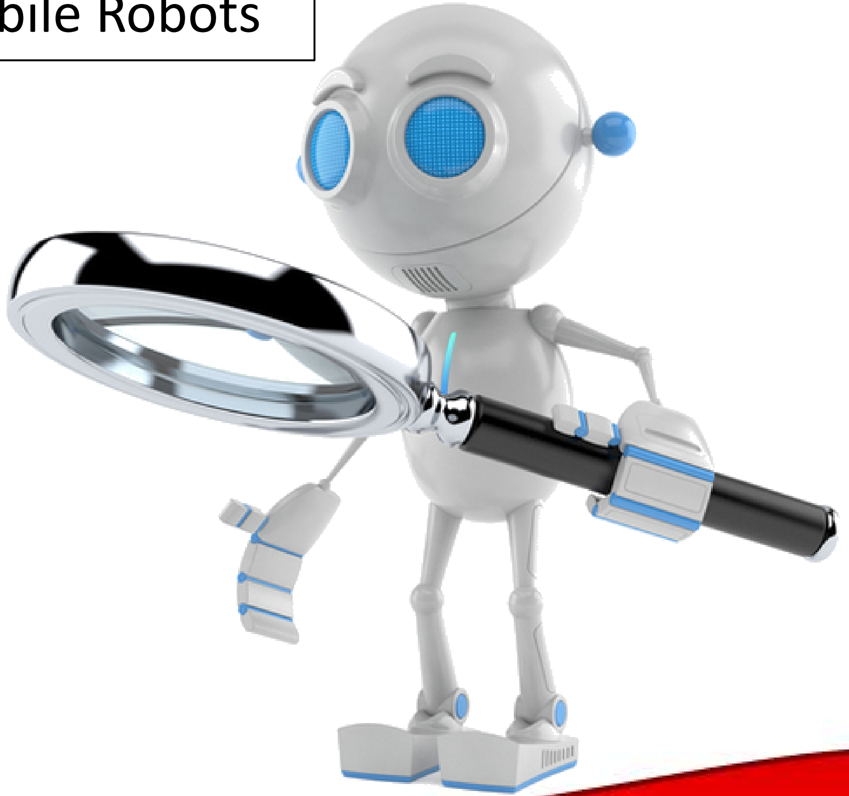
Algorithms and Path Planning

Topics

- Simple Search
- Depth First Search
- Breadth First Search
- Dijkstra’s Search
- Greedy Search
- A* Search

Classes of interest

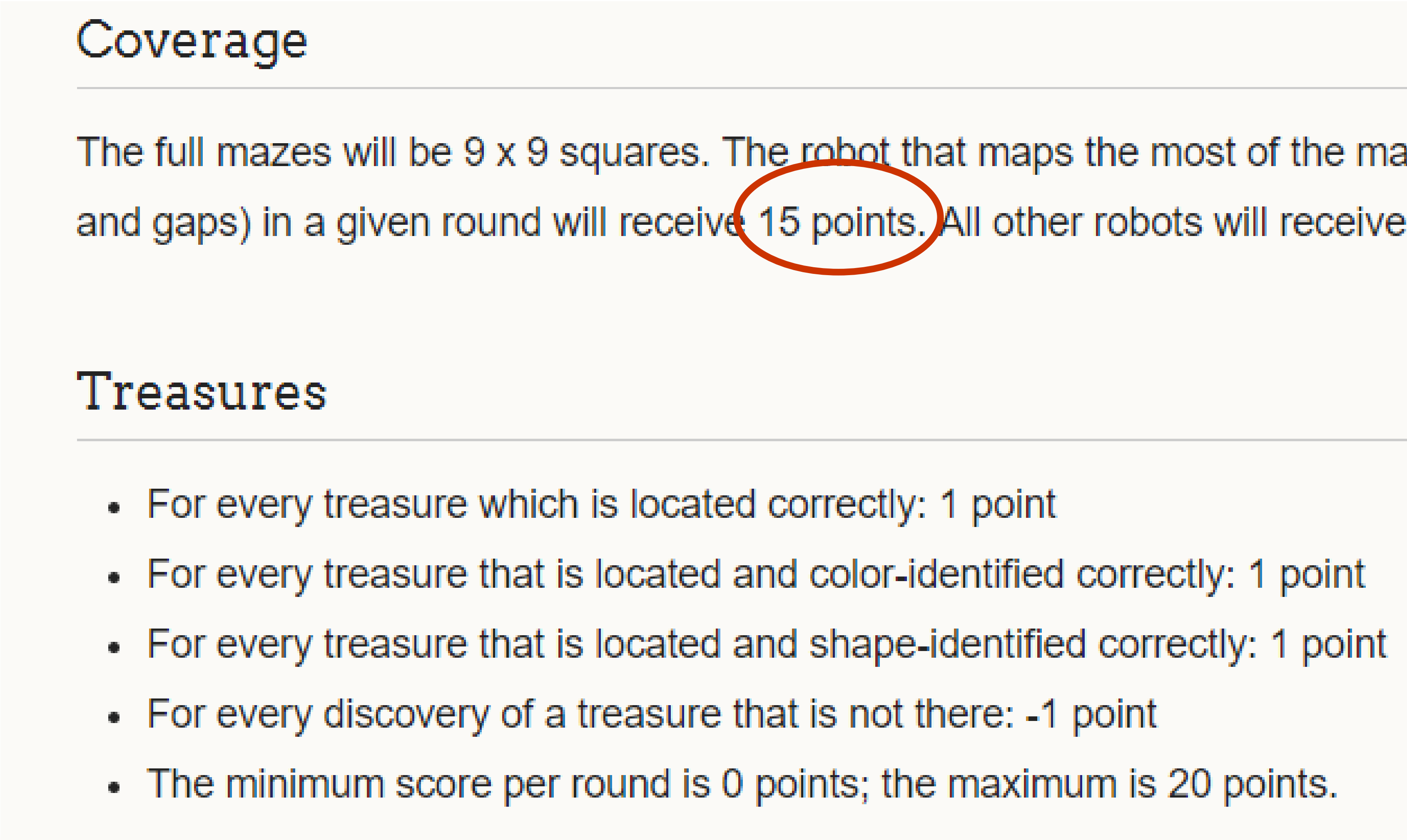
- ECE2400: Computer Systems Programming
- CS4700: Foundations of Artificial Intelligence
- CS4701: Practicum of Artificial Intelligence
- CS3758/MAE4180: Autonomous Mobile Robots



ECE 3400: Intelligent Physical Systems

End Game

5 weeks left till competition day



Coverage

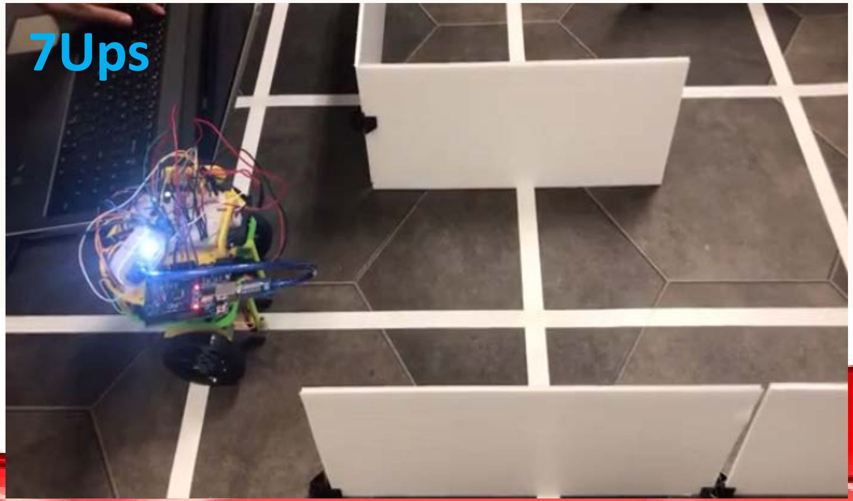
The full mazes will be 9 x 9 squares. The robot that maps the most of the maze correctly (wrt to walls and gaps) in a given round will receive 15 points. All other robots will receive scaled values thereof.

Treasures

- For every treasure which is located correctly: 1 point
- For every treasure that is located and color-identified correctly: 1 point
- For every treasure that is located and shape-identified correctly: 1 point
- For every discovery of a treasure that is not there: -1 point
- The minimum score per round is 0 points; the maximum is 20 points.

Can you explore the entire maze?

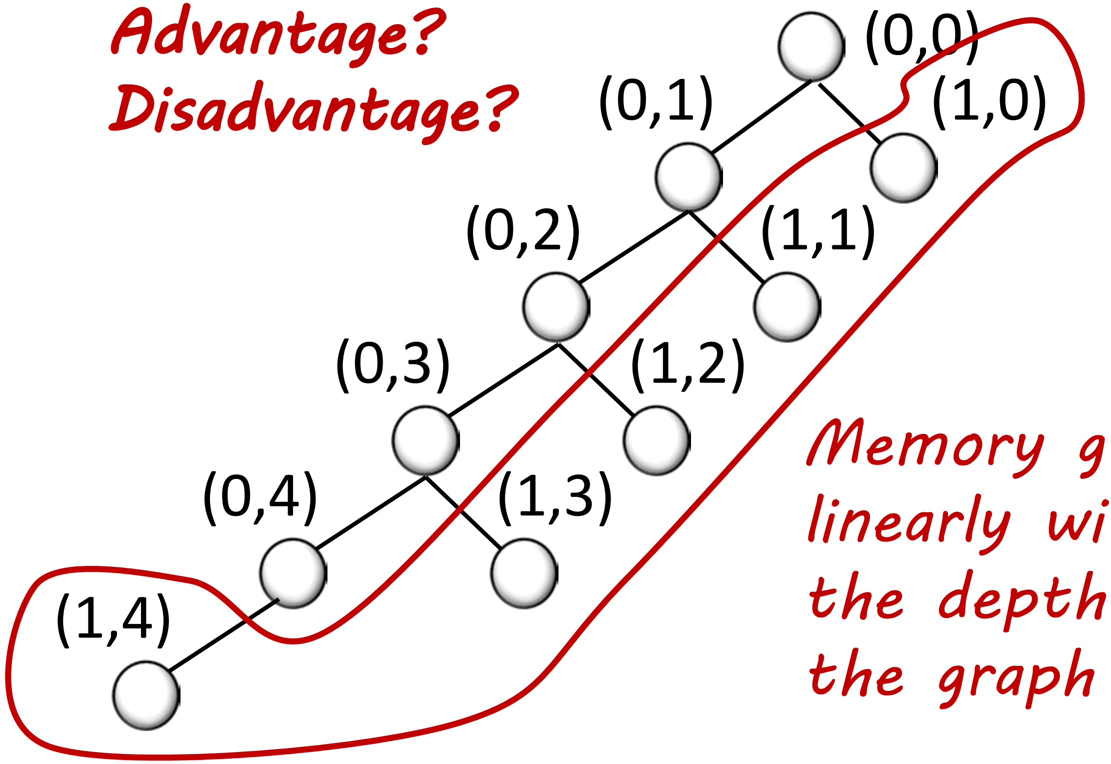
- 15s avg. for 6 squares
- 3.4min for 81 squares
- Unlikely, but possible.



Algorithms and Search

- Depth First Search (DFS)

Advantage?
Disadvantage?

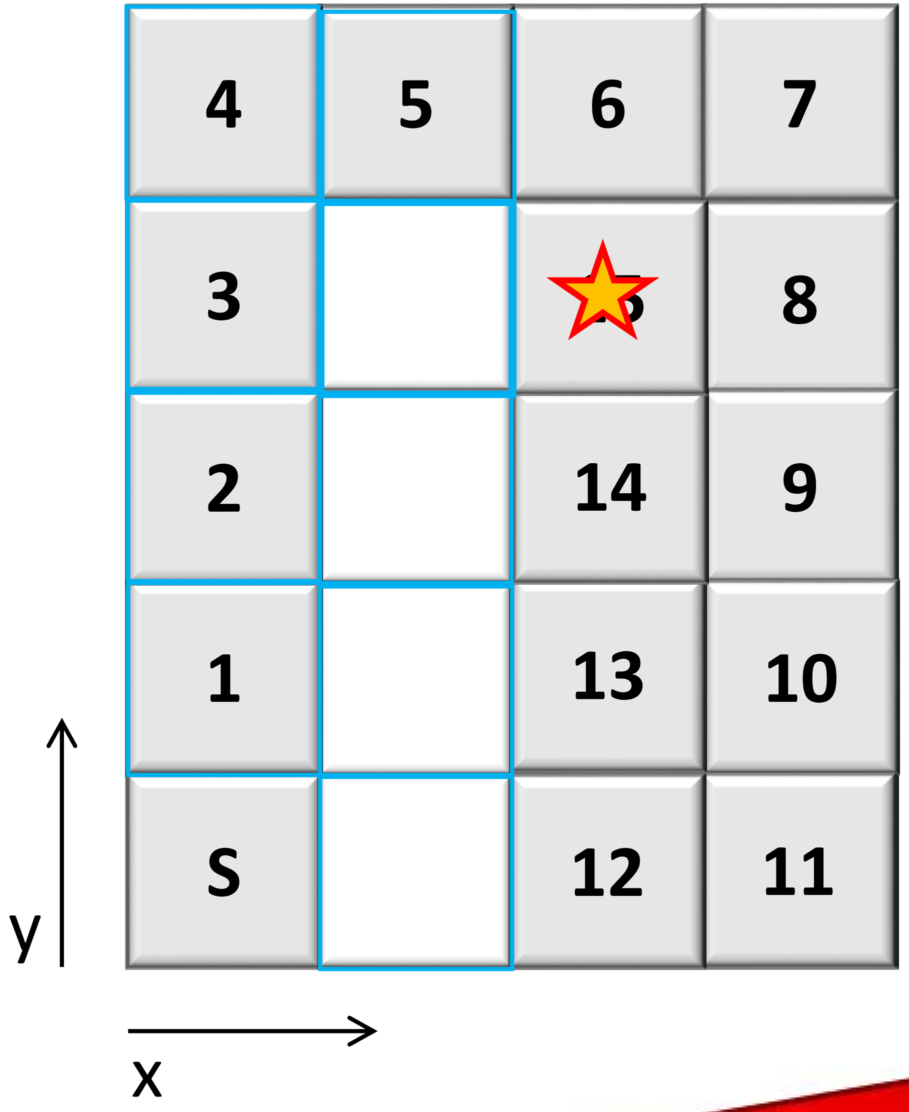


Memory grows linearly with the depth of the graph

and so on…

Search order: N, E, S, W

Find a treasure



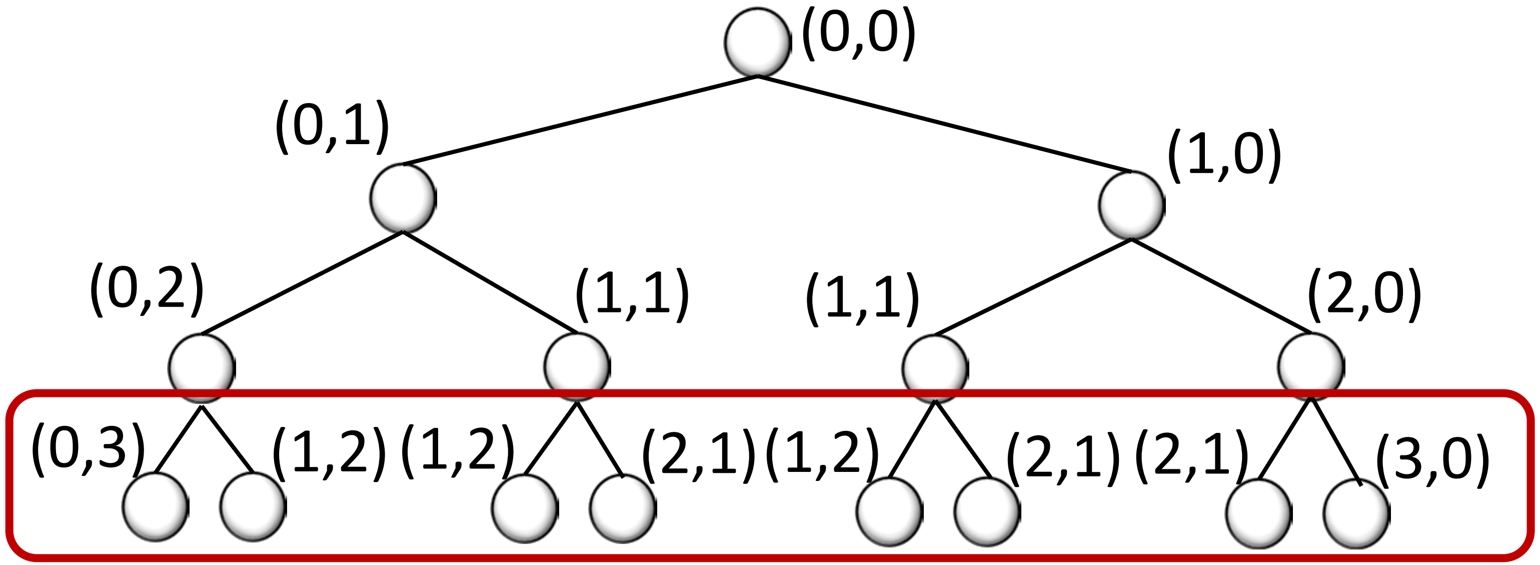
4	5	6	7
3		15	8
2		14	9
1		13	10
S		12	11

Search order: N, E, S, W

Algorithms and Search

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

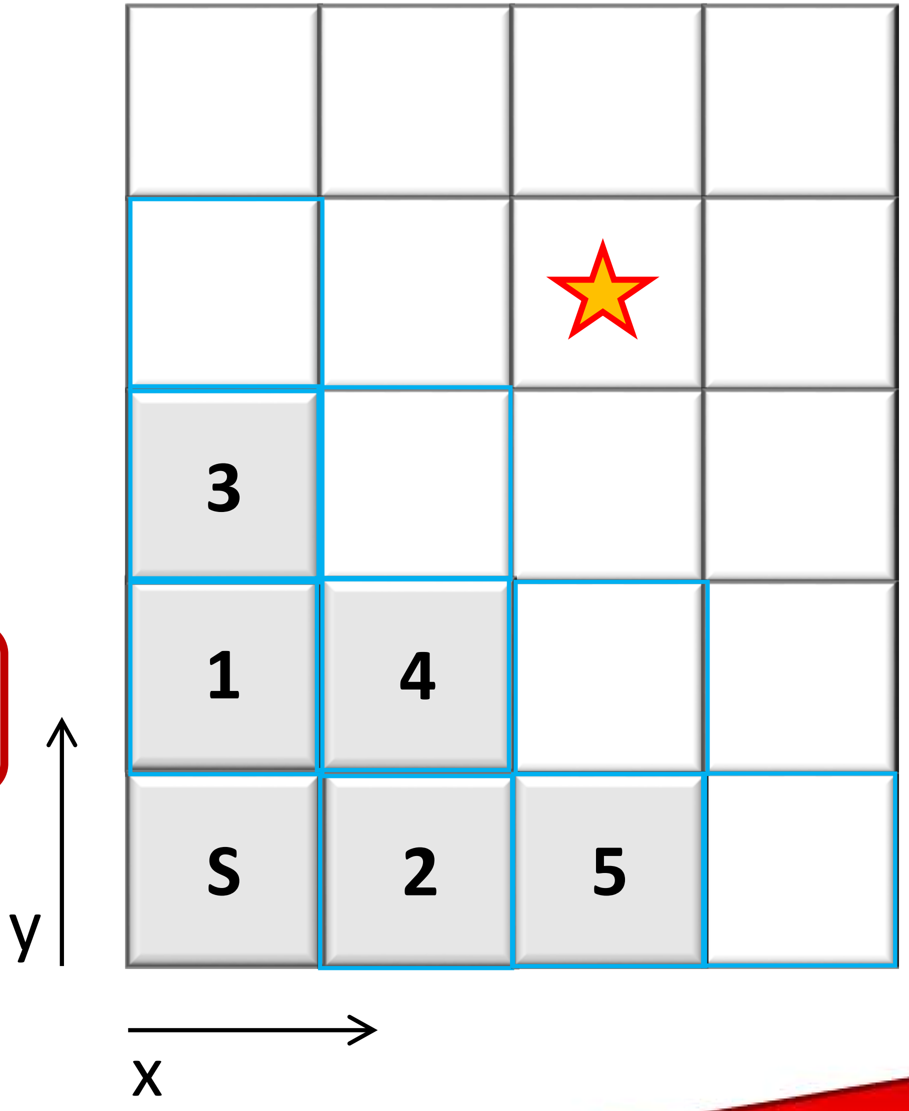
Advantage?
Disadvantage?



and so on…

Memory grows exponentially with the depth of the graph

Find a treasure

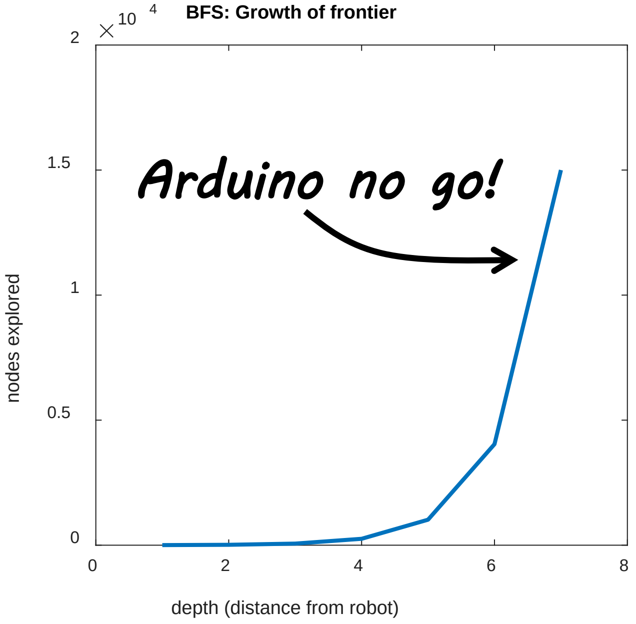


BFS: Memory and Computation

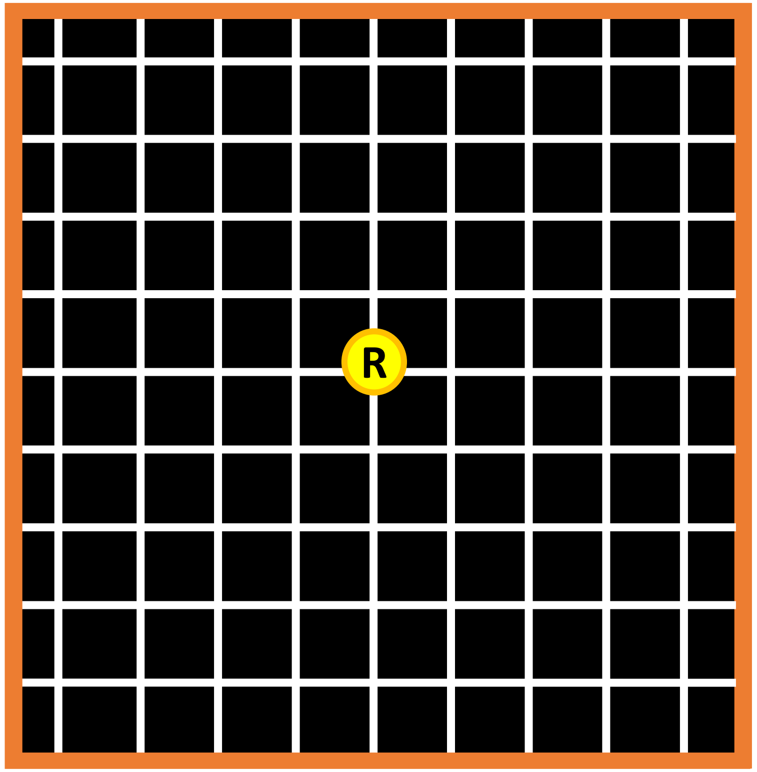
What do we need?

- Locations
- Example issue from last year…

```
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)
      frontier.append(n’)
```



Arduino no go!



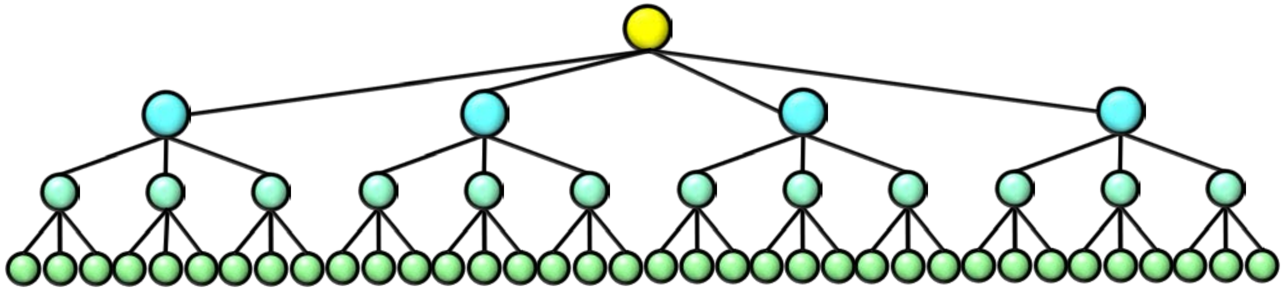
BFS: Memory and Computation

What do we need?

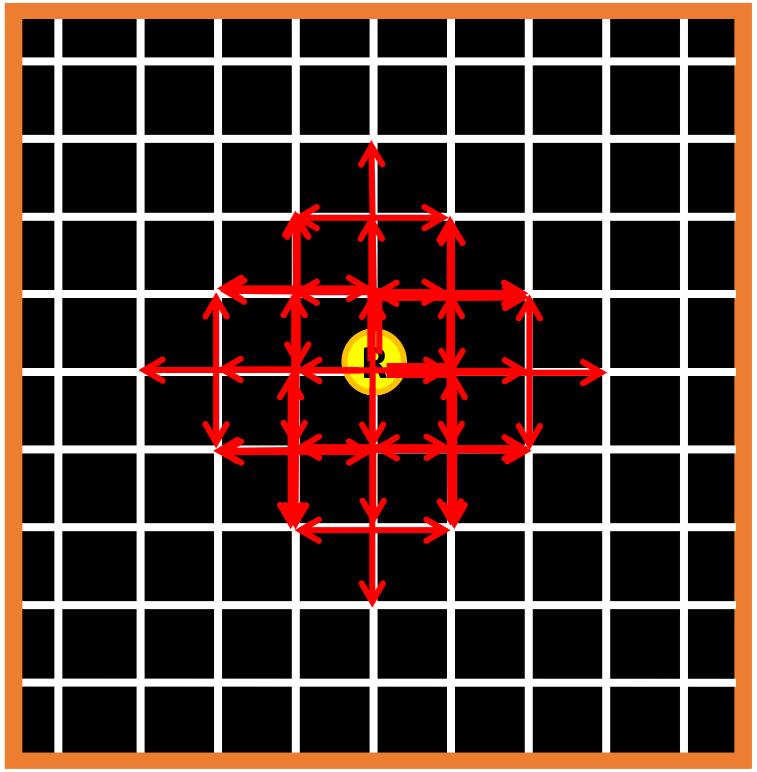
- Locations
- Parents

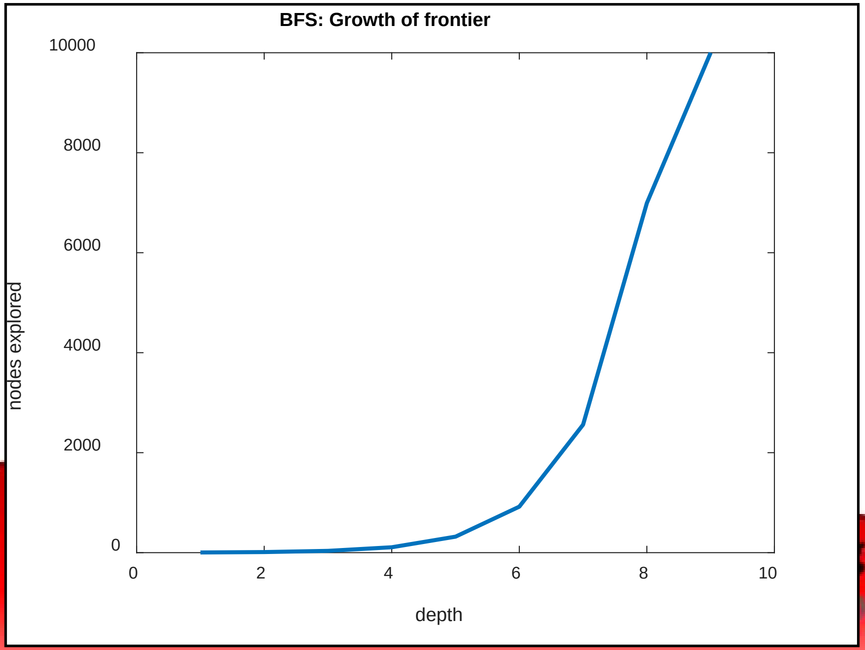
Frontier size:

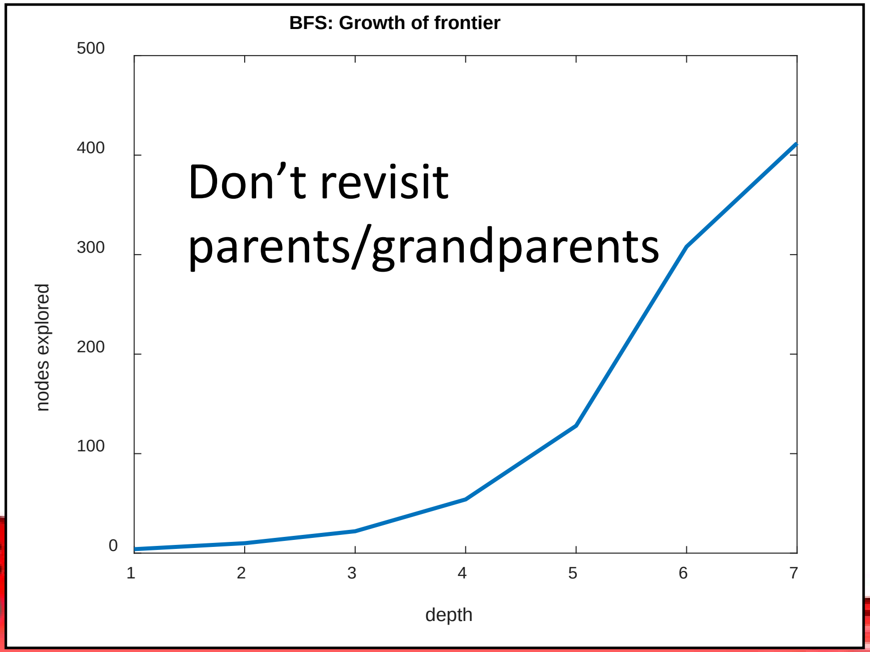
- 4
- 12
- 36
- mem = 4 · 3^{n-1} (n = depth)



etc…





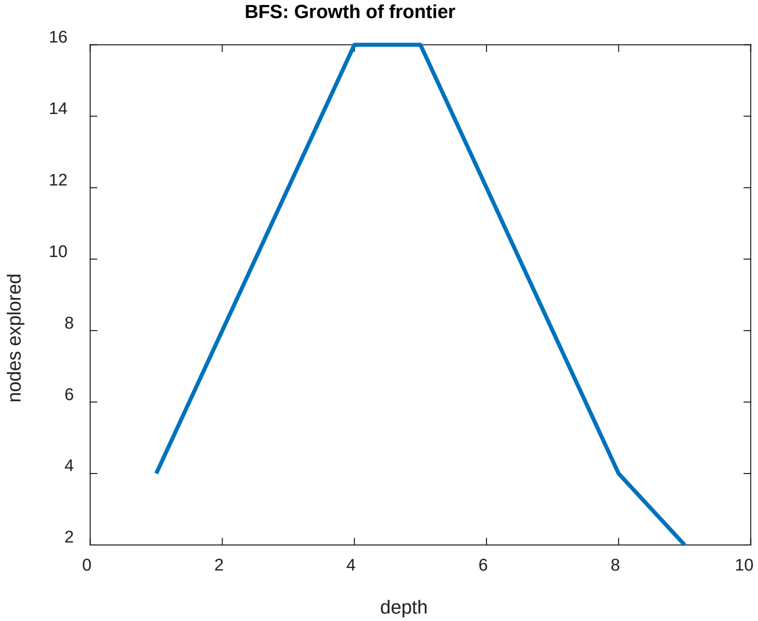


Don’t revisit parents/grandparents

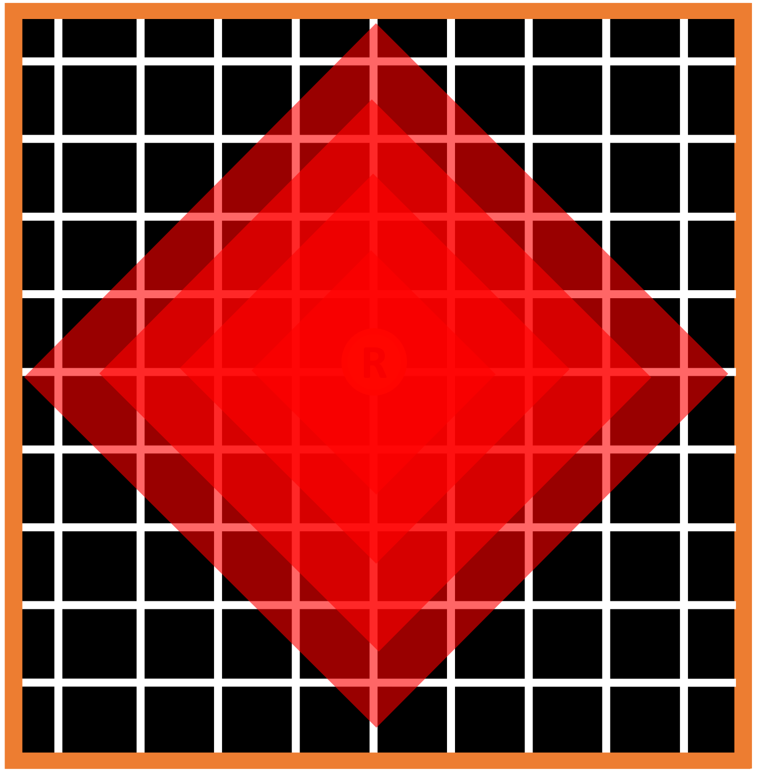
BFS: Memory and Computation

What do we need?

- Locations
- Parents
- Visited
  - *What is the maximum size of the frontier now?*



- *What is the issue with this approach?*
  - Store branches with lowest motion cost!

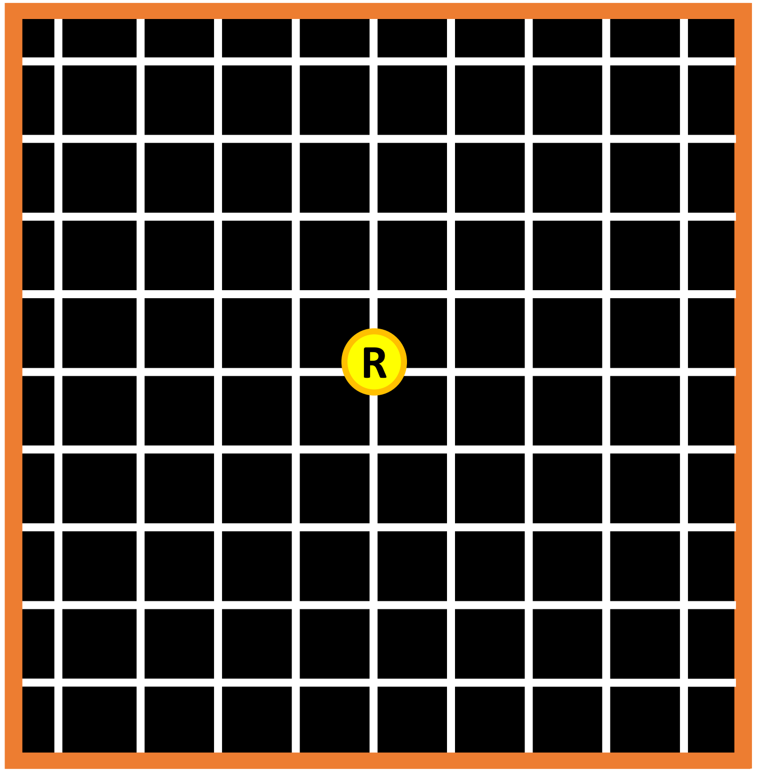


BFS: Memory and Computation

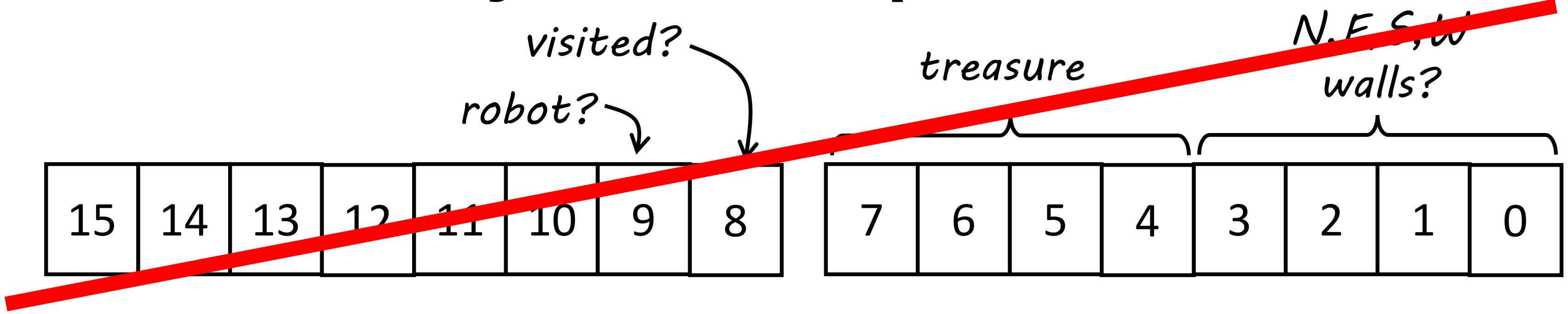
What do we need?

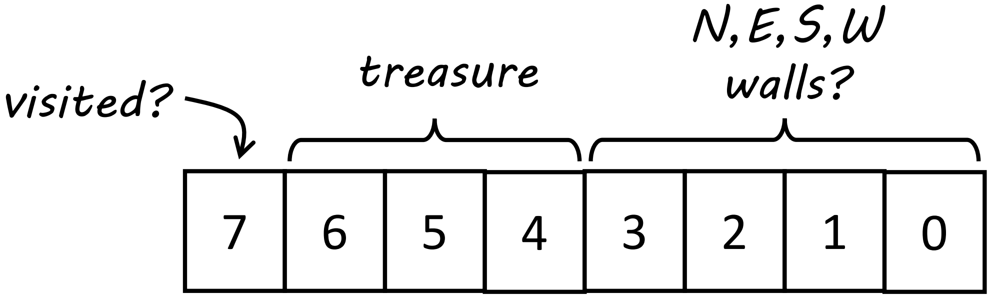
- Locations*
- Parents*
- Visited
- Cost*
- Action*

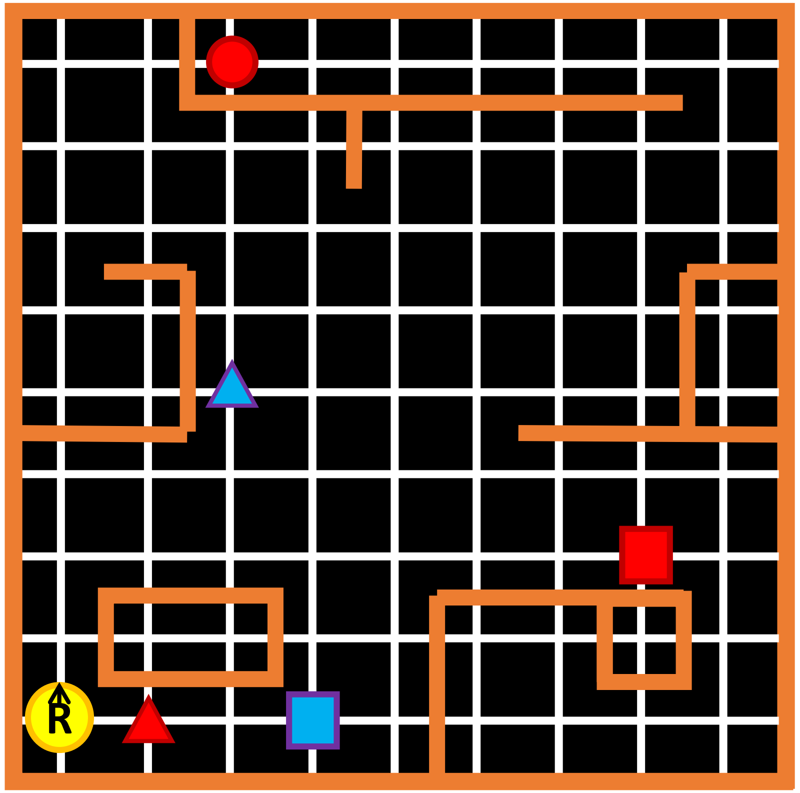
```
n = state(init)
frontier.append(n)
visited.append(n)
while(frontier not empty)
   n = pull state* from frontier
   if n = goal, return solution
   for all actions in n
      n' = a(n)
      if n' not visited or cost is lower
            frontier.append(n')
            visited.append(n')
```



BFS: Memory and Computation



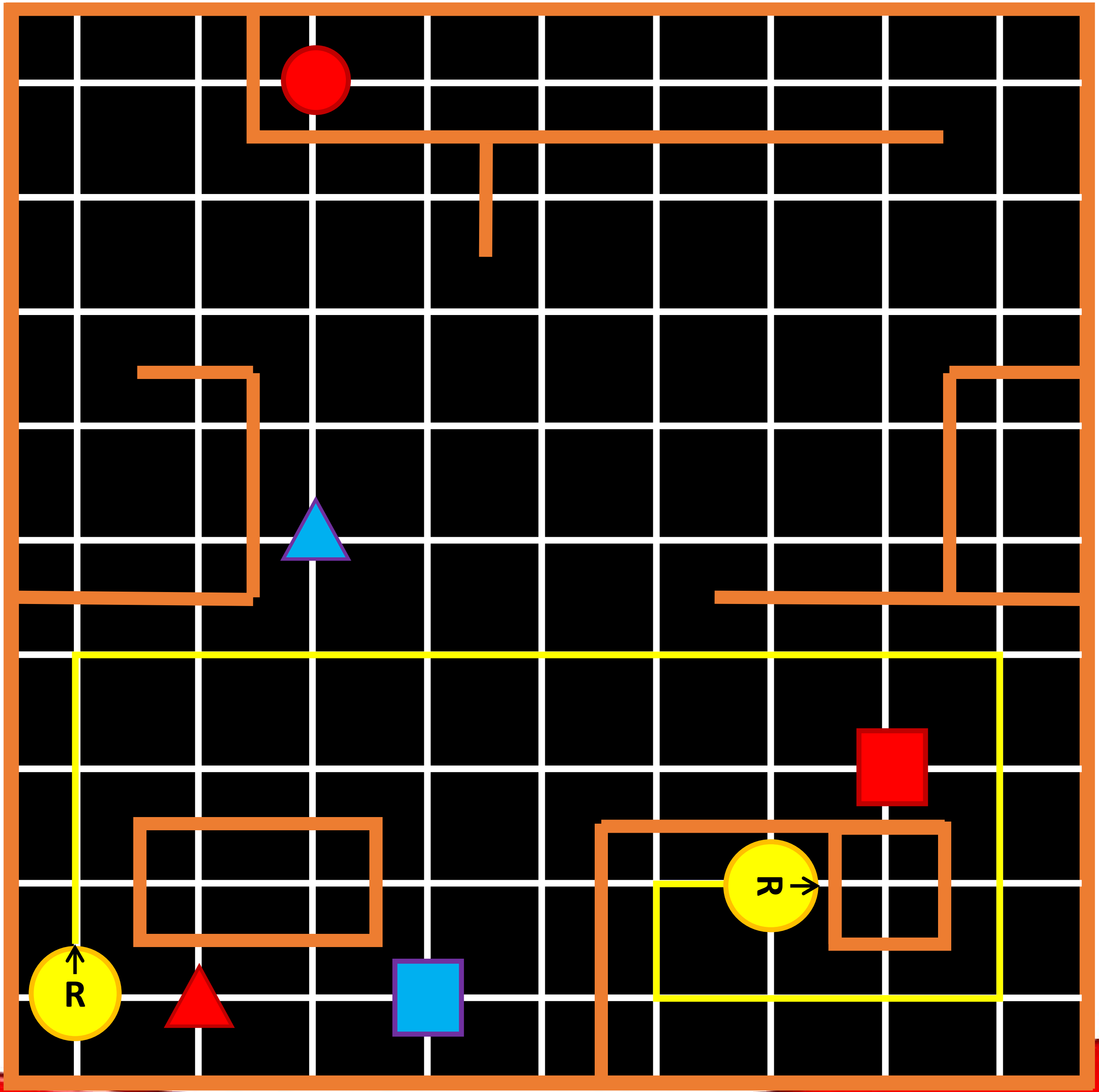




- …Teams are at 75-35% capacity (512B-1331B)
- Frontier (x,y-locations + parent + cost): 80B
- Visited: 81B, or 0B!
- ~~Maze: 162B~~
- Maze: 81B

Maze Exploration

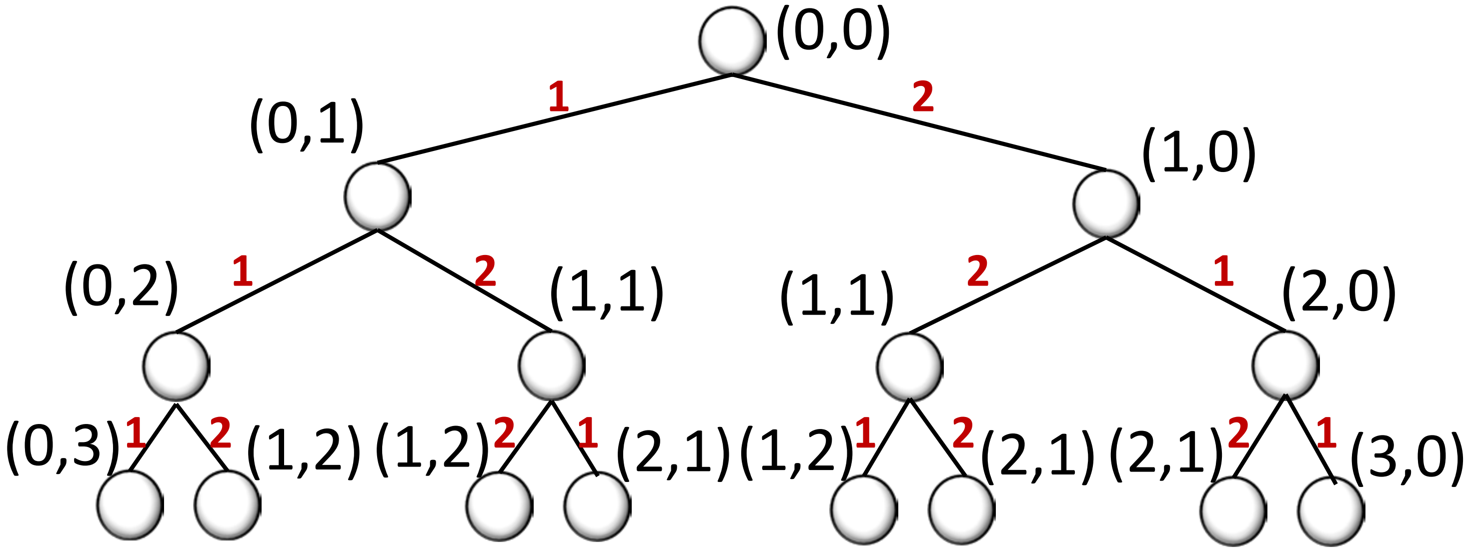
- **Depth First Search**
  - Search order:
    - Straight
    - Left, then straight
    - Right, then straight
    - U-turn, then straight
- **If stuck, find shortest path to the next frontier in the tree**
- *What treasure does the robot find first?*



Algorithms and Search

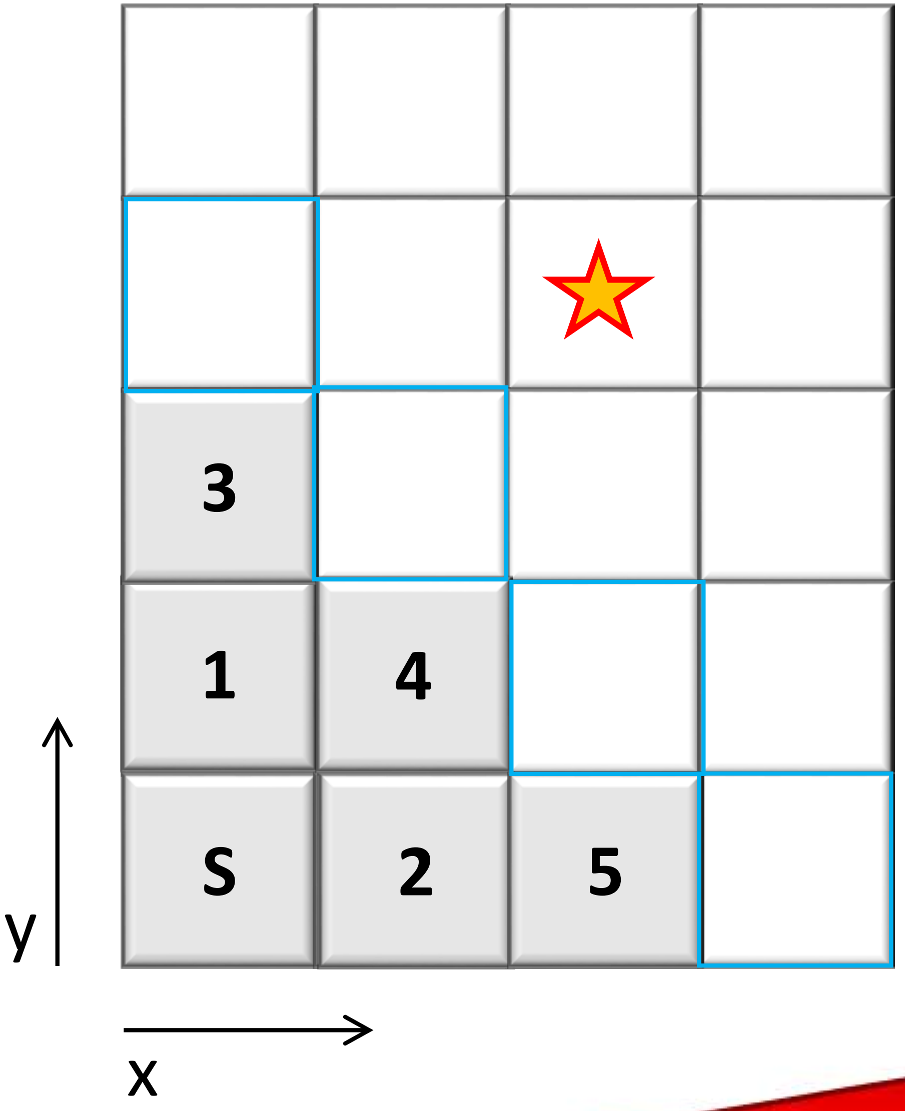
Search order: N, E, S, W

- Depth First Search (DFS)
- Breadth First Search (BFS)
- Add motion cost
- Dijkstra's to save computation/memory



and so on...

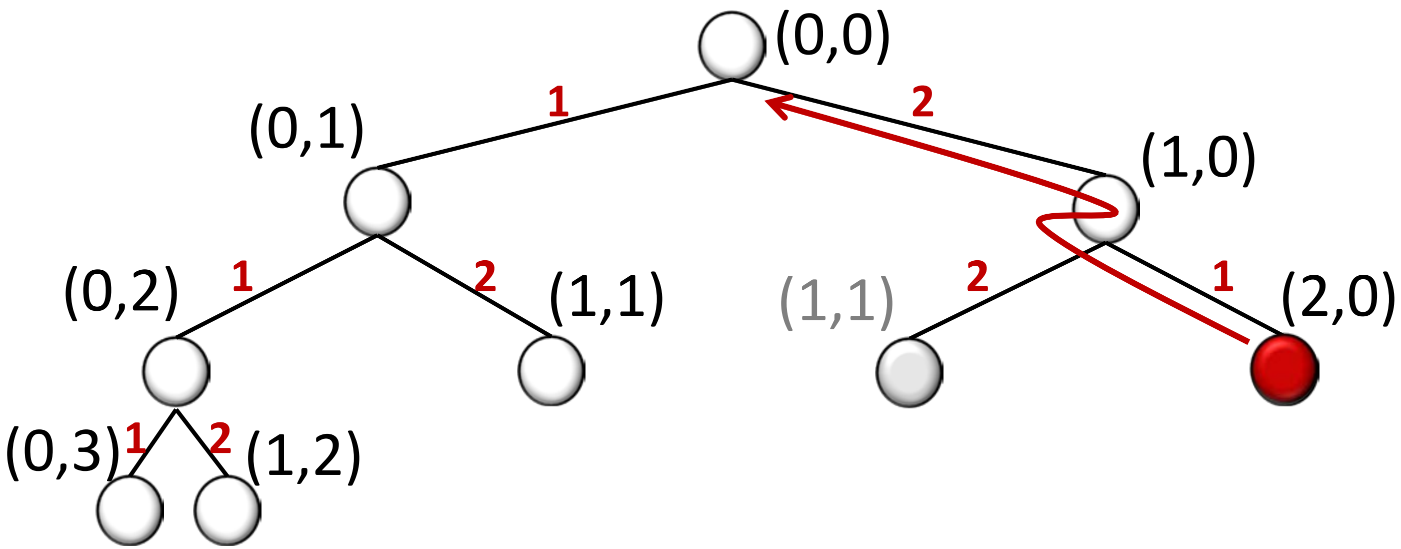
Find a treasure



Algorithms and Search

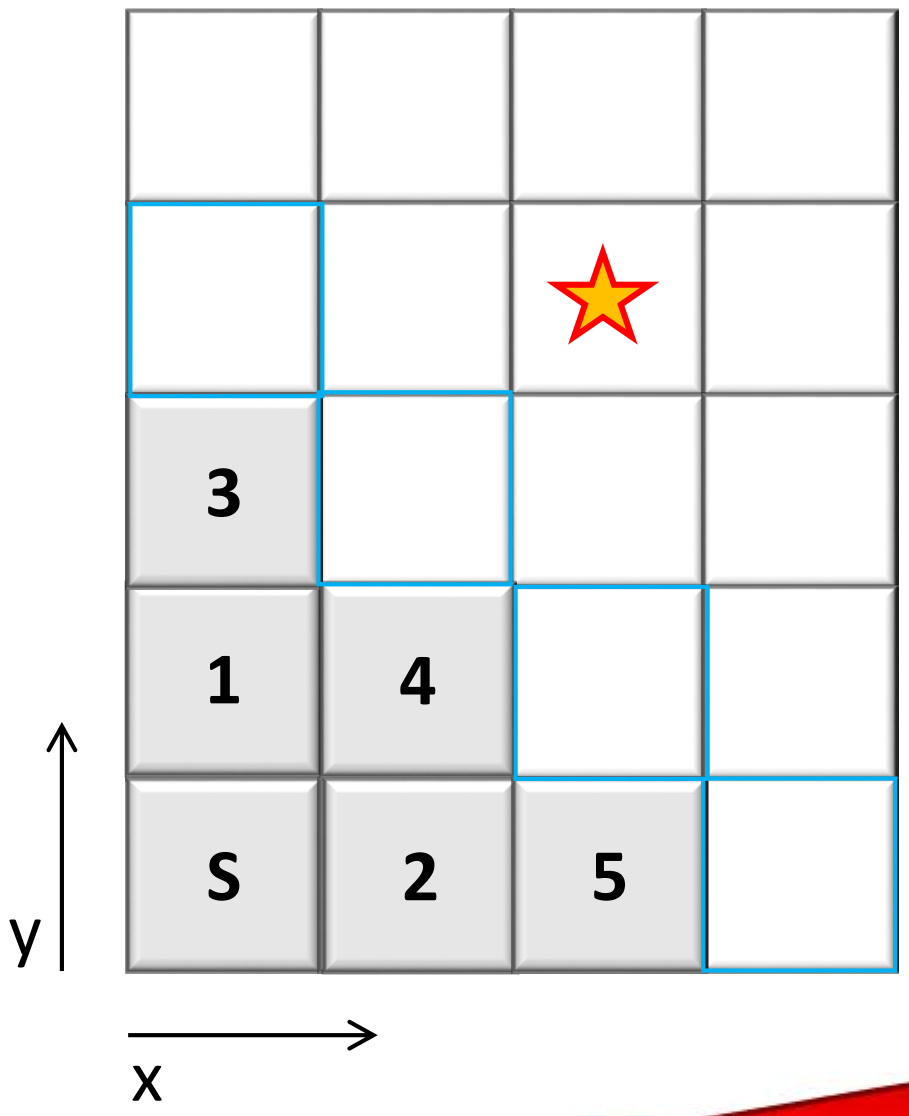
Search order: N, E, S, W

- Depth First Search (DFS)
- Breadth First Search (BFS)
- Add motion cost
- Dijkstra’s to save computation/memory



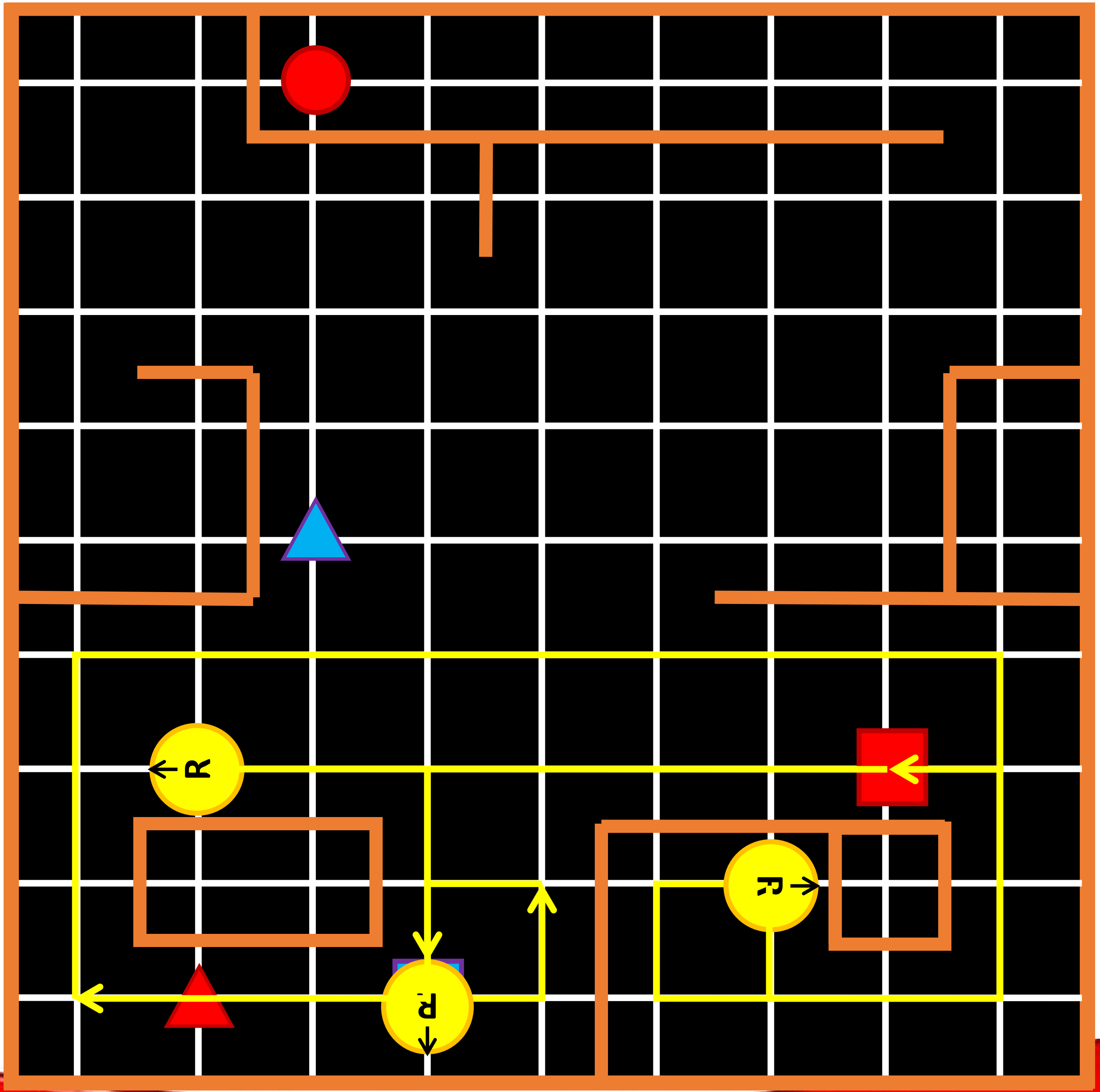
and so on…

Find a treasure



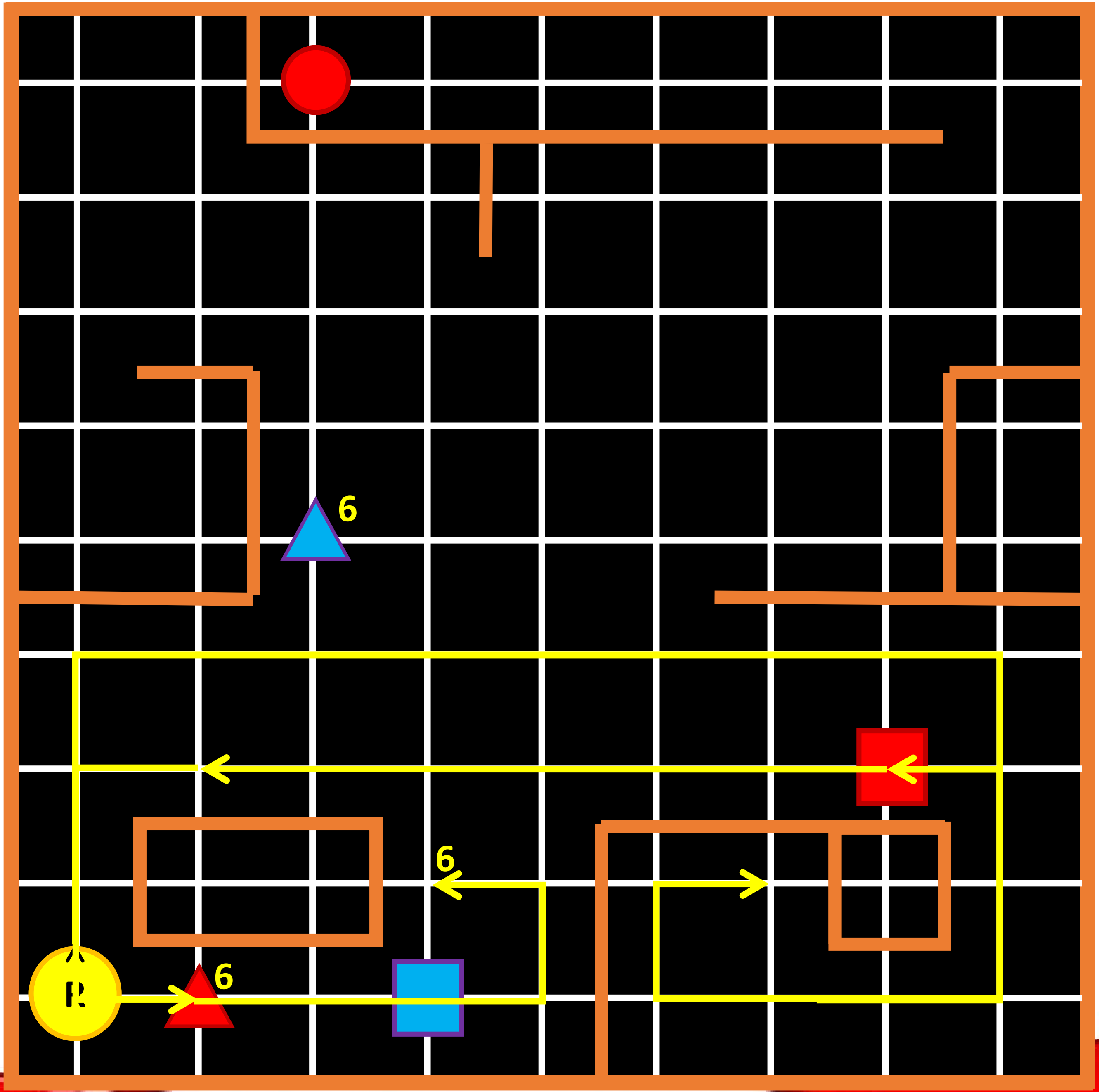
Maze Exploration

- **Depth First Search**
  - Search order:
    - Straight
    - Left, then straight
    - Right, then straight
    - U-turn, then straight
- **If stuck, find shortest path to the next frontier in the tree**
- *What treasure does the robot find first?*
- *What treasure does the robot find second?*
- *Could we be more efficient?*



Maze Exploration

- **Dijkstra to find the next frontier**
  - Search order:
    - Straight
    - Left, then straight
    - Right, then straight
    - U-turn, then straight
- *What treasure does the robot find first?*
- *Next treasure?*
- Extra computation (Dijkstra for every square), but *maybe better*
- *Better path planning?*
  - Add cost for revisiting nodes

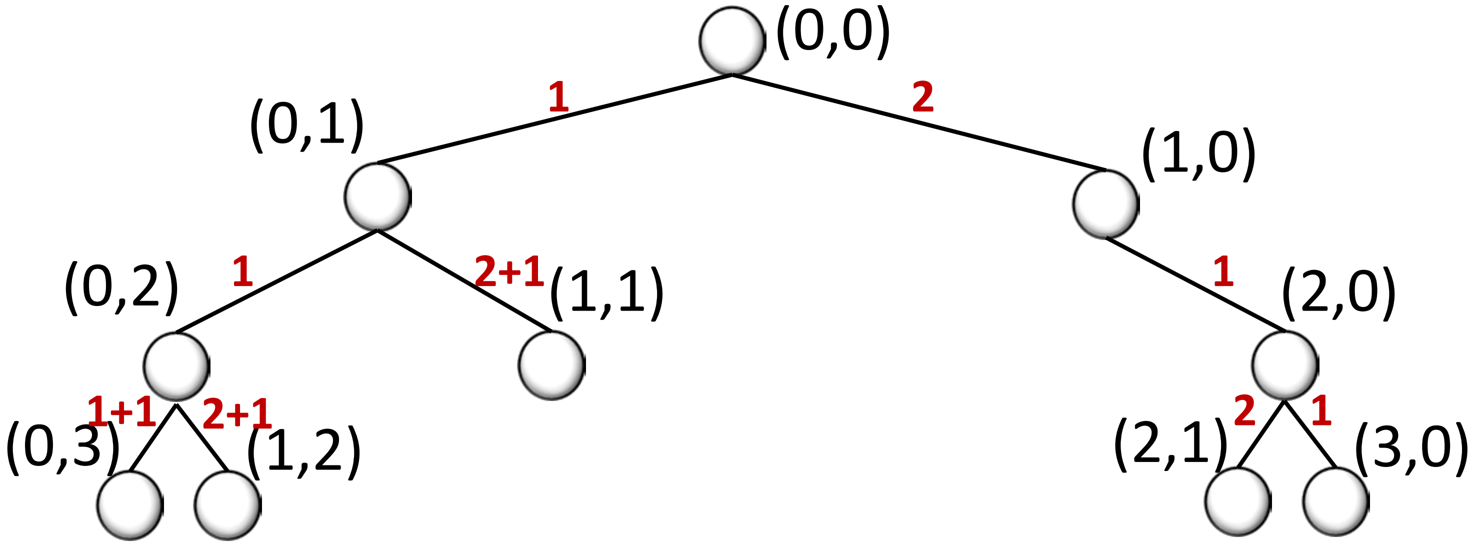


Search order: N, E, S, W

Maze Exploration

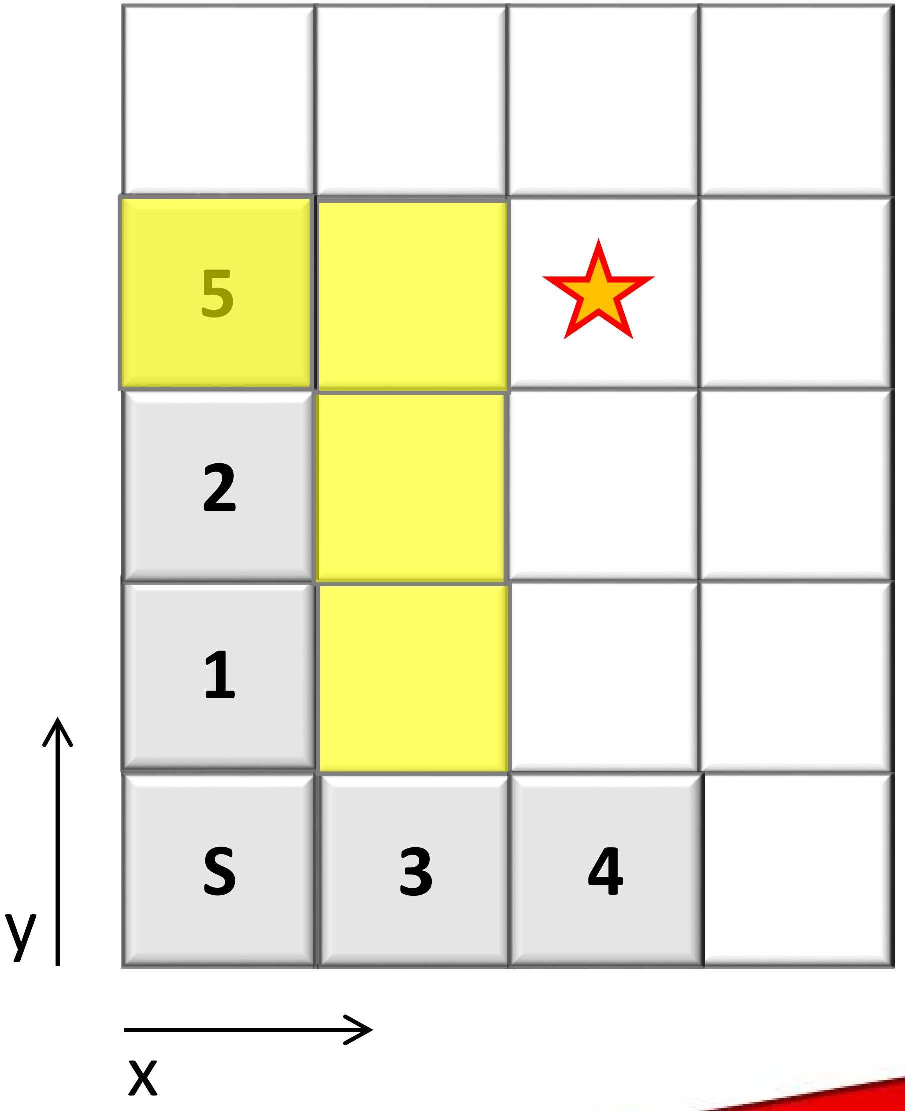
- Dijkstra's Search
- *Could you be more efficient by looking ahead?*

Only reasons about the cost to get there...



and so on...

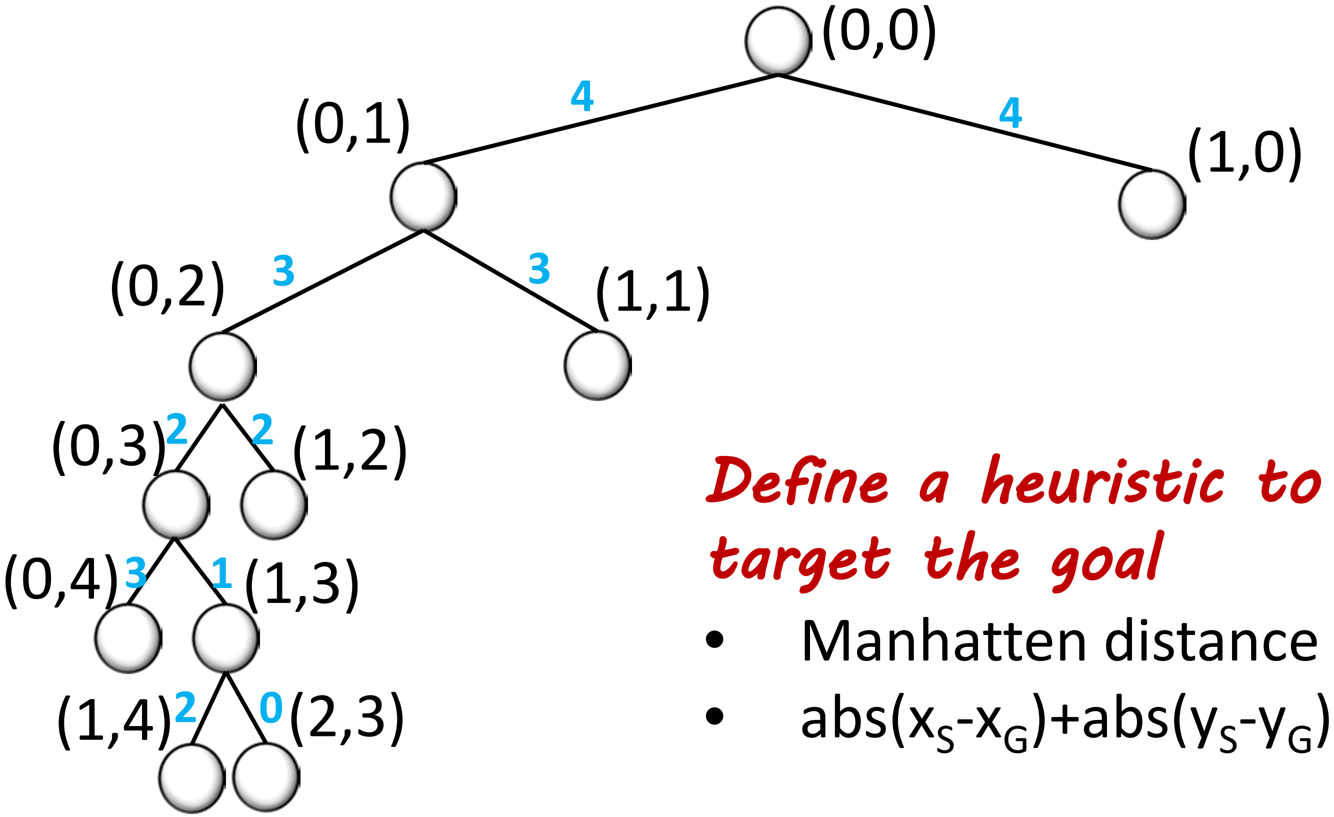
Find a treasure



Informed Search

Search order: N, E, S, W

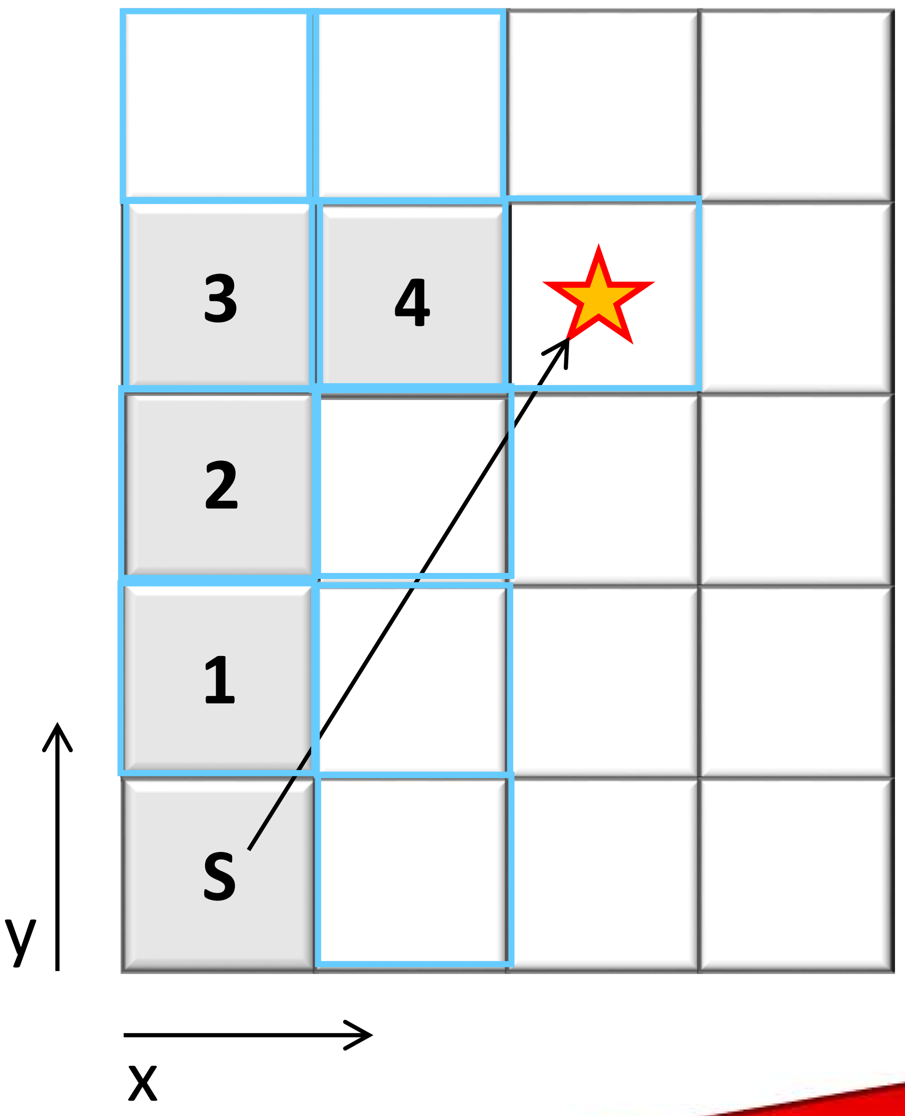
- Greedy Search



Define a heuristic to target the goal

- Manhatten distance
- abs(x_S-x_G)+abs(y_S-y_G)

Find a treasure



Informed Search

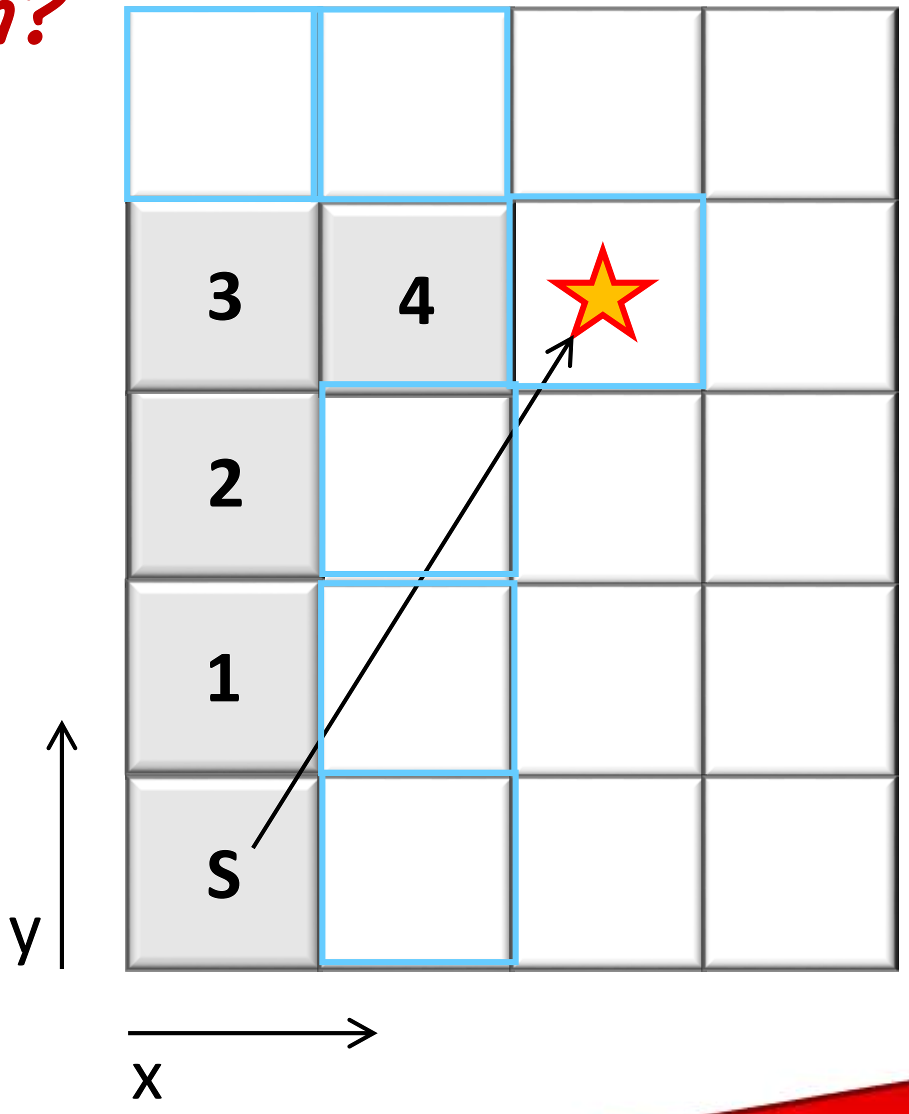
Search order: N, E, S, W

- Greedy Search

Cause for concern?

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

Find a treasure



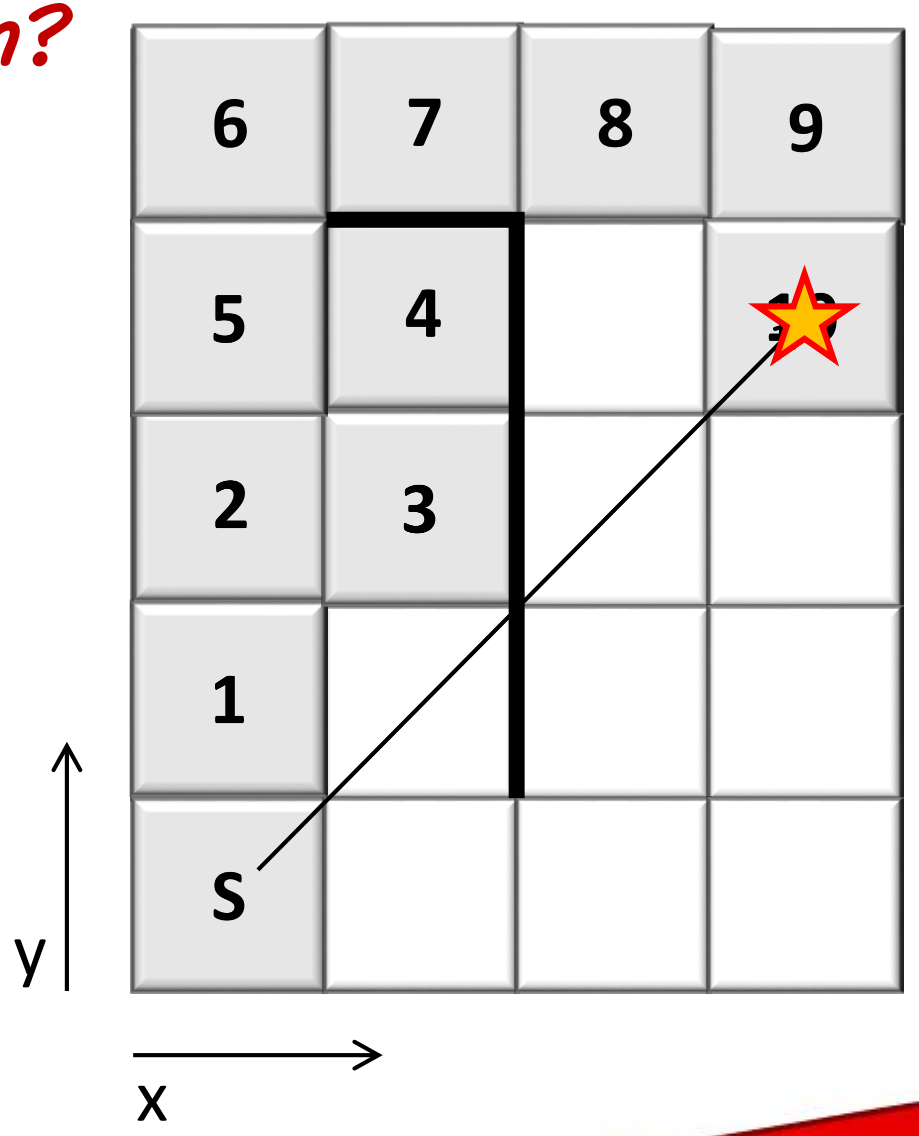
Search order: N, E, S, W

Informed Search

- Greedy Search
  - Faster, but does not guarantee optimal

Cause for concern?

Find a treasure



Informed Search

- Breadth First Search
  - Guarantee: Finds a path
  - Searches *everything*
- Dijkstra's Algorithm Considers parent cost
  - Guarantee: Finds the shortest path
  - …but it wastes time exploring in directions that aren't promising
- Greedy Search Considers goal
  - Guarantee: Finds a path
  - …only explores promising directions

A*

Can we do better?

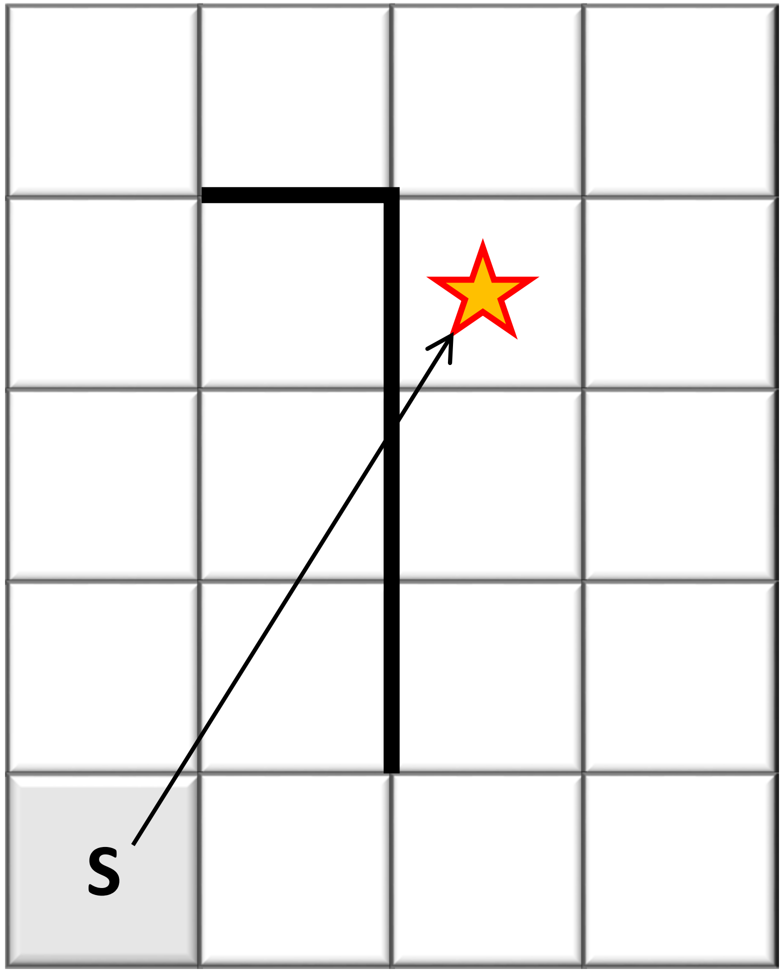
Informed Search

Search order: N, E, S, W

- A* (“A-star”)

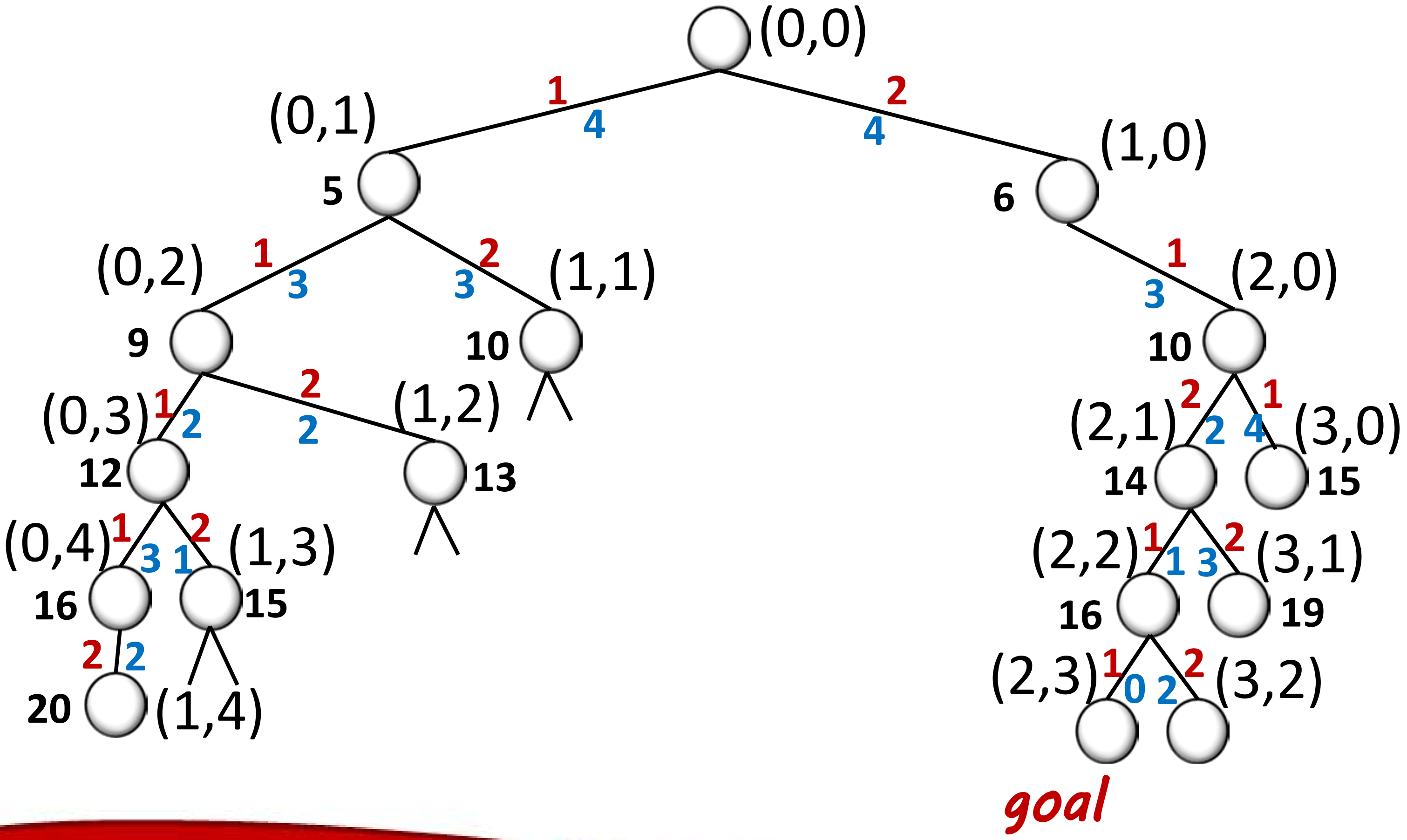
```
n = state(init)
frontier.append(n)
while(frontier not empty)
   n = pull state from frontier
   if n = goal, return solution
   for all actions in n
      n’ = a(n)
      if ((n’ not visited or
         (visited and n’.cost < n_old.cost))
            priority = heuristic(goal,n’)+cost
            frontier.append(priority)
            visited.append(n’)
```

Find a treasure



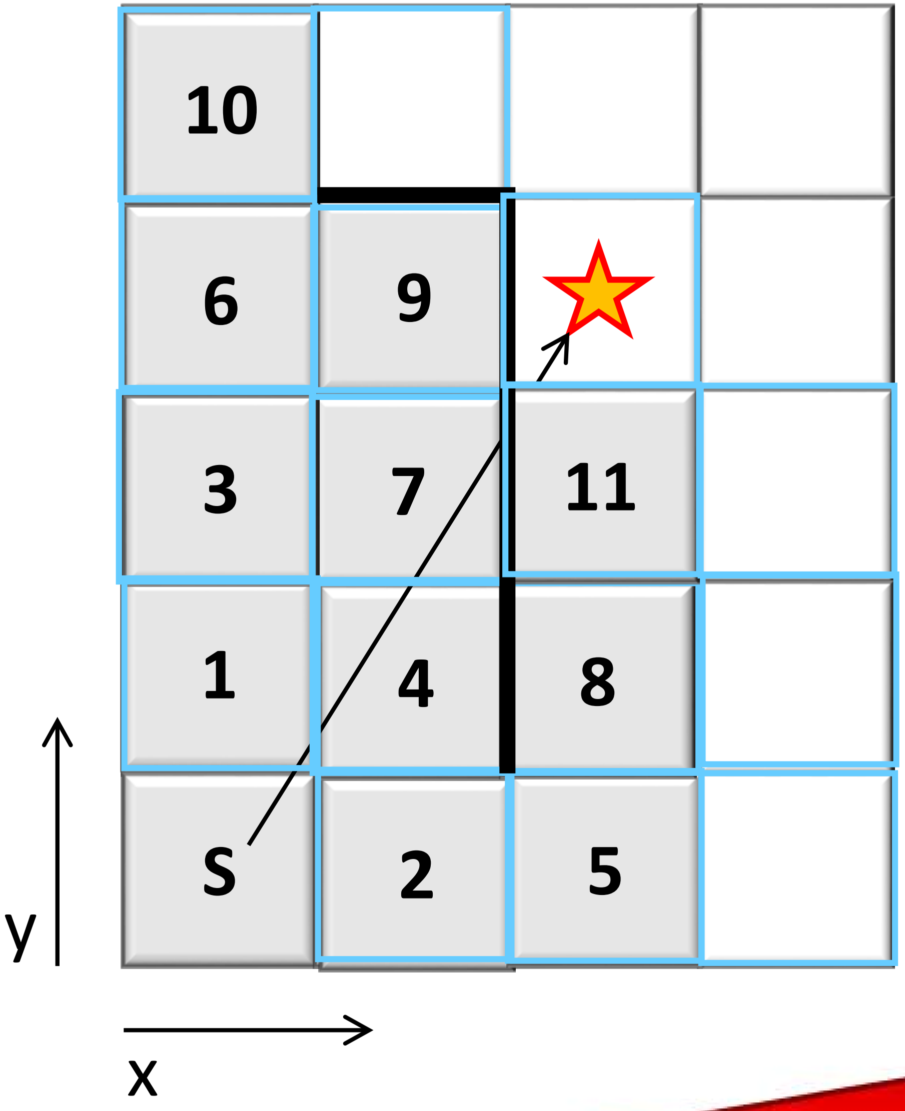
Informed Search

- A* ("A-star")
  - Cost and goal heuristic



Search order: N, E, S, W

Find a treasure



Informed Search

admissible heuristic

- What if the heuristic is too optimistic?
    - Estimated cost < true cost

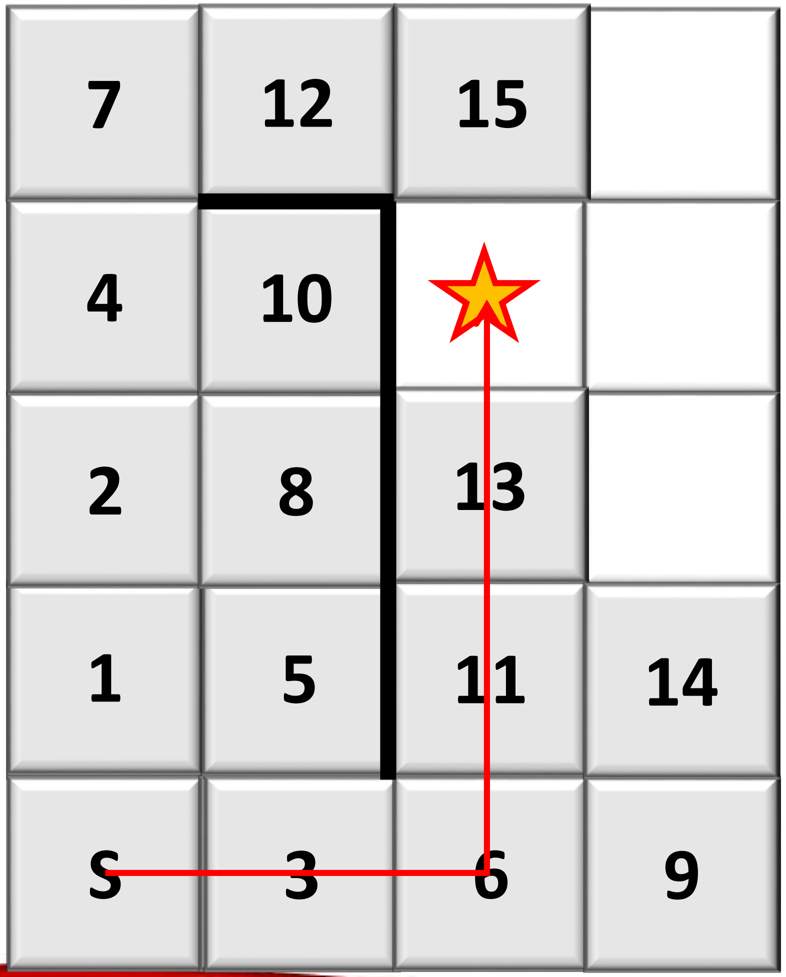
inadmissible heuristic

- What if the heuristic is too pessimistic?
    - Estimated cost > true cost
    - No longer guaranteed to be optimal
- What if the heuristic is just right?
    - Pre-compute the cost between all nodes
    - Feasible for you?

Summary

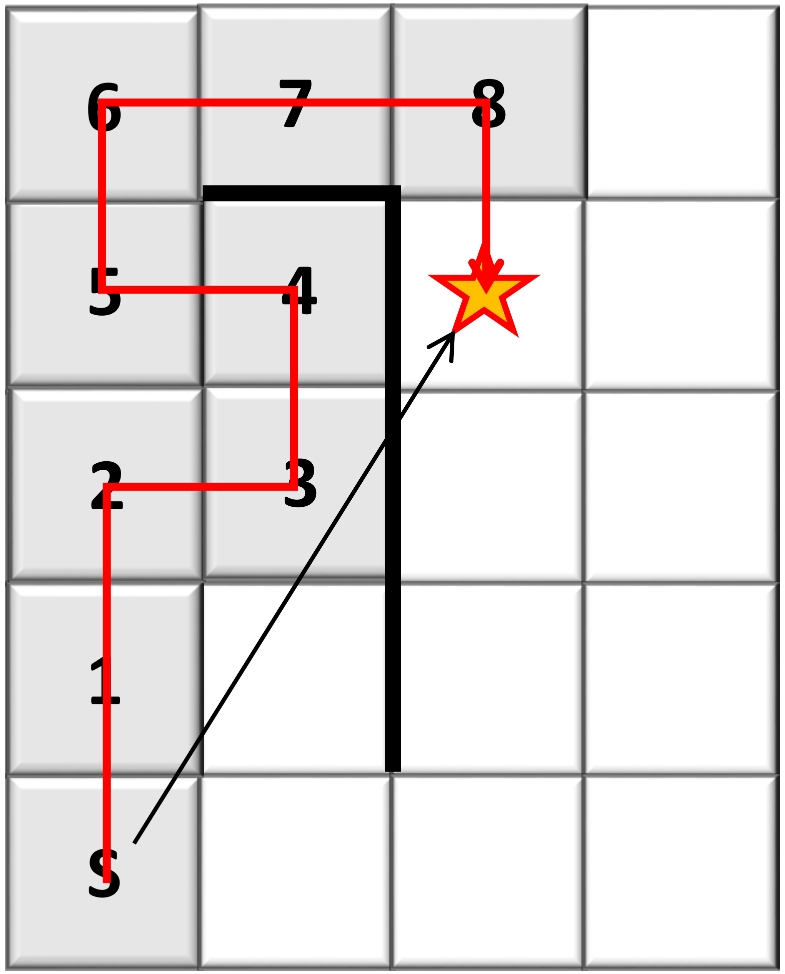
Dijkstra

minimum path



7	12	15	
4	10	★	
2	8	13	
1	5	11	14
S	3	6	9

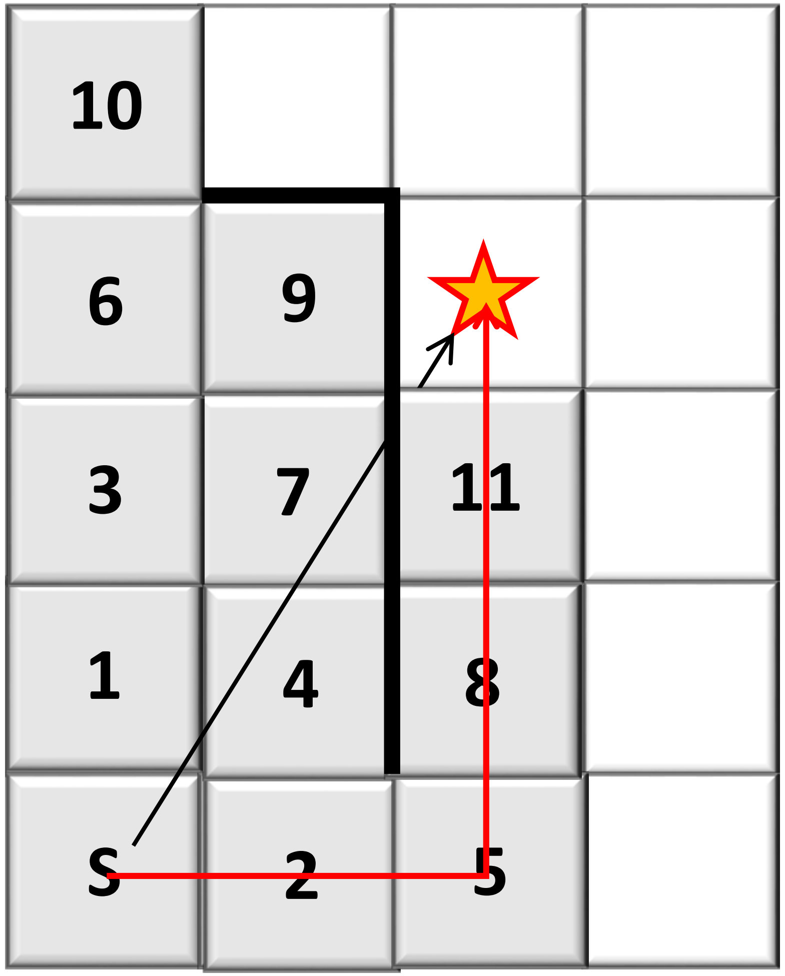
Greedy



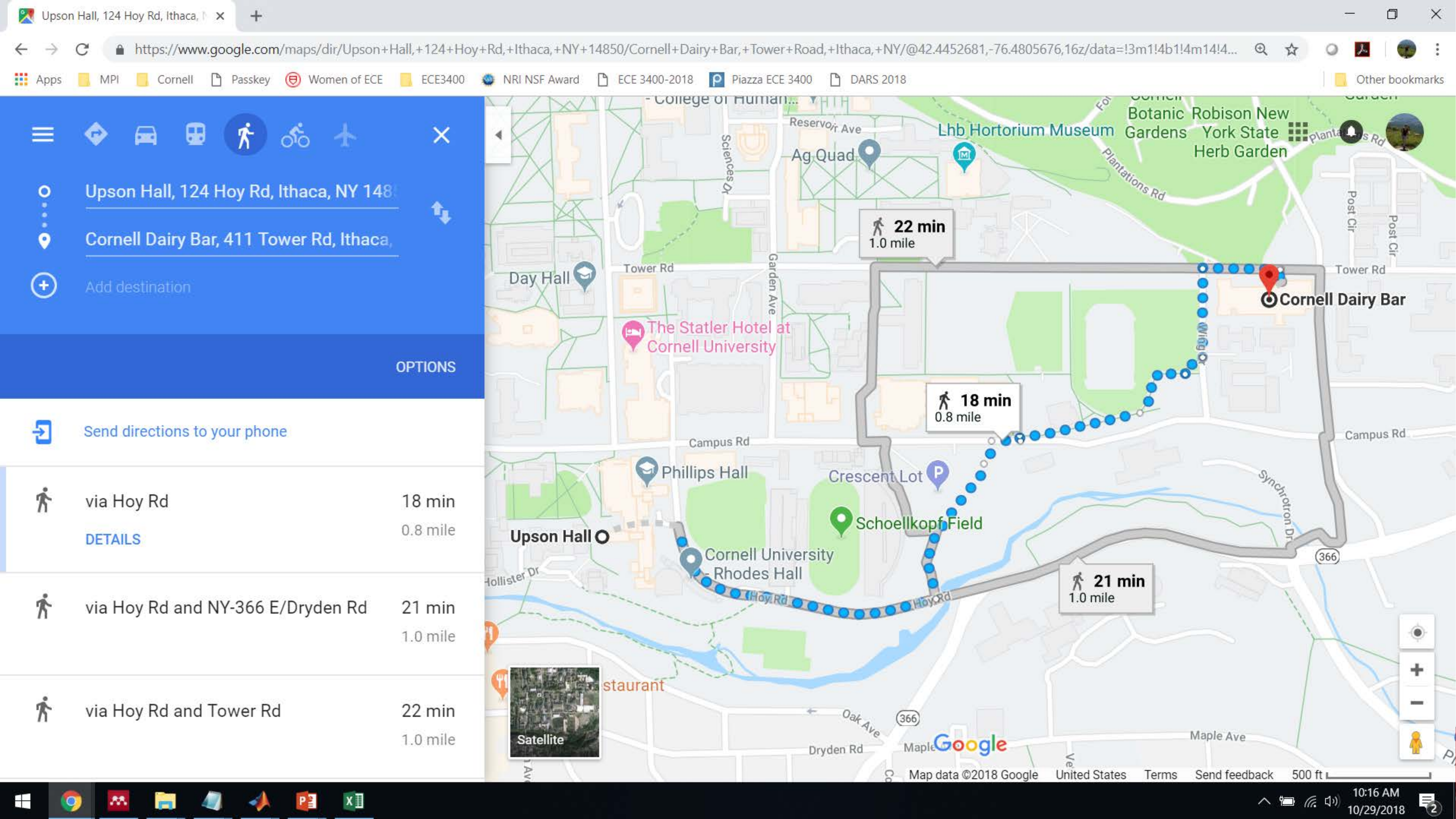
6	7	8	
5	4	★	
2	3		
1			
S			

A*

minimum path and efficient



10			
6	9	★	
3	7	11	
1	4	8	
S	2	5	



Upson Hall, 124 Hoy Rd, Ithaca, N

https://www.google.com/maps/dir/Upson+Hall,+124+Hoy+Rd,+Ithaca,+NY+14850/Cornell+Dairy+Bar,+Tower+Road,+Ithaca,+NY/@42.4452681,-76.4805676,16z/data=!3m1!4b1!4m14!4...

Upson Hall, 124 Hoy Rd, Ithaca, NY 148

Cornell Dairy Bar, 411 Tower Rd, Ithaca,

Add destination

OPTIONS

Send directions to your phone

via Hoy Rd DETAILS	18 min	0.8 mile
via Hoy Rd and NY-366 E/Dryden Rd	21 min	1.0 mile
via Hoy Rd and Tower Rd	22 min	1.0 mile

Game Theory

- Pick a whole number between 1 and 100.
- The winner is the person who picks the value which is closest to two thirds of the class average.
- E.g.
  - [10, 20, 60].
  - Class average 30.
  - Winner: 20.

- https://bit.ly/2z9R56F
- The poll will close at the end of the class (12.10pm 10/29th)

Go Build Robots!

Class website: https://cei-lab.github.io/ece3400-2018/

Piazza: https://piazza.com/cornell/fall2018/ece3400/home