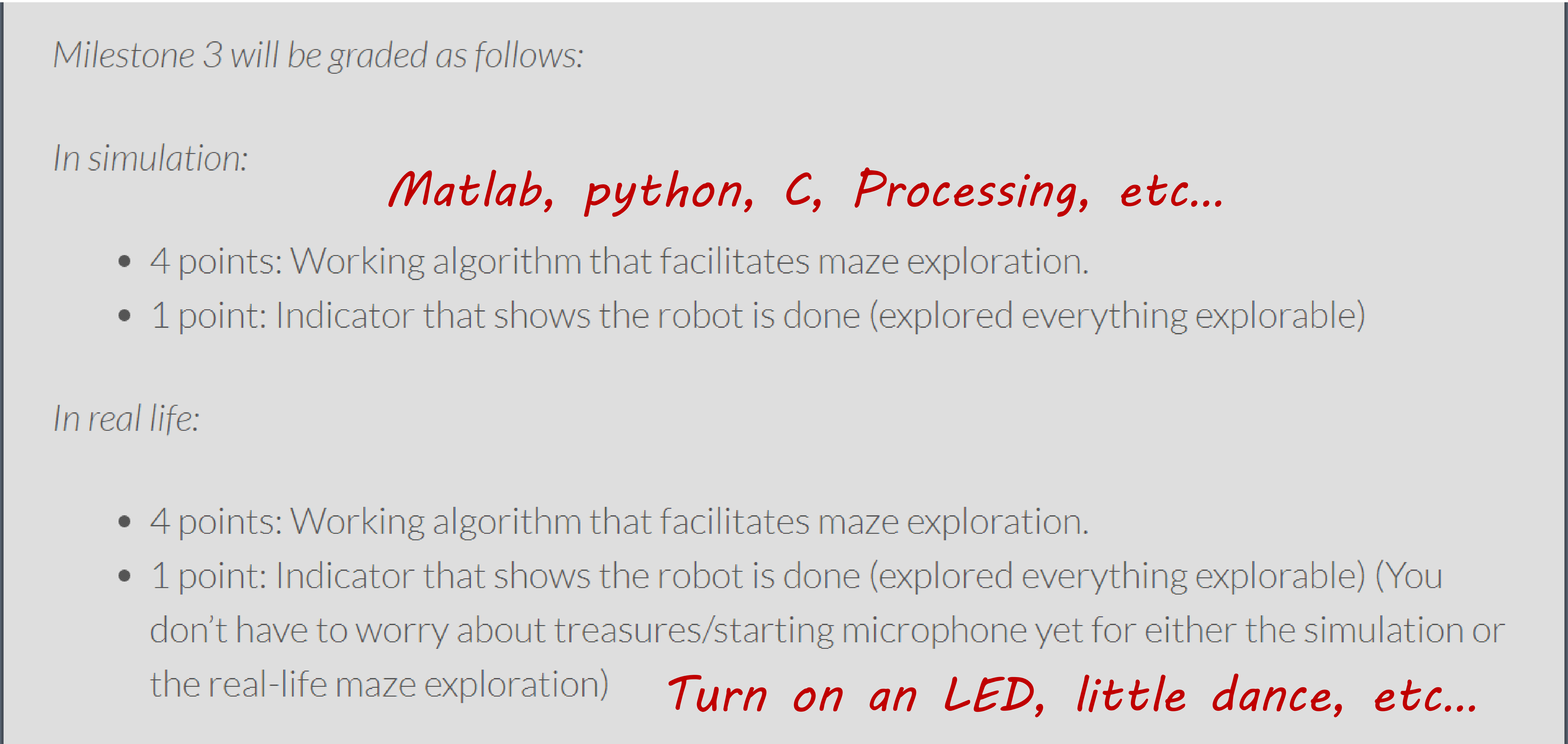
Algorithms, Exploration and Search

- Nov 6th-Nov 10th



Milestone 3 will be graded as follows:

In simulation:

Matlab, python, C, Processing, etc...

- 4 points: Working algorithm that facilitates maze exploration.
- 1 point: Indicator that shows the robot is done (explored everything explorable)

In real life:

- 4 points: Working algorithm that facilitates maze exploration.
- 1 point: Indicator that shows the robot is done (explored everything explorable) (You don't have to worry about treasures/starting microphone yet for either the simulation or the real-life maze exploration)

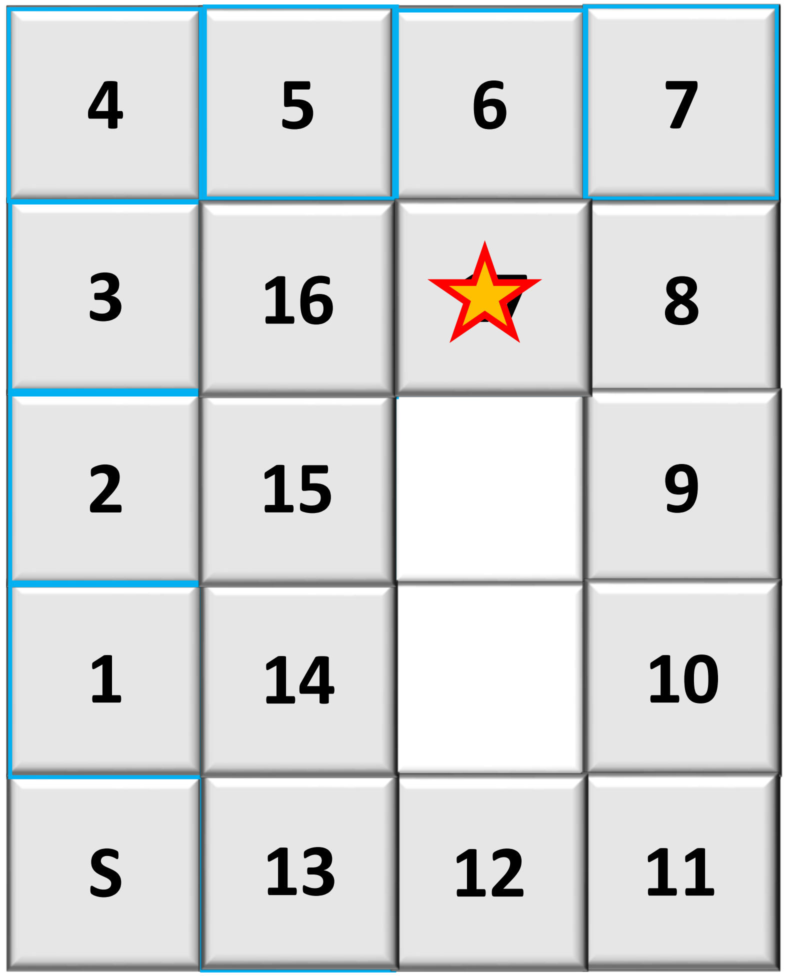
Turn on an LED, little dance, etc...

Algorithms and Search

- Brute force search
- Depth First Search (DFS)

Search order: N, E, S, W

Find a treasure

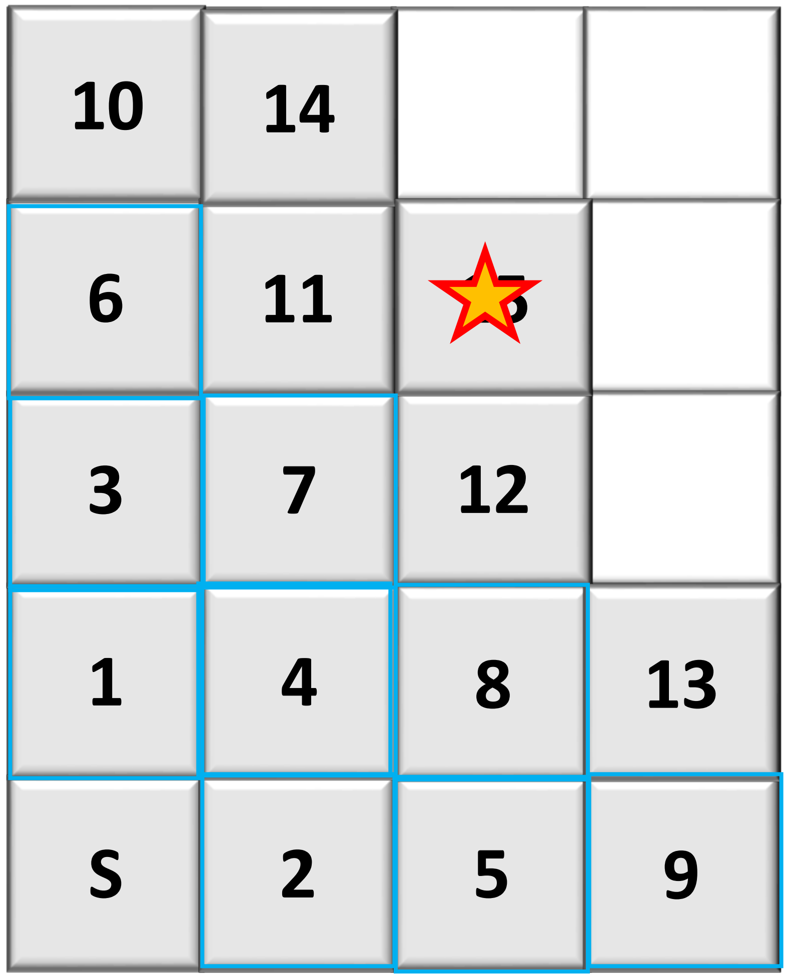


Search order: N, E, S, W

Algorithms and Search

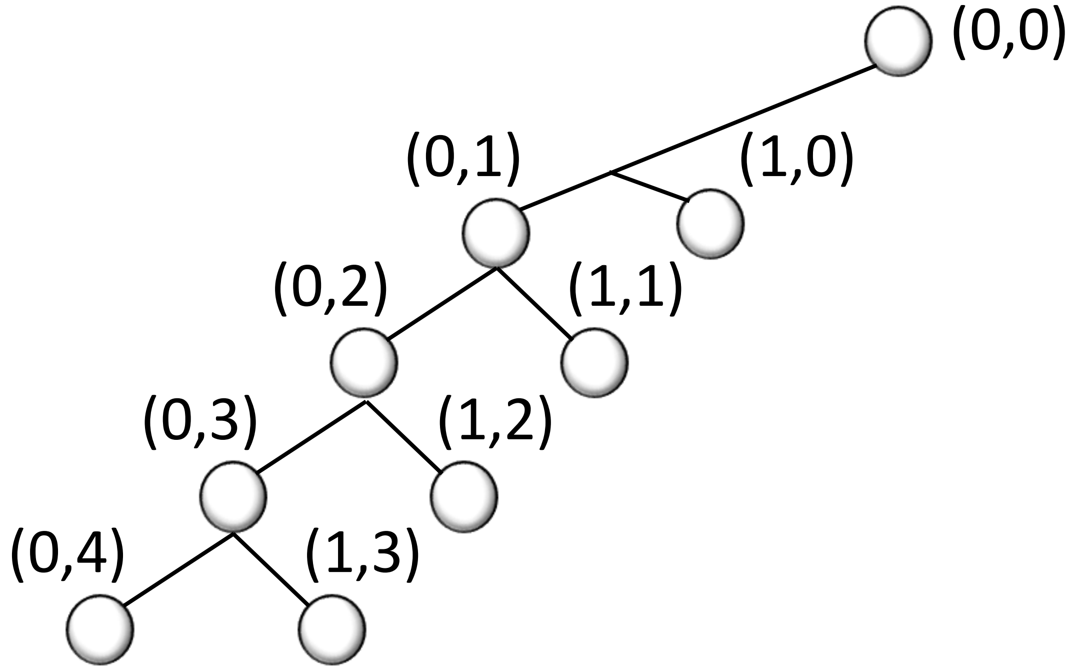
- ~~Brute force search~~
- Depth First Search (DFS)
- Breadth First Search (BFS)

Find a treasure



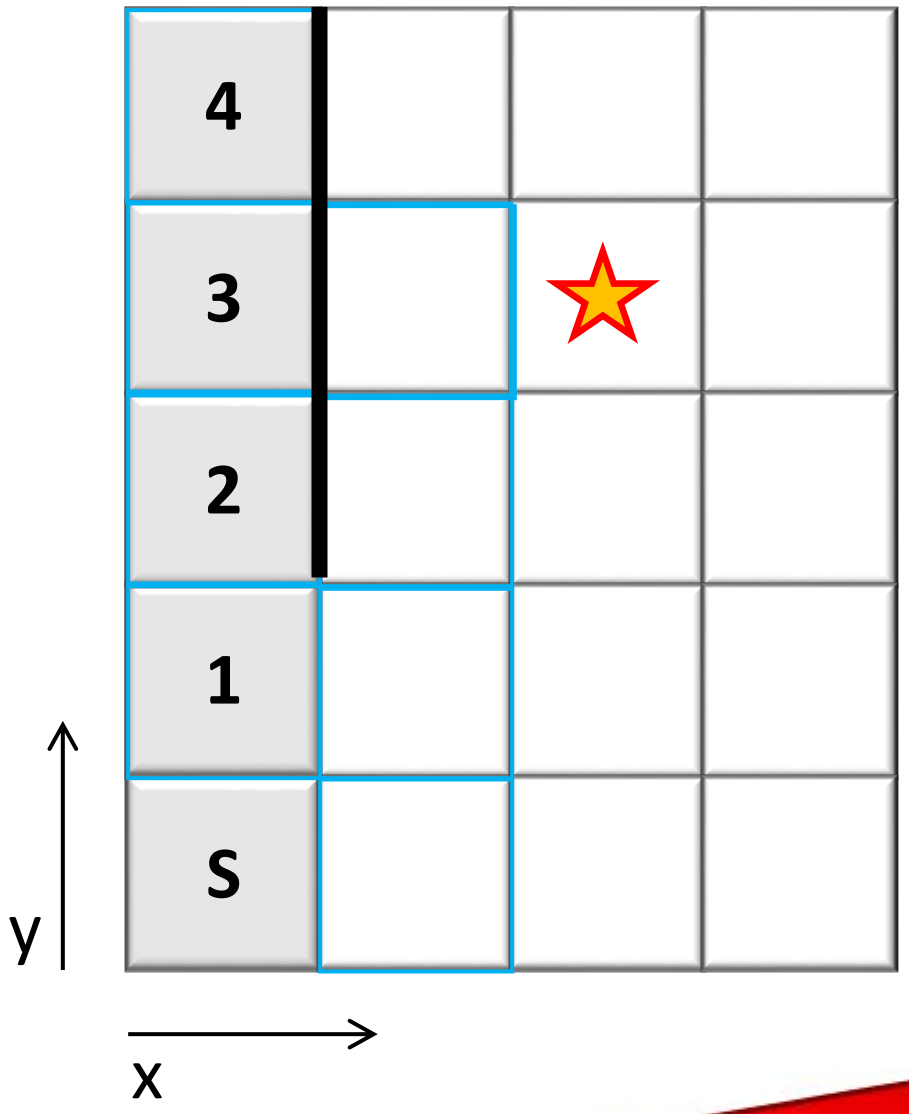
Algorithms and Search

- ~~Brute force search~~
- Depth First Search (DFS)



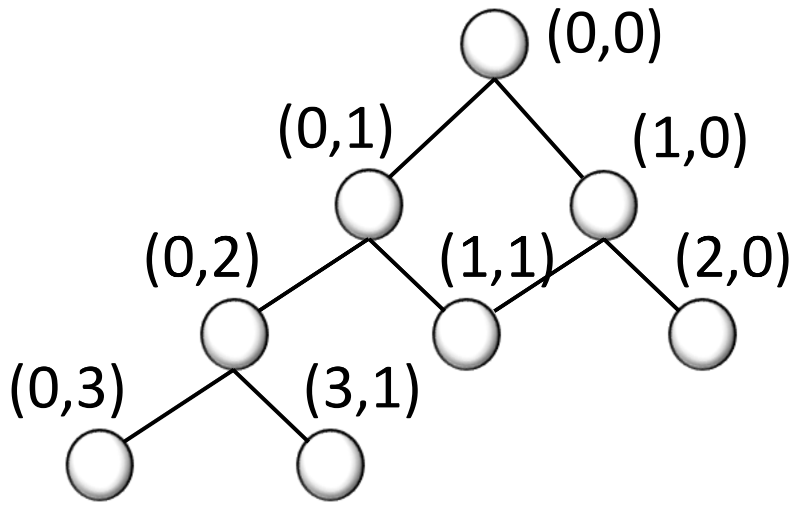
and so on…

Find a treasure



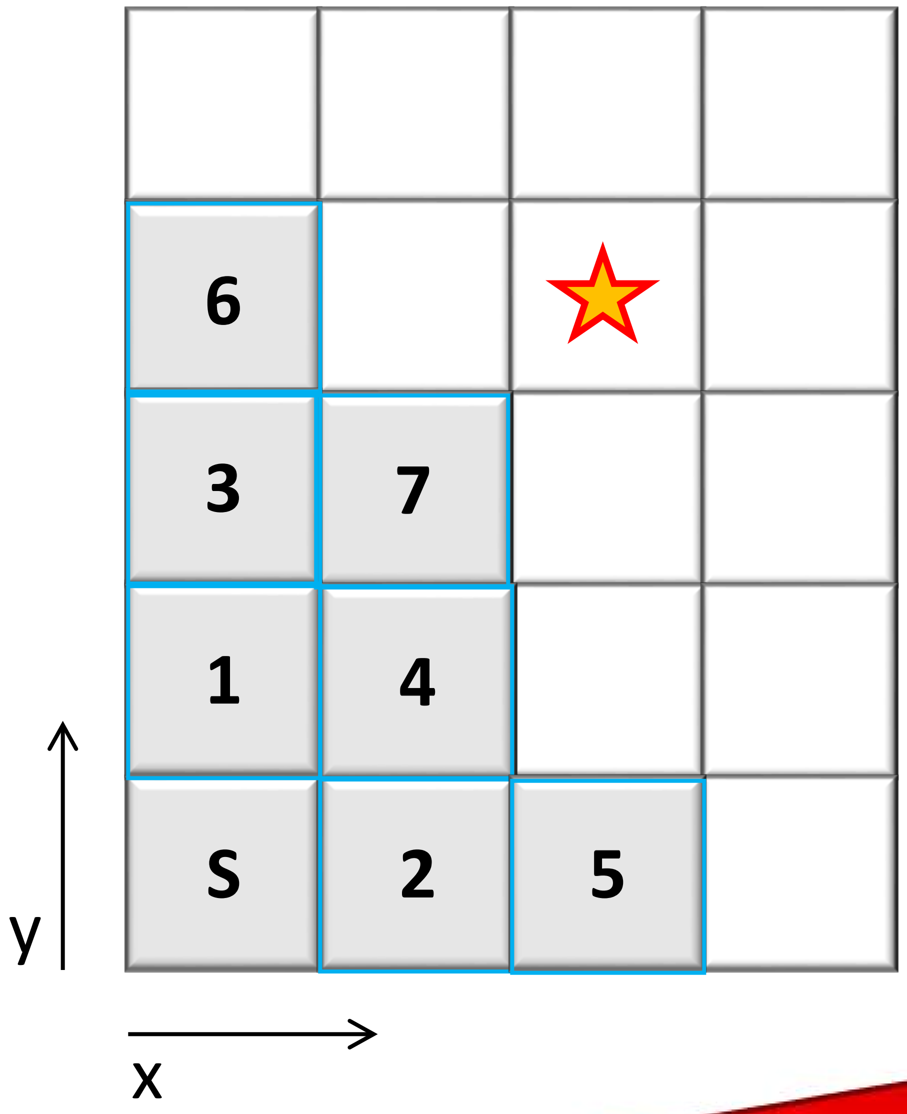
Algorithms and Search

- ~~Brute force search~~
- Depth First Search (DFS)
- Breadth First Search (BFS)



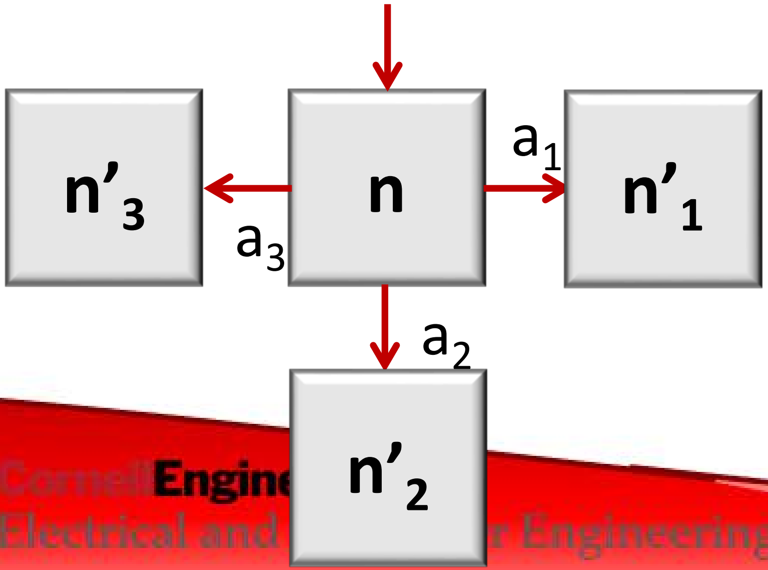
and so on…

Find a treasure

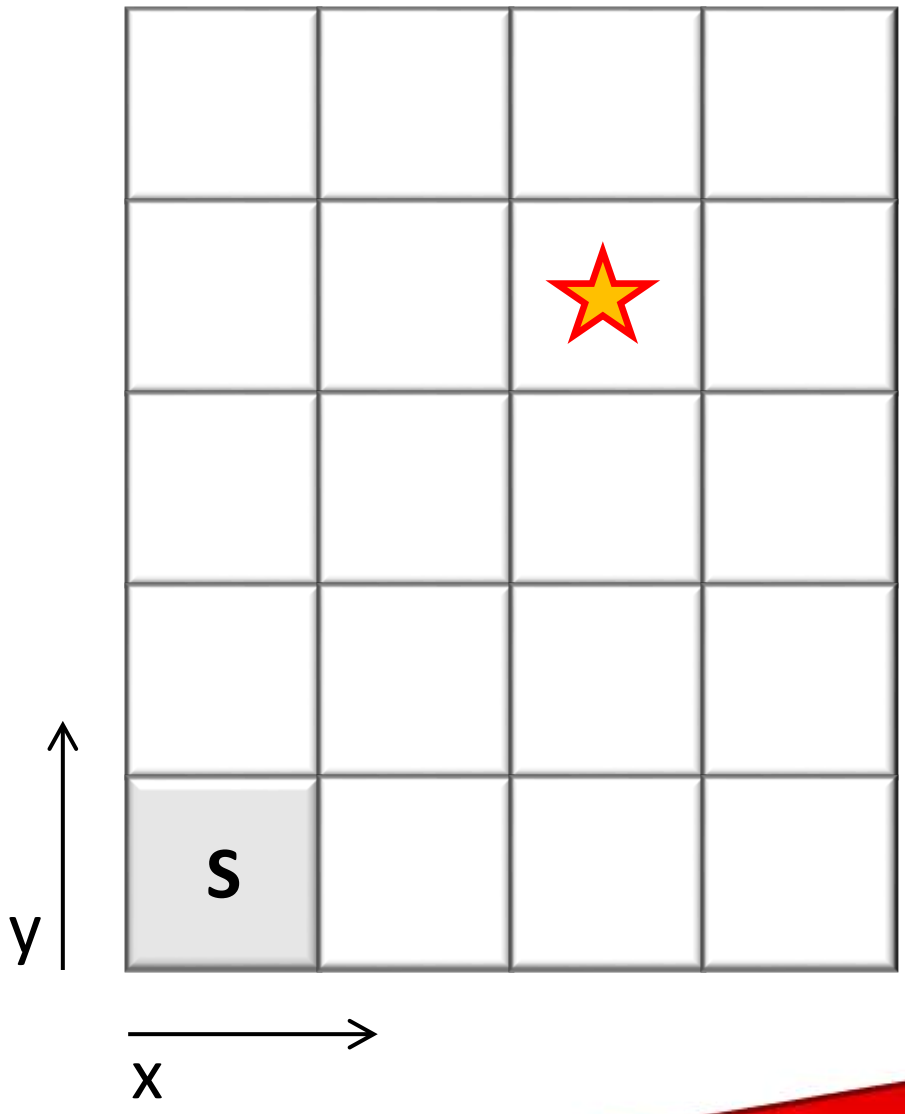


Algorithms and Search

- ~~Brute force search~~
- Depth First Search (DFS)
- Breadth First Search (BFS)
- Common structure
  - For every node, n
  - you have a set of actions, a
  - that moves you to a new node, n’

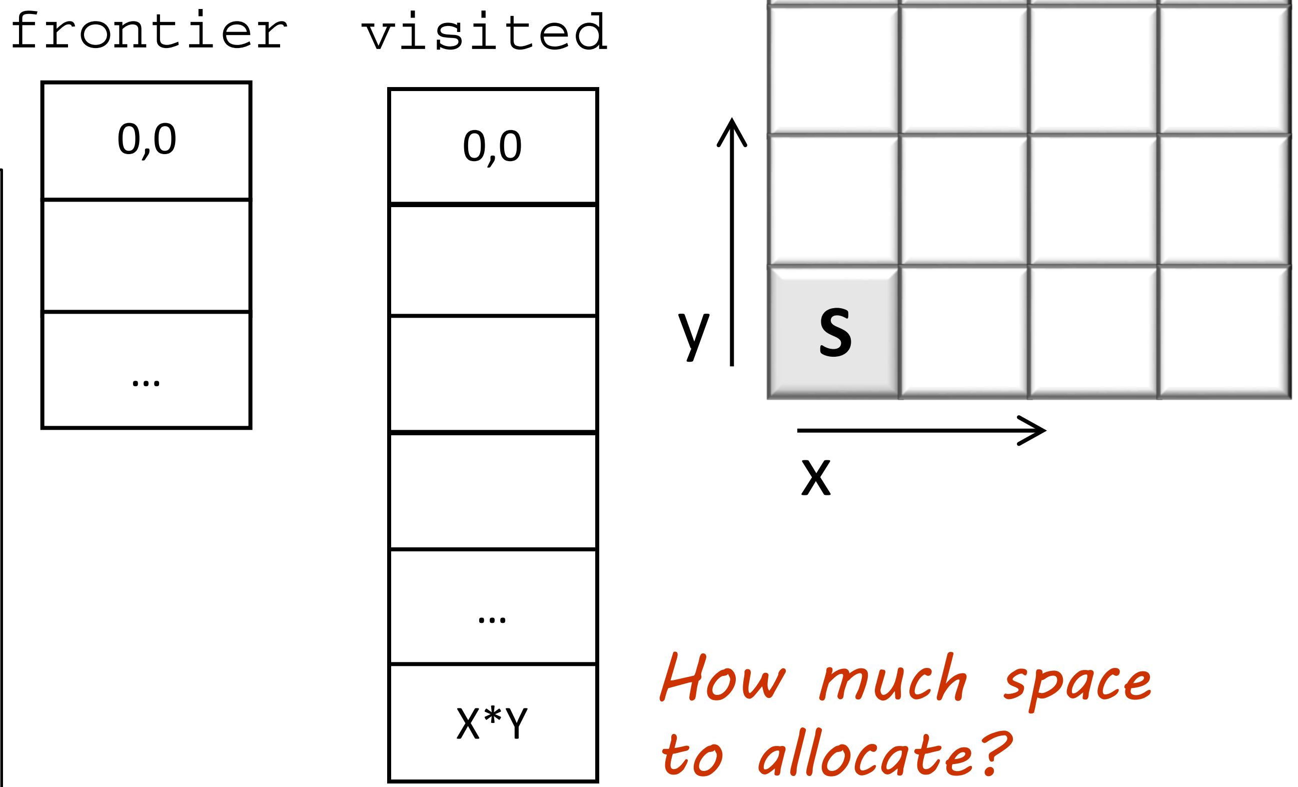


Find a treasure



General Search Algorithm

```
n = state(init)
frontier.append(n)
while(frontier not empty)
    n = pull state from frontier
    if n = goal, return solution
    for all actions in n
        n' = a(n)
        if n' not visited
            append n' to visited
            append n' to frontier
```

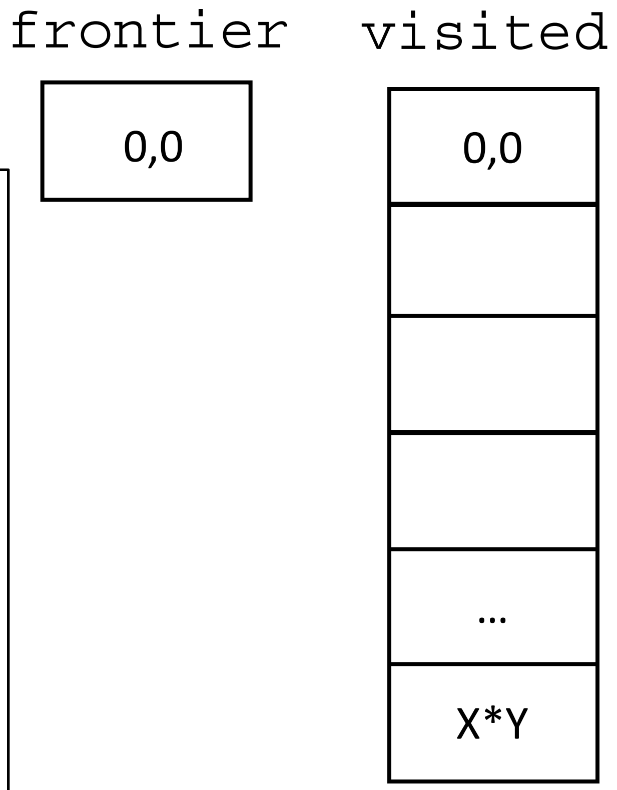


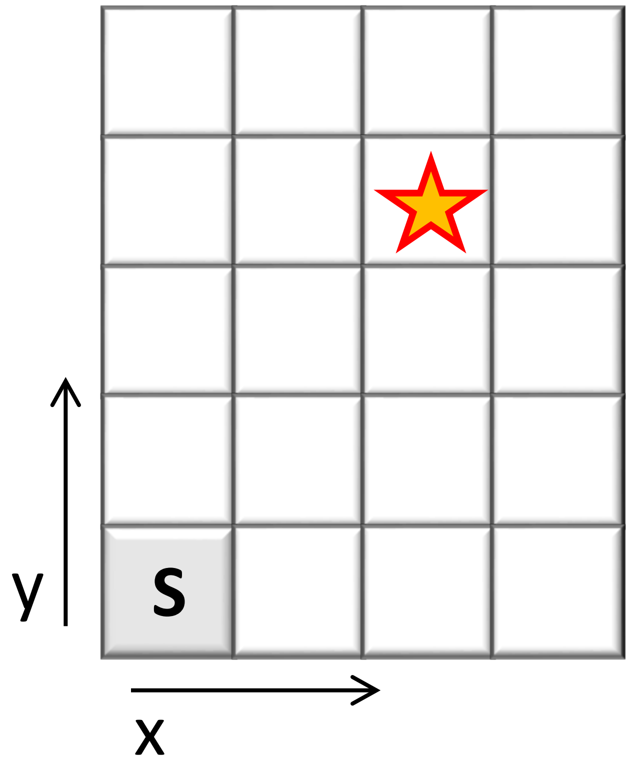
How much space to allocate?

General Search Algorithm

- **Depth First Search (DFS)**

```
n = state(init)
frontier.append(n)
while(frontier not empty)
    n = pull state from frontier
    if n = goal, return solution
    for all actions in n
        n' = a(n)
        if n' not visited
            append n' to visited
            append n' to frontier
```

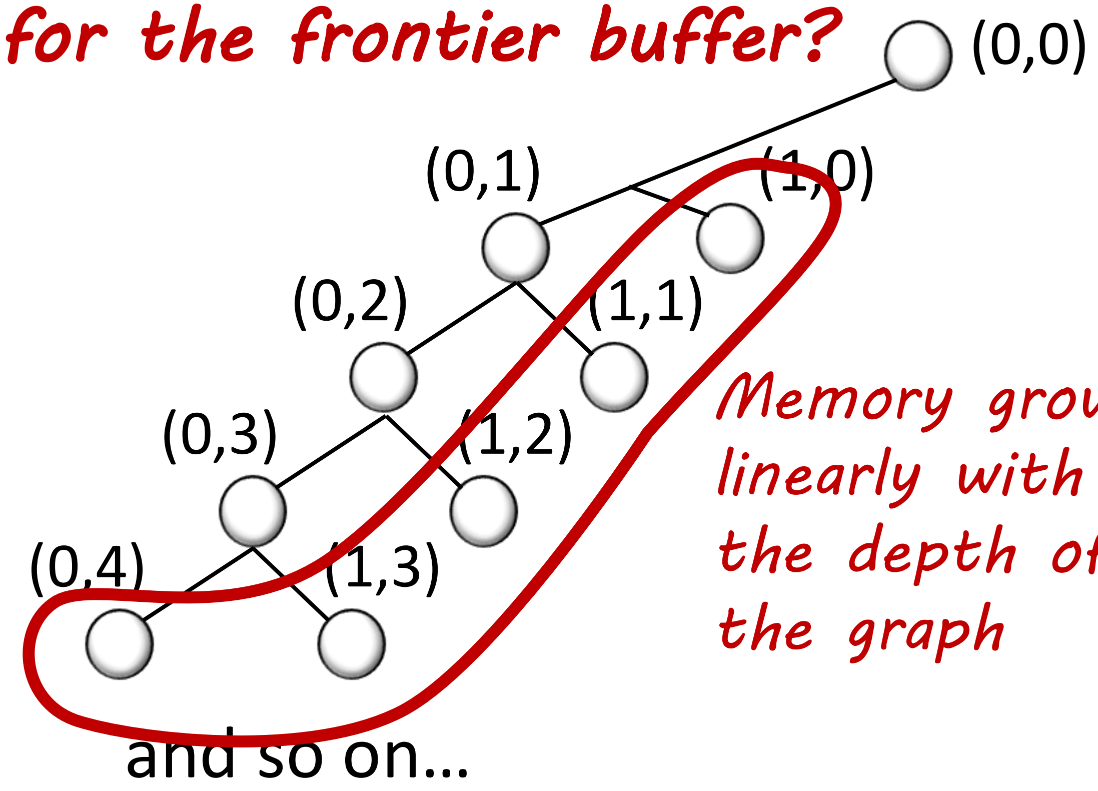




General Search Algorithm

- **Depth First Search (DFS)**

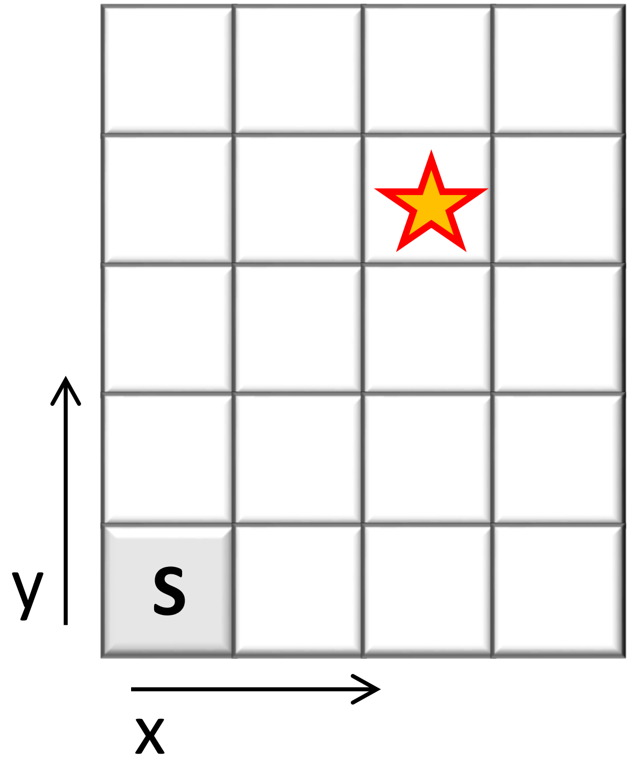
How much memory to allocate for the frontier buffer?



Memory grows linearly with the depth of the graph

frontier
0,4
1,3
1,2
1,1
1,0

visited
0,0
0,1
0,2
0,3
…
X*Y



Type of Buffer?

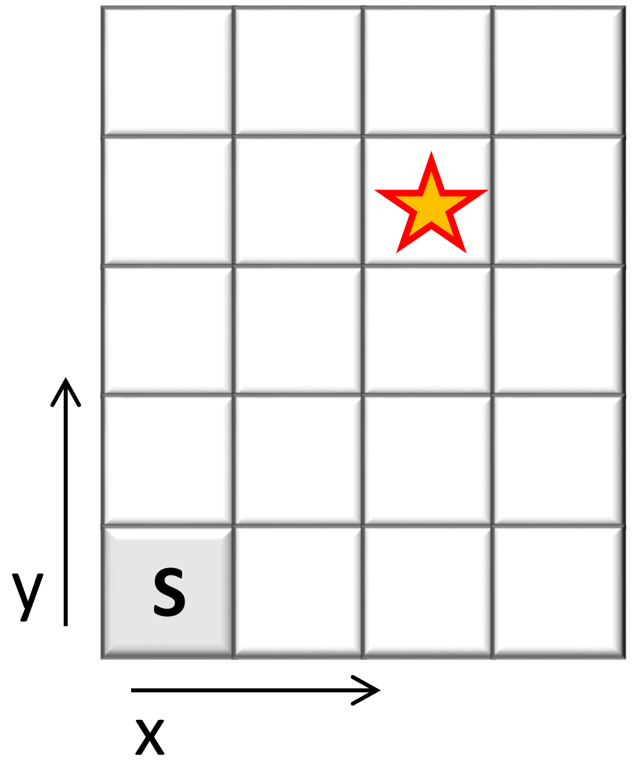
Last-In First-Out (LIFO) Buffer

General Search Algorithm

- Depth First Search (DFS)
- **Breadth First Search (BFS)**

```
n = state(init)
frontier.append(n)
while(frontier not empty)
    n = pull state from frontier
    if n is goal, return solution
    for all actions in n
        n' = a(n)
        if n' not visited
            append n' to visited
            append n' to frontier
```

frontier	visited
0,0	0,0
0,1	0,1
1,0	1,0
0,2	0,2
1,1	
2,0	…
0,3	



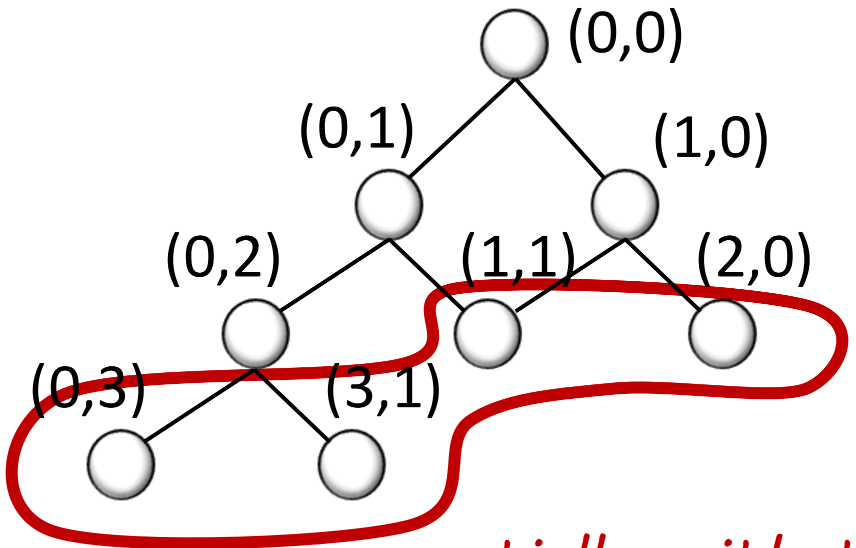
Type of Buffer?

First-In First-Out (FIFO) Buffer

General Search Algorithm

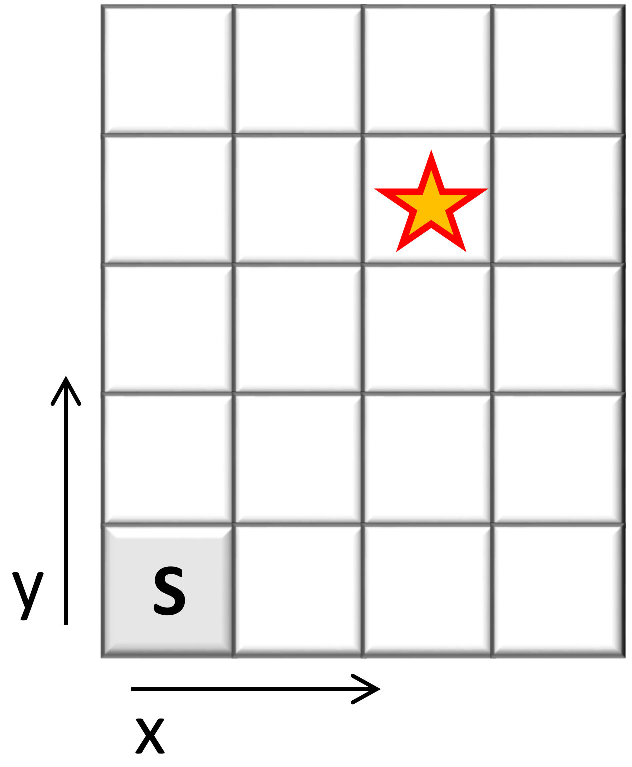
- Depth First Search (DFS)
- **Breadth First Search (BFS)**

How much memory to allocate for the frontier buffer?



Memory grows exponentially with the depth of the graph

frontier	visited
0,0	0,0
0,1	0,1
1,0	1,0
0,2	0,2
1,1	
2,0	…
0,3	



Type of Buffer?

First-In First-Out (FIFO) Buffer

Algorithms and Search

- What is the most efficient way to explore the full maze?
  - Hint: Your robot takes time to move!
  - Double hint: Your robot takes time to turn!

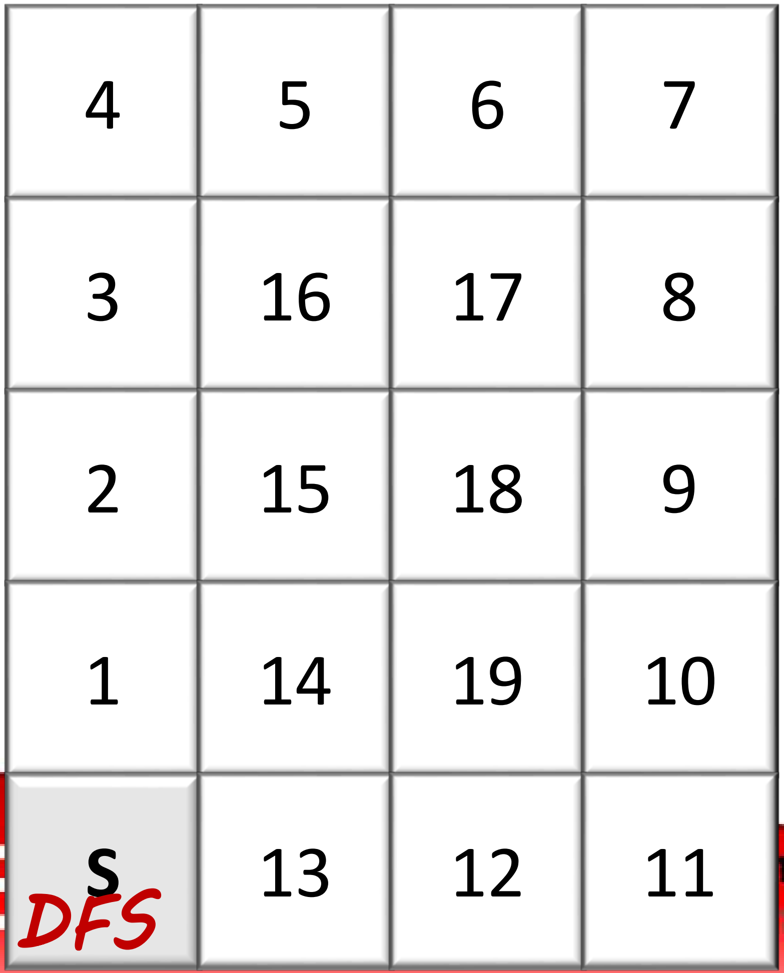
4	5	6	7
3	16	17	8
2	15	18	9
1	14	19	10
S DFS	13	12	11

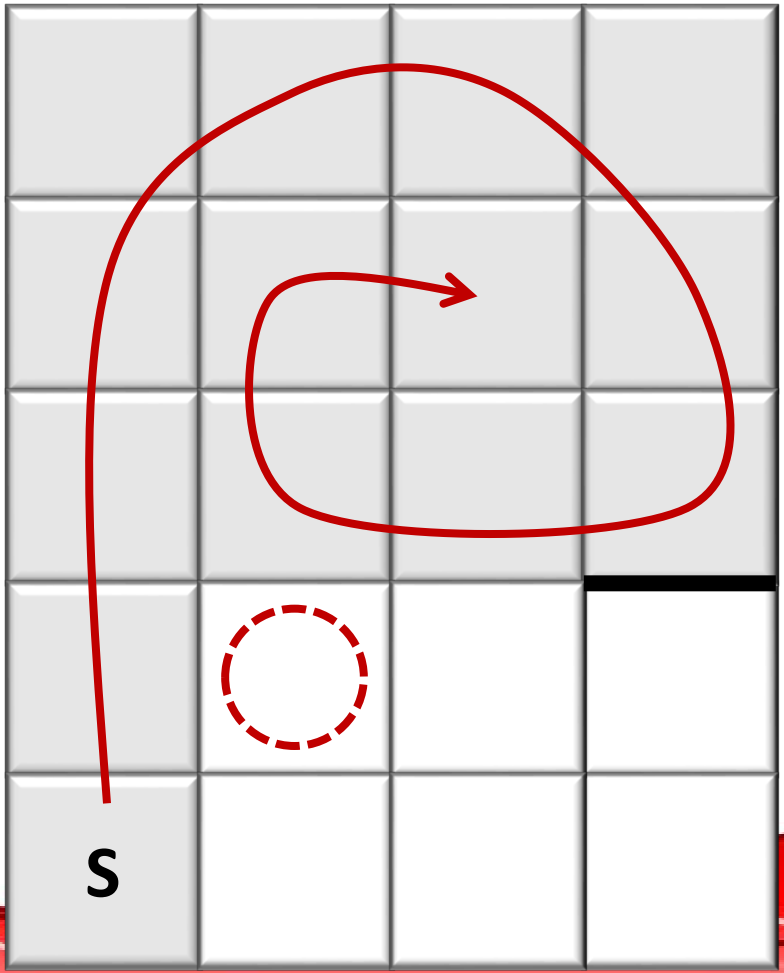
13	14	18	19
6	12	15	17
3	7	11	16
1	4	8	10
S BFS	2	5	9

S			

Algorithms and Search

- Can we be done already?...
  - No! Your robot also has to get to the frontier.



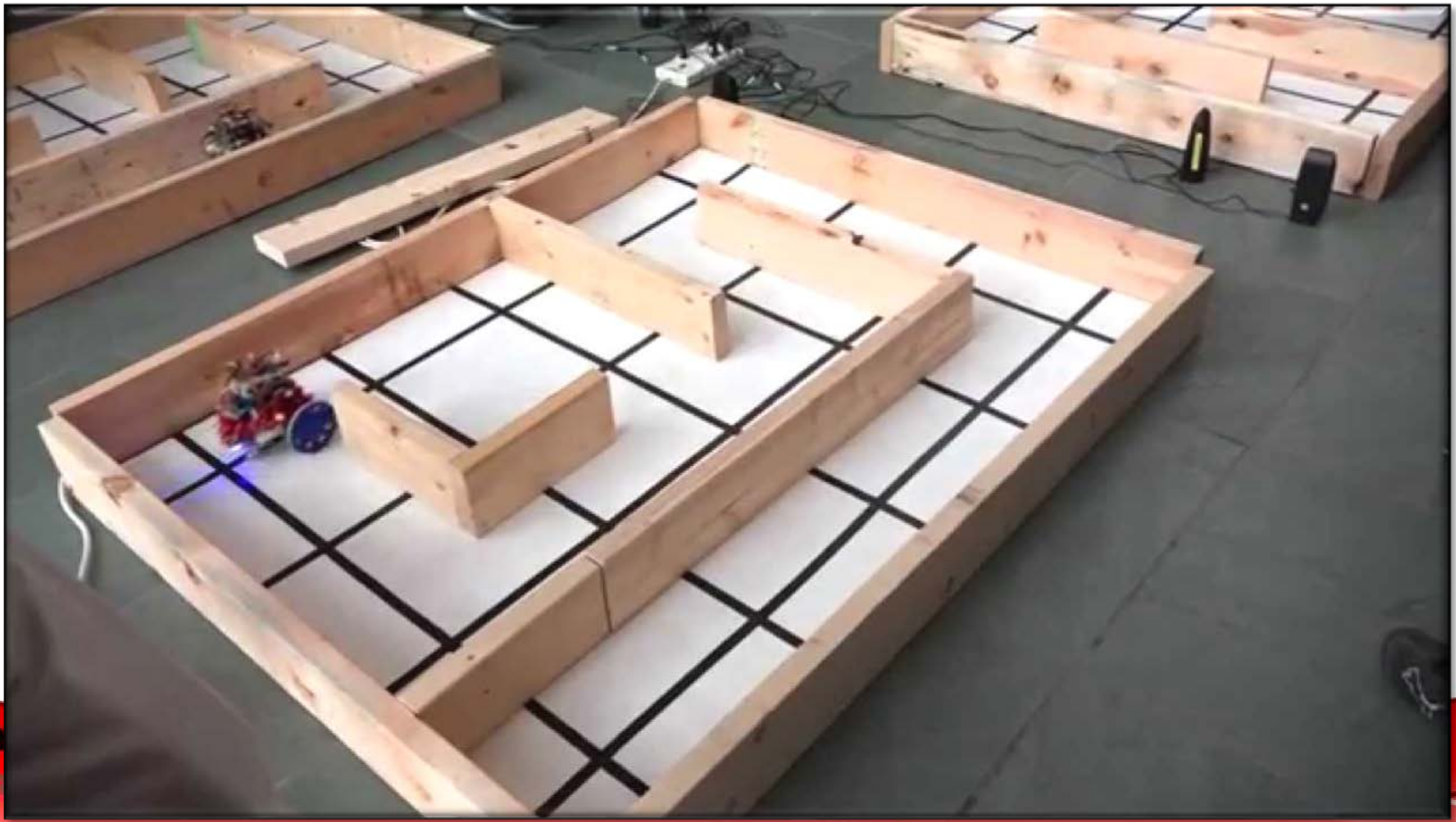


Algorithms and Search

Nomenclature:

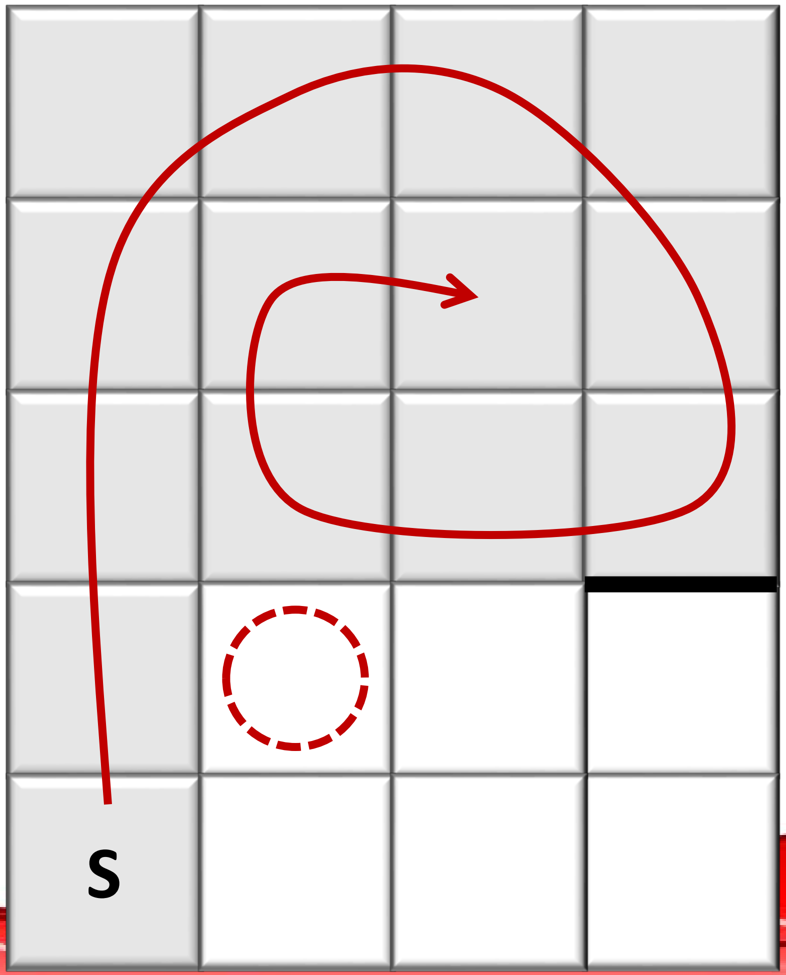
- *Exploration:* pick next site to search
- *Search problem:* find the shortest path to that site

Sequence of actions to get there



Algorithms and Search

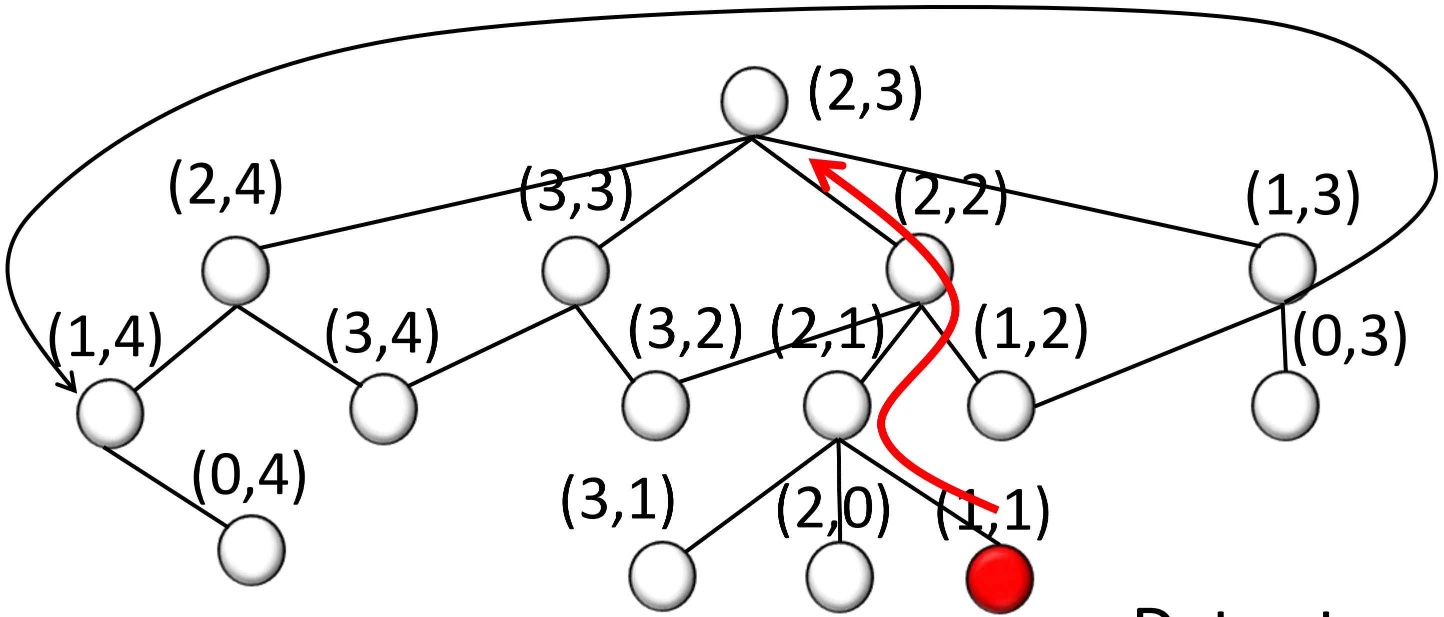
- Can we be done already?...
  - No! Your robot also has to get to the frontier.
  - What algorithm is more efficient to get you to the frontier?



Search order: N, E, S, W

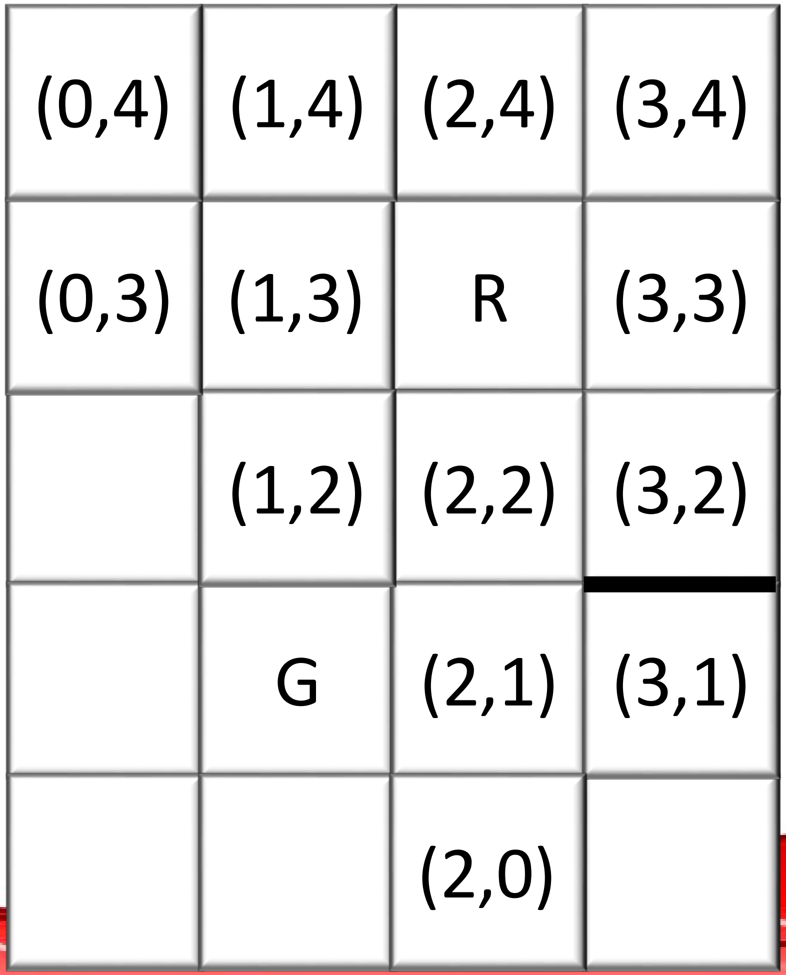
Breadth First Search and Dijkstra’s Algorithm

- Breadth-First Search



Data structure

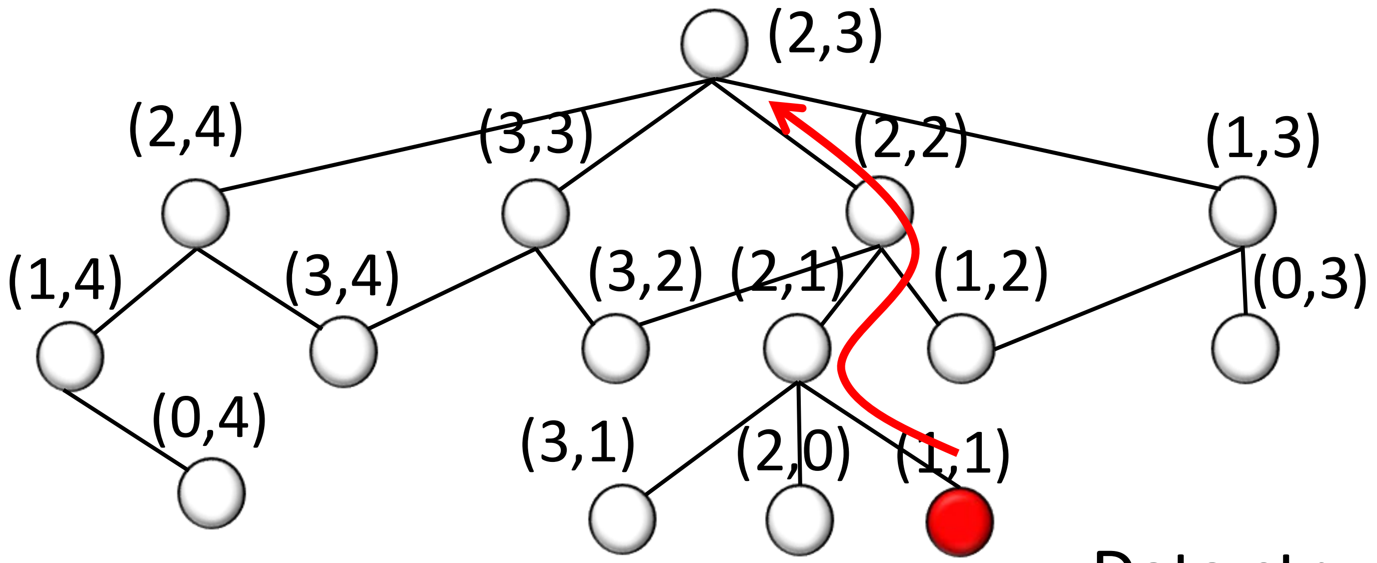
- n.state
- n.parent



(0,4)	(1,4)	(2,4)	(3,4)
(0,3)	(1,3)	R	(3,3)
	(1,2)	(2,2)	(3,2)
	G	(2,1)	(3,1)
		(2,0)	

Breadth First Search and Dijkstra's Algorithm

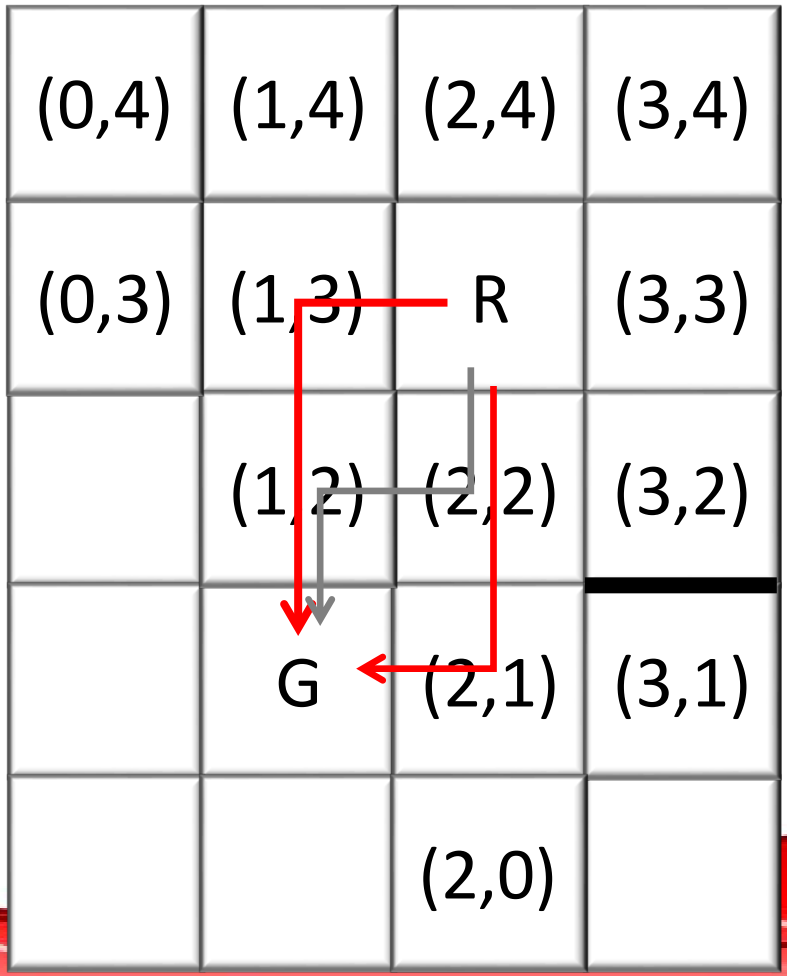
- Breadth-First Search



Does not include the cost to get there...

Data structure

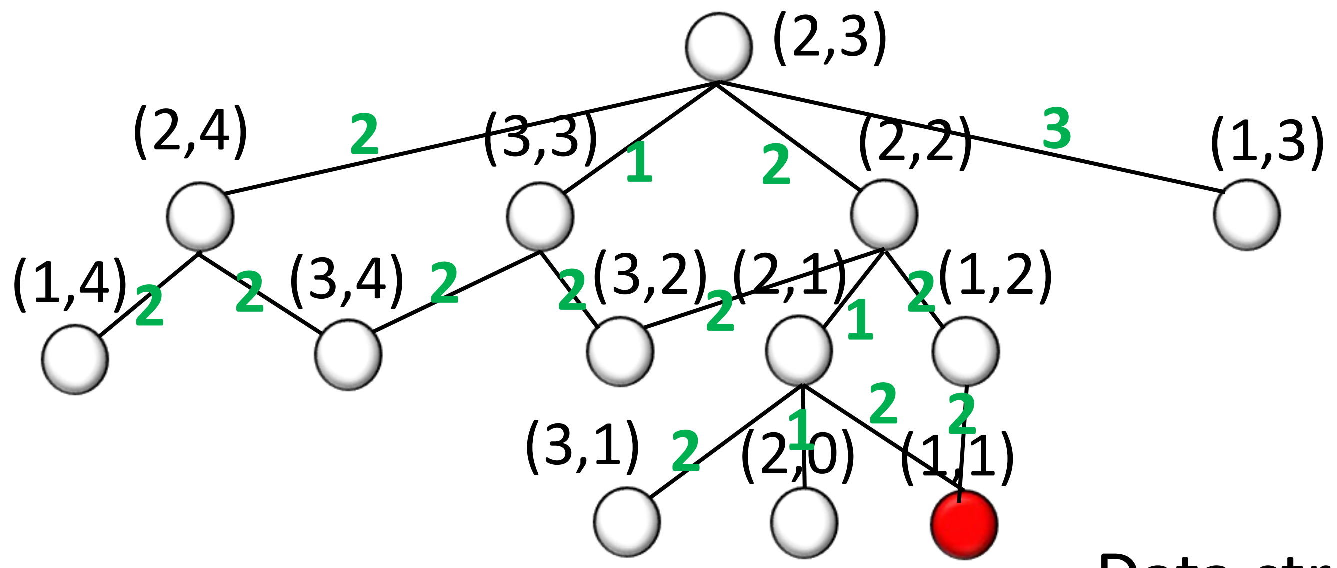
- n.state
- n.parent



Search order: N, E, S, W

Breadth First Search and Dijkstra's Algorithm

- Dijkstra's Algorithm: consider parent cost



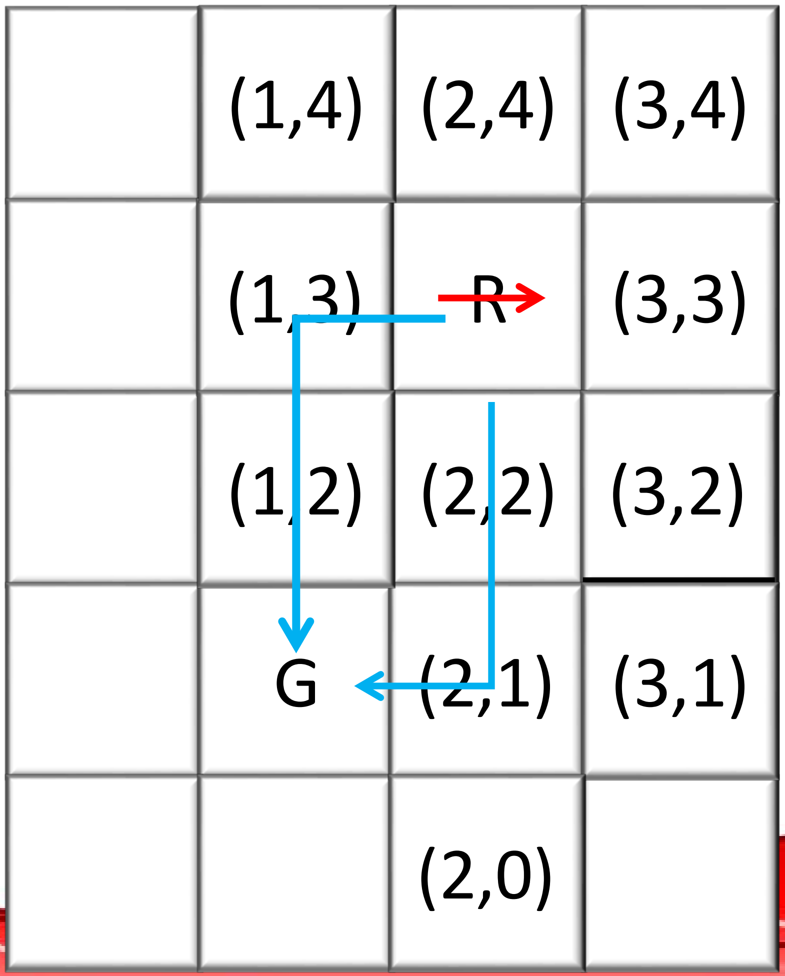
May save some computation!

Data structure

- n.state
- n.parent
- n.cost
- n.action

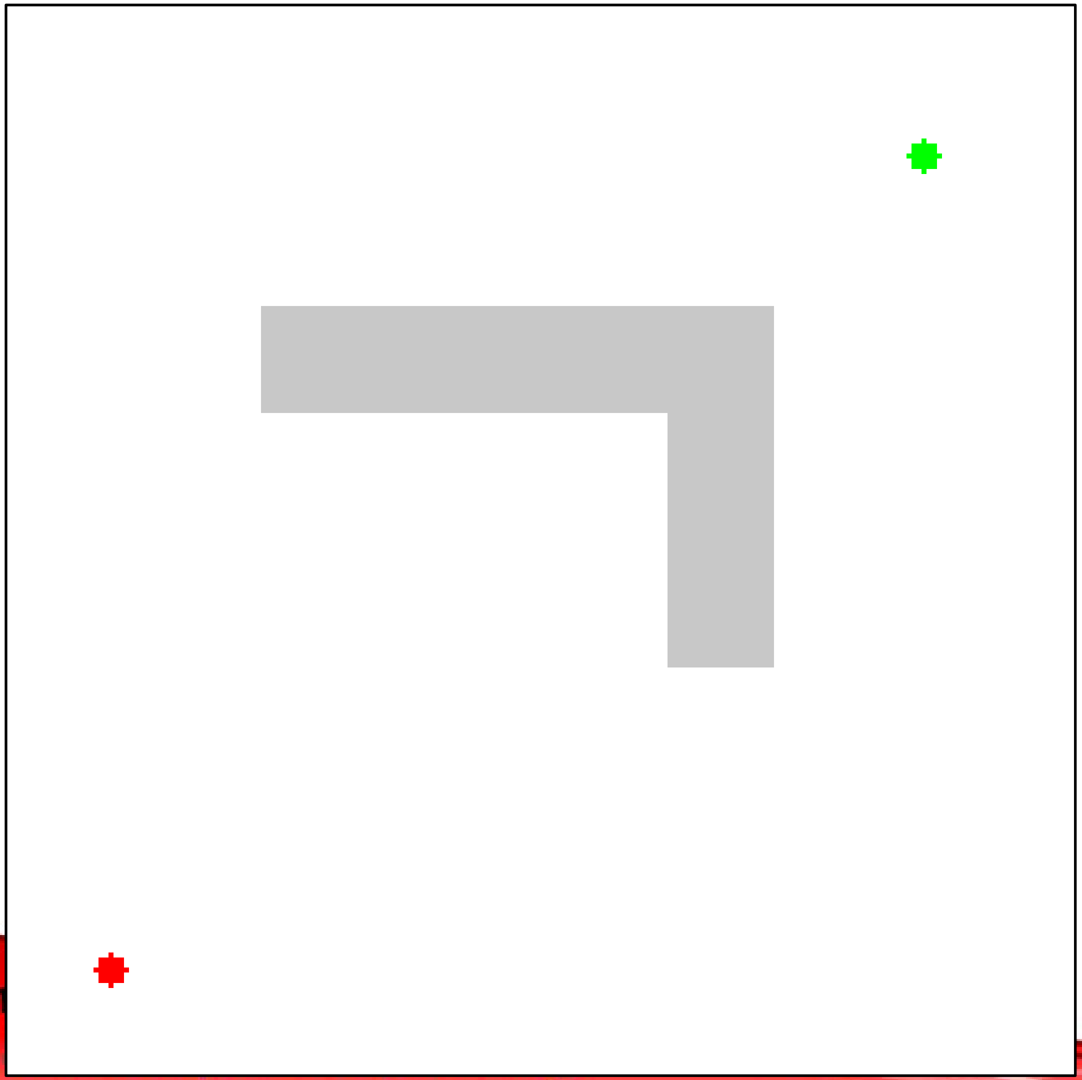
Go straight, cost 1

Turn quadrant, cost 1



Breadth First Search and Dijkstra's Algorithm

- Dijkstra's Algorithm: consider parent cost

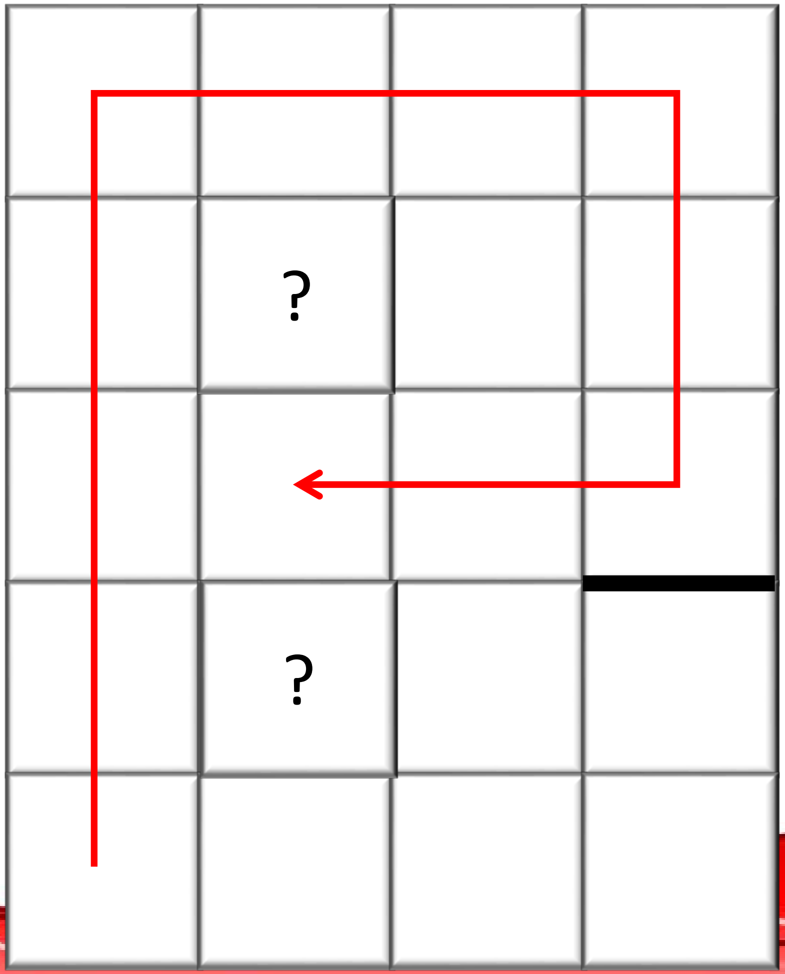


(Wikipedia)

Could we be smarter?

- Sure!
  - Detect walls to the side
  - Detect treasures/walls from further away
  - Pick closest frontier (minimize distance)
  - Pick a path to frontier that traverses most possible unknown space

- …Informed algorithms
  - Consider parent cost, and
  - estimates the shortest path to the "goal"



Prof Allison Okamura, Stanford

ECE Colloquium tonight: “Let’s be Flexible!”

- 4.30-6pm (PH233)

Opportunities for Graduate Study in Engineering

- 7.30-8pm (PH203)
- Pizza served from 7pm

Stanford

Go Build Robots!

Class website: https://cei-lab.github.io/ece3400/
Piazza: https://piazza.com/cornell/fall2017/ece3400/home