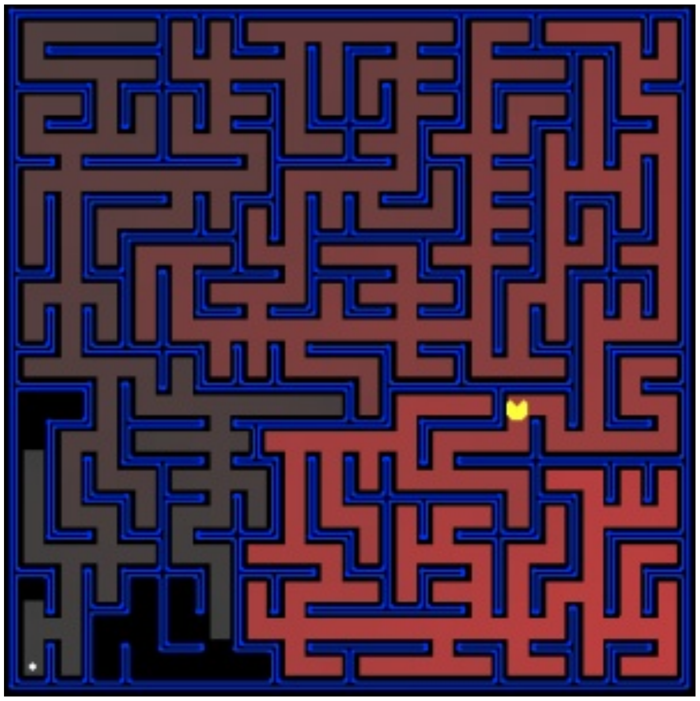


ASSIGNMENT 1 SEARCH – 4 CREDIT

Pacman Agent and Rubik’s Cube

ABSTRACT

In the first part of this assignment we solved the single and multiple goals search problem for Pacman agent within a decent amount of time by using delicate Heuristic functions. In the second part, we solved the 2x2x2 Rubik’s Cube using A* search with a well-designed hashing mechanism for repeated state detection.

You Wu, Qixin Zhu

CS 440 / ECE 448

Table of Contents

1-Basic Pathfinding

State representation: Each reachable location in the maze (aka not a wall) is represented as a node in the graph. For a certain location, if a neighbor location is also reachable (not a wall), then an undirected edge between these two nodes is added to the graph, of which the edge weight is 1. Eventually, the whole maze is represented by a graph that includes all reachable location and unit length edges, but exclude all wall locations. For each state in the frontier, it only has the information of its current location.

Transition model: Given a current node (location), by picking one of its neighbor edges, the Pacman is treated as moving along that edge and arriving at the destination node, which is the corresponding neighbor node.

Goal test (for the single goal problem): Check if the current Node is the same as the goal node. (Goal test for the multiple goal problem will be introduced later in section 2.)

1.1 Depth First Search

Due to the nature of DFS, the returned path is not guaranteed to be the shortest path, this is especially obvious in the Open Maze problem.

Medium Maze: the path cost is 80, the number of nodes expanded is 93, the total runtime is 1ms.

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

Big Maze: the path cost is 266, the number of nodes expanded is 484, the total runtime is 3ms.

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

Open Maze: the path cost is 270, the number of nodes expanded is 282, the total runtime is 0ms.

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

1.2 Breadth First Search

BFS is guaranteed to find the shortest path for these maze problems, because all edge weight is unit length.

Medium Maze: the path cost is 68, the number of nodes expanded is 339, the total runtime is 3ms.

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

Big Maze: the path cost is 266, the number of nodes expanded is 795, the total runtime is 3ms.

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

Open Maze: the path cost is 74, the number of nodes expanded is 573, the total runtime is 2ms.

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

1.3 Greedy Best First Search

Greedy algorithm is not guaranteed to find the shortest path. This greedy algorithm uses the Manhattan distance to determine which Node on the frontier should be expanded. As noticed, the runtime of Greedy algorithm is slightly larger than BFS or DFS. The reason is that this Greedy algorithm uses Priority Queue to maintain the frontier, which takes log(n) time to extract the front element of the queue, while BFS uses Queue and DFS uses Stack, both of which takes constant time to extract the next element.

Medium Maze: the path cost is 68, the number of nodes expanded is 76, the total runtime is 5ms.

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

Big Maze: the path cost is 266, the number of nodes expanded is 617, the total runtime is 7ms.

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

Open Maze: the path cost is 82, the number of nodes expanded is 347, the total runtime is 1ms.

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

1.4 A* Search

Manhattan distance is used in this problem to be the heuristic function.

Medium Maze: the path cost is 68, the number of nodes expanded is 184, the total runtime is 7ms.

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

Big Maze: the path cost is 266, the number of nodes expanded is 779, the total runtime is 14ms.

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

Open Maze: the path cost is 74, the number of nodes expanded is 390, the total runtime is 2ms.

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

2-Search with Multiple dots

State representation: The Graph composition is the same as before. For each state in the frontier, it not only contains its current location, but also knows which goals have been visited which have not. That is to say, for each location in the maze, there are 2^n different states, in which n is the number of goals.

Transition model: Basics are same as before. The difference is that, when reaching a goal, the current state will remember that information and pass to all its successor states.

Goal test (for the single goal problem): Check if the current state has visited all goals, the current location does not matter anymore.

2.1 Result Summary

At the very beginning, BFS is used to solve the Tiny Search problem and the results are used as a guideline to debug and verify the A* algorithm. The BFS algorithm for multiple dots problem is simply enumerating all possible visiting orders, hence the time complexity is O(n!), in which n is the number of goals. For comparison reason, the BFS results is also shown here: the path cost is 34, the number of nodes expanded is 10,699,779,821, the total runtime is about 75 minutes.

Tiny Search: the path cost is 34, the number of nodes expanded is 245, the total runtime is 54ms.

```
%%%%%%%%%
%2  %dc %
% % %%% %
% % 1   b%
% %%P%% %
%3 89  a%
% % %%%%%
%4 5  67%
%%%%%%%%%
```

Small Search: the path cost is 191, the number of nodes expanded is 6561, the total runtime is 1511ms.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      P      1%      b  c   %
% %%%%%%% %%%%%% %  % % %%   %
%   %2       % 9 %  % %  %d e%
%3           4   %a   %  %%% %
%%%%%%%%%%%%%%  %%%%%%%     f%
%65              7   8% %%%  %
%%%%%%%%%%%%%%%%%%%%%%% %   g%
%      l%         %     % %%%%
%       %% %%     %i    %   h%
%m% %%%    % %%   %%%%% %%%%%%
%            k%             j%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Medium Search: the path cost is 308, the number of nodes expanded is 1,871,262, the total runtime is 171,752ms.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      P      2%             %  a% c   %d%  g%k %
% %%%%%%% %%%%%% %  % % %%         %%%%  %  %%% %
%    %5%    % 7% %  % %  %   %   %       %  %   %
%   1%    6      %9   % %%%% %   %% e%%%%%  %   %
%%%%%%% %%%%%%  %%%%%%%          b%    %l %     %
%4          3%       8% %%%  %%%%%%  % %  % % % %
%%%%%%%%%%%%%%%%%%%%%%% %    %  n %% % %    %h% %
%    %  %      %  %     % %%%%       %f  %%%%i% %
%q      %%  %p %  %     %          %       m%   %
% % %%%    % %    %%%%% %%%%%% %  % %%%%%%%%%%% %
%   %    %    %             o%    %      % j    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

2.2 Heuristic Function

The heuristic function used to come up with the above solution is the maximum of the following two functions:

(1) Minimum Spanning Tree (MST) cost of all the unvisited goals + the minimum shortest path length from current location to any unvisited goals

Admissibility discussion: By definition, MST connected all unvisited goals with the minimum possible cost. In the best case, the MST connect all unvisited goals in one continuous path without any bifurcation, then the MST cost is the actual cost to visit all of them. In other cases, if there exists a bifurcation (one node has 3 or more connections in the MST), then the actual visiting path will go through either a duplicate edge in the MST or an edge not belonging to the MST. So the actual path length must be larger than the MST cost. The second term in the heuristic function means the minimum path length from the current location to "touch" (or connect) the MST. Thus, the summation of the two terms must be less than or equal to the actual path cost. **Admissibility is proved**.

Time complexity: Let e be the number of edges and v be the number of vertices in the complete graph for all unvisited goals, then e is $O(v^2)$. Considering the graph implementation uses adjacency list, the time complexity for calculating MST is $O(e*\log(v))$, aka $O(v^2*\log(v))$. (Note: this could have been reduced to $O(v^2)$ by using an adjacency matrix implementation.) Let n be the number of goals and k be the number of reachable location in the maze, then the time complexity of this whole A* search algorithm is: $\mathbf{O(n^2 * \log(n) * k * 2^n)}$.

(2) The shortest path length between the two furthest unvisited goals + the minimum shortest path length from current location to those two furthest unvisited goals

Admissibility discussion: To visit all the remaining goals, it takes at least the shortest path between the two furthest goals. The reach either one of those two, it takes at least the shortest path from current location to reach the closer one of those two. Thus **this heuristic is also admissible**. Note that only in very few extreme each cases, this heuristic will govern the MST method.

Time complexity: Assume finding shortest path between any two given nodes take constant time. (The implementation to achieve this is discussed in later section.) Let n be the number of goals, finding the two furthest unvisited goals takes $O(n^2)$ time, resulting in a total time complexity to be $\mathbf{O(n^2 * k * 2^n)}$. Thus when combined with MST, the overall time complexity does not change and will be governed by MST.

2.3 Evolution of Heuristic Function

The section summarizes the evolution of our heuristic function:

(0) As mentioned above, brute force BFS is listed here again for comparison reason: the number of nodes expanded to solve Tiny Search is **10,699,779,82**.

(1) The first naïve heuristic function is the minimum of Manhattan distance to all unvisited goals, which is admissible. The number of nodes expanded to solve Tiny Search is **249,559**.

(2) The second heuristic function is the minimum Manhattan distance + the number of unvisited goals. The thought behind this is that it will take at least "the number of goals" steps to visit them even they are right next to each other, thus the heuristic is still admissible. The number of nodes expanded to solve Tiny Search is **66,203**.

(3) The third heuristic function is to replace the Manhattan distance above with the actual shortest path length pre-computed by BFS, which is still admissible. The number of nodes expanded to solve Tiny Search is **58,496**.

(4) The forth heuristic function is the two furthest unvisited goals + distance to reach the closer one of those two. As explained in section 2.2, the heuristic is admissible. The number of nodes expanded to solve Tiny Search is **1,064**.

(5) The fifth heuristic function is the MST cost of unvisited goals + the minimum distance to connect the tree. As explained in section 2.2, the heuristic is admissible. The number of nodes expanded to solve Tiny Search is **255**.

(6) The final heuristic function is the maximum value of (4) and (5), as explained in section 2.2. The number of nodes expanded to solve Tiny Search is **245**.

2.4 Other non-trivial improvements

(1) Use “lastStep” variable to reduce frontier size

For each state, a “lastStep” variable is added to record the location where the current state comes from. This variable reduced the potential size of frontier by eliminating 1 successor of current state and also terminate the search at dead end immediately. Although this approach does not affect the actual number of nodes expanded, since the repeated state with higher cost is ignored, it affects the size of frontier and hence affect the time of extract element from the Priority Queue. The “lastStep” variable is cleaned and set to null when an unvisited goal is newly visited, because the remaining unvisited goals changed, and even we return to a previous location, the state will be different, so the new state should be allowed. This also allows the algorithm to go back after visiting a goal locating in a dead end.

(2) Pre-compute all single goal shortest path length and use look-up array in A* search

To achieve the constant time of knowing the shortest path length between any two nodes in the maze graph, a lookup matrix of k * n is pre-constructed, in which k is the number of all reachable nodes and n is the number of goals. The matrix is computed by running BFS at each node until all other nodes are visited. Thus, one run of BFS will find the shortest path length to all n goals.

(3) Pre-construct complete graph for all goals and speed up MST calculation

Similarly, to reduce the repetitive work of building a complete graph for the unvisited node, the graph was pre-constructed using the actual shortest path length, which is also pre-computed using BFS, as the edge weight between any two goals. Every time the heuristic need to be evaluated, the visited goals of that state is excluded from the MST computation.

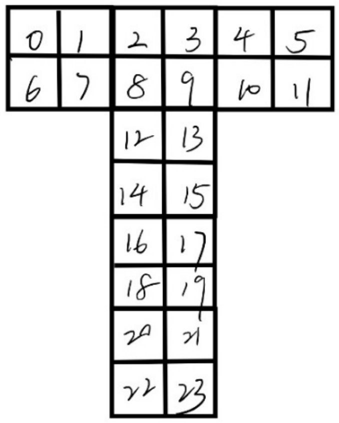
As stated in section 2.2, the time complexity of A* search for multiple goals problem is much more than the pre-computation. Thus, the above preparation work is trivial compare to the actual search algorithm.

2.4 Extra Credit Problem

To make admissible heuristic function non-admissible and speed up the search process, we could simply add a multiplier to the current heuristic function. In this case, after trial and error, we chose to multiply our heuristic in section 2.2 by 1.30. Due to the too many dots, a path representation as before does not work, so the maze figure is skipped here. The number of nodes expanded to solve the Big Dots is 4,745 and the suboptimum path length is **262**.

3-Rubik's Cube without Rotation Invariance

Implementation Design Highlight:
When we rotate 2*2*2 cube for one time, the number of the elements (one-color face contain four little faces, which is called element, therefore, 4*6=24 in total) that changed from their original positon is always 12. When we move one face, the 4 elements' position in that face is changed (we call faces element), and any elements attached to these four elements also changed their positions because they are contacted, namely,4*2=8(we call such 8 elements sides for specific face); therefore, 8+4=12 in total. Therefore, we numbered all the elements, which is universally suitable for all state.



Therefore, any rotate operation for ***element i***:
Face element: CurrentPosition [i+1*turn%4]= OriginalPosition [i]
Sides element: CurrentPosition [i+2*turn%8]= OriginalPosition [i]

Any counter rotate operation ***element i***:
Face element: CurrentPosition [i+3*turn%4]= OriginalPosition [i]
Sides element: CurrentPosition [i+6*turn%8]= OriginalPosition [i]

State representation: Each reachable cube state is represented as an array of characters. For a certain state, if a successor state is reachable in one step operation (12 choices in total, including clockwise/counterclockwise rotation for all 6 faces of the cube), then the successor state will be added to the frontier. For each state in the frontier, it has the information of its current cost *g* and heuristic value h.

Transition model: Given a current state, by picking one of its 12 possible operations, the Cube is treated as performing the chosen operation and reaching the next state.

Goal test (for no rotation invariance): Check if the current state has the same string representation of char-array as that of the goal state. (Goal test will be modified for the Rubik's Cube problem with Rotation Invariance in section 4)

3.1 Result Summary

As required by the instruction, we denote the moves by a single capital letter for the face turned (**T**op, **Bo**ttom, **F**ront, **Ba**ck, **L**eft, **R**ight) and a prime mark if the face was turned CCW instead of CW. The operation rules are defined as follows:

a. Use the character of one face for one time clockwise rotation of this face
b. Use the character of one face plus ' for one time counterclockwise rotation of this face.
c. When operate a specific cube, the operator must face the operating face to unify the rotation direction (It doesn't affect anything if computer is the case)

For example: Ba L' (clockwise rotate the BACK face once and counterclockwise rotate the LEFT face once)

Input 1.1: Number of expanded node is: 57. Runtime: 44ms

The path is: Ba F' L' T'.

From: To:

```
y o r r p b          rrbbgg
y g o o r b          rrbbgg
    b b                oo
    o g                oo
    g p                yy
    g p                yy
    p r                pp
    y y                pp
```

Input 1.2: Number of expanded node is: 628. Runtime: 83ms

The path is: F' L' L' T' F'.

From:

```
oygbpg
groryb
  yp
  og
  bp
  ry
  bp
  or
  or
```

To:

```
rrbbgg
rrbbgg
  oo
  oo
  yy
  yy
  pp
  pp
```

Input 1.3: Number of expanded node is: 4331. Runtime: 186ms

This is the path: F' T' R L T' R'

From:

```
oyogog
gprbrb
  yp
  og
  bp
  ry
  bp
  ry
```

To:

```
rrbbgg
rrbbgg
  oo
  oo
  yy
  yy
  pp
  pp
```

3.2 Heuristic Function

The heuristic function used is:

$$Heuristic(NowState) = \frac{1}{12}\sum_{i=1}^{24} Compare(NowState\_Char[i], GoalState\_Char[i])$$

In which, "Compare" is 1 if the character at location [i] of NowState is different from the character at location [i] of GoalState; "Compare" is 0 if those two characters are the same.

For Example:

If the NowState is char [ ]: {r,r,b,o,g,g,r,r,b,o,g,g,o,y,o,y,y,p,y,p,p,b,p,b}

The GoalState is char [ ]: {r,r,b,b,g,g,r,r,b,b,g,g,o,o,o,o,y,y,y,y,p,p,p,p}

Then heuristic value for NowState is 8/12=0.666667

Admissibility discussion: A universally acknowledged fact is: when we rotate 2*2*2 cube for only one time, the number of the elements (one big face contain four little faces, which is called element, therefore, 4*6=24 in total) that changed from their original positon is always 12(when we move one face, the 4 elements' position in that face is changed, and any elements attached to these four elements also changed their positions because they are contacted, namely,4*2=8; therefore, 8+4=12 in total). So if there exist any state that need only one step rotation to the Goal-State, the elements differences must be less than or equal to 12, obviously, the heuristic value calculated by the formula (elements differences divided by 12) is always less than or equal to 1, which means it will be always less than or equal to the actual path cost when it is 1. If the state now need more than 1 steps, because the maximum value of heuristic formula is always less than or equals to 2(equals only when all elements are different from the Goal-State), the value calculated by heuristic formula will be always less than or equal to the actual path cost that more than 1. Therefore, the heuristic value given by the formula is guaranteed to be less than or equal to actual path cost. **Admissibility is proved**.

3.3 Hashing Strategy

Without any repeated state detection, the time complexity of the Rubik's Cube problem is O(12^n). But there are only a limited number of total states for a 2x2x2 cube, which is 8! * 3^7 = 88 million (for this section, without rotation invariance). If we can eliminate all repeated states, the time complexity will be significantly reduced. Hence, in the A* search, we maintain a closed set by using the hash code of the string representation of each state.

4- Rubik’s Cube with Rotation Invariance

State representation: In this case we have a repeated state detector for any given state A and its 23 3D equivalent pseudo states. So one state at this stage includes all 24 possible 3D rigid body rotation of the same Rubik’s Cube.

Transition model: Same as before.

Goal test (including rotation invariance): The principle is still to check if the current state is the same as the goal state. Here one state in defined to include all 24 pseudo states (with 3D rigid body rotation) of the given Rubik’s Cube.

4.1 Result Summary

Input 2.1: Number of expanded node is: 5. Runtime: 157ms

The path is: T' R'

```
From:                                   To:
        yorrpb                                  yyrrbb
        ygoorb                                  yyrrbb
          bb                                      oo
          og                                      oo
          gp                                      gg
          gp                                      gg
          pr                                      pp
          yy                                      pp
```

Since the Input 2.1 is the same as the Input 1.1, we can see major change after introducing the repeated state detector. The path steps reduced from 4 steps to 2 steps, and the node explored reduced from 57 to only 5. This means the repetition caused by 3D rotation exists in the solution path in Section 3.

Input 2.2: Number of expanded node is: 1834. Runtime: 10360ms

The path is: F' T' L' L' T' Ba'

```
From:                         To:
      opgbob                        ggyyrr
      byobrr                        ggyyrr
        gp                            oo
        op                            oo
        ry                            bb
        yg                            bb
        rp                            pp
        yg                            pp
```

Input 2.3: Number of expanded node is: 374. Runtime: 2406ms

The path is: Ba' Bo' L L Bo'

```
From:                         To:
      ybopyy                        ppbboo
      rgybog                        ppbboo
        or                            rr
        pp                            rr
        bb                            yy
        go                            yy
        pr                            gg
        gr                            gg
```

4.2 Heuristic Function

The heuristic function shares the same concept as in Section 3.2. The only difference is that, with rotation invariance, all 24 pseudo-states (one state with 24 3D rotations) are recognized as one state. So the heuristic value for a given state is calculated the same way as in Section 3.2, but repeated for all 24 pseudo-state of the goal. The returned heuristic value is the minimum of those 24 values, hence **the admissibility is maintained**.

4.3 Hashing Strategy

To facilitate the repeated state detection, the new hashing strategy should be able to return the same hash code for 24 pseudo states of 1 real state. To achieve this goal, all 24 string representation of those pseudo states are generated and sorted. The lexicographical order of those 24 strings are unique, hence the first element of the sorted strings is unique. That first string's hash code is returned as the hash code of the given state no matter which pseudo state it is in.

4.4 Extra Credit Problem

Our heuristic function is good enough to handle any 7-step puzzles within 30 seconds, and here is the result of the given Input.

Input 3.1:
```
Number of expanded node is: 4433. Runtime: 25265ms
The path is: F R' Bo F T' R Ba
```
```
From:                                   To:
         goggop                                  bbggyy
         yyrprb                                  bbggyy
           oy                                      oo
           gr                                      oo
           po                                      rr
           br                                      rr
           pb                                      pp
           by                                      pp
```

5- Statement of individual contribution

Both team members participate and interact in programming, debugging, optimizing algorithm, discussing high-level ideas and designing heuristic functions. You Wu is in mainly charge of coding for Rubik's Cube problem. Qixin Zhu is mainly in charge of coding for Pacman Agent problem.

6-References

[1] https://en.wikipedia.org/wiki/Travelling_salesman_problem

[2] http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

[3] https://courses.edx.org/courses/BerkeleyX/CS188x_1/1T2013/info

[4] https://heuristicswiki.wikispaces.com/Rubik%27s+cube

[5] http://www.cube20.org/

[6] https://en.wikipedia.org/wiki/Pocket_Cube

[7] https://ruwix.com/the-rubiks-cube/gods-number/