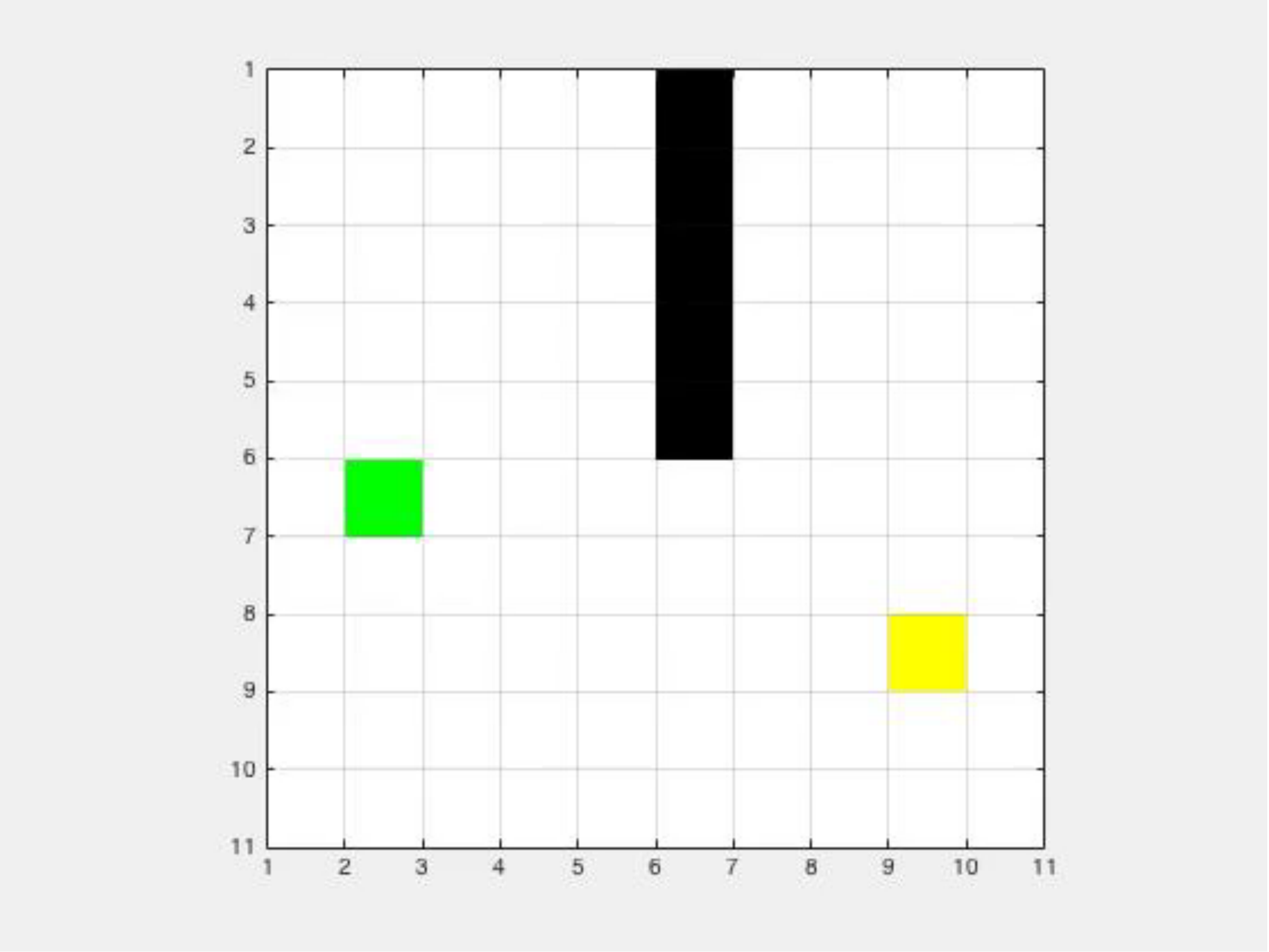
A-Star Procedure

SECTION 1.4

Dijkstra/Grassfire Algorithm



Dijkstra/Grassfire Algorithm

- When applied on a grid graph where all of the edges have the same length, Dijkstra's algorithm and the grassfire procedure have similar behaviors.
- They both explore nodes in order based on their distance from the starting node until they encounter the goal.

A* Search

- A* Search attempts to improve upon the performance of grassfire and Dijkstra by incorporating a heuristic function that guides the path planner.

Heuristic Functions

- Heuristic functions are used to map every node in the graph to a non-negative value
- Heuristic Function Criteria:
  - o $H(goal) = 0$
  - o For any 2 adjacent nodes x and y

    $H(x) <= H(y) + d(x,y)$

    $d(x,y)$ = weight/length of edge from x to y
- These properties ensure that for all nodes, n
  - o $H(n) <=$ length of shortest path from n to goal.

Example Heuristic Functions

- For path planning on a grid the following 2 heuristic functions are often used
  - Euclidean Distance

$$H(x_n, y_n) = \sqrt{((x_n - x_g)^2 + (y_n - y_g)^2)} \tag{1}$$

  - Manhattan Distance

$$H(x_n, y_n) = |(x_n - x_g)| + |(y_n - y_g)| \tag{2}$$

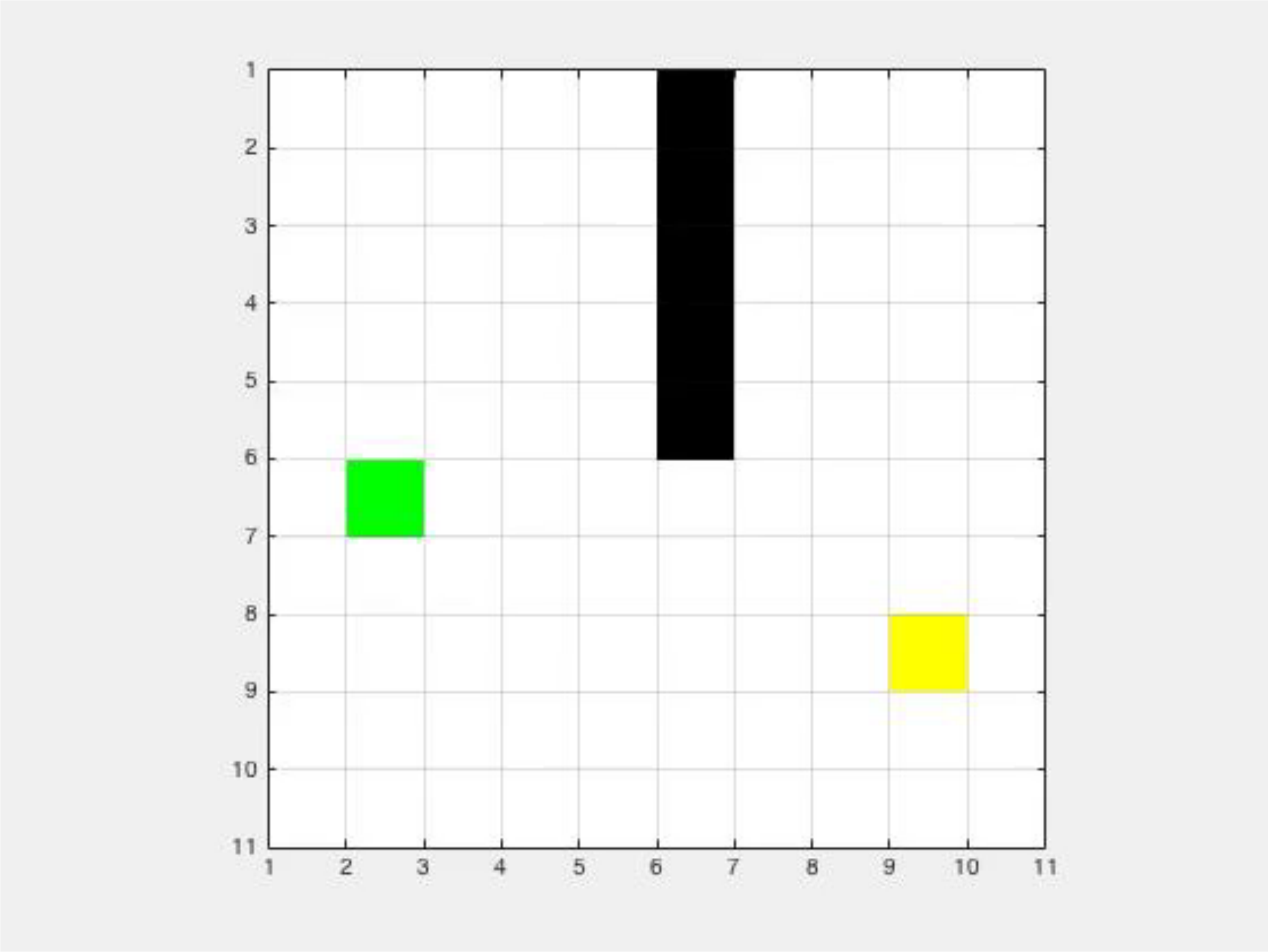
  - where (x_n, y_n) denotes the coordinates of the node n and (x_g, y_g) denotes the coordinate of the goal

A* algorithm – pseudo code

- For each node n in the graph
  - n.f = Infinity, n.g = Infinity
- Create an empty list.
- start.g = 0, start.f = H(start) add start to list.
- While list not empty
  - Let current = node in the list with the smallest f value, remove current from list
  - If (current == goal node) report success
  - For each node, n that is adjacent to current

```
If (n.g > (current.g + cost of edge from n to current))
    n.g = current.g + cost of edge from n to current
    n.f = n.g + H(n)
    n.parent = current
    add n to list if it isn't there already
```

Dijkstra’s Algorithm



A* Algorithm