   % %   %   % %                 % %   % % %       %
%.%%% %%% %%% %%% % %%% % %%% % %%%%%% %%%% % %%%%% % % %%% %
%...%         %   % %   % %   %         %   % %     % % %   %
%%%.%%% % %%%%% %%%%% %%%%% %%% %%%%%%% % %%% % %%%%%%% % %%%
%P..  %       %             %         %     %           %   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Computed path: See dots above

Solution cost: 94

Number of expanded nodes: 335

Big:



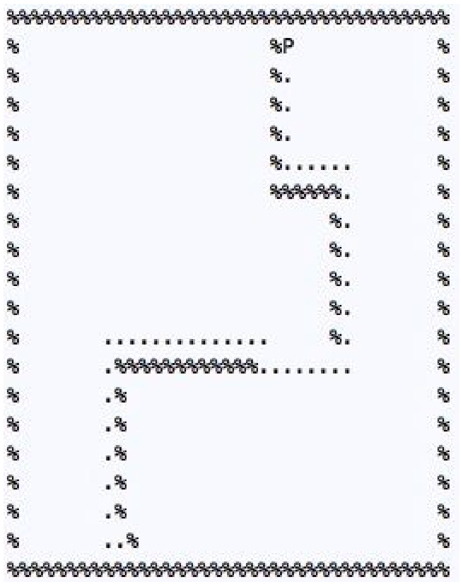
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%P....  %.......%...% %     %   %   %   %             % %   %         %     %   %
%   %.%%%.% %%%.%.%.% % % %%% % % % % %%% %%% %% %%%% % % % % % %%%%% % %%% %%% %
%   %.....% % %...%.%   %   % % % % % %   % % %     % %   %   %     %   % %     %
%%% %%%% %% % %  %%.%% %% % %%% % % % % %%% % %%%%% % % %%%%% %   % % %%% % % % %
%   % %     %     %.    % %   %   % % %     %     % % %     %   %       %   % % %
%   % % % %   % % %.    % % % %%% % % %%%%% %%%%% % % %%%%% % % % %%% % %%% % %%%
% %   % % %   % % %...  %   %     % %       %     % %     %   % %     %   %     %
% %%% % %%% % %%% % %. %% %%% %%% % %% %%%% % %%%%% %     % %   %% %% %%% % %%% %
% %       % % %   % %...% %.....% %     %     %.........  % %       % %   % %   %
% % %%%%% % % % %%% % %.%%%.%%%.% %%    % %%%%%.%%% %%%.%%% % %%%%% %%% %%% % % %
%   %   %   % %     % %.....%...        % %   %.% %   %.... %     %     %   % % %
% % %%% %%%%% % % %%% % % %%%.%%%%%%% %%% % %%%.% %%% % %%.%%%%  ...   %% %%% % %
% %         % % %   % %   % %.%       %   % ....  % %   % ........%..   %   % % %
% % %%% %%%%% % %%% % %%%%% %.% %%%%%%%%% %%.%%%% % %%%%% %% %%%%%%%.%% %%% % %%%
%   %   %     %   %   %      .% ........%   ..  %   %   % %   %     ...   % %   %
%%% % %%% %%%%% %%%%%%% %%% %....%% %%%.%%%%%.% %%% % %       % % %%%%.%%%% %%% %
%       % %           % %   %         %.......  % %   %   % % % %   % ....% %   %
% %%%%%%% % %%%%% %%% % %%% % %%%%%%% %%%%%%%%%%% %%% %%% % % % %%% % %%%.% % %%%
%         %     %   % %   % %   %   %   %   %     %     %   %   % %     %.% %   %
%%%%% % %%%%%   %%% % %%% % %%% %%%     % % %%% % % %%% % %%%%%%% %%% %%%.% % % %
%     %   %   %   % % %   %   %           %     %   %   %         %   %...% % % %
% %%%%%%% % % % % % % % %%% % %%%%% % %%%%%%% %%%%%%% %%%%%%% %%% %%% %.%%% % % %
%       % % %   % % % % %   %     % %       % %   %   %   %     %   % %...% % % %
%%% %% %% % %%%%% % %%% % % %  %% % %%%% %% %%% % %%%%% % % % % %%% % %%%.% %%% %
%         %   %   %   % % % %   % %       %     %       % % % %   % %   %.%     %
% %%%%    %%% % %%% % % %%% %%% % % %%% % % % %%% % %%%%% % %     % % % %.%%%%% %
%   %     %   % %   %       %   %   %   % % %   % % %     % %     % % % %.....% %
% % %%% %%% %%% %% %%%%%%%%%% %%%%%%% %%% % %%% % % %%% % % % %%%%% %%% %%%%%.% %
% %           %                       %   %     %       %   %               %...%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Computed path:See dots above

Solution cost: 148

Number of expanded nodes: 1113

Open:



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     %P            %
%                     %.            %
%                     %.            %
%                     %.            %
%                     %......       %
%                     %%%%%%.       %
%                          %.       %
%                          %.       %
%                          %.       %
%                          %.       %
%       ..............     %.       %
%       .%%%%%%%%%%%%........       %
%       .%                          %
%       .%                          %
%       .%                          %
%       .%                          %
%       .%                          %
%       ..%                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Computed path:See dots above

Solution cost: 45

Number of expanded nodes: 198

Part 1.2

A*

Algorithm description:

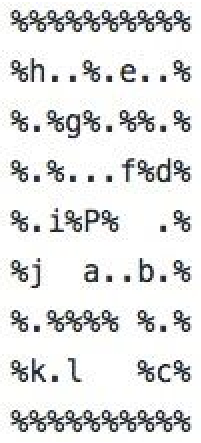
Similar to part 1.1, the A* algorithm expands the node with the least cost plus heuristic value. The center process is realized by a PQ (priority queue). The path so far to reach the current point is also pushed onto the queue.

Once the status of the graph changes (a certain goal point are found), the explored array will be refreshed. A list of status (currently undetected goals, path, explored array, cost …..) will be pushed into queue to record the state of current node.

The key difference with part 1.1 is in the choice of heuristics. In part 1.2, we choose the heuristic of a particular position as the sum of 1) the Manhattan distance to reach the closest goal, and 2) the total edge length (i.e. Manhattan distance) of the minimum spanning tree (MST), formed by the goals still to be found.

Now we argue that the above heuristic is admissible. The cost for a particular position to reach all the goals includes two parts: 1) the cost for the current position to reach one of the goals, and 2) the total cost to traverse all the rest of the goals from that one goal. The first part is larger or equal to the cost reaching the closest goal, while the second part is larger or equal to the sum of edges of MST, which is true by definition. Therefore, the heuristic value is always less than or equal to the true path value (and is a good estimate), so the heuristic is admissible.

Tiny:



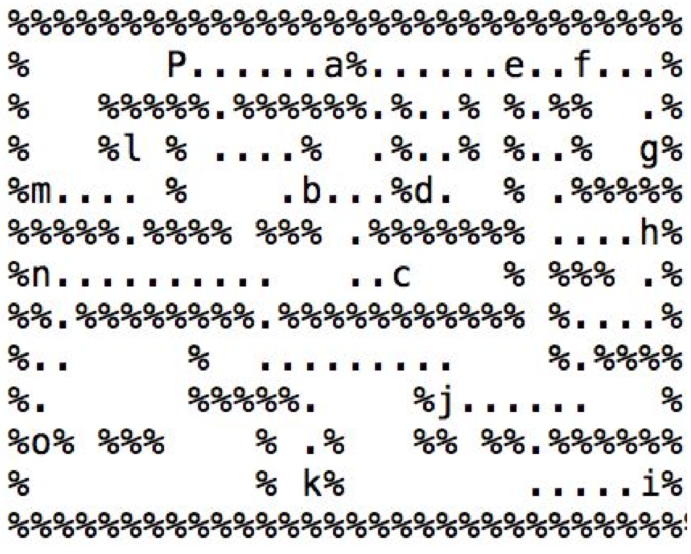
```
%%%%%%%%%%
%h..%.e..%
%.%g%.%%.%
%.%...f%d%
%.i%P%  .%
%j  a..b.%
%.%%%% %.%
%k.l   %c%
%%%%%%%%%%
```

Computed path: See dots above

Solution cost: 36

Number of expanded nodes: 599

Small:



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      P......a%......e..f...%
%   %%%%%.%%%%%%.%..% %.%%  .%
%   %l % ....%  .%..% %..%  g%
%m.... %    .b...%d.  % .%%%%%
%%%%%.%%%% %%% .%%%%%%% ....h%
%n..........   ..c    % %%% .%
%%.%%%%%%%%.%%%%%%%%%%% %....%
%..     %  .........    %.%%%%
%.      %%%%%.    %j......   %
%o% %%%    % .%   %% %%.%%%%%%
%          % k%        .....i%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Computed path: See dots above

Solution cost: 143

Number of expanded nodes: 16206003

Medium:

The running time of medium maze is too long, and no longer to generate the solution path.

Computed path: N/A

Solution cost: N/A

Number of expanded nodes: N/A

Extra credit

The suboptimal search used is the weighted A* search, where the evaluation function is the cost plus the A* heuristic multiplied by a factor. The resulting searching heuristic is not admissible, but the efficiency greatly increases.

Specifically, the new heuristic function is five times the A* heuristic in part 1.2. The factor 5 is a result of manual testing. As the factor is incremented from 1, the efficiency increases greatly before reaching 5, and would change little as it goes beyond 5. The cost and number of nodes expanded is shown below.

Solution cost: 338

Number of expanded nodes: 1332

Part 2

Overall Structure

The structure for sokoban is similar to the maze, we continue to use the maze class built for part 1. The start and goal (points used to place boxes) points are firstly searched from the graph array and then we start the searching algorithm. The canTravel function (mentioned in part 1 overall structure) is modified, so that when no wall around or facing a box that is not beside wall or another box, we can travel to a certain direction. When changing the position, the status of the graph will also change (unlike the maze), since the boxes in the graph can be moved. Therefore for each iteration, the goal (bullet point) needs to be searched again. The whole process will be terminated when all of the bullet points disappear (turn into 'B').

Part 2.1

BFS plus A* with one heuristic

Algorithm description:

During the search, A list of status (currently undetected goals, path, explored array, cost,graph) will be pushed into queue to record the state of current node. The graph will be stored into the list since the status of the graph might be changed. To prevent from repetition, a 2-D array named *explored* with the same dimensions of the graph will be applied to denote whether a certain position has been detected. Once the status of the graph changes (boxes moves), the array will be refreshed.

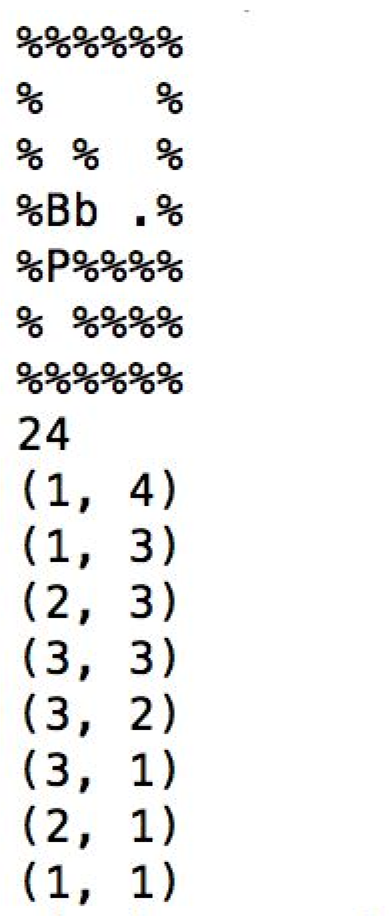
We use purely BFS to complete the code, the algorithm is not optimized, so only the first input has been generated. When we want to use the algorithm in part 1.2 for this part, the algorithm is not even admissible.

Input1:

Computed path: (1, 4) (1, 3) (2, 3) (3, 3) (3, 2) (3, 1) (2, 1) (1, 1)

Solution cost:8

Number of expanded nodes:24



```
%%%%%%
%    %
% %  %
%Bb .%
%P%%%%
% %%%%
%%%%%%
24
(1, 4)
(1, 3)
(2, 3)
(3, 3)
(3, 2)
(3, 1)
(2, 1)
(1, 1)
```

Extra credit

Animation:

When preparing the frames for the animation, we firstly store the portion of graph (space or bullet point) at position we have reached, and then replace it with character “a” to denote the current position. The modified graph then is used as the frame. Since the path has been stored, in the next state, the original portion will be restored and the graph will be change according to the path stored.

In the actual implementation, we use curse class to ensure the upper left corner will always be drawn on the top left of terminal. If a next step is been made, we will generate a new image of sokoban and overwrite the original picture. We use time library in python, so there will be 2 seconds break after every picture is generated so that every step can be seen.

Statement of individual contribution

Jiaying Wu

- Finished part 1.1 BFS search
- Worked on part 1.2 of finding heuristics
- Worked on animation of part 2

Qin Li

- Finished part 1.1 Astar and greedy search
- Worked on part 1.2 of finding heuristics
- Worked on algorithm of part 2

Yuchen Liang

- Finished part 1.1 DFS search
- Worked on part 1.2 A* heuristics and optimized the data structures from 1.1 A*
- Solved part 1 extra credit by using weighed A*