

EECS 367 & ROB 320 Lab

KinEval Path Planning code overview

Administrative

- Let me know if you are not on the Slack workspace
- Assignment #0: ROS Pub/Sub
 - Due 11:59pm, Wednesday, January 18
- CI-Grader has run
 - Need: workflow survey (GitHub username, unique name), Github Classroom or specific private repo
- Next CI-Grader Run: Late Tonight(I will post on Slack when it's done)

Lab Takeaways

1. Stencil overview
2. Walk through heap insert function
3. Validate implementation
4. Search canvas introduction
5. Data structure considerations

Assignment 1 Overview

Assignment 1: 2D Path Planning			
4	All	Heap implementation	
6	All	A-star search	
2	511	BFS	
2	511	DFS	
2	511	Greedy best-first	

KinEval Stencil

The screenshot shows the GitHub repository page for 'KinEval Stencil'. The repository has 4 watchers, 7 stars, and 3 forks. The navigation bar includes links for Code, Issues (1), Pull requests, Actions, Projects, Wiki, Security, and Insights. The main content area shows a list of files and folders. A red box on the left contains the text 'Bulk of the code needed to complete Assignment 1' with an arrow pointing to the 'project_pathplan' folder. Another red box at the bottom contains the text 'Starter code for Javascript intro and heap debugging' with an arrow pointing to the 'tutorial_heapsort' folder. The right sidebar contains sections for 'About', 'Releases', 'Packages', and 'Contributors'.

Code **Issues** 1 **Pull requests** **Actions** **Projects** **Wiki** **Security** **Insights**

0 tags [Go to file](#) [Add file](#) [Code](#)

#3 from cxt98/master [b8f51ea](#) yesterday 9 commits

js	initial commit Fall 2018	2 years ago
kinEval	initial commit Fall 2018	2 years ago
project_pathplan	Adds refactored stencil files for project 1.	9 days ago
project_pendularm	add refactor of assignment2, tested with CI grader	5 days ago
robots	initial commit Fall 2018	2 years ago
tutorial_heapsort	initial commit Fall 2018	2 years ago
tutorial_js	initial commit Fall 2018	2 years ago
	initial commit Fall 2018	2 years ago
	add refactor of assignment2, tested with CI grader	5 days ago
	initial commit Fall 2018	2 years ago
	initial commit Fall 2018	2 years ago

About

Stencil code for KinEval (Kinematic Evaluator) for robot control, kinematics, decision, and dynamics in JavaScript/HTML5

[Readme](#) [View license](#)

Releases

No releases published

Packages

No packages published

Contributors 4

Heapsort Tutorial

Watch

4

Star

7

Fork

3

<> Code

Issues 1

Pull requests

Actions

Projects

Wiki

Security

Insights

master

kineval-stencil / tutorial_heapsort /

Go to file

Add file

odestcj initial commit Fall 2018

6cd9f47 on 10 Sep 2018 History

..

heap.js

initial commit Fall 2018

2 years ago

heapsort.html

initial commit Fall 2018

2 years ago

Heapsort Tutorial HTML

heapsort.html

```
57  <!--  
58      A script element contains JavaScript code for the browser to execute.  
59      Script code could be in another file, specified in the "src" attribute,  
60      as in the case below for your heap code.  
61  
62      This external source file should produce the object "minheaper" that has  
63      two function methods for inserting and extracting heap elements.  
64  -->  
65  
66  <!-- you will want to uncomment this tag  
67  <script src="heap.js"></script>  
68  -->
```

Specifies JavaScript
source file to make
available in HTML.
Uncomment this line!

Heapsort Tutorial HTML

heapsort.html

Represent heap as
a JavaScript array

```
console.log("building min binary heap from number array");  
numbers_heap = []; // create array for heap  
for (i=0; i<numbers.length; i++) {
```

Repeatedly call
heap insert method

```
console.log("inserting number "+numbers[i]+" into the heap");  
minheaper.insert(numbers_heap, numbers[i]);
```

```
151 console.log("appending current heap state to output object");  
152 output_string = "heap (insert " + numbers[i] + "): "; //  
153 for (j=0; j<numbers_heap.length; j++) {  
154     output_string += numbers_heap[j] + " "; //  
155 }  
156 addHTMLLine("output", output_string);  
157 }  
158
```

Print state of
heap to screen

Heapsort Tutorial Results

My Heap Sort

File | /Users/student/Documents/AutoRob/git/tutorial_heapsort/heapsort.html

[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 heapng functions are not yet implemented

Without heap.js implementation

My Heap Sort

file:///Users/student/Documents/EECS567-master/tutorial_heapsort/heapsort.html

[My Heap Sort](#)

2698

504

4275

3138

1559

4500

5664

4872

4020

3515

6512

4361

5817

9333

4752

6585

6073

8241

7735

4747

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 4275 4020 3138 4500 5664 8241 4747 6512 4361 5817 9333 4752 6585 6073

heap (insert 7735): 504 2698 1559 4275 4020 3138 4500 5664 7735 4747 6512 4361 5817 9333 4752 6585 6073 8241

heap (insert 4872): 504 2698 1559 4275 4020 3138 4500 5664 4872 4747 6512 4361 5817 9333 4752 6585 6073 8241 7735

heap (insert 3515): 504 2698 1559 4275 3515 3138 4500 5664 4872 4020 6512 4361 5817 9333 4752 6585 6073 8241 7735 4747

heap (extract 504): 1559 2698 3138 4275 3515 4361 4500 5664 4872 4020 6512 4747 5817 9333 4752 6585 6073 8241 7735

heap (extract 1559): 2698 3515 3138 4275 4020 4361 4500 5664 4872 7735 6512 4747 5817 9333 4752 6585 6073 8241

heap (extract 2698): 3138 3515 4361 4275 4020 4747 4500 5664 4872 7735 6512 8241 5817 9333 4752 6585 6073

heap (extract 3138): 3515 4020 4361 4275 6073 4747 4500 5664 4872 7735 6512 8241 5817 9333 4752 6585

heap (extract 3515): 4020 4275 4361 4872 6073 4747 4500 5664 6585 7735 6512 8241 5817 9333 4752

heap (extract 4020): 4275 4752 4361 4872 6073 4747 4500 5664 6585 7735 6512 8241 5817 9333

heap (extract 4275): 4361 4752 4500 4872 6073 4747 9333 5664 6585 7735 6512 8241 5817

heap (extract 4361): 4500 4752 4747 4872 6073 5817 9333 5664 6585 7735 6512 8241

heap (extract 4500): 4747 4752 5817 4872 6073 8241 9333 5664 6585 7735 6512

heap (extract 4747): 4752 4872 5817 5664 6073 8241 9333 6512 6585 7735

heap (extract 4752): 4872 5664 5817 6512 6073 8241 9333 7735 6585

heap (extract 4872): 5664 6073 5817 6512 6585 8241 9333 7735

heap (extract 5664): 5817 6073 7735 6512 6585 8241 9333

heap (extract 5817): 6073 6512 7735 9333 6585 8241

heap (extract 6073): 6512 6585 7735 9333 8241

heap (extract 6512): 6585 8241 7735 9333

heap (extract 6585): 7735 8241 9333

heap (extract 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 6585 7735 8241 9333

With heap.js implementation

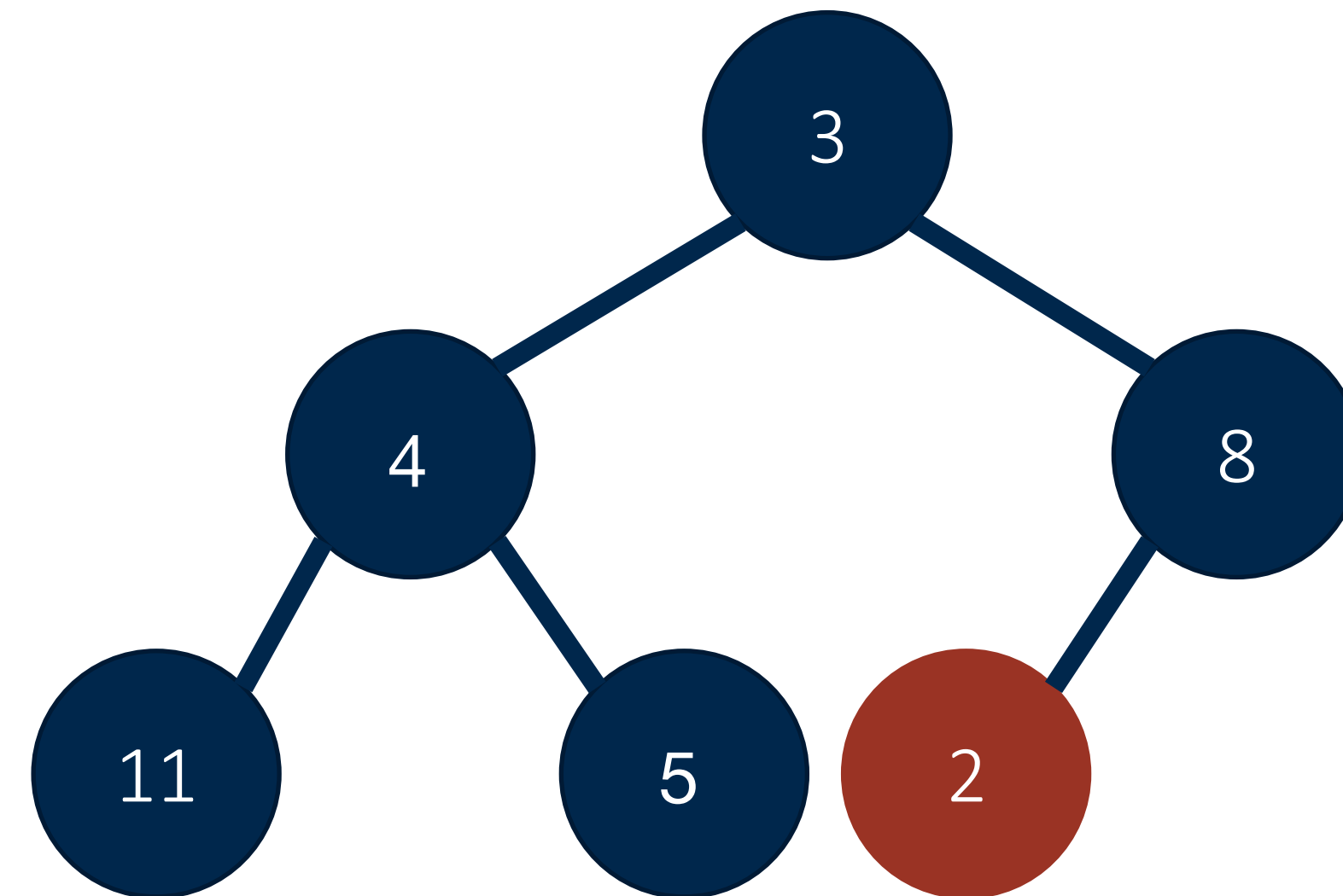
Heapsort Tutorial JavaScript

heap.js

```
25 // create empty object
26 minheaper = {};
27
28 // define insert function for min binary heap
29 function minheap_insert(heap, new_element) {
30
31     // STENCIL: implement your min binary heap insert operation
32 }
33
```

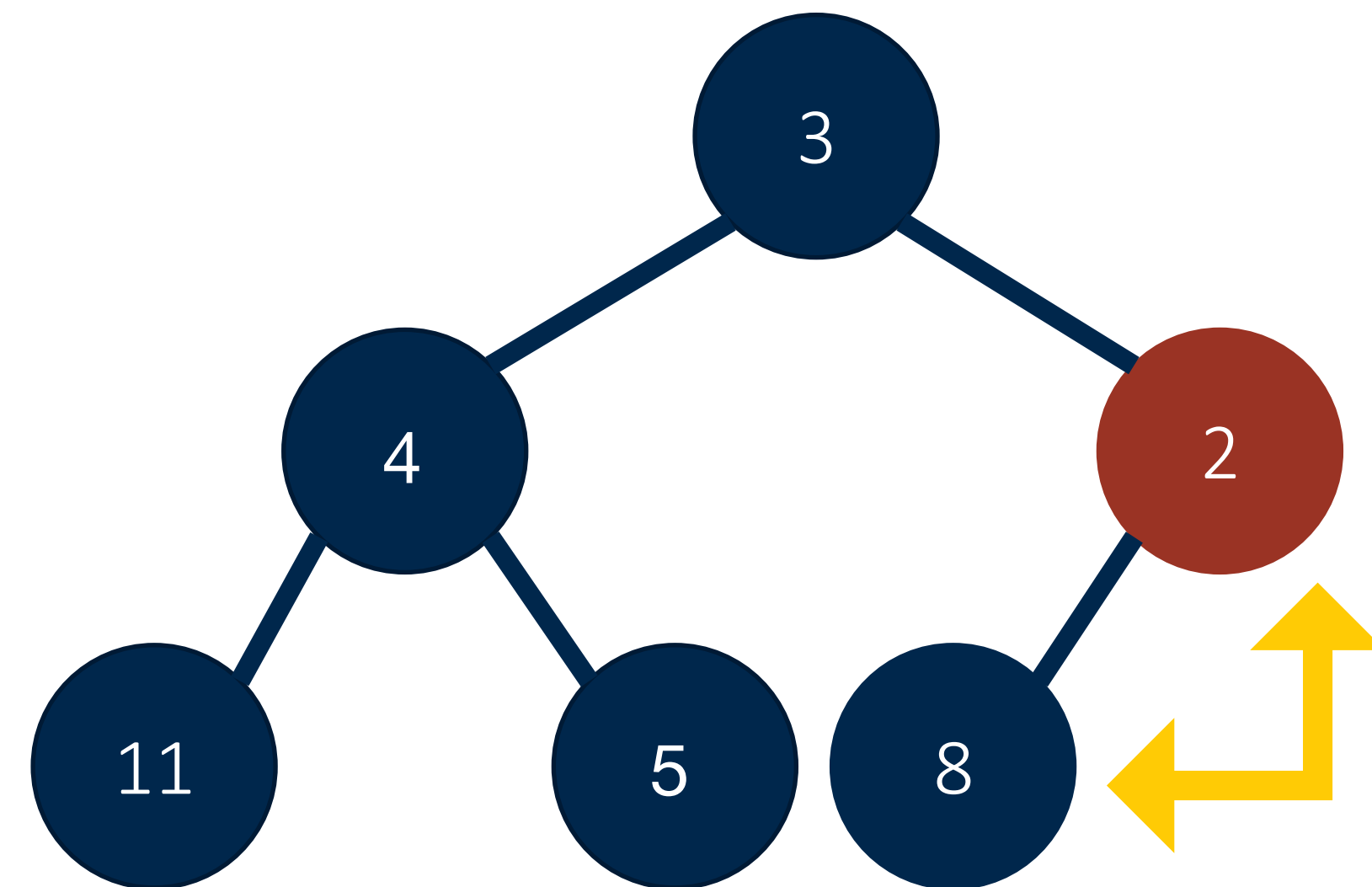
Heap Insert

1. Add new element into first open spot in tree (end of heap)



Heap Insert

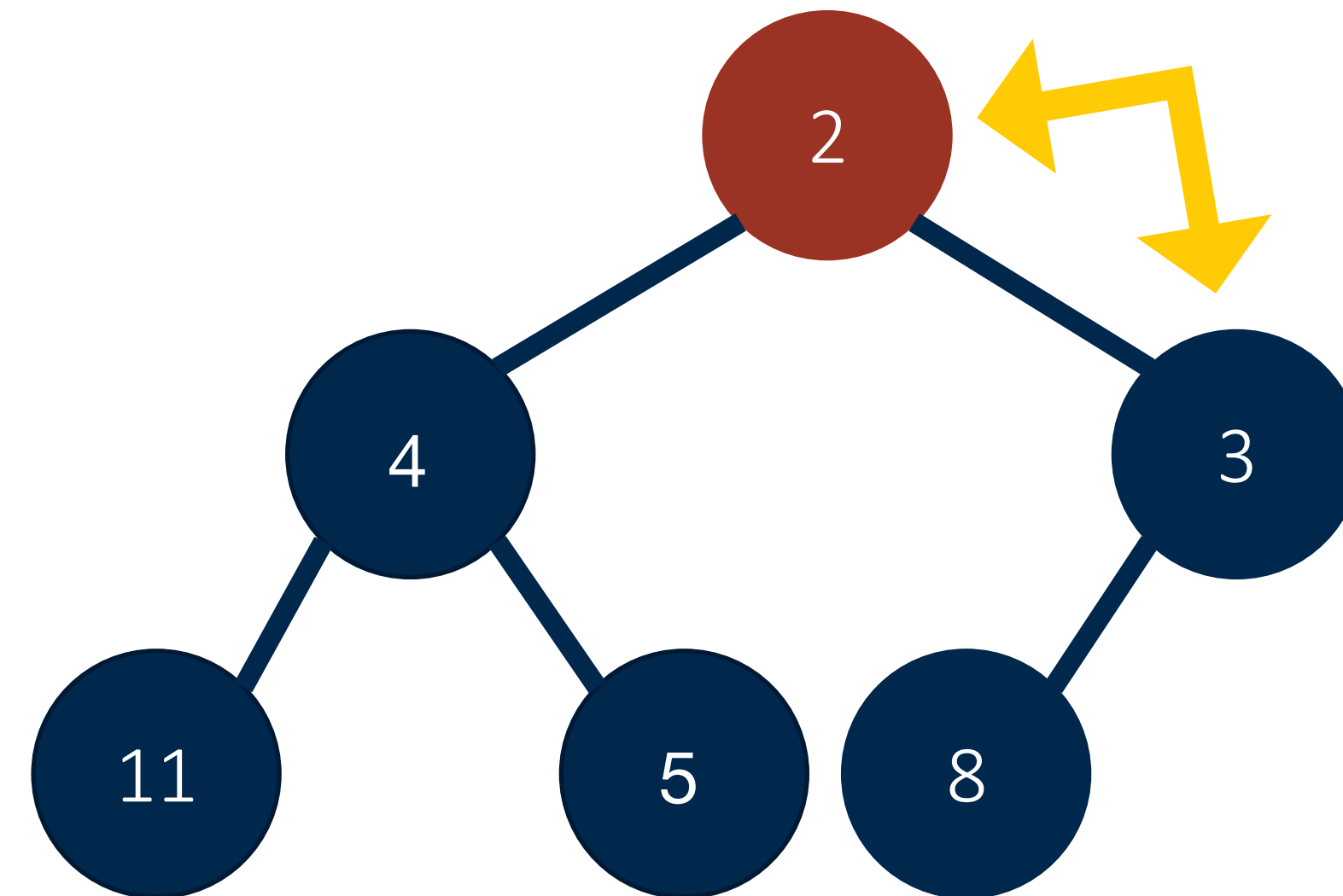
1. Add new element into first open spot in tree (end of heap)
2. If new element is smaller than parent, swap with parent



Heap Insert

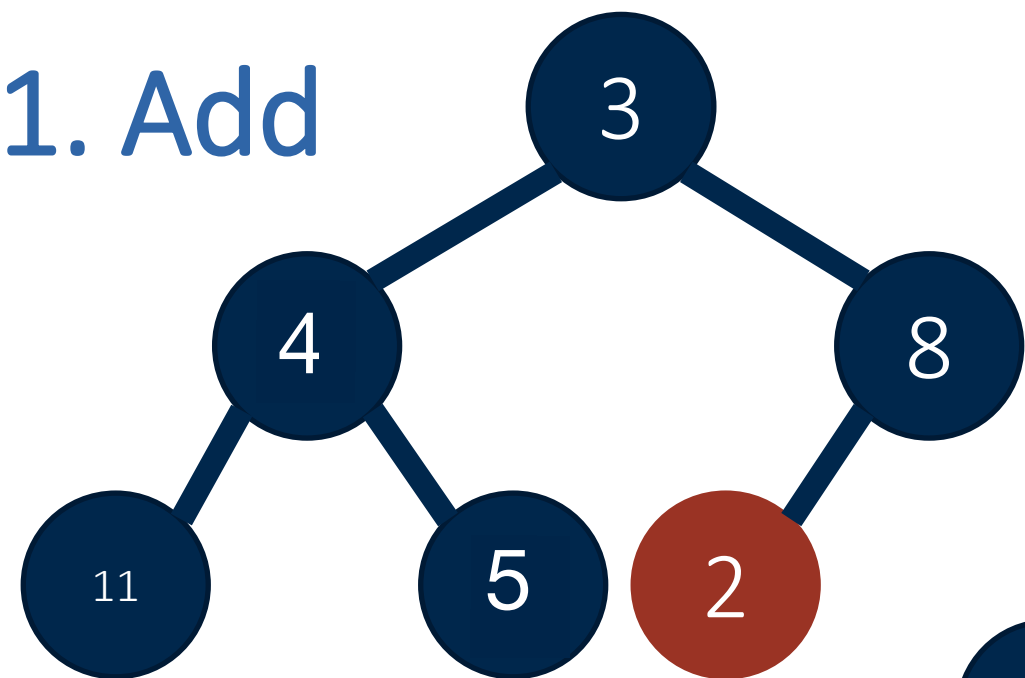
1. Add new element into first open spot in tree (end of heap)
2. If new element is smaller than parent, swap with parent
3. Repeat step 2 until heap property holds

Heap property: For min heaps, the value of the parent node must always be less than or equal to the value of the child node

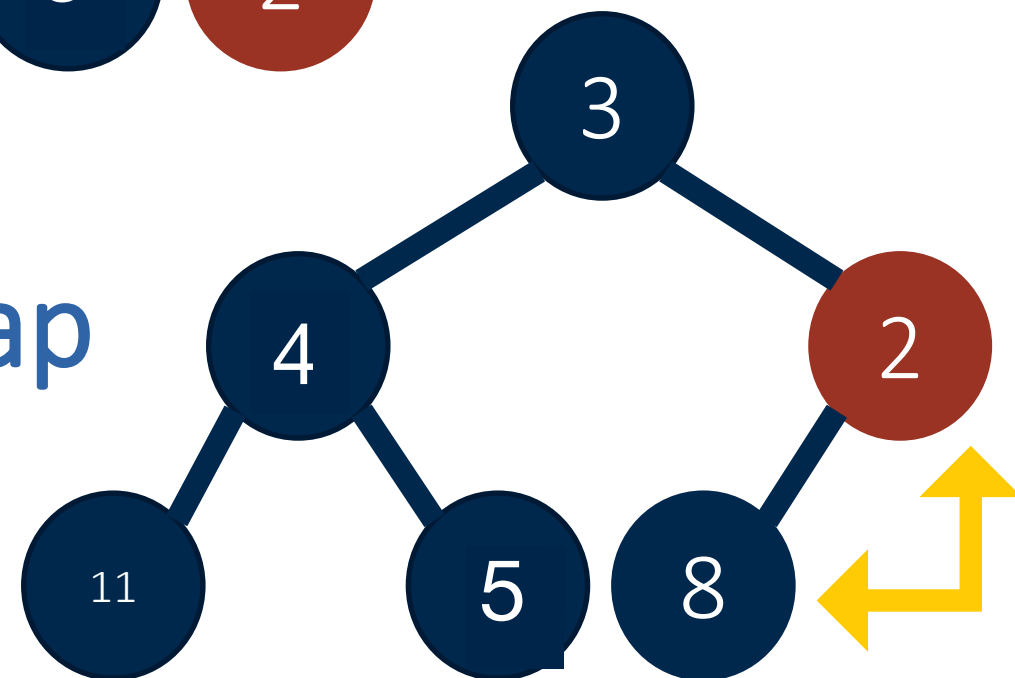


Heap Insert

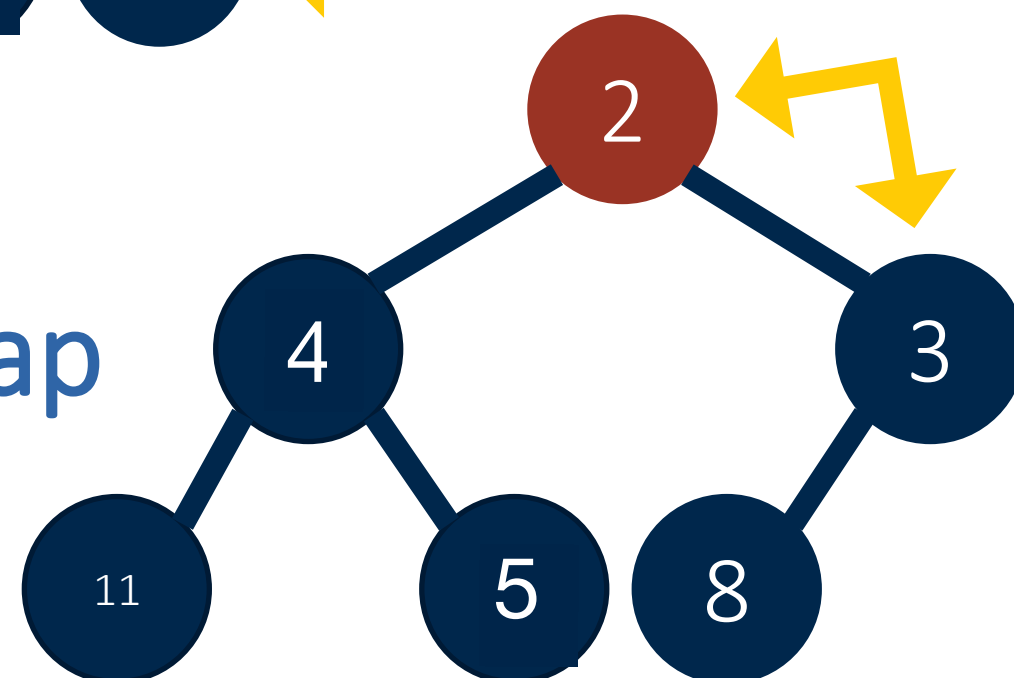
1. Add



2. Swap



3. Repeat swap



```
// 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

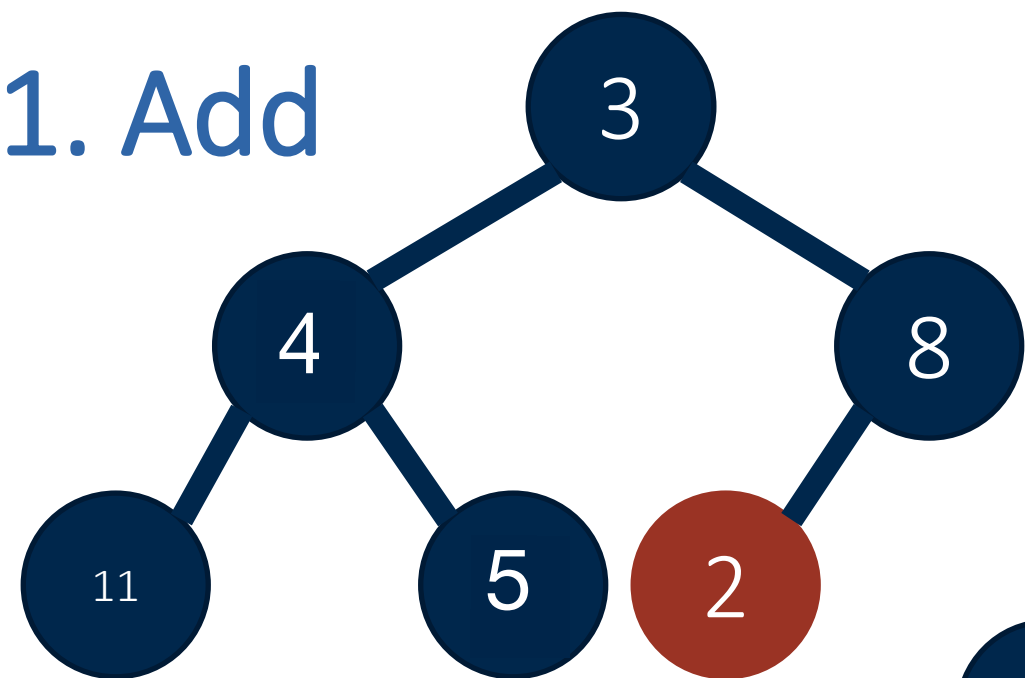
    // TODO: Add new_element to the heap array

    // TODO: Initialize heap condition check

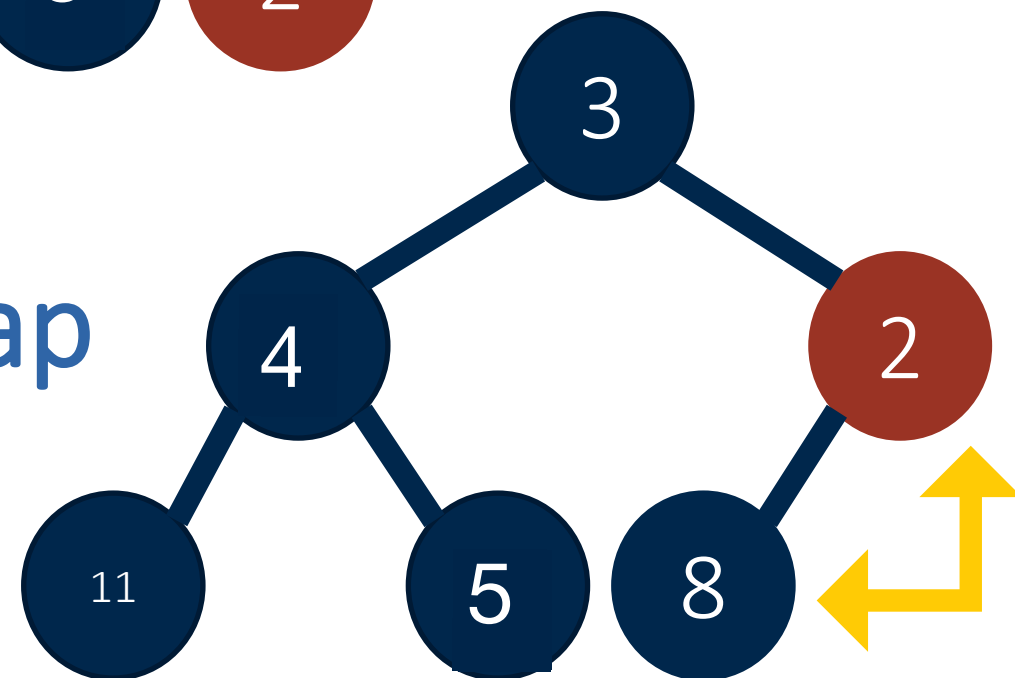
    // TODO: As long as heap condition not satisfied
    //     TODO: Swap new_element with parent
    //
    //     TODO: Update index for new_element
    //     TODO: Update index for new_element's parent
    //
    //     TODO: Update heap condition check
}
```

Heap Insert

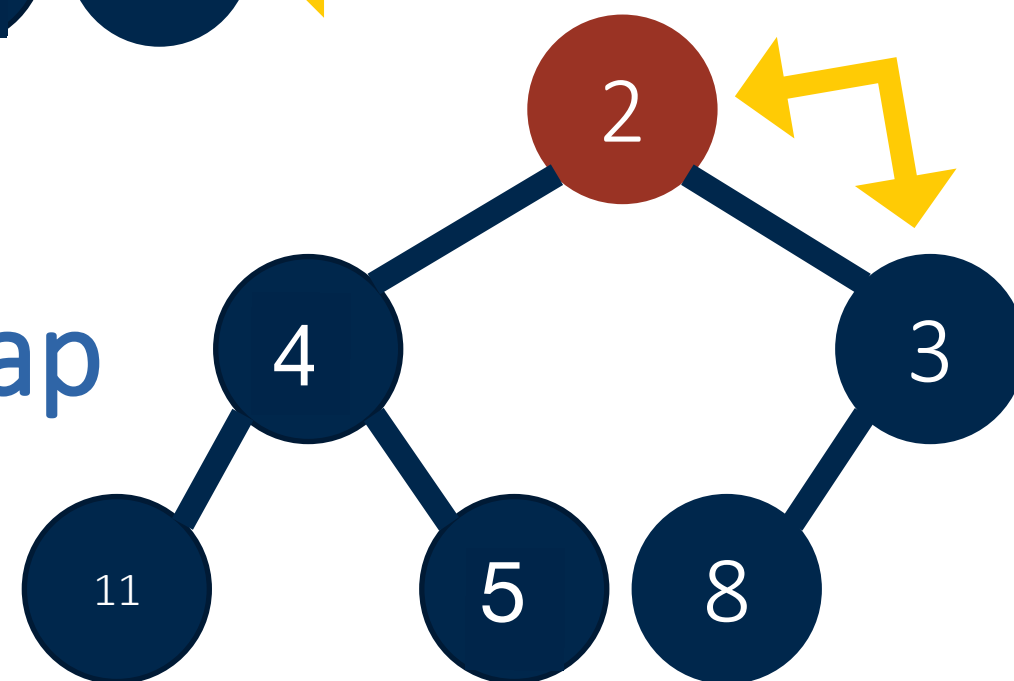
1. Add



2. Swap



3. Repeat swap



```
// 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

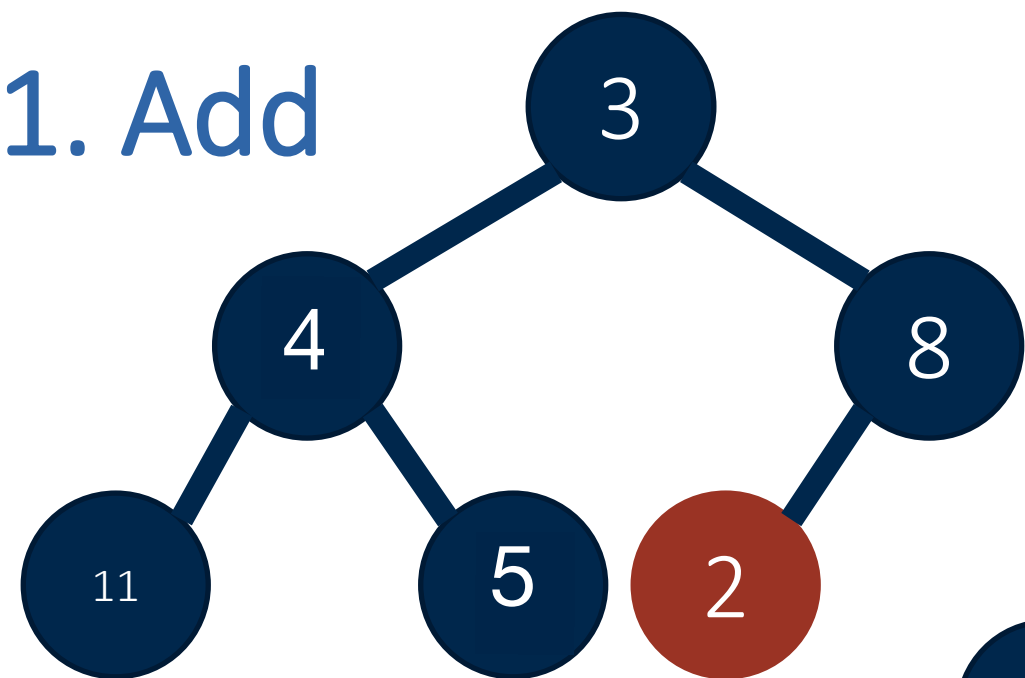
  // TODO: Add new_element to the heap array

  // TODO: Initialize heap condition check

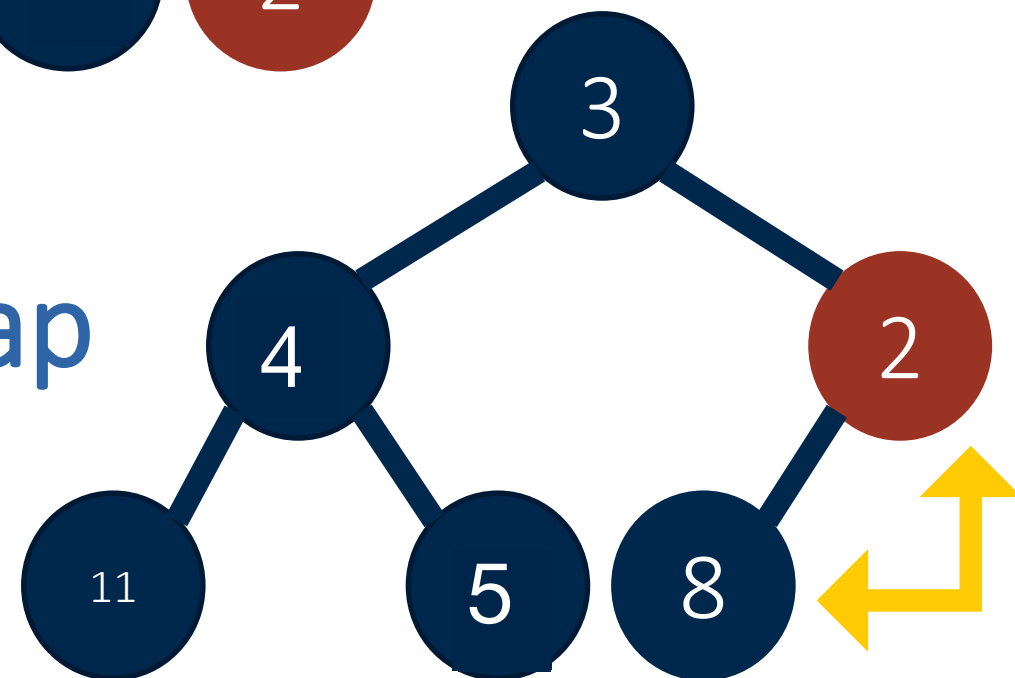
  // TODO: As long as heap condition not satisfied
  //   TODO: Swap new_element with parent
  //
  //   TODO: Update index for new_element
  //   TODO: Update index for new_element's parent
  //
  //   TODO: Update heap condition check
}
```

Heap Insert

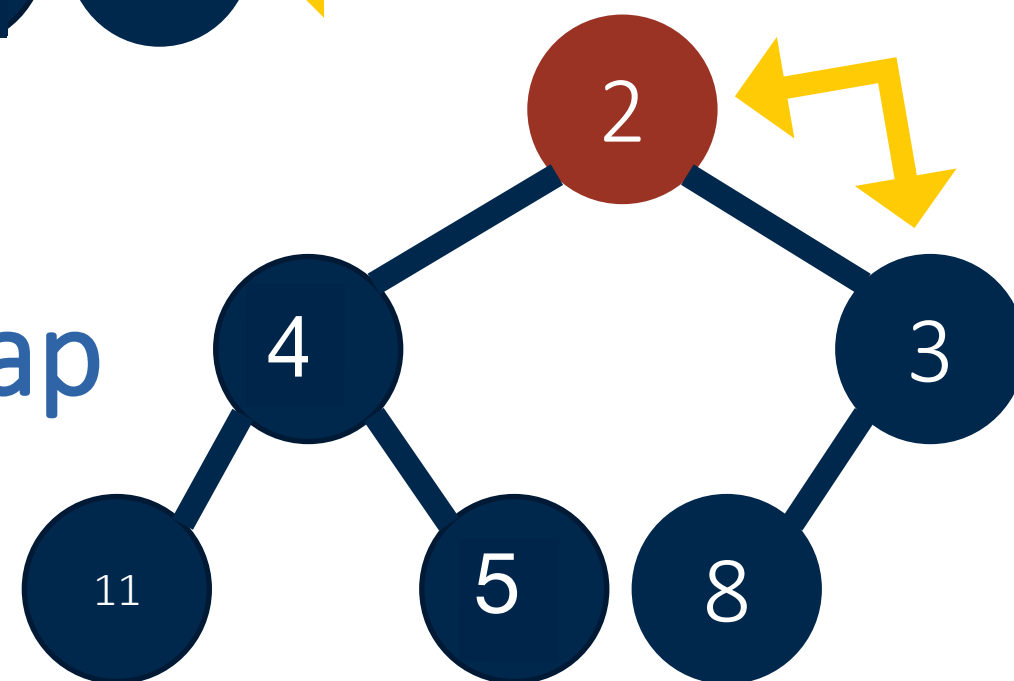
1. Add



2. Swap



3. Repeat swap



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

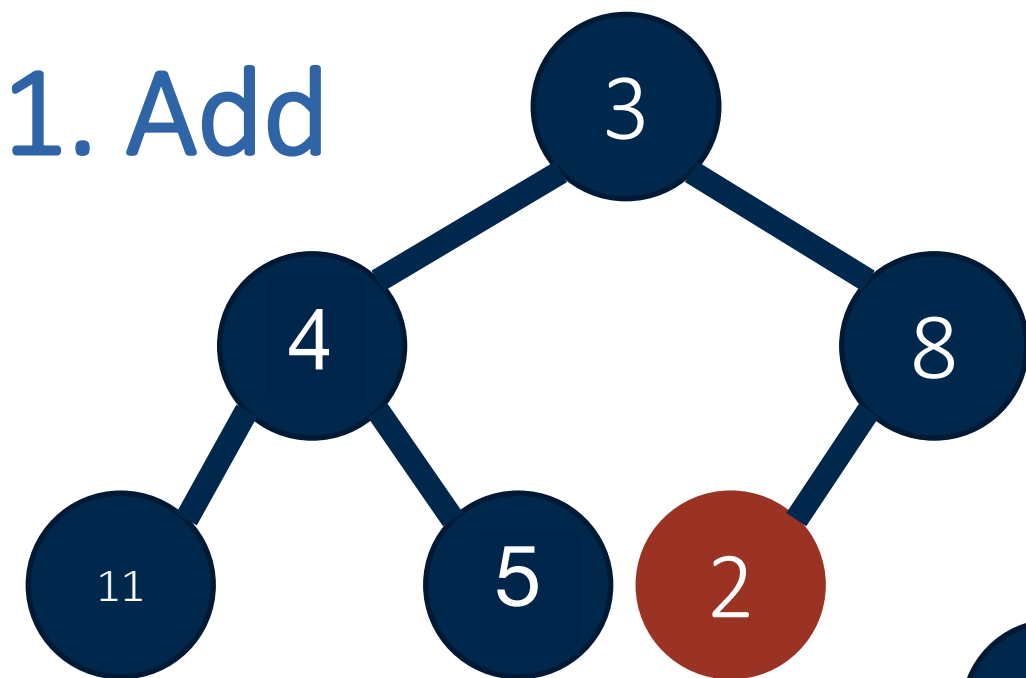
  // TODO: Add new_element to the heap array

  // TODO: Initialize heap condition check

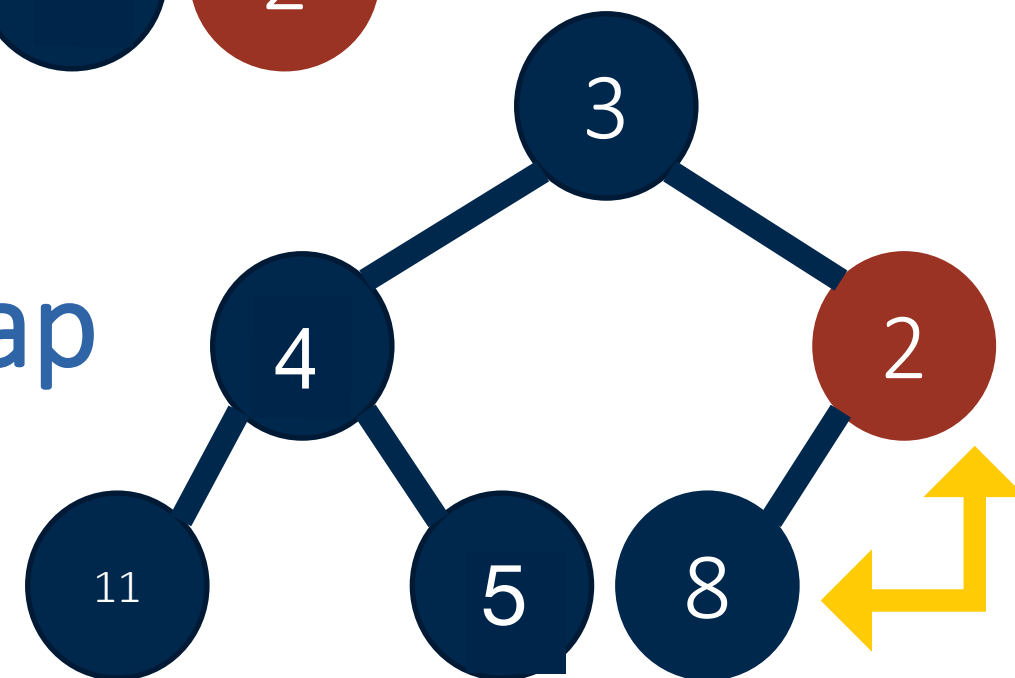
  // TODO: As long as heap condition not satisfied
  //   TODO: Swap new_element with parent
  //
  //   TODO: Update index for new_element
  //   TODO: Update index for new_element's parent
  //
  //   TODO: Update heap condition check
}
```

Heap Insert

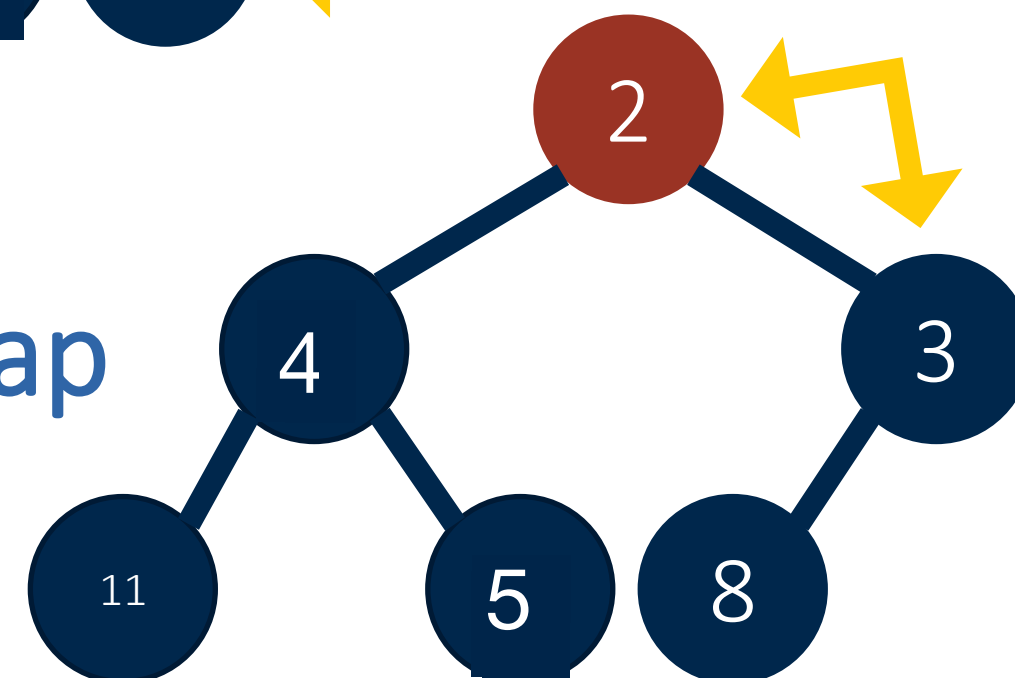
1. Add



2. Swap



3. Repeat swap



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

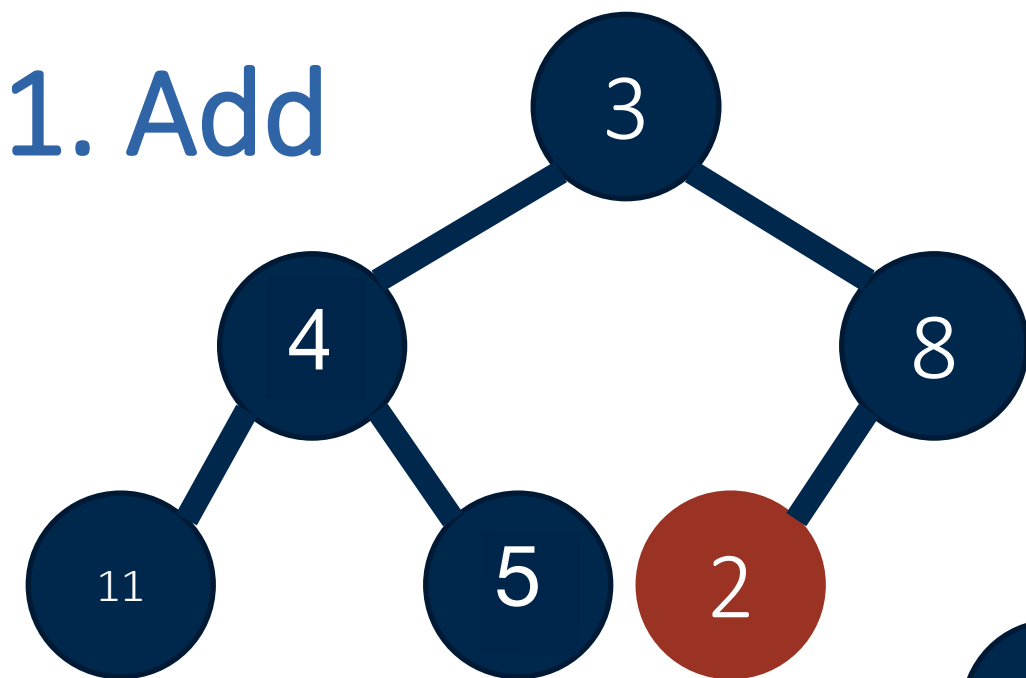
  heap.push(new_element);

  // TODO: Initialize heap condition check

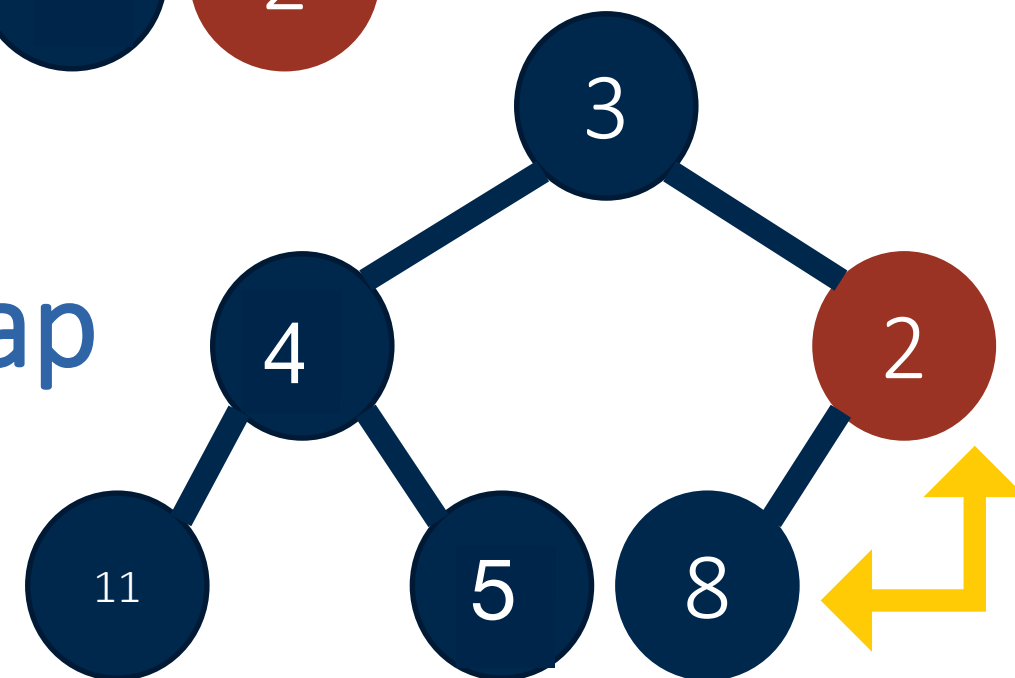
  // TODO: As long as heap condition not satisfied
  //   TODO: Swap new_element with parent
  //
  //   TODO: Update index for new_element
  //   TODO: Update index for new_element's parent
  //
  //   TODO: Update heap condition check
}
```


Heap Insert

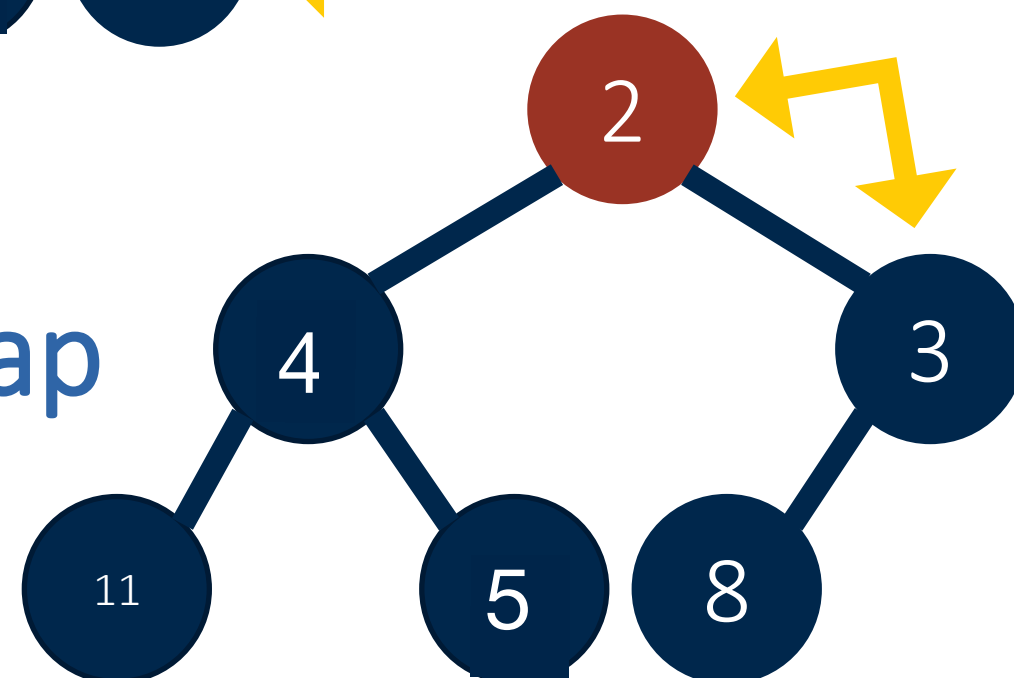
1. Add



2. Swap



3. Repeat swap



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

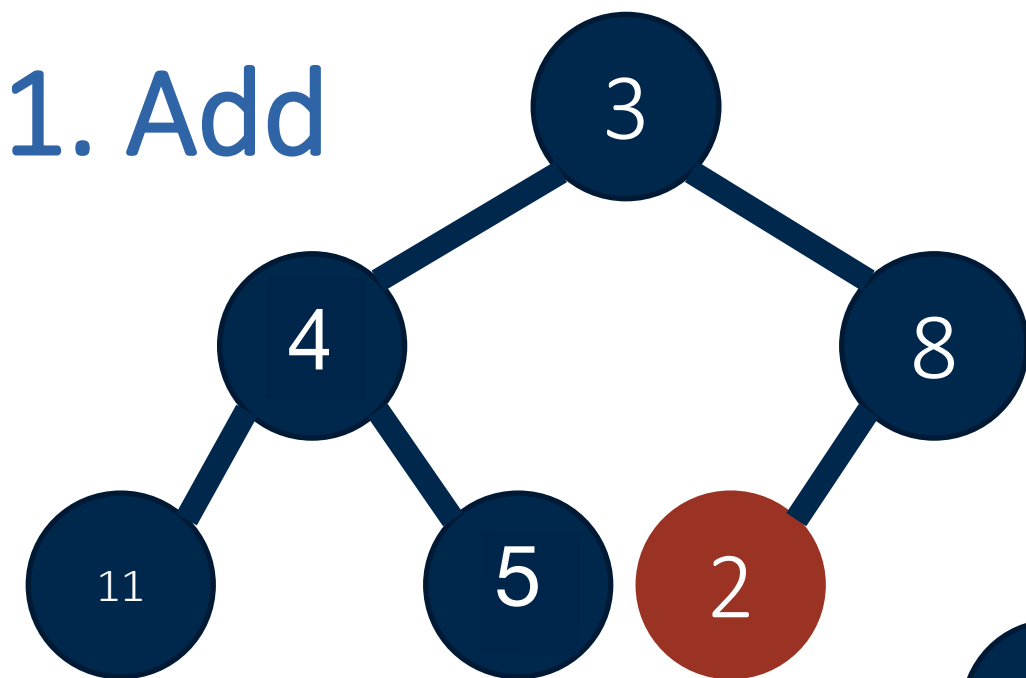
    heap.push(new_element);

    // 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]);

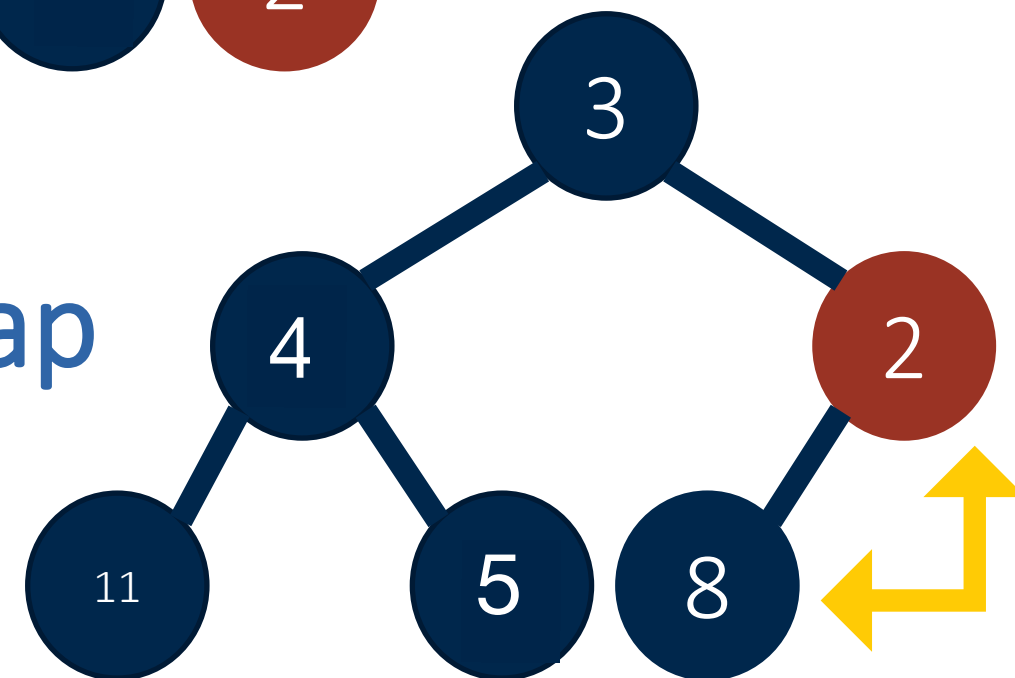
    // TODO: As long as heap condition not satisfied
    //      TODO: Swap new_element with parent
    //
    //      TODO: Update index for new_element
    //      TODO: Update index for new_element's parent
    //
    //      TODO: Update heap condition check
}
```


Heap Insert

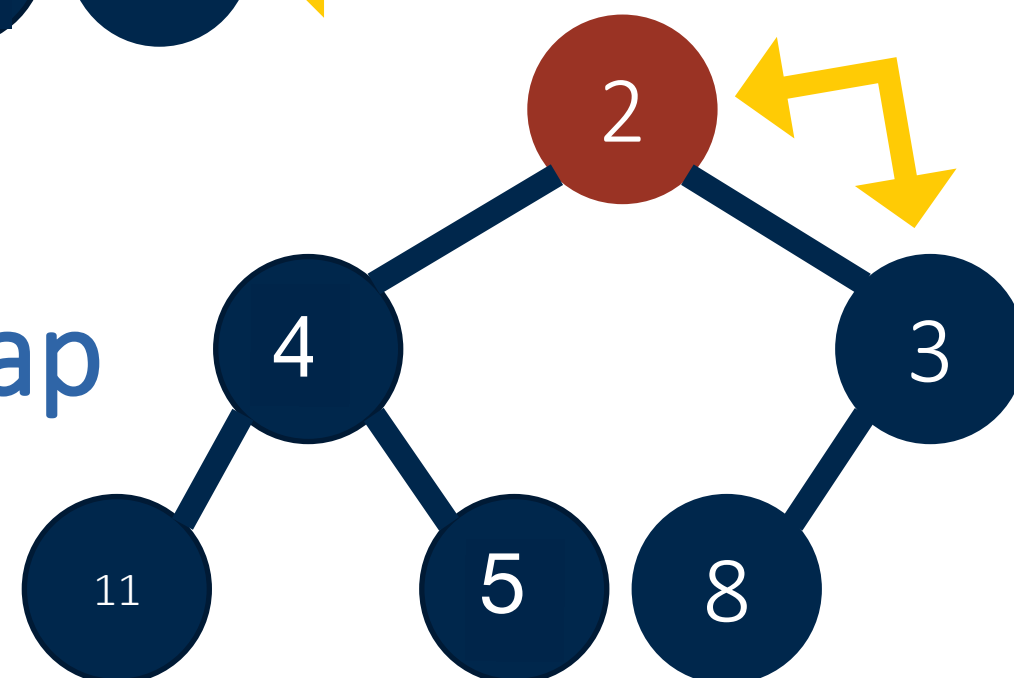
1. Add



2. Swap



3. Repeat swap



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

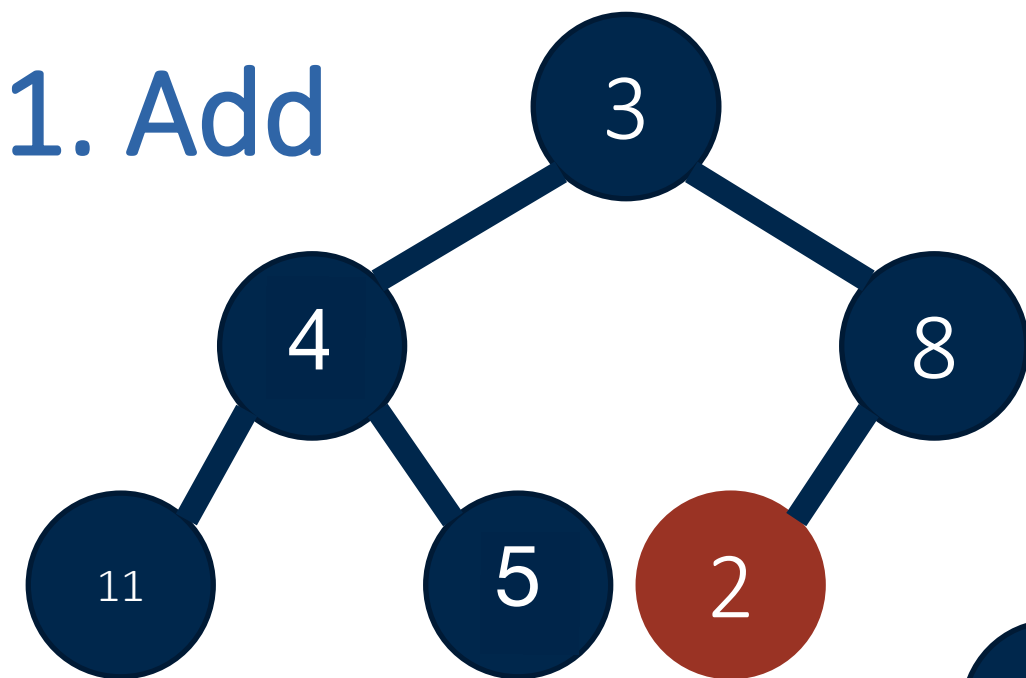
    heap.push(new_element);

    // 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]);

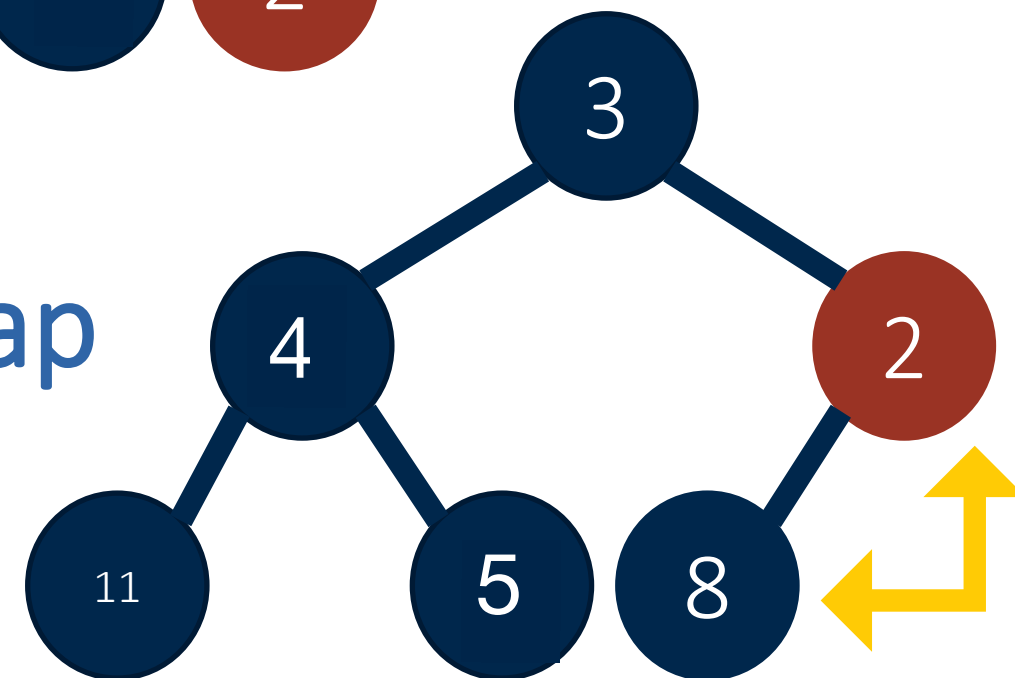
    while (!heaped) {
        // TODO: Swap new_element with parent
        //
        // TODO: Update index for new_element
        // TODO: Update index for new_element's parent
        //
        // TODO: Update heap condition check
    }
}
```

Heap Insert

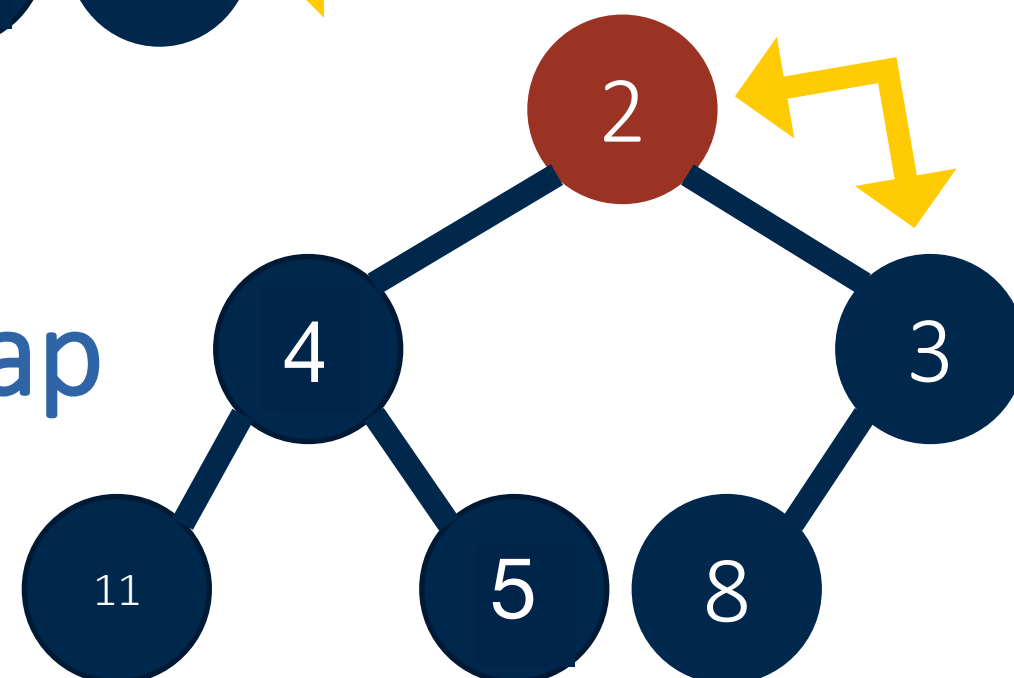
1. Add



2. Swap



3. Repeat swap



```
// 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);

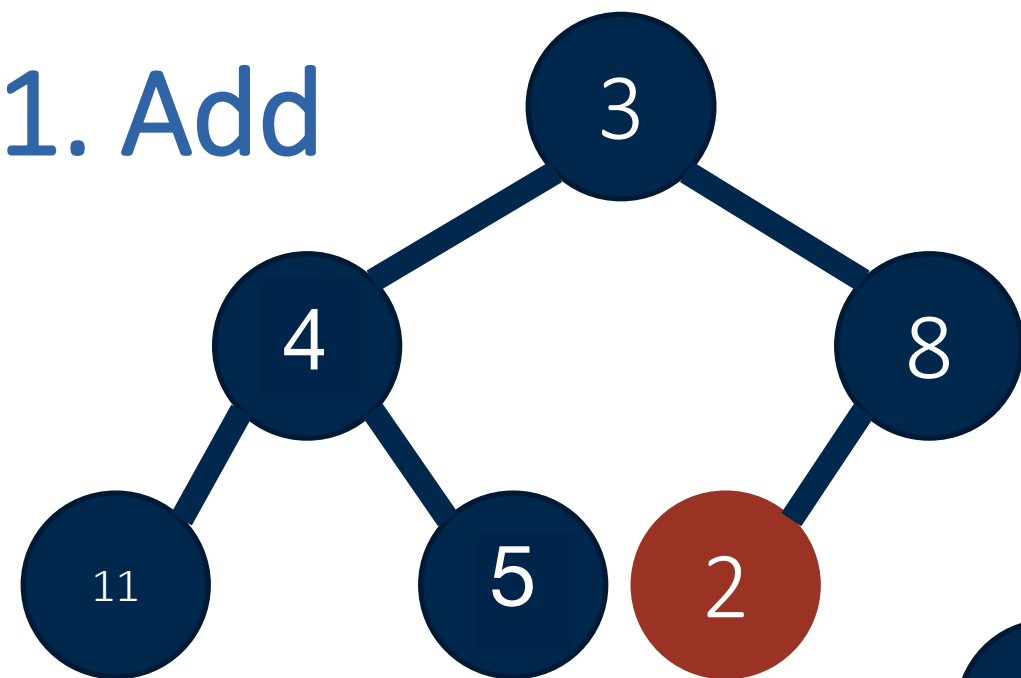
    // Heap condition is true if new element added as root, or if
    // new element is less than or equal to its parent element
    var heaped = (elntIdx <= 0) || (heap[prntIdx] <= heap[elntIdx]);

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

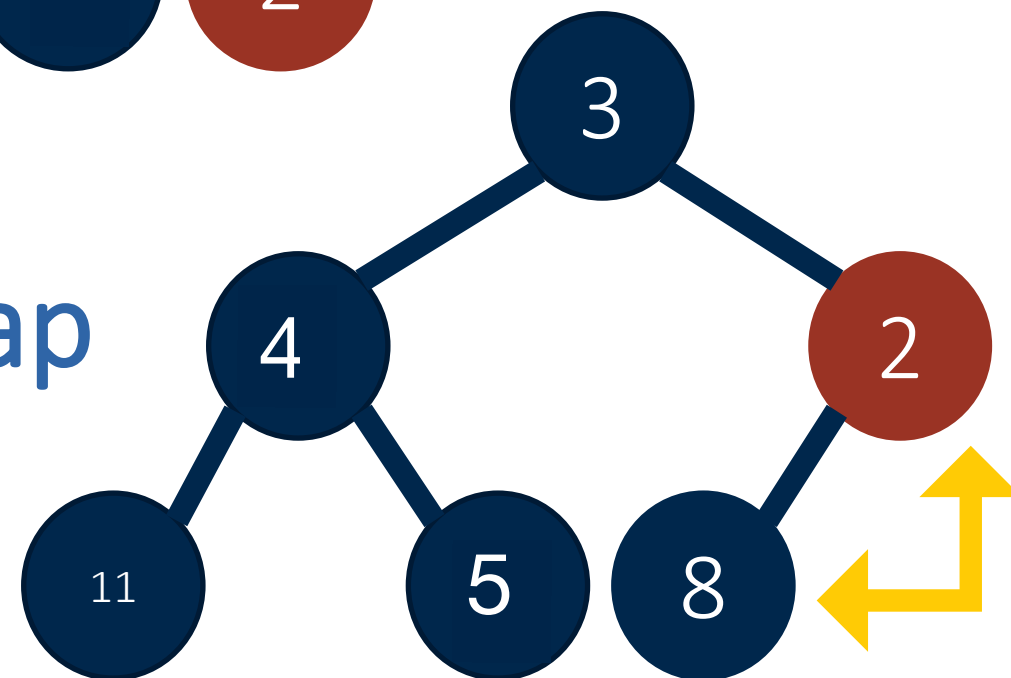
        // TODO: Update index for new_element
        // TODO: Update index for new_element's parent
        //
        // TODO: Update heap condition check
    }
}
```

Heap Insert

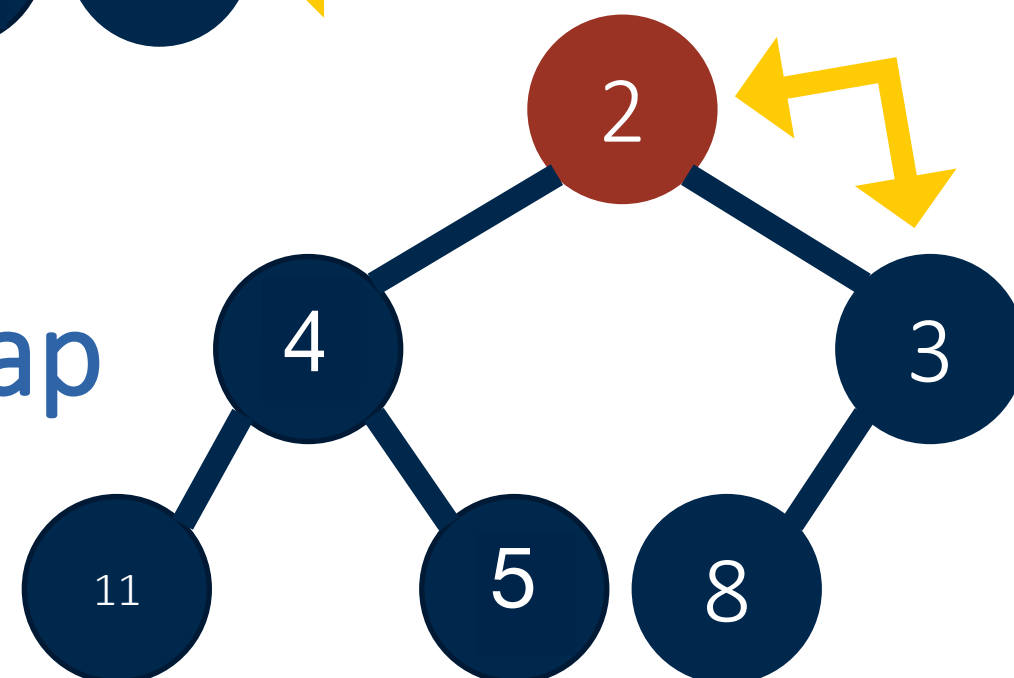
1. Add



2. Swap



3. Repeat swap



```
// 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]);

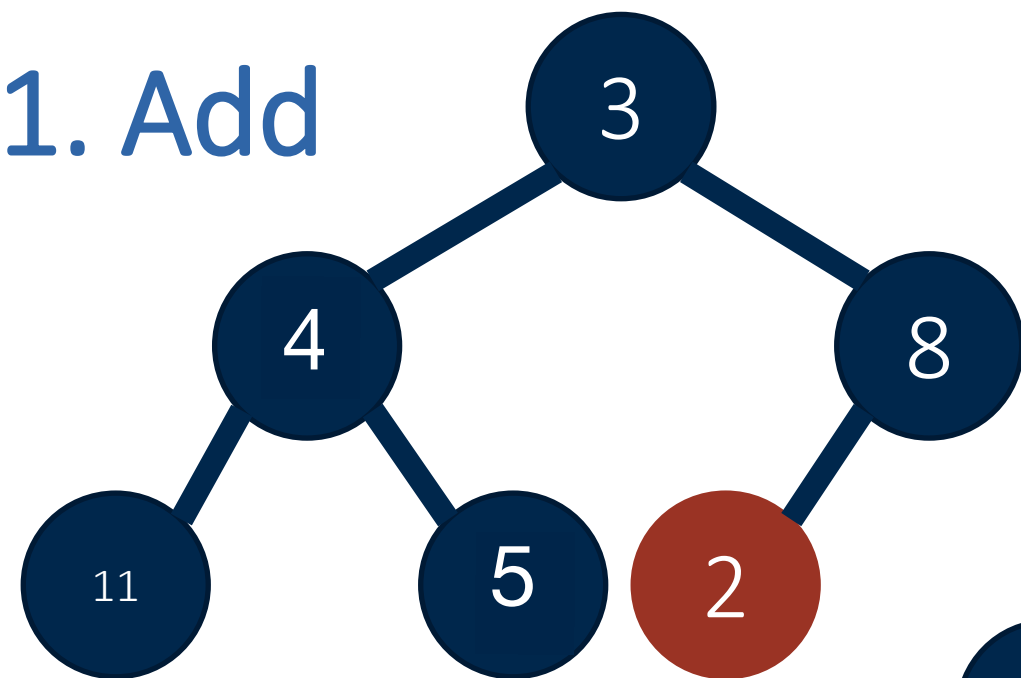
    while (!heaped) {
        // Swap element and parent
        var tmp = heap[prntIdx];
        heap[prntIdx] = heap[elntIdx];
        heap[elntIdx] = tmp;

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

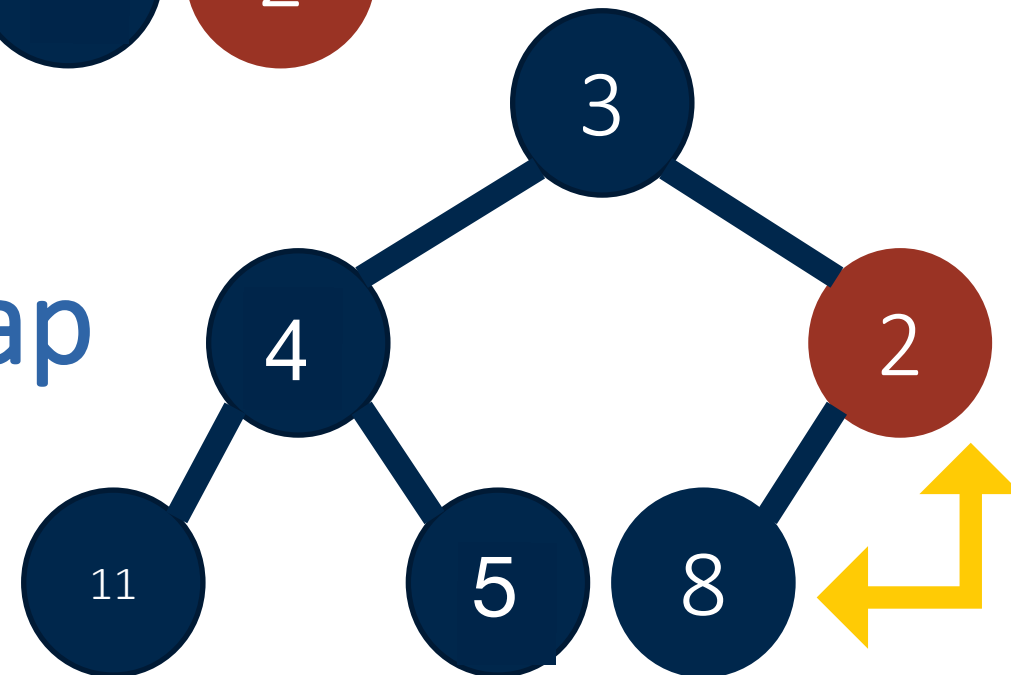
        //
        //      TODO: Update heap condition check
    }
}
```


Heap Insert

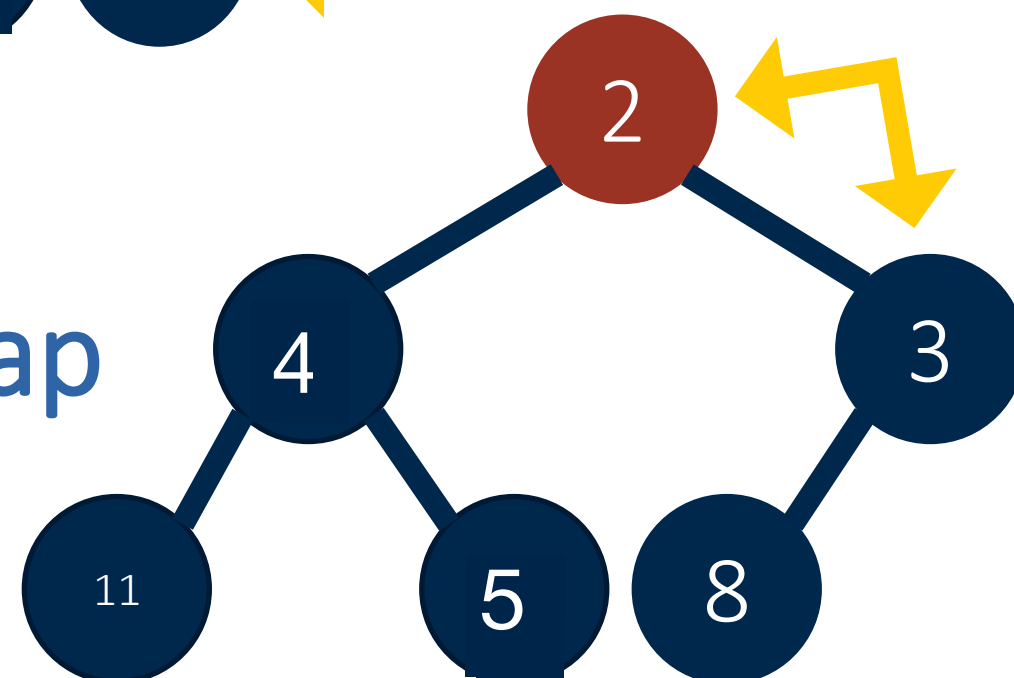
1. Add



2. Swap



3. Repeat swap



```
// 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);

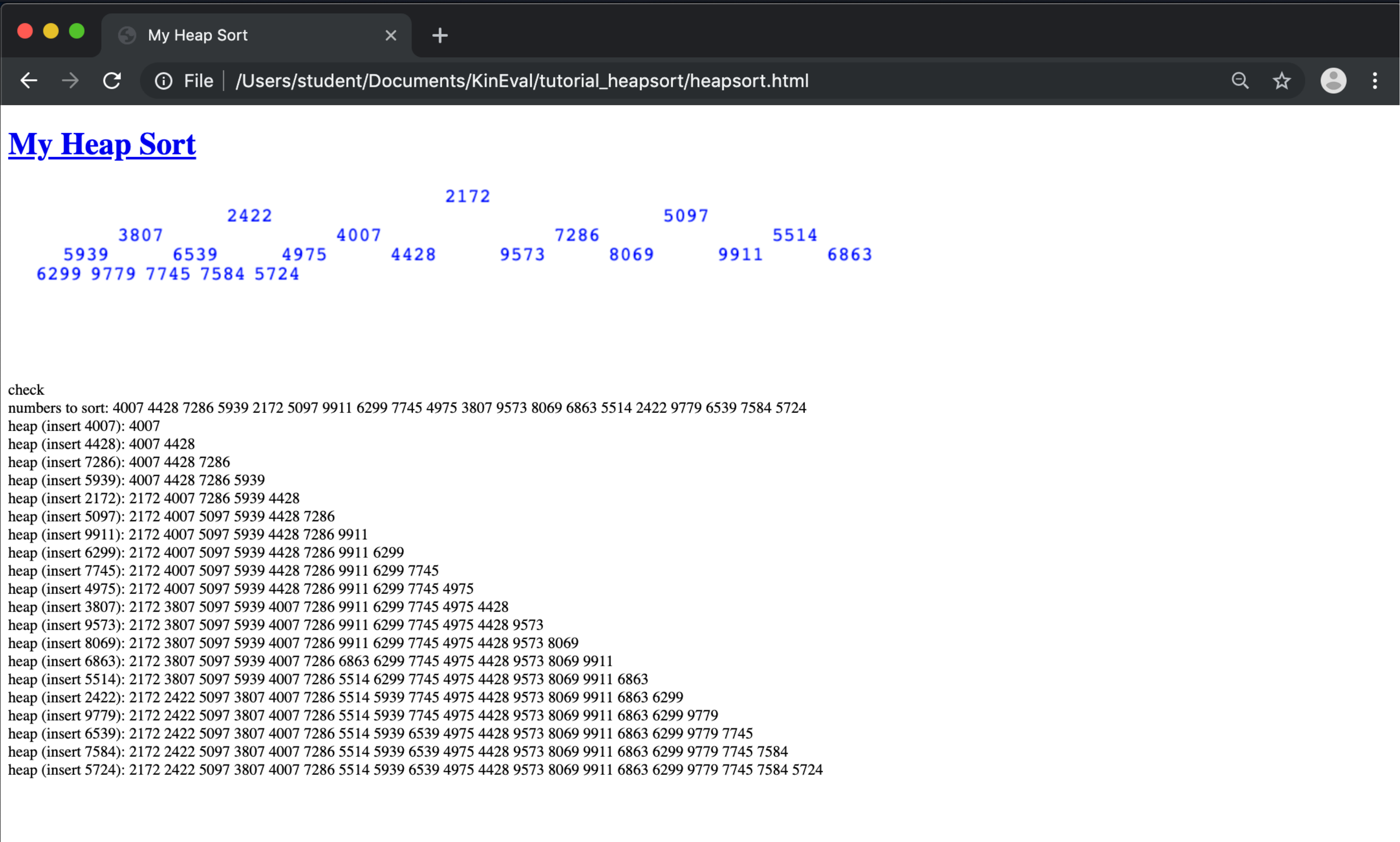
    // Heap condition is true if new element added as root, or if
    // new element is less than or equal to its parent element
    var heaped = (elntIdx <= 0) || (heap[prntIdx] <= heap[elntIdx]);

    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]);
    }
}
```

Heap Insert Result



Lab Takeaways

1. Stencil overview
2. Walk through heap insert function
3. Validate implementation
4. Search canvas introduction
5. Data structure considerations

Search Canvas Infrastructure

infrastructure.js

```
237 // specify start and goal configurations
238 q_start_config = [0,0];
239 q_goal_config = [4,4];
240 q_init = q_start_config;
241 q_goal = q_goal_config;
242
// keep track of the last goal drawn on the canvas
goal = [10000,1000];
scheme = "default";
```

In the `initSearch()` function, which instantiates global variables and starts your algorithms

`q_start_config = q_init`
start location in world

`q_goal_config = q_goal`
goal location in world

`q_init` and `q_goal` can be specified in URL

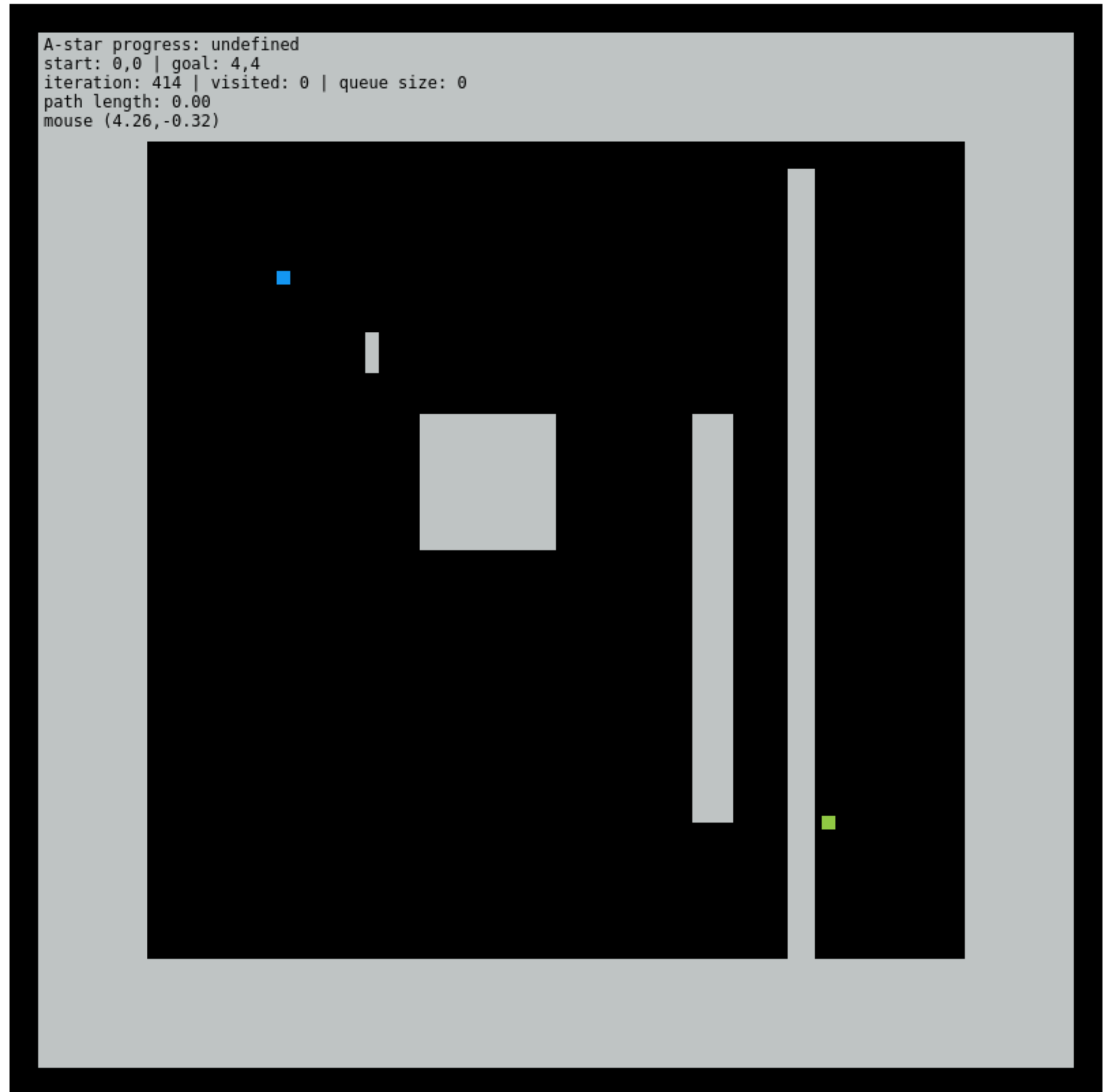
Search Canvas

Open `search_canvas.html` in your browser

Available URL parameters described in `search_canvas.html` file

World coordinates go from $(-2, -2)$ to $(7, 7)$

2D Search Canvas



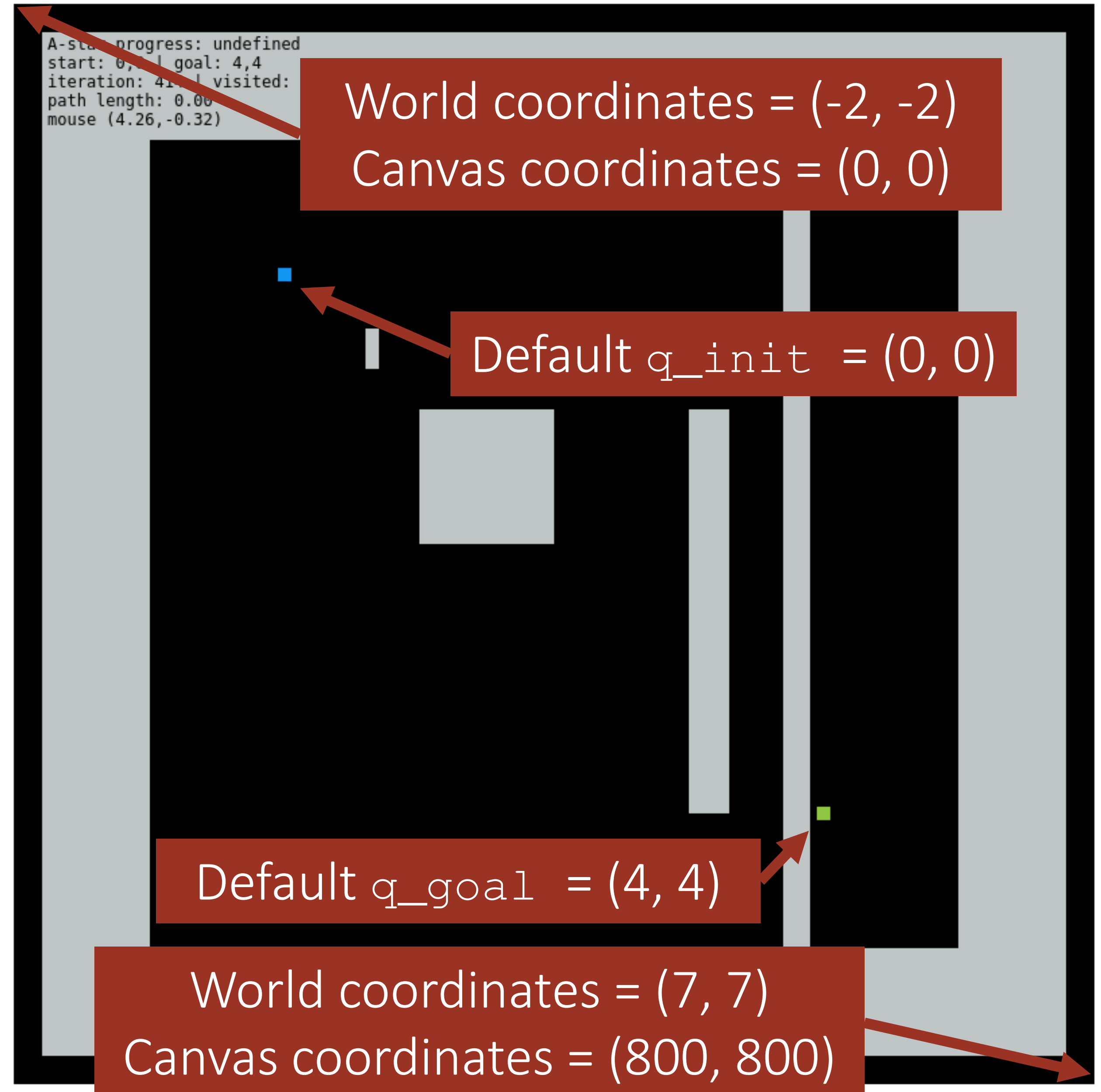
Search Canvas

Open search_canvas.html in your browser

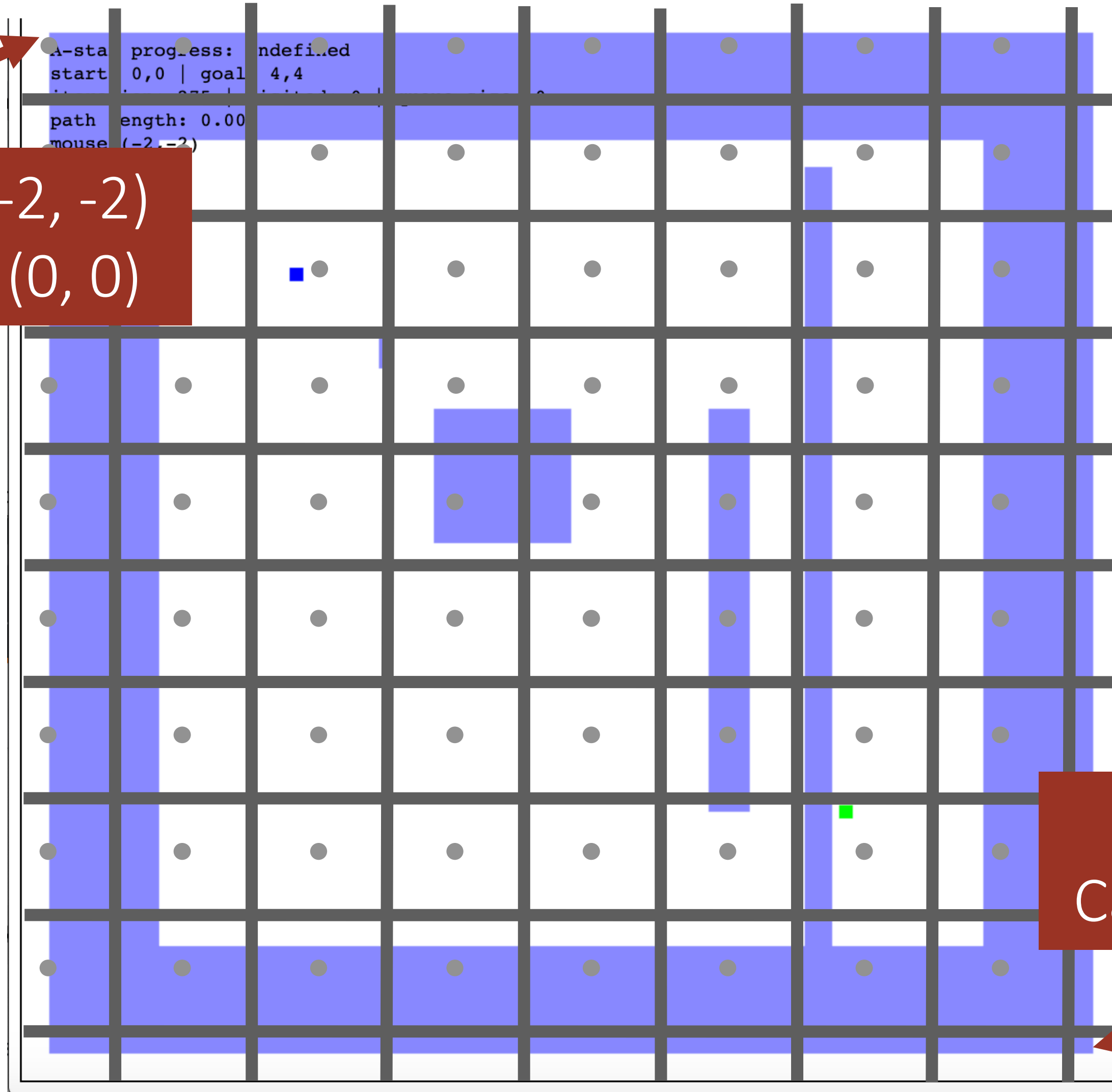
Available URL parameters described in search_canvas.html file

World coordinates go from $(-2, -2)$ to $(7, 7)$

2D Search Canvas



Search Graph

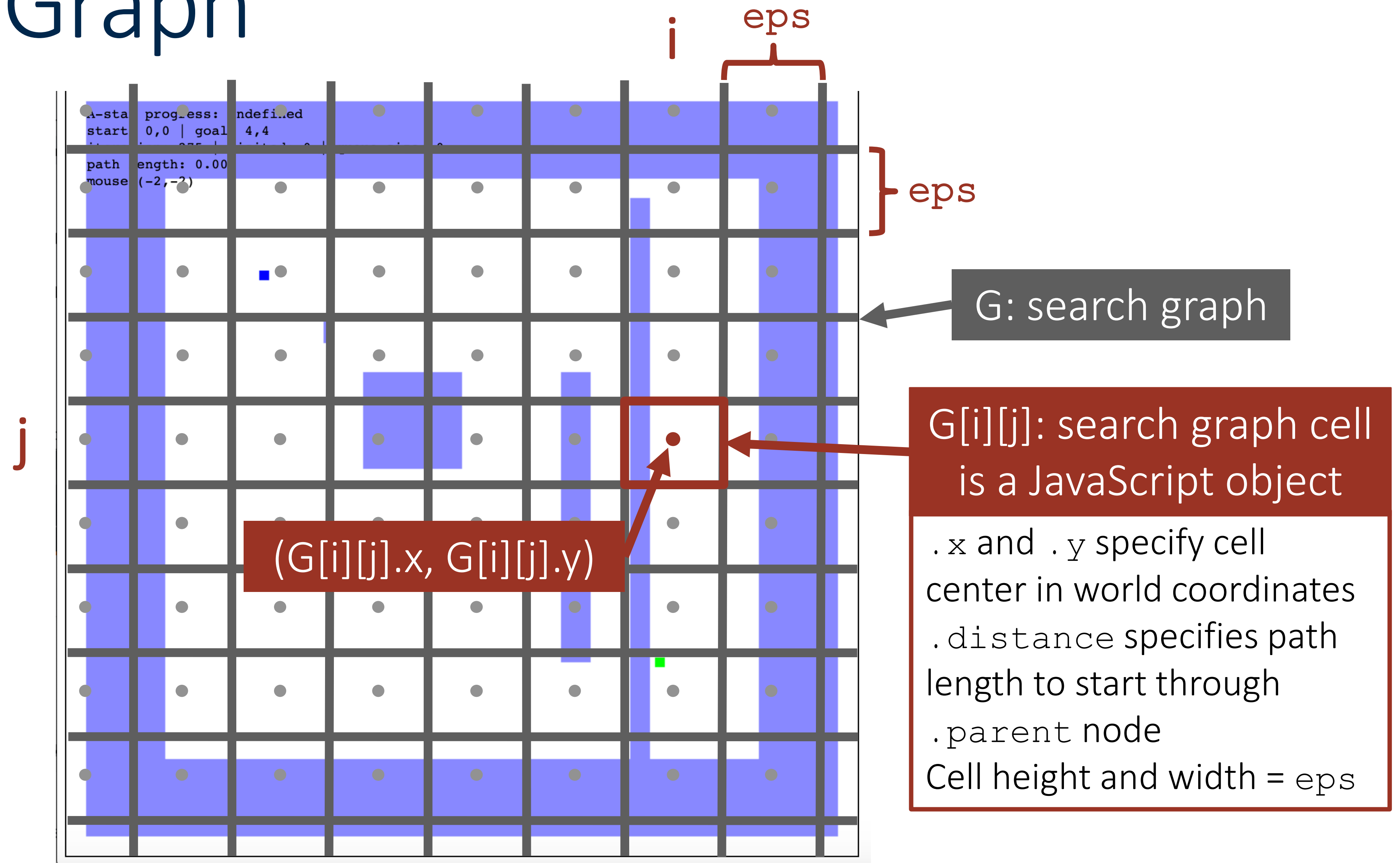


World coordinates = $(-2, -2)$
Canvas coordinates = $(0, 0)$

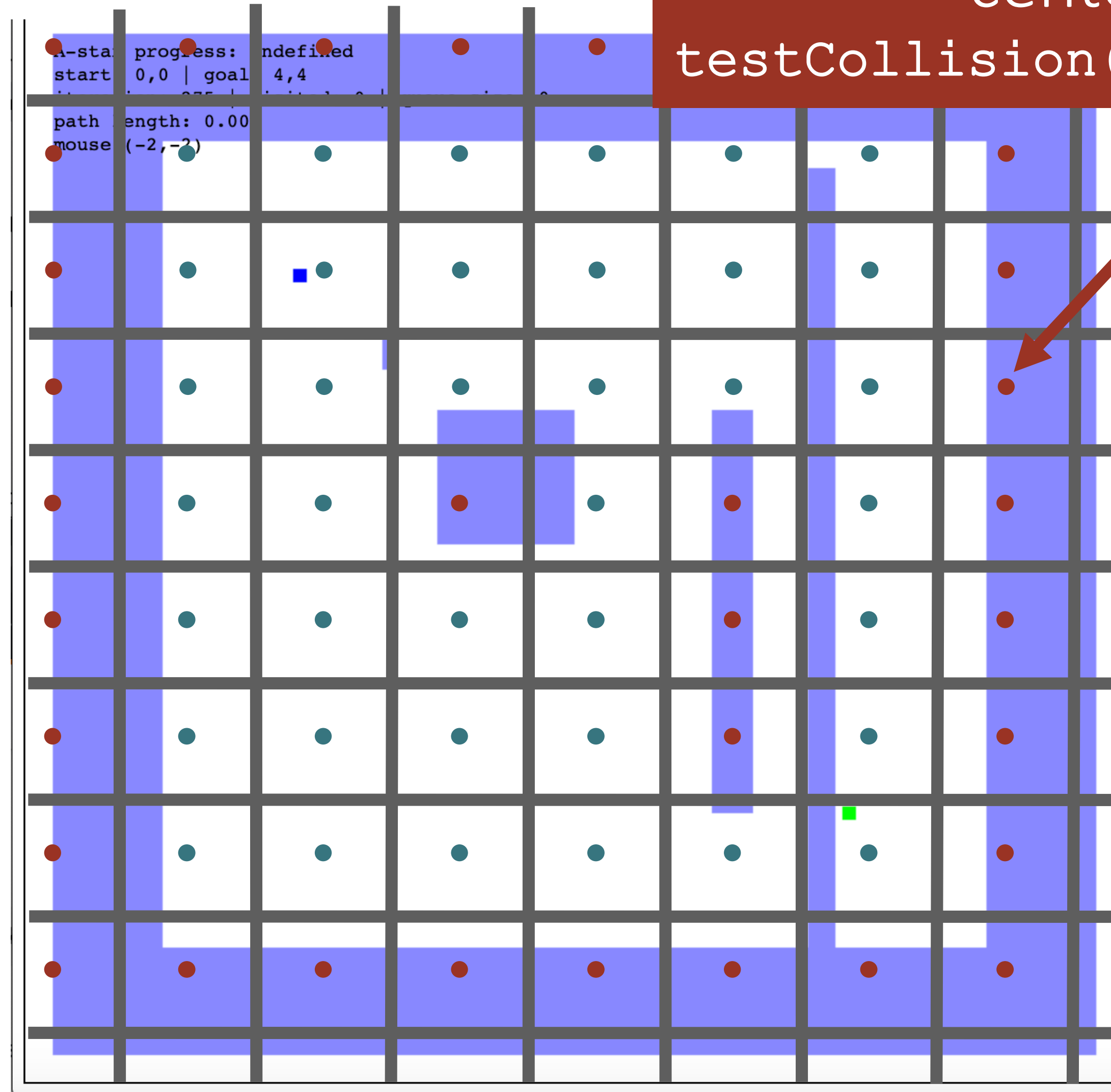
G: search graph

World coordinates = (7, 7)
Canvas coordinates = (800, 800)

Search Graph



Collisions



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

Graph Search Initialization

graph_search.js

```
26 function initSearchGraph() {
27
28     // create the search queue
29     visit_queue = [];
30
31     // initialize search graph as 2D array over configuration space
32     //   of 2D locations with specified spatial resolution
33     G = [];
34     for (iind=0,xpos=-2;xpos<7;iind++,xpos+=eps) {
35         G[iind] = [];
36         for (jind=0,ypos=-2;ypos<7;jind++,ypos+=eps) {
37             G[iind][jind] = {
38                 i:iind,j:jind, // mapping to graph array
39                 x:xpos,y:ypos, // mapping to map coordinates
40                 parent:null, // pointer to parent in graph along motion path
41                 distance:10000, // distance to start via path through parent
42                 visited:false, // flag for whether the node has been visited
43                 priority:null, // visit priority based on fscore
44                 queued:false // flag for whether the node has been queued for visiting
45             };
46
47             // STENCIL: determine whether this graph node should be the start
48             //   point for the search
49         }
```

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

Graph Search Iteration

draw.js

```
217 // render the world to the canvas element
218 drawRobotWorld();
219
220 // make sure the rrt iterations are not running faster than animation update
221 if (search_iterate && (Date.now()-cur_time > min_msec_between_iterations)) {
222
223     // update time marker for last iteration update
224     cur_time = Date.now();
225
226     // update iteration count
227     search_iter_count++;
228
229     // call iteration for the selected search algorithm
230     switch (search_alg) {
231         case "depth-first":
232         case "breadth-first":
233         case "greedy-best-first":
234         case "A-star":
235             search_result = iterateGraphSearch();
236             break;
237         case "RRT":
```

In `animate()` function,
which is responsible for
calling your iterate functions

Graph Search Iteration

graph_search.js

```
53 function iterateGraphSearch() {  
54  
55  
56     // STENCIL: implement a single iteration of a graph search algorithm  
57     //   for A-star (or DFS, BFS, Greedy Best-First)  
58     //   An asynch timing mechanism is used instead of a for loop to avoid  
59     //   blocking and non-responsiveness in the browser.  
60     //  
61     //   Return "failed" if the search fails on this iteration.  
62     //   Return "succeeded" if the search succeeds on this iteration.  
63     //   Return "iterating" otherwise.  
64     //  
65     //   Provided support functions:  
66     //  
67     //   testCollision - returns whether a given configuration is in collision  
68     //   drawHighlightedPathGraph - draws a path back to the start location  
69     //   draw_2D_configuration - draws a square at a given location  
70 }
```

Ensure your implementations are isolated to **single search steps!** Including excessive loops may cause browser to become unresponsive.

Once search has completed, turn off iteration: `search_iterate = false;`

Lab Takeaways

1. Stencil overview
2. Walk through heap insert function
3. Validate implementation
4. Search canvas introduction
5. Data structure considerations