```
function path = A_star_search(map,MAX_X,MAX_Y)
```

```
%This part is about map/obstacle/and other settings
    %pre-process the grid map, add offset
    size_map = size(map,1);
    Y_offset = 0;
    X_offset = 0;

    %Define the 2D grid map array.
    %Obstacle=-1, Target = 0, Start=1
    MAP=2*(ones(MAX_X,MAX_Y));

    %Initialize MAP with location of the target 最后一个是target
    xval=floor(map(size_map, 1)) + X_offset;
    yval=floor(map(size_map, 2)) + Y_offset;
    xTarget=xval;
    yTarget=yval;
    MAP(xval,yval)=0;

    %Initialize MAP with location of the obstacle 中间的是障碍物
    for i = 2: size_map-1
        xval=floor(map(i, 1)) + X_offset;
        yval=floor(map(i, 2)) + Y_offset;
        MAP(xval,yval)=-1;
    end

    %Initialize MAP with location of the start point 最后一个是起点
    xval=floor(map(1, 1)) + X_offset;
    yval=floor(map(1, 2)) + Y_offset;
    xStart=xval;
    yStart=yval;
    MAP(xval,yval)=1;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %LISTS USED FOR ALGORITHM
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %OPEN LIST STRUCTURE
    %--------------------------------------------------------------------------
    %IS ON LIST 1/0 |X val |Y val |Parent X val |Parent Y val |h(n) |g(n)|f(n)|
    %--------------------------------------------------------------------------
    OPEN=[];
    %CLOSED LIST STRUCTURE
    %--------------
    %X val | Y val |
    %--------------
    % CLOSED=zeros(MAX_VAL,2);
    CLOSED=[];

    %Put all obstacles on the Closed list
    k=1;%Dummy counter
    for i=1:MAX_X
        for j=1:MAX_Y
            if(MAP(i,j) == -1)
                CLOSED(k,1)=i;
                CLOSED(k,2)=j;
                k=k+1;
            end
        end
    end
    CLOSED_COUNT=size(CLOSED,1);
    %set the starting node as the first node
    xNode=xval;
    yNode=yval;
    OPEN_COUNT=1;
    goal_distance=distance(xNode,yNode,xTarget,yTarget);
    path_cost=0;
    OPEN(OPEN_COUNT,:)=insert_open(xNode,yNode,xNode,yNode,goal_distance,path_cost,goal_distance);
    OPEN(OPEN_COUNT,1)=1;
    CLOSED_COUNT=CLOSED_COUNT+1;
    CLOSED(CLOSED_COUNT,1)=xNode;
    CLOSED(CLOSED_COUNT,2)=yNode;
    NoPath=1;



    STORRING_OPEN = [];
    STORRING_OPEN_CNT = 1;
    get_xNode_Open = 2;
    get_yNode_Open = 3;
    get_parent_xNode_Open = 4;
    get_parent_yNode_Open = 5;
    get_h_Open = 6;
    get_g_Open = 7;
    get_f_OPen = 8;

    get_scanning_xNode_expand = 1;
    get_scanning_yNode_expand = 2;
    get_scanning_h_expand = 3;
    get_scanning_g_expand = 4;
    get_scanning_f_expand = 5;
```

```
输入参数的数目不足。

出错 A_star_search_2 (第 5 行)
    size_map = size(map,1);
```

```
%This part is your homework
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% START ALGORITHM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    path_1 = [xStart,yStart];
    Path_Cnt = 1;



    while(~isempty(OPEN)) %you have to dicide the Conditions for while loop exit finish the while loop
       row = size(OPEN,1);
       i_min = min_fn(OPEN,row,xTarget,yTarget);
       if i_min == -1
           path = [];
           fprintf("No road to go\n")
           return
       end



       % Getting the OPEN array which is the smallest in the Whole Open
       % List
       current_smallest_OPEN_Node = OPEN(i_min,:);                       % Array
       current_x = current_smallest_OPEN_Node(get_xNode_Open);           % Current Node X
       current_y = current_smallest_OPEN_Node(get_yNode_Open);           % Current Node Y
       parent_x  = current_smallest_OPEN_Node(get_parent_xNode_Open);    % Current Parent X of Node X
       parent_y  = current_smallest_OPEN_Node(get_parent_yNode_Open);    % Current Parent Y of Node Y
       current_h = current_smallest_OPEN_Node(get_h_Open);               % Current Euler distance
       current_g = current_smallest_OPEN_Node(get_g_Open);               % Current Cost
       current_f = current_smallest_OPEN_Node(get_f_OPen);               % Current Heuristic Number

       % Storing All the NODE with smallest f
       STORRING_OPEN(STORRING_OPEN_CNT,:) = OPEN(i_min,:);
       STORRING_OPEN_CNT = STORRING_OPEN_CNT + 1;
       STORRING_OPEN;



       % Remove the Current Node
       OPEN(i_min,:) = [];
       CLOSED_COUNT = CLOSED_COUNT + 1;
       CLOSED(CLOSED_COUNT,1) = current_x;
       CLOSED(CLOSED_COUNT,2) = current_y;
       CLOSED_COUNT = CLOSED_COUNT + 1;
       CLOSED(CLOSED_COUNT,1) = parent_x;
       CLOSED(CLOSED_COUNT,2) = parent_y;






       % Storring Path
       % scanning the destination
       if (current_x == xTarget && current_y == yTarget)
           Path_Cnt = Path_Cnt + 1;
           path_1(Path_Cnt,:) = [current_x,current_y];

           node_current = [current_x,current_y];     %storring end point
           start_node   = [xStart,yStart];           %storring start point
```

```matlab
        parent_node_in_store  = [STORRING_OPEN(:,get_parent_xNode_Open),STORRING_OPEN(:,get_parent_yNode_Open)];
        current_node_in_store = [STORRING_OPEN(:,get_xNode_Open),STORRING_OPEN(:,get_yNode_Open)];

        path = [current_x,current_y];
        path_cnt_out = 1;

        while(~isequal(node_current,start_node))
            location_in_store = ismember(current_node_in_store,node_current,"rows");
            [rowIndex_in_store,~] = find(location_in_store);
            node_current = [STORRING_OPEN(rowIndex_in_store,get_parent_xNode_Open),STORRING_OPEN(rowIndex_in_store,get_parent_yNode_Open)];
            path_cnt_out = path_cnt_out + 1;
            path(path_cnt_out,:) = node_current
        end

        fprintf("SUCCESS\n")
        return
    end
    path_x = current_x;
    path_y = current_y;
    path_node = [path_x,path_y];
    if(~any(ismember(path_1,path_node,"rows")))
        Path_Cnt = Path_Cnt + 1;
        path_1(Path_Cnt,:) = [path_x,path_y];
    end

    %fprintf("path is %0.2f and yNode is %0.2f.\n",path(Path_Cnt,1),path(Path_Cnt,2))




    % Expand the current Open Array from current_x and current_y
    expand_array_list = expand_array(current_x,current_y,current_g,xTarget,yTarget,CLOSED,MAX_X,MAX_Y);

    % Looping for all expand_array_list
    for index_expand_array_list = 1:1:size(expand_array_list,1)
        scanning_node = expand_array_list(index_expand_array_list,:);
        scanning_x = scanning_node(get_scanning_xNode_expand);
        scanning_y = scanning_node(get_scanning_yNode_expand);
        scanning_h = scanning_node(get_scanning_h_expand);
        scanning_g = scanning_node(get_scanning_g_expand);
        scanning_f = scanning_node(get_scanning_f_expand);

        % Check if scanning node already in the OPEN List
        scanning_node_location      = [scanning_x,scanning_y];
        OPEN_List_ALL_Node_Location = [OPEN(:,get_xNode_Open) OPEN(:,get_yNode_Open)];
        location = ismember(OPEN_List_ALL_Node_Location,scanning_node_location,"rows");

        % get the row of it
        [rowIndex_SameNode,~] = find(location);
        OPEN_In_List_xNode = OPEN(rowIndex_SameNode,get_xNode_Open);
        OPEN_In_List_yNode = OPEN(rowIndex_SameNode,get_yNode_Open);


        %fprintf("Scanning_g is %0.2f and Current_g is %0.2f\n",scanning_g,current_g);
        fprintf("I am starting scanning from %0.2f and %0.2f.\n",current_x,current_y);
        fprintf("The scanning node is %0.2f and %0.2f\n",scanning_x,scanning_y);
        fprintf("The g used to in list is %0.2f and %0.2f. \n",OPEN_In_List_xNode,OPEN_In_List_yNode);



        % If We have not check this node
        if (~any(ismember(OPEN_List_ALL_Node_Location,scanning_node_location,"rows")))

            fprintf("Did not find same node here\n")
            OPEN_COUNT = OPEN_COUNT + 1;
            OPEN(OPEN_COUNT,:) = insert_open(scanning_x, ...
                                             scanning_y, ...
                                             current_x, ...
                                             current_y, ...
                                             scanning_h, ...
                                             scanning_g, ...
                                             scanning_f);

            %fprintf("h is %0.2f and g is %0.2f and f is %0.2f. \n",current_h,current_g,current_f)
            %fprintf("new h is %0.2f and new g is %0.2f and new f is %0.2f in New Node. \n\n\n\n",scanning_h,scanning_g,scanning_f)
```

```
           elseif (scanning_g <= OPEN(rowIndex_SameNode,get_g_Open))
               % finding the same node location
               %fprintf("Find the Same Node here. \n");

               location = ismember(OPEN_List_ALL_Node_Location,scanning_node_location,"rows");
               % get the row of it
               [rowIndex_SameNode,~] = find(location);
               % OPEN_In_List_xNode = OPEN(rowIndex_SameNode,get_xNode_Open);
               % OPEN_In_List_yNode = OPEN(rowIndex_SameNode,get_yNode_Open);
               % OPEN_IN_List_g = OPEN(rowIndex_SameNode,get_g_Open);
               % OPEN_IN_List_f = OPEN(rowIndex_SameNode,get_f_OPen);
               %fprintf("Old h is %0.2f, Old g is %0.2f and Old f is %0.2f. \n",current_h,current_g,current_f)
               %fprintf("New h is %0.2f, New g is %0.2f and New f is %0.2f. \n\n\n\n",scanning_h,scanning_g,scanning_f)
               %Update the newest cost and heutic value
               OPEN(rowIndex_SameNode,get_h_Open) = scanning_h;
               OPEN(rowIndex_SameNode,get_g_Open) = scanning_g;
               OPEN(rowIndex_SameNode,get_f_OPen) = scanning_h + scanning_g;
               continue;

           else
               path = [];
               fprintf("No Road to Go.\n")
           end
        end



    end %End of While Loop
    path = [];
```

```
    end


        %Once algorithm has run The optimal path is generated by starting of at the
        %last node(if it is the target node) and then identifying its parent node
        %until it reaches the start node.This is the optimal path

        %
        %How to get the optimal path after A_star search?
        % 首先将OPEN List 和 Close List 初始化 在第一个for loop中:
            % 首先寻找最小的 f 点
            % 记录并删除此点
            % 将这个点展开寻找
            % 用for loop把所有点经过一次
                %如果此点没去过记录并更新
                %如果词典去过保留更好路径
            % 找到终点后回溯
        %please finish it
        %
```

Published with MATLAB® R2023a