

AutoRob

Introduction to Autonomous Robotics  
Michigan EECS 367

Robot Kinematics and Dynamics  
Michigan ME 567 EECS 567 ROB 510

Fall 2019

# EECS 367 Lab: Assignment 1 Code Overview

# Administrative

- Next **Wednesday, September 18**
  - Quiz 1 during regular lecture time and location
  - Assignment #1: Path Planning
    - Due 11:59pm, Wednesday, September 18
- Send Professor Jenkins your Git account/repo (through slack)

# Lab Takeaways

1) Stencil  
Overview

3) Validate  
Implementation

5) Data Structure  
Considerations

1

2

3

4

5

2) Walkthrough  
Heap Insert

4) Search Canvas  
Introduction

How to start  
assignment 1

# Assignment 1

Points	Sections	Feature
Assignment 1: 2D Path Planning		
4	All	Heap implementation
8	All	A-star search
2	Grad	BFS
2	Grad	DFS
2	Grad	Greedy best-first

# KinEval Stencil

The screenshot shows a GitHub repository page for 'AutoRobStudent / KinEval'. The repository is private. The commit history is as follows:

Commit	Type	Date
Student and Student initial commit	Initial commit	7 days ago
js	Initial commit	7 days ago
kineval	Initial commit	7 days ago
project_pathplan	Initial commit	7 days ago
project_pendulum	Initial commit	7 days ago
robots	Initial commit	7 days ago
tutorial_heapsort	Initial commit	7 days ago
tutorial_js	Initial commit	7 days ago
worlds	Initial commit	7 days ago
LICENSE	Initial commit	7 days ago
README.md	Initial commit	7 days ago
home.html	Initial commit	7 days ago

A red box highlights the commits for 'project\_pathplan' and 'tutorial\_heapsort'. A red arrow points from a callout box labeled 'Bulk of code needed to complete Assignment 1' to these highlighted commits. Another red arrow points from a callout box labeled 'Starter code to help debug heap' to the same two commits.

# KinEval Stencil

AutoRobStudent / KinEval Private

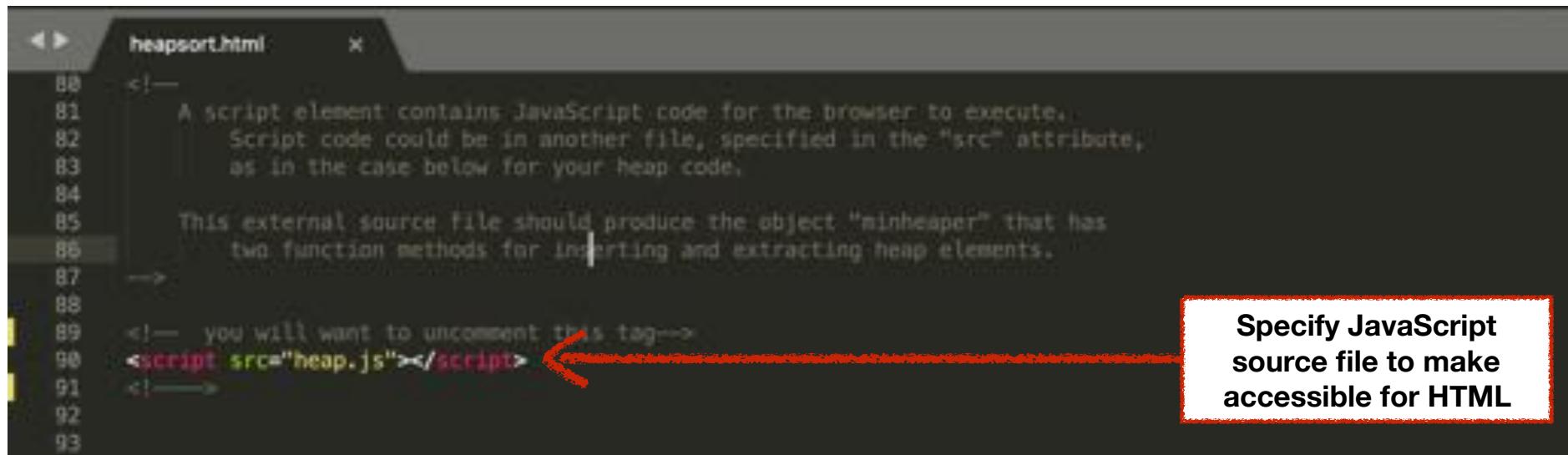
Unwatch 2 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Security Insights Settings

Branch: master KinEval / tutorial\_heapsort / Create new file Upload files Find file History

		Latest commit f5e3944 7 days ago
Student and Student	Initial commit	
→		
heap.js	Initial commit	7 days ago
heapsort.html	Initial commit	7 days ago

# heapsort.html



```
heapsort.html *  
80 <!--  
81     A script element contains JavaScript code for the browser to execute.  
82     Script code could be in another file, specified in the "src" attribute,  
83     as in the case below for your heap code.  
84  
85     This external source file should produce the object "minheaper" that has  
86     two function methods for inserting and extracting heap elements.  
87 -->  
88  
89 <!-- you will want to uncomment this tag-->  
90 <script src="heap.js"></script>  
91 <!-->  
92  
93
```

Specify JavaScript source file to make accessible for HTML

# heapsort.html

```
heapsort.html      x  
167 console.log("building min binary heap from number array");  
168 numbers_heap = []; // create array for heap ←  
169 for (i=0;i<numbers.length;i++) {  
170  
171     console.log("inserting number "+numbers[i]+" into the heap");  
172     minheaper.insert(numbers_heap,numbers[i]); ←  
173  
174     console.log("appending current heap state to output object");  
175     output_string = "heap (insert " + numbers[i] + "): "; //  
176     for (j=0;j<numbers_heap.length;j++) {  
177         output_string += numbers_heap[j] + " "; //  
178     }  
179     addHTMLLine("output",output_string); ←  
180 }  
181  
182 console.log("before sorting, crudely draw current heap state to canvas");  
183 // grab canvas element and its drawing context  
184 console.log("obtaining reference to canvas object in the document");  
185 c = document.getElementById("myCanvas");  
186 console.log("obtaining the drawing context of the canvas object");  
187 ctx = c.getContext("2d");  
188
```

Represent heap as JS array

Repeatedly call heap insert method

Print state of heap to screen

# heapsort.html

## Without heap.js implementation



[My Heap Sort](#)

```
check
numbers to sort: 4055 8917 6224 8831 7815 9098 7526 3088 7537 7958 9402 4169 4304 3771 8151 2545 289 2486 6758 8685
my heaping functions are not yet implemented
```

## With heap.js implementation



[My Heap Sort](#)

```
2698      504
4275      3515      1559
5664      4020      6512      3138      4500
6585      6073      8241      7735      4752
```

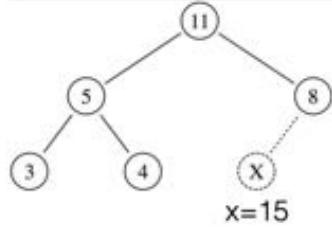
```
check
numbers to sort: 3138 4020 2698 6073 1559 504 4752 6585 8241 4747 6512 4361 5817 9333 4500 5664 4275 7735 4872 3515
heap (insert 3138): 3138
heap (insert 4020): 3138 4020
heap (insert 2698): 2698 4020 3138
heap (insert 6073): 2698 4020 3138 6073
heap (insert 1559): 1559 2698 3138 6073 4020
heap (insert 504): 504 2698 1559 6073 4020 3138
heap (insert 4752): 504 2698 1559 6073 4020 3138 4752
heap (insert 6585): 504 2698 1559 6073 4020 3138 4752 6585
heap (insert 8241): 504 2698 1559 6073 4020 3138 4752 6585 8241
heap (insert 4747): 504 2698 1559 6073 4020 3138 4752 6585 8241 4747
heap (insert 6512): 504 2698 1559 6073 4020 3138 4752 6585 8241 4747 6512
heap (insert 4361): 504 2698 1559 6073 4020 3138 4752 6585 8241 4747 6512 4361
heap (insert 5817): 504 2698 1559 6073 4020 3138 4752 6585 8241 4747 6512 4361 5817
heap (insert 9333): 504 2698 1559 6073 4020 3138 4752 6585 8241 4747 6512 4361 5817 9333
heap (insert 4500): 504 2698 1559 6073 4020 3138 4500 6585 8241 4747 6512 4361 5817 9333 4752
heap (insert 5664): 504 2698 1559 5664 4020 3138 4500 6073 8241 4747 6512 4361 5817 9333 4752 6585
heap (insert 4275): 504 2698 1559 5664 4020 3138 4500 6073 8241 4747 6512 4361 5817 9333 4752 6585 6073
heap (insert 3515): 504 2698 1559 4275 4020 3138 4500 6073 8241 4747 6512 4361 5817 9333 4752 6585 6073 8241
heap (insert 4752): 504 2698 1559 4275 3515 4020 3138 4500 6073 8241 4747 6512 4361 5817 9333 4752 6585 6073 8241 7735
heap (extract 904): 1559 2698 3138 4275 3515 4020 3138 4500 6073 8241 4747 6512 4361 5817 9333 4752 6585 6073 8241 7735 4747
heap (extract 1559): 2698 3515 3138 4275 3515 4020 3138 4500 6073 8241 4747 6512 4361 5817 9333 4752 6585 6073 8241 7735
heap (extract 2698): 3138 3515 3138 4275 3515 4020 3138 4500 6073 8241 4747 6512 4361 5817 9333 4752 6585 6073 8241 7735
heap (extract 1138): 3515 4020 4361 4275 6073 4747 4500 5664 6872 4020 6512 4747 5817 9333 4752 6585 6073 8241 7735
heap (extract 3515): 4020 4275 4361 4872 6073 4747 4500 5664 6872 4020 6512 4747 5817 9333 4752 6585 6073 8241 7735
heap (extract 4275): 4361 4752 4500 4872 6073 4747 4500 5664 6872 4020 6512 4747 5817 9333 4752 6585 6073 8241 5817 9333
heap (extract 4361): 4500 4752 4747 4872 6073 5817 9333 5664 6872 7735 6512 4747 5817 9333 4752 6585 7735 6512
heap (extract 4500): 4747 4752 5817 9333 6512 6073 8241 9333 5664 6872 7735 6512 4747 5817 9333 4752 6585 7735
heap (extract 4747): 4752 4872 5817 5664 6872 7735 6512 6073 8241 9333 7735 6585
heap (extract 4872): 5664 6073 5817 6512 6872 8241 9333 7735
heap (extract 5664): 5817 6073 7735 6512 6872 8241 9333
heap (extract 5817): 6073 6512 7735 9333 6872 8241
heap (extract 6073): 6512 6872 7735 9333 8241
heap (extract 6512): 6872 7735 8241 9333
heap (extract 6585): 7735 8241 9333
heap (extract 8241): 9333
heap (extract 9333):
```

sorted: 504 1559 2698 3138 3515 4020 4275 4361 4500 4747 4752 4872 5664 5817 6073 6512 6872 7735 8241 9333

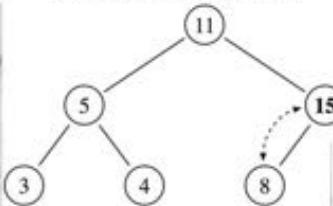
# Heap Insert

## Heap operations: Insert

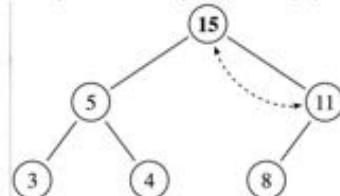
1) add new element to end of tree



2) swap with parent



3) until heaped, do (2)



For priority queue, previously non-queued locations will be inserted with f\_score priority

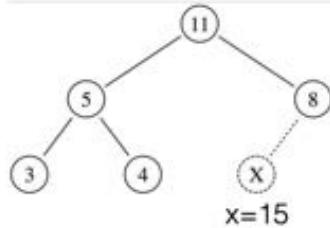
Michigan Robotics 367/510/567 - autorob.org

```
// define insert function for min binary heap
function minheap_insert(heap, new_element) {
    // TODO: Find index for new_element
    // TODO: Find index of new_element's parent
    // TODO: Add new_element to the heap array
    // TODO: Initialize heap condition check
    // TODO: As long as heap condition not satisfied
    //     // Swap new_element with parent
    //     // ...
    //     // Update index for new_element
    //     // Update index for new_element's parent
    // ...
}
```

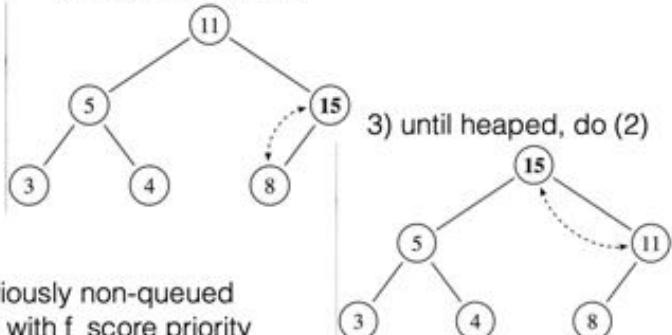
# Heap Insert

## Heap operations: Insert

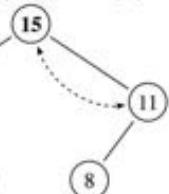
1) add new element to end of tree



2) swap with parent



3) until heaped, do (2)



For priority queue, previously non-queued locations will be inserted with f\_score priority

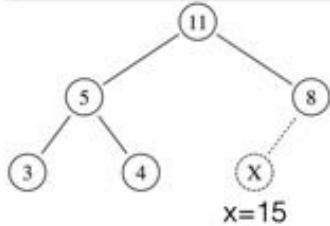
Michigan Robotics 367/510/567 - autorob.org

```
// define insert function for min binary heap
function minheap_insert(heap, new_element) {
    var elntIdx = heap.length;
    // TODO: Find index of new_element's parent
    // TODO: Add new_element to the heap array
    // TODO: Initialize heap condition check
    // TODO: As long as heap condition not satisfied
    // // TODO: Swap new_element with parent
    // // TODO: Update index for new_element
    // // TODO: Update index for new_element's parent
    // // TODO: Update heap condition check
}
```

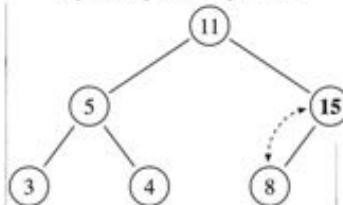
# Heap Insert

## Heap operations: Insert

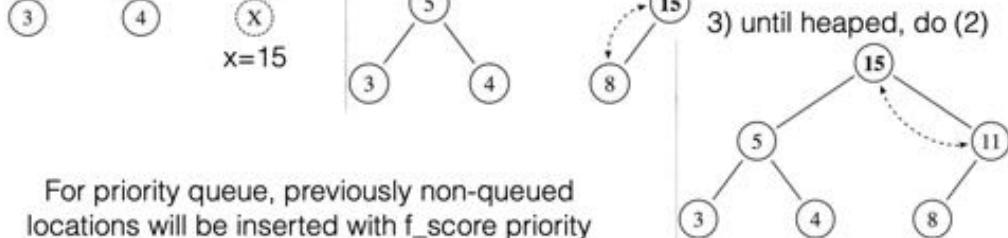
1) add new element to end of tree



2) swap with parent



3) until heaped, do (2)



For priority queue, previously non-queued locations will be inserted with f\_score priority

Michigan Robotics 367/510/567 - autorob.org

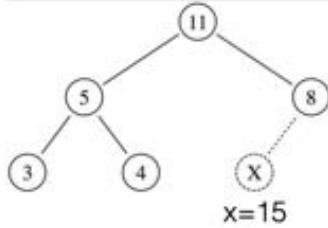
```
// define insert function for min binary heap
function minheap_insert(heap, new_element) {
    var elntIdx = heap.length;
    var prntIdx = Math.floor( (elntIdx - 1) / 2 );

    // TODO: Add new_element to the heap array
    // TODO: Initialize heap condition check
    // TODO: As long as heap condition not satisfied
    //       Swap new_element with parent
    //       Update index for new_element
    //       Update index for new_element's parent
    //       Update heap condition check
}
```

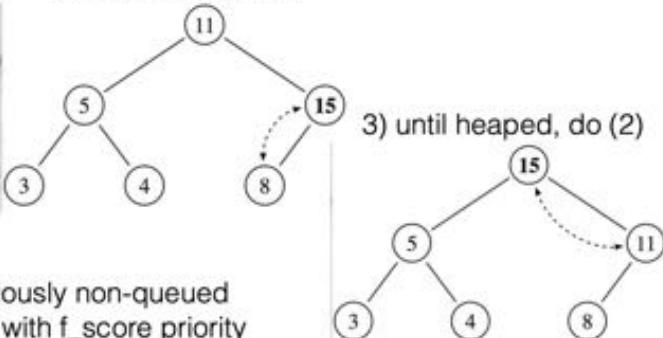
# Heap Insert

## Heap operations: Insert

1) add new element to end of tree



2) swap with parent



For priority queue, previously non-queued locations will be inserted with f\_score priority

Michigan Robotics 367/510/567 - autorob.org

```
// define insert function for min binary heap
function minheap_insert(heap, new_element) {
    var elntIdx = heap.length;
    var prntIdx = Math.floor( (elntIdx - 1) / 2 );

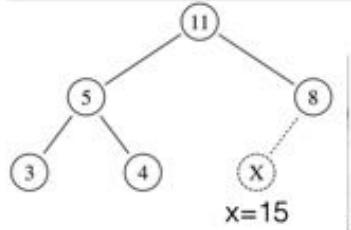
    heap.push(new_element);

    // TODO: Initialize heap condition check
    // TODO: As long as heap condition not satisfied
    //       Swap new_element with parent
    //       ...
    //       ...
    //       Update index for new_element
    //       Update index for new_element's parent
    //       ...
    //       ...
    //       Update heap condition check
}
```

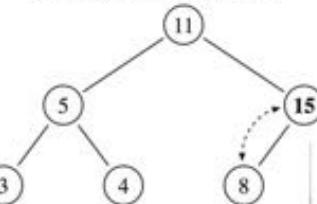
# Heap Insert

## Heap operations: Insert

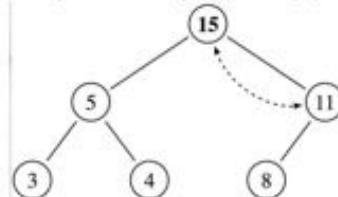
1) add new element to end of tree



2) swap with parent



3) until heaped, do (2)



For priority queue, previously non-queued locations will be inserted with f\_score priority

Michigan Robotics 367/510/567 - autorob.org

```
// define insert function for min binary heap
function minheap_insert(heap, new_element) {
    var elntIdx = heap.length;
    var prntIdx = Math.floor( (elntIdx - 1) / 2 );

    heap.push(new_element);

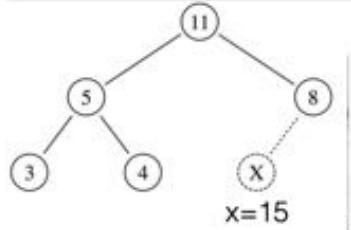
    // Heap condition is true if new element added as root, or if
    // new element is less than or equal to its parent element
    var heaped = (elntIdx <= 0) || (heap[prntIdx] >= heap[elntIdx]);

    // TODO: As long as heap condition not satisfied
    // TODO: Swap new_element with parent
    //
    // TODO: Update index for new_element
    // TODO: Update index for new_element's parent
    //
    // TODO: Update heap condition check
}
```

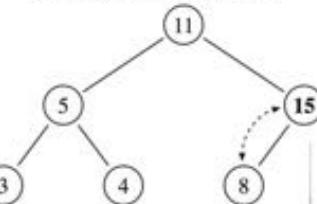
# Heap Insert

## Heap operations: Insert

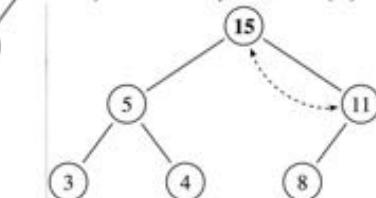
1) add new element to end of tree



2) swap with parent



3) until heaped, do (2)



For priority queue, previously non-queued locations will be inserted with f\_score priority

Michigan Robotics 367/510/567 - autorob.org

```
// define insert function for min binary heap
function minheap_insert(heap, new_element) {
    var elntIdx = heap.length;
    var prntIdx = Math.floor( (elntIdx - 1) / 2);

    heap.push(new_element);

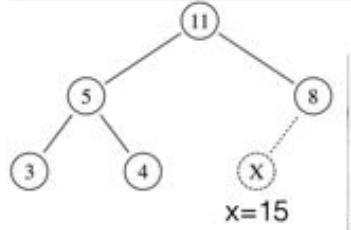
    // Heap condition is true if new element added as root, or if
    // new element is less than or equal to its parent element
    var heaped = (elntIdx <= 0) || (heap[prntIdx] >= heap[elntIdx]);

    while (!heaped) {
        // TODO: Swap new_element with parent
        //
        //
        // TODO: Update index for new_element
        // TODO: Update index for new_element's parent
        //
        //
        // TODO: Update heap condition check
    }
}
```

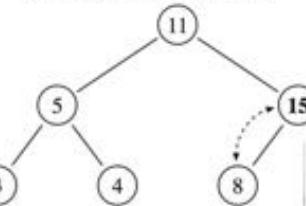
# Heap Insert

## Heap operations: Insert

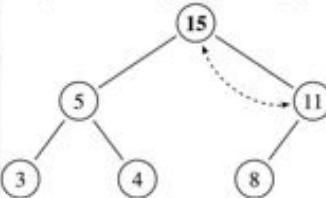
1) add new element to end of tree



2) swap with parent



3) until heaped, do (2)



For priority queue, previously non-queued locations will be inserted with f\_score priority

Michigan Robotics 367/510/567 - autorob.org

```

// define insert function for min binary heap
function minheap_insert(heap, new_element) {
    var elntIdx = heap.length;
    var prntIdx = Math.floor( (elntIdx - 1) / 2 );

    heap.push(new_element);

    // Heap condition is true if new element added as root, or if
    // new element is less than or equal to its parent element
    var heaped = (elntIdx <= 0) || (heap[prntIdx] >= heap[elntIdx]);

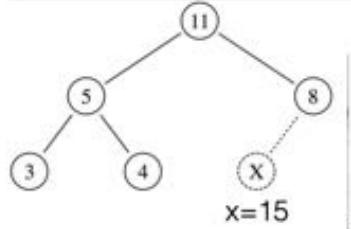
    while (!heaped) {
        // Swap element and parent
        var tmp = heap[prntIdx];
        heap[prntIdx] = heap[elntIdx];
        heap[elntIdx] = tmp;

        // TODO: Update index for new_element
        // TODO: Update index for new_element's parent
        // ...
        // TODO: Update heap condition check
    }
}
  
```

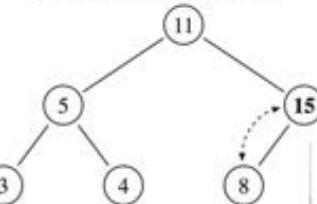
# Heap Insert

## Heap operations: Insert

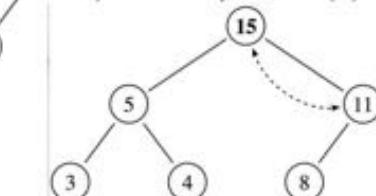
1) add new element to end of tree



2) swap with parent



3) until heaped, do (2)



For priority queue, previously non-queued locations will be inserted with f\_score priority

Michigan Robotics 367/510/567 - autorob.org

```
// define insert function for min binary heap
function minheap_insert(heap, new_element) {
    var elntIdx = heap.length;
    var prntIdx = Math.floor( (elntIdx - 1) / 2 );

    heap.push(new_element);

    // Heap condition is true if new element added as root, or if
    // new element is less than or equal to its parent element
    var heaped = (elntIdx <= 0) || (heap[prntIdx] >= heap[elntIdx]);

    while (!heaped) {
        // Swap element and parent
        var tmp = heap[prntIdx];
        heap[prntIdx] = heap[elntIdx];
        heap[elntIdx] = tmp;

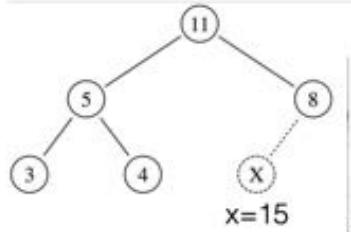
        // Update element and parent index
        elntIdx = prntIdx;
        prntIdx = Math.floor( (elntIdx - 1 ) / 2 );

        // TODO: Update heap condition check
    }
}
```

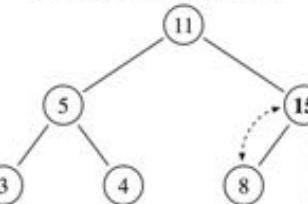
# Heap Insert

## Heap operations: Insert

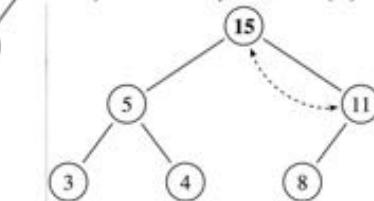
1) add new element to end of tree



2) swap with parent



3) until heaped, do (2)



For priority queue, previously non-queued locations will be inserted with f\_score priority

Michigan Robotics 367/510/567 - autorob.org

```
// define insert function for min binary heap
function minheap_insert(heap, new_element) {
    var elntIdx = heap.length;
    var prntIdx = Math.floor( (elntIdx - 1) / 2 );

    heap.push(new_element);

    // Heap condition is true if new element added as root, or if
    // new element is less than or equal to its parent element
    var heaped = (elntIdx == 0) || (heap[prntIdx] >= heap[elntIdx]);

    while (!heaped) {
        // Swap element and parent
        var tmp = heap[prntIdx];
        heap[prntIdx] = heap[elntIdx];
        heap[elntIdx] = tmp;

        // Update element and parent index
        elntIdx = prntIdx;
        prntIdx = Math.floor( (elntIdx - 1) / 2 );

        // Re-evaluate heap condition
        heaped = (elntIdx == 0) || (heap[prntIdx] >= heap[elntIdx]);
    }
}
```

# heapsort.html

The screenshot shows a web browser window titled "My Heap Sort". The address bar indicates the file is located at "/Users/student/Documents/KinEval/tutorial\_heapsort/heapsort.html". The main content area displays the title "My Heap Sort" in blue, followed by a 2D matrix representation of a heap and a series of log messages from the heap sort process.

2172  
2422      3807      4007      5097  
5939      6539      4975      4428      7286  
6299      9779      7745      7584      5724

```
check
numbers to sort: 4007 4428 7286 5939 2172 5097 9911 6299 7745 4975 3807 9573 8069 6863 5514 2422 9779 6539 7584 5724
heap (insert 4007); 4007
heap (insert 4428); 4007-4428
heap (insert 7286); 4007-4428 7286
heap (insert 5939); 4007-4428 7286 5939
heap (insert 2172); 2172-4007 7286 5939-4428
heap (insert 5097); 2172-4007 5097 5939-4428 7286
heap (insert 9911); 2172-4007 5097 5939-4428 7286 9911
heap (insert 6299); 2172-4007 5097 5939-4428 7286 9911 6299
heap (insert 7745); 2172-4007 5097 5939-4428 7286 9911 6299 7745
heap (insert 4975); 2172-4007 5097 5939-4428 7286 9911 6299 7745-4975
heap (insert 3807); 2172-3807 5097 5939-4007 7286 9911 6299 7745-4975-4428
heap (insert 9573); 2172-3807 5097 5939-4007 7286 9911 6299 7745-4975-4428-9573
heap (insert 8069); 2172-3807 5097 5939-4007 7286 9911 6299 7745-4975-4428-9573-8069
heap (insert 6863); 2172-3807 5097 5939-4007 7286-6863 6299 7745-4975-4428-9573-8069 9911
heap (insert 5514); 2172-3807 5097 5939-4007 7286-5514 6299 7745-4975-4428-9573-8069 9911-6863 6299
heap (insert 3422); 2172-2422 5097 3807-4007 7286-5514 5939 7745-4975-4428-9573-8069 9911-6863 6299
heap (insert 9779); 2172-2422 5097 3807-4007 7286-5514 5939 7745-4975-4428-9573-8069 9911-6863 6299-9779 7745
heap (insert 6539); 2172-2422 5097 3807-4007 7286-5514 5939-6539 4975-4428-9573-8069 9911-6863 6299-9779 7745
heap (insert 7584); 2172-2422 5097 3807-4007 7286-5514 5939-6539-4975-4428-9573-8069 9911-6863 6299-9779 7745-7584 5724
heap (insert 5724); 2172-2422 5097 3807-4007 7286-5514 5939-6539-4975-4428-9573-8069 9911-6863 6299-9779 7745-7584 5724
```

# Lab Takeaways

1) Stencil  
Overview

3) Validate  
Implementation

5) Data Structure  
Considerations

1

2

3

4

5

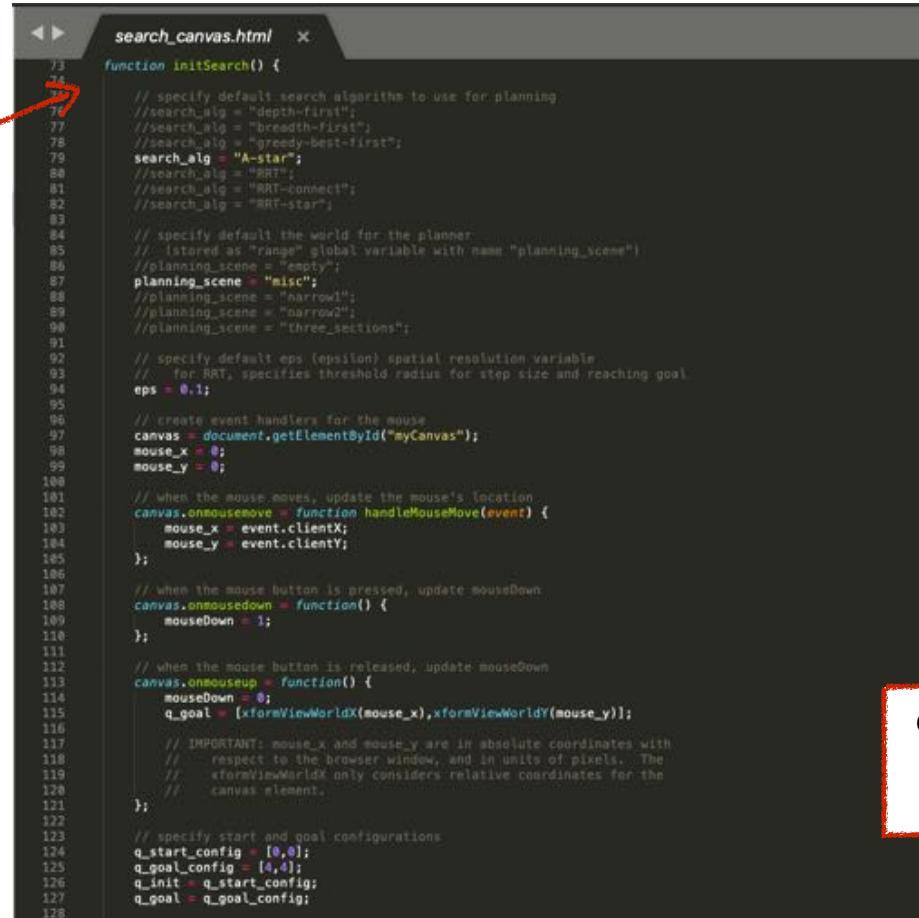
2) Walkthrough  
Heap Insert

4) Search Canvas  
Introduction

How to start  
assignment 1

# search\_canvas.html

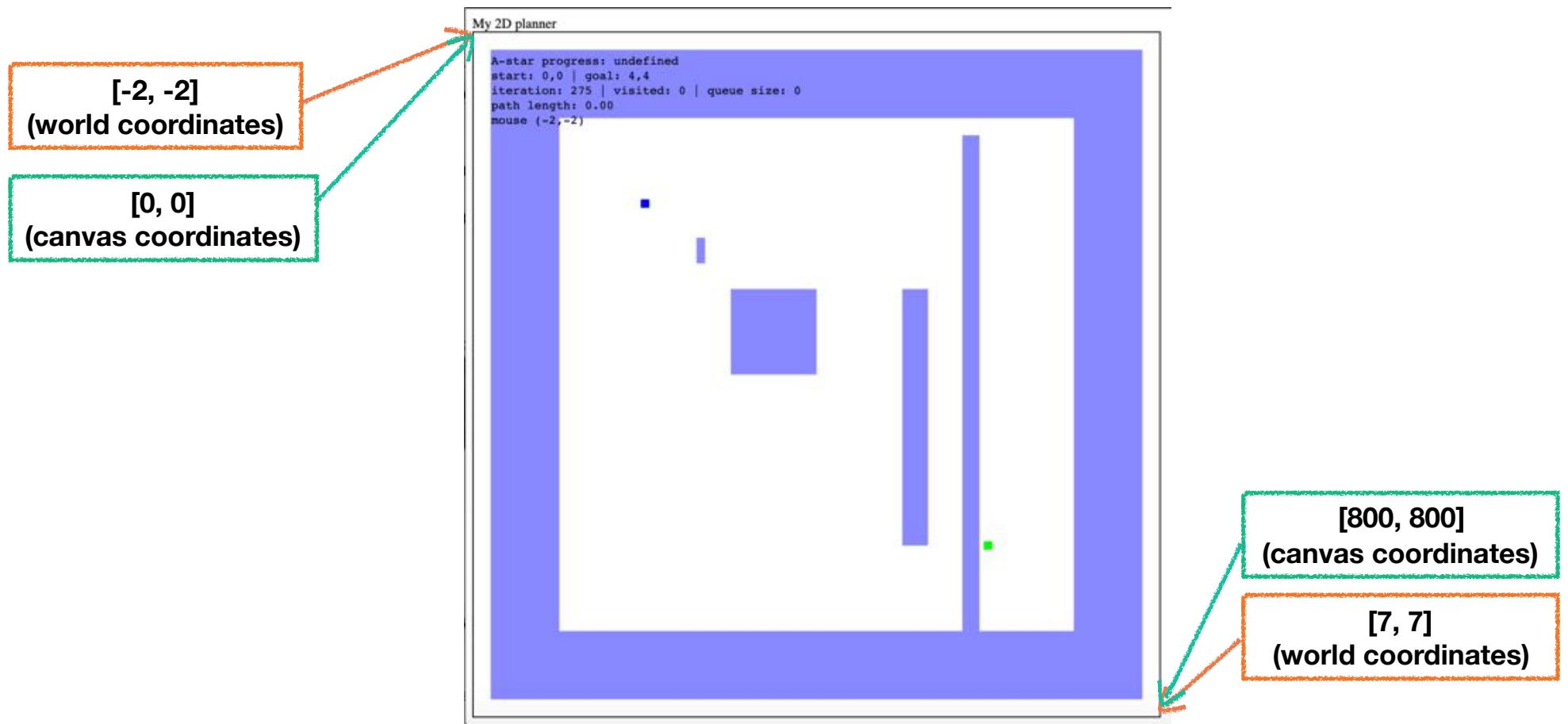
**initSearch() function**  
Instantiates global variables, commence algorithms



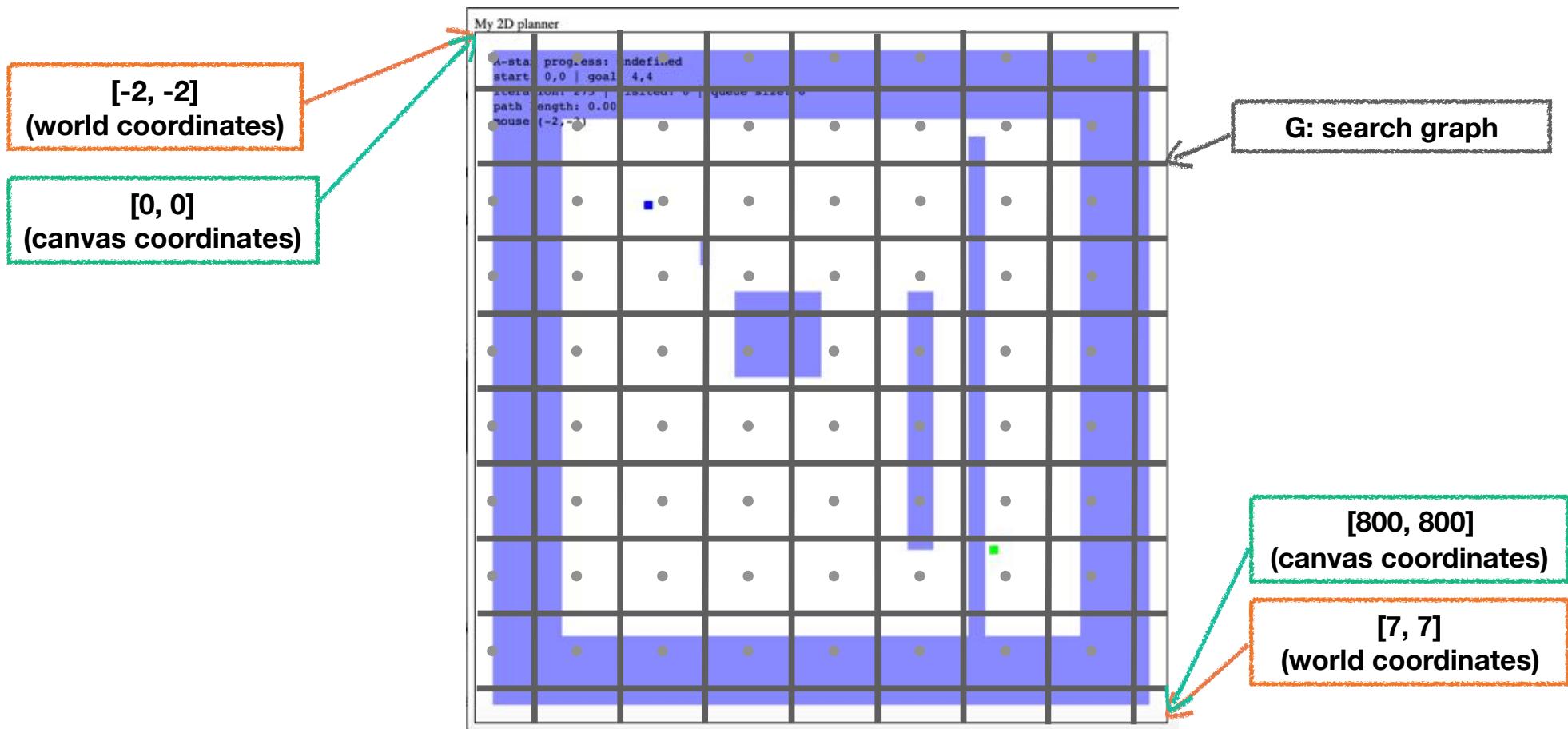
```
function initSearch() {
  // specify default search algorithm to use for planning
  //search_alg = "depth-first";
  //search_alg = "breadth-first";
  //search_alg = "greedy-best-first";
  search_alg = "A-star";
  //search_alg = "RRT";
  //search_alg = "RRT-connect";
  //search_alg = "RRT-star";
  ...
  // specify default the world for the planner
  //... listed as "range" global variable with name "planning_scene"
  //planning_scene = "empty";
  planning_scene = "misc";
  //planning_scene = "narrow1";
  //planning_scene = "narrow2";
  //planning_scene = "three_sections";
  ...
  // specify default eps (epsilon) spatial resolution variable
  //... for RRT; specifies threshold radius for step size and reaching goal
  eps = 0.1;
  ...
  // create event handlers for the mouse
  canvas = document.getElementById("myCanvas");
  mouse_x = 0;
  mouse_y = 0;
  ...
  // when the mouse moves, update the mouse's location
  canvas.onmousemove = function handleMouseMove(event) {
    mouse_x = event.clientX;
    mouse_y = event.clientY;
  };
  ...
  // when the mouse button is pressed, update mouseDown
  canvas.onmousedown = function() {
    mouseDown = 1;
  };
  ...
  // when the mouse button is released, update mouseDown
  canvas.onmouseup = function() {
    mouseDown = 0;
    q_goal = [xformViewWorldX(mouse_x),xformViewWorldY(mouse_y)];
    ...
    // IMPORTANT: mouse_x and mouse_y are in absolute coordinates with
    // respect to the browser window, and in units of pixels. The
    // xformViewWorld() only considers relative coordinates for the
    // canvas element.
  };
  ...
  // specify start and goal configurations
  q_start_config = [0,0];
  q_goal_config = [4,4];
  q_init = q_start_config;
  q_goal = q_goal_config;
}
```

**q\_start\_config** = start cell in grid  
**q\_goal\_config** = goal cell in grid  
**q\_init**

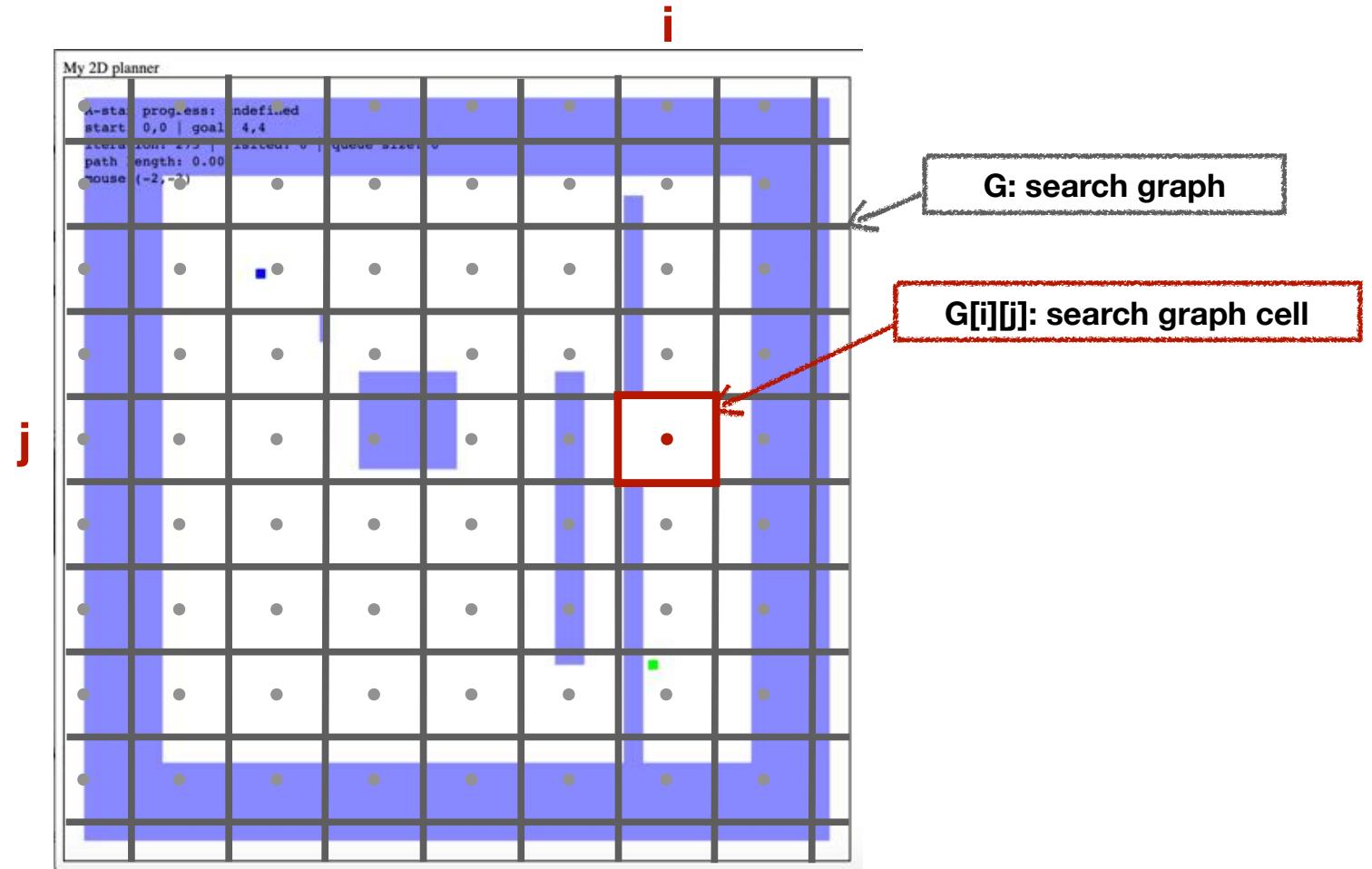
# search\_canvas.html



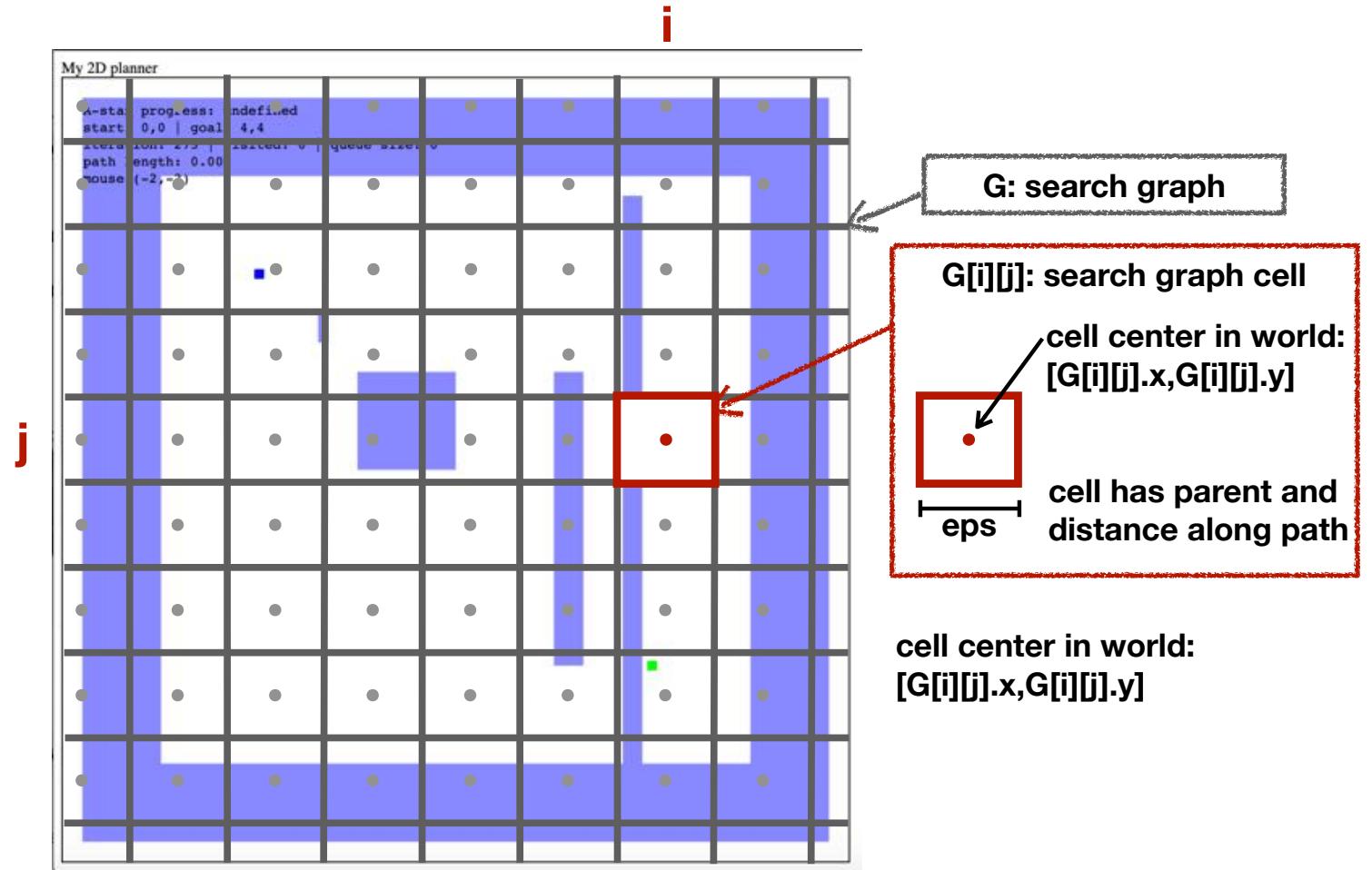
# search\_canvas.html



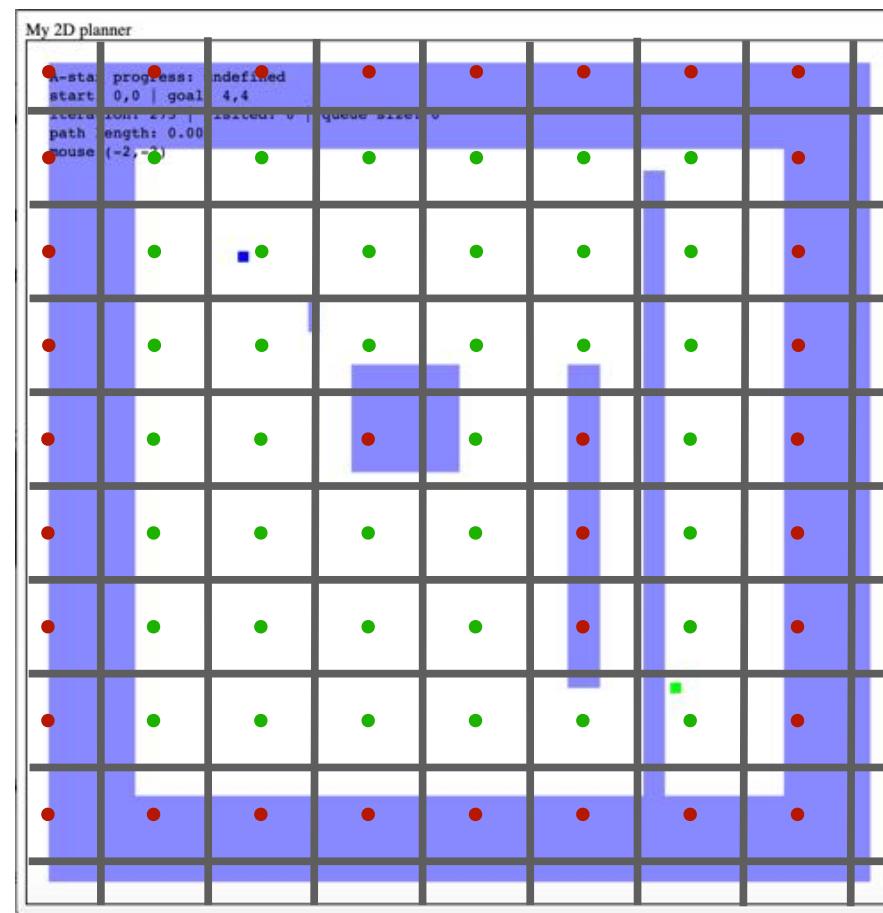
# search\_canvas.html



# search\_canvas.html

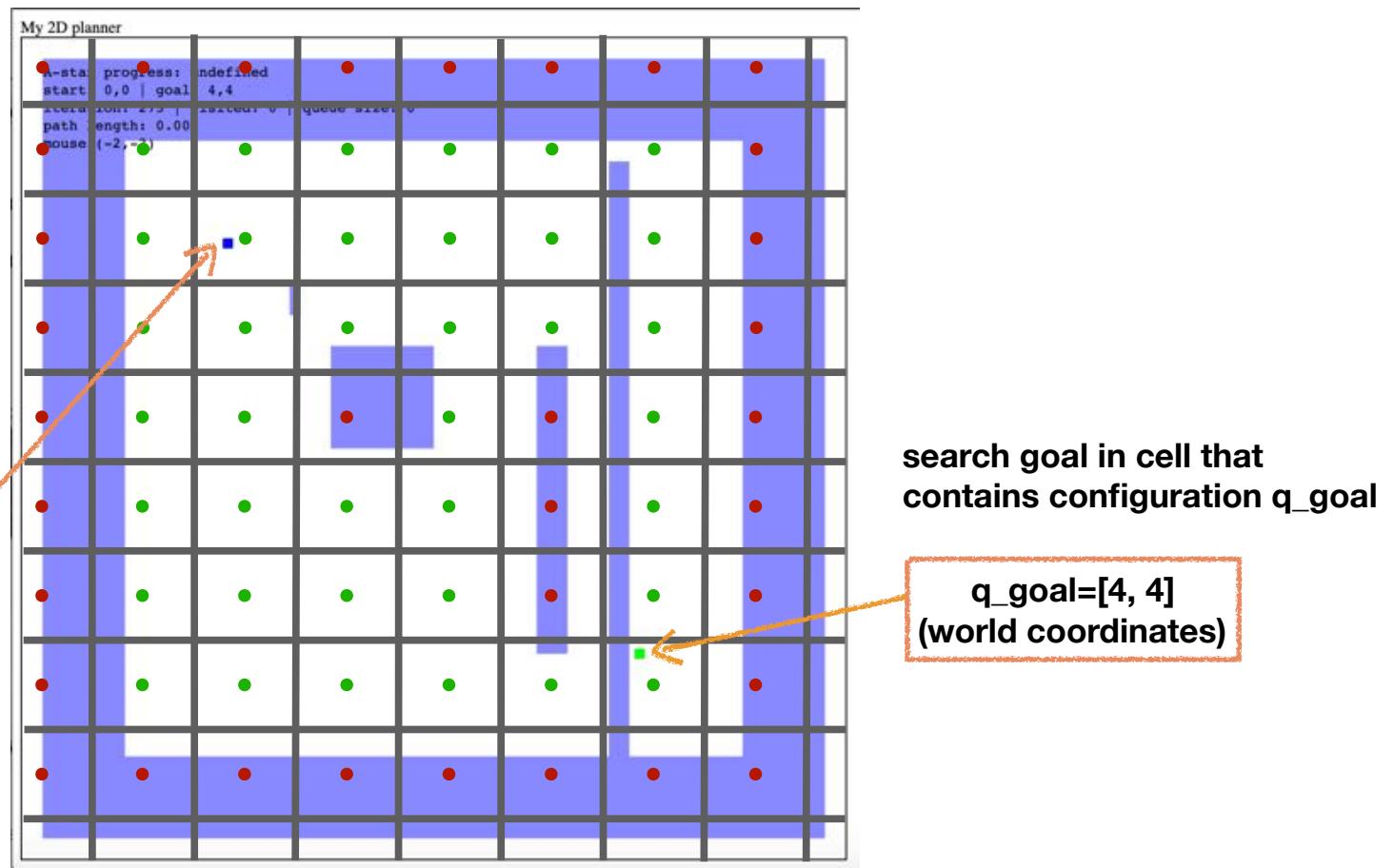


# search\_canvas.html



test configuration at visited cell  
center for collision:  
testCollision([G[i][j].x,G[i][j].y])

# search\_canvas.html



# search\_canvas.html

Important to identify discrete start indices within graph from continuous world position

```
function initSearchGraph() {  
    // KE: visit queue not created for certain values of eps  
    visit_queue = [];  
  
    // initialize search graph as 2D array over configuration space  
    // of 2D locations with specified spatial resolution  
    G = [];  
    for (iind=0,xpos=-2;xpos<7;iind++,xpos+=eps) {  
        G[iind] = [];  
        for (jind=0,ypos=-2;ypos<7;jind++,ypos+=eps) {  
            G[iind][jind] = {  
                i:iind,j:jind, // mapping to graph array.  
                x:xpos,y:ypos, // mapping to map coordinates  
                parent:null, // pointer to parent in graph along motion path  
                distance:10000, // distance to start via path through parent  
                visited:false, // flag for whether the node has been visited  
                priority:null, // visit priority based on fscore  
                queued:false // flag for whether the node has been queued for visiting  
            };  
            // STENCIL: determine whether this graph node should be the start  
            // point for the search  
            visit_queue = [];  
        }  
    }  
}
```

# search\_canvas.html

**animate() function  
responsible for  
calling your iterate  
functions**

```
// render the world to the canvas element
drawRobotWorld();

// make sure the rrt iterations are not running faster than animation update
if (search_iterate && (Date.now() - cur_time > min_msec_between_iterations)) {

    // update time marker for last iteration update
    cur_time = Date.now();

    // update iteration count
    search_iter_count++;

    // call iteration for the selected search algorithm
    switch (search_alg) {
        case "depth-first":
        case "breadth-first":
        case "greedy-best-first":
        case "A-star":
            search_result = iterateGraphSearch();
            break;
        case "RRT":
            search_result = "failed";
            // (hack to speed viz)
            while (search_result == "failed")
                search_result = iterateRRT();
            break;
        case "RRT-connect":
            // (hack to speed viz) while (search_result == "failed")
            search_result = iterateRRTConnect();
            break;
        case "RRT-star":
            search_result = iterateRRTStar();
            break;
        default:
            console.warn('search_canvas: search algorithm not found, using rrt as default');
            search_result = iterateRRT();
            break;
    }
}
```

Animation accomplished by iterative  
drawUpdate()->computeUpdate()

# search\_canvas.html

Ensure your implementations are isolated to single search steps

```
function iterateGraphSearch() {  
  
    // STENCIL: implement a single iteration of a graph search algorithm  
    // for A-star (or DFS, BFS, Greedy Best-First)  
    // An async timing mechanism is used instead of a for loop to avoid  
    // blocking and non-responsiveness in the browser.  
  
    // Return "failed" if the search fails on this iteration.  
    // Return "succeeded" if the search succeeds on this iteration.  
    // Return "iterating" otherwise.  
  
    // Provided support functions:  
  
    // testCollision - returns whether a given configuration is in collision  
    // drawHighlightedPathGraph - draws a path back to the start location  
    // draw_2D_configuration - draws a square at a given location  
}
```

Including excessive loops may cause browser to become unresponsive