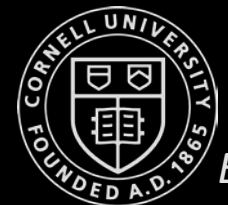
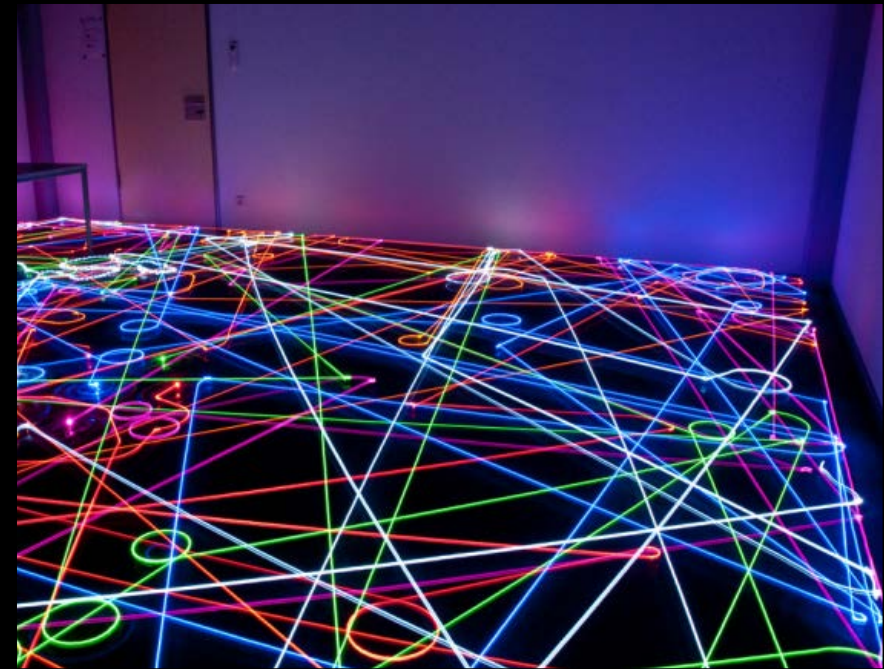


Fast Robots



Algorithms and Search

- **What is the simplest thing to do?**
 - Random or brute force search
- **Other methods?**
 - Uninformed search
 - Depth First Search (DFS)
 - Breadth First Search (BFS)
 - Dijkstra's Search
 - Informed Search
 - Greedy
 - A*
 - (and many more)



Comparing Search Algorithms

Definitions

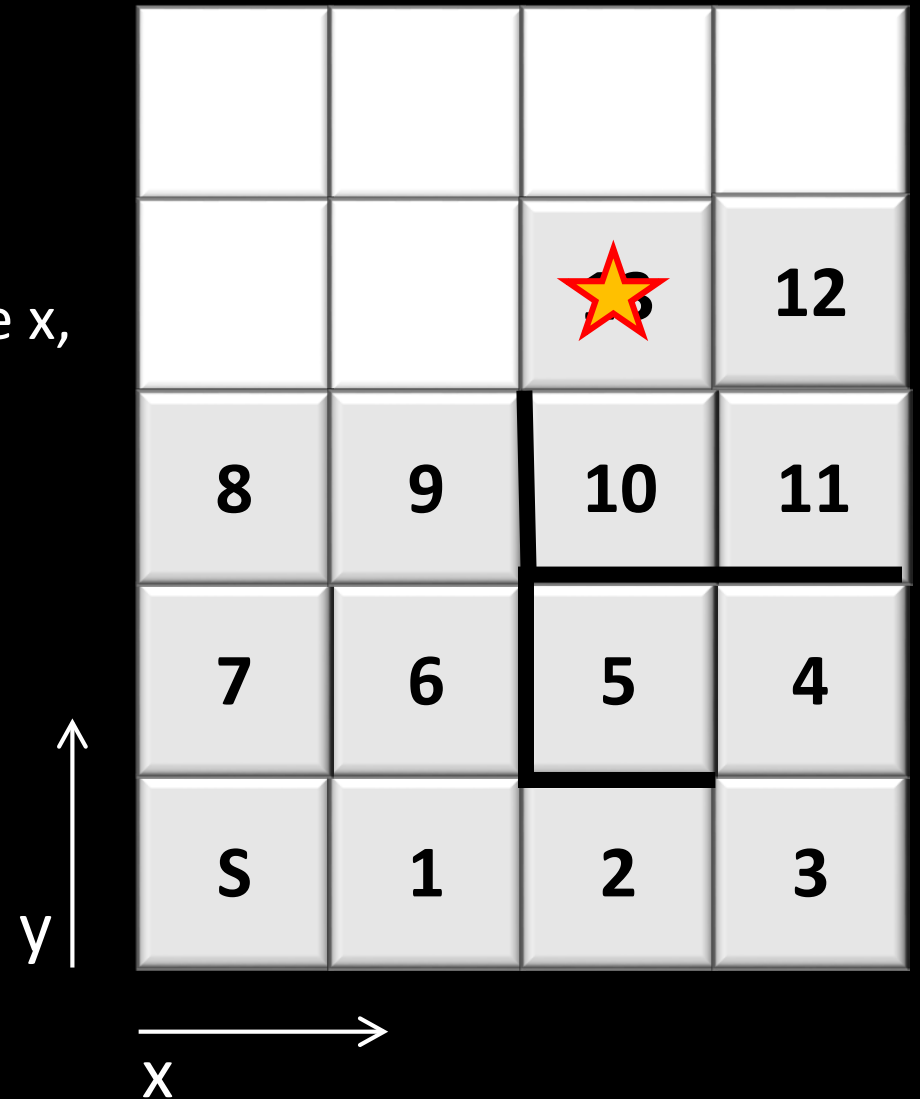
- Complete
 - A search algorithm is complete if, whenever at least one solution exists, the algorithm is guaranteed to find a solution within a finite amount of time
- Time complexity
 - The time complexity of a search algorithm is an expression for the worst-case amount of time it will take to run, expressed in terms of the maximum path length, m , and the maximum branching factor, b
- Space complexity
 - The space complexity of a search algorithm is an expression for the worst-case amount of memory that the algorithm will use, expressed in terms of m and b
- Optimality
 - A search is optimal if it is complete, and only returns cost-minimizing solutions

Algorithms and Search

- **What is the simplest thing to do?**
 - Random or brute force search
 - **How many grid traversals will brute force take?**
 - First establish a search order
 - Advance x first, then increment y and decrease x, etc.

Search order: N, E, S, W

Find a goal

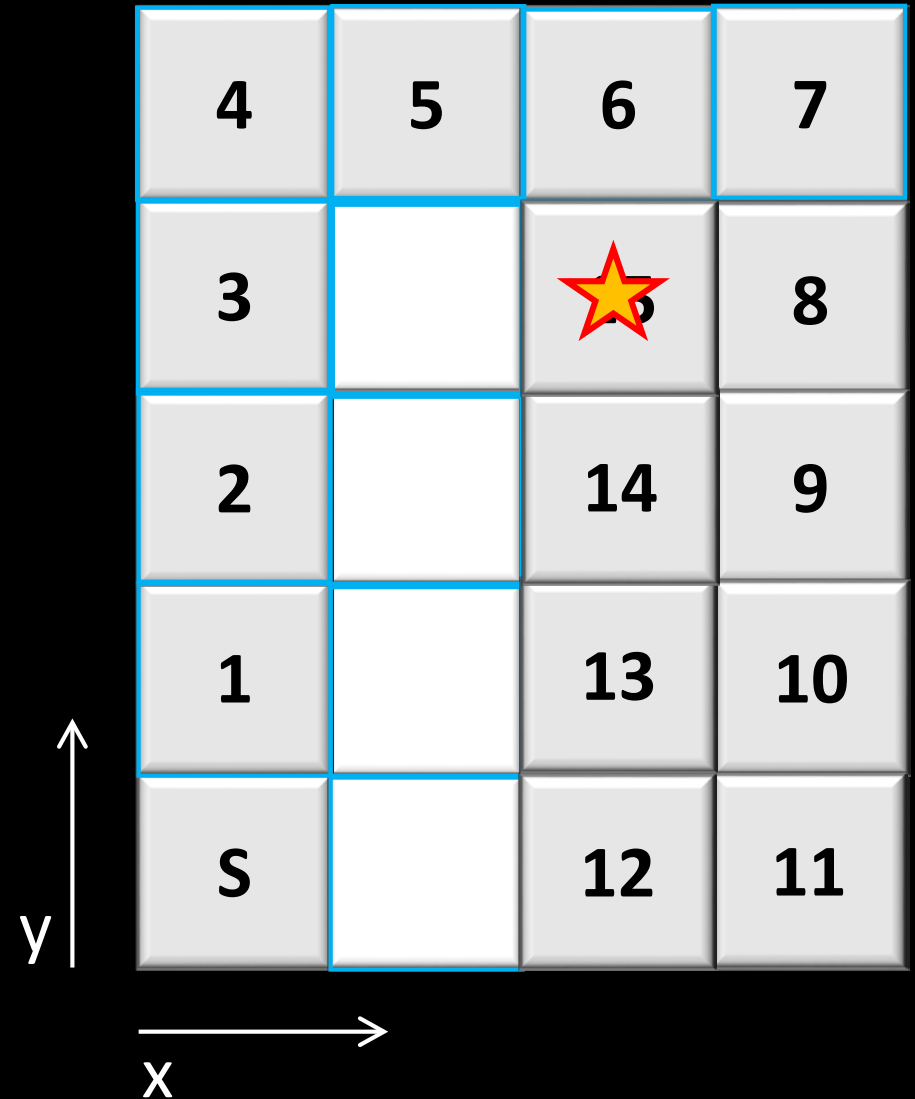


Algorithms and Search

- **What is the simplest thing to do?**
 - Random or brute force search
- **Other methods?**
 - Depth First Search (DFS)

Search order: N, E, S, W

Find a goal

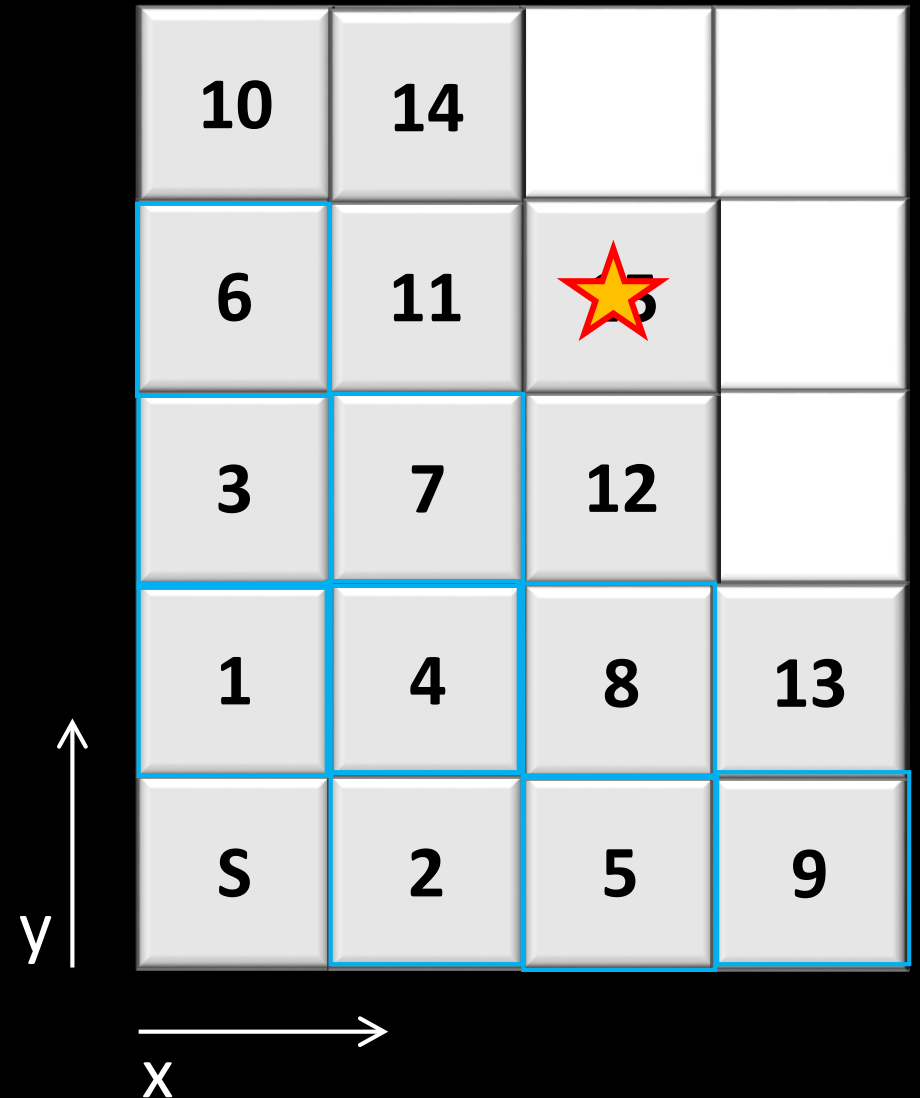


Algorithms and Search

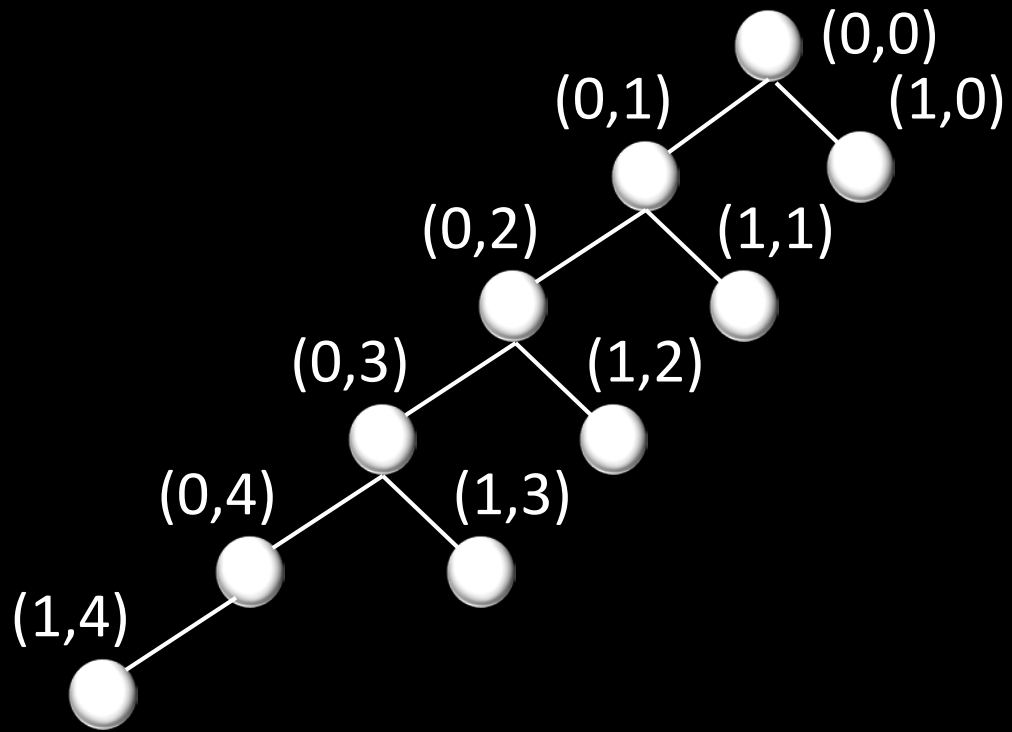
- **What is the simplest thing to do?**
 - Random or brute force search
- **Other methods?**
 - Depth First Search (DFS)
 - Breadth First Search (BFS)

Search order: N, E, S, W

Find a goal



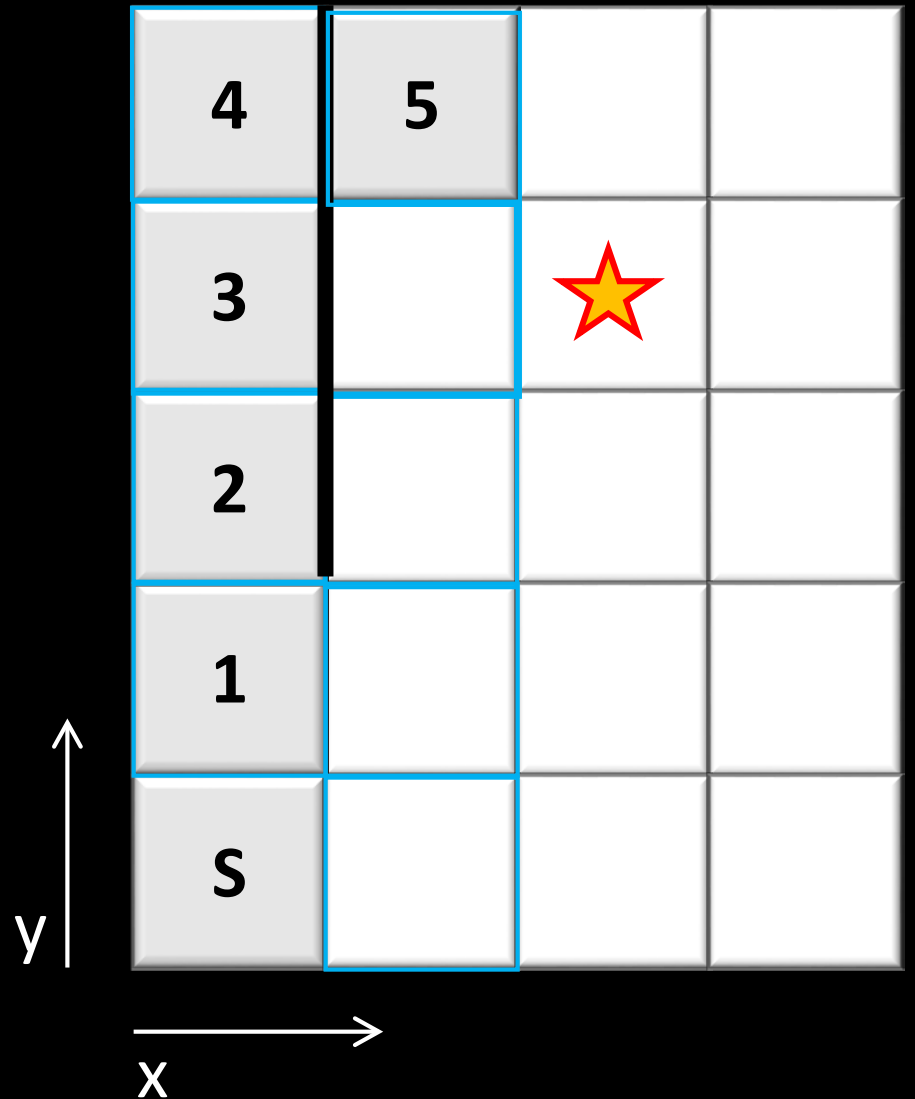
Depth First Search (DFS)



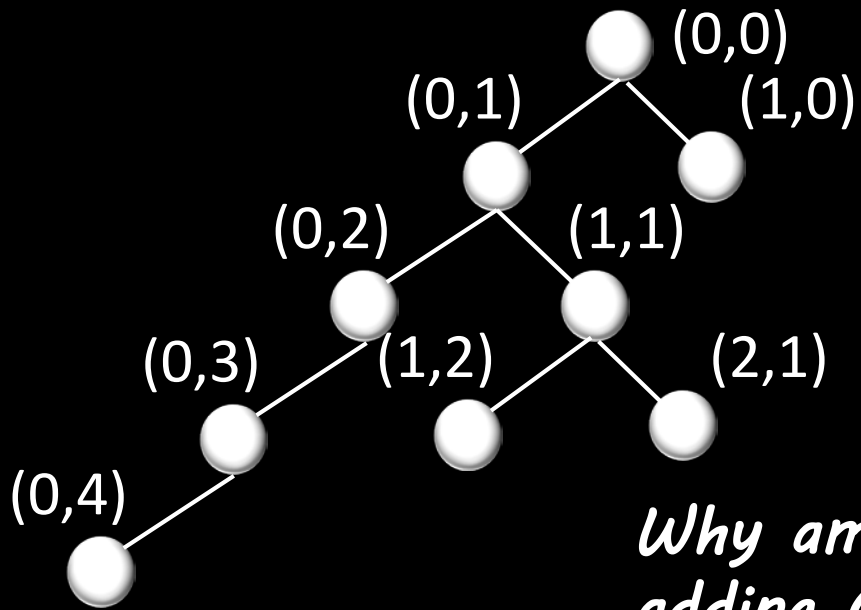
and so on...

Search order: N, E, S, W

Find a goal



Depth First Search (DFS)

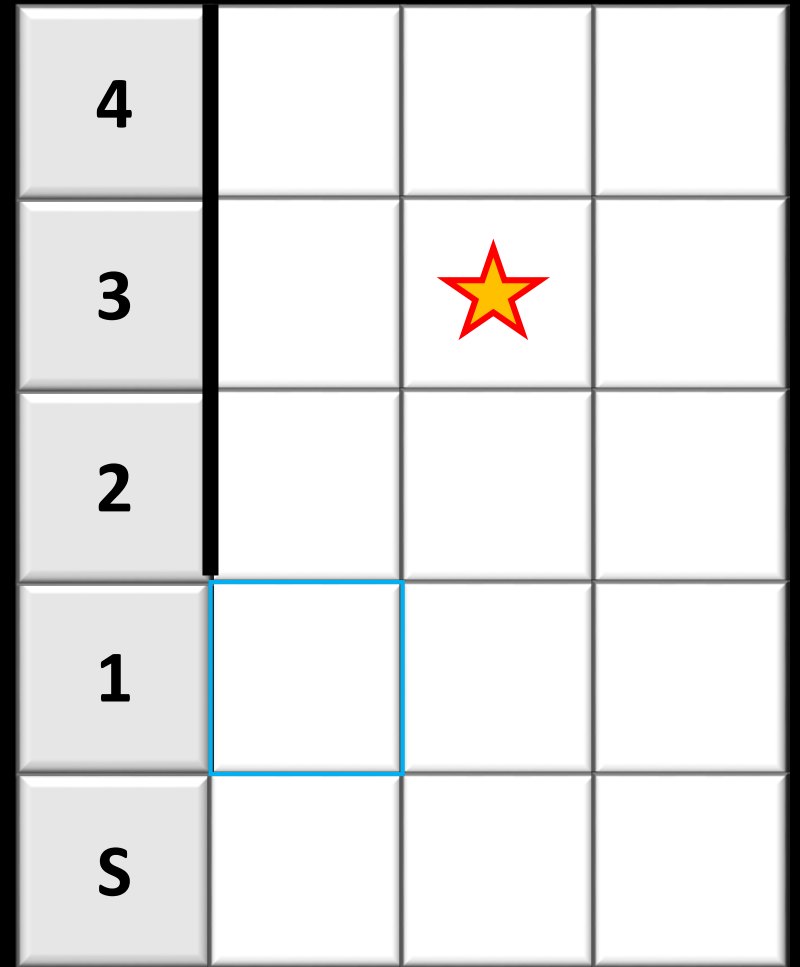


and so on...

Why am I not also adding (1,0)?

- Keep track of what is already on the frontier

Find a goal

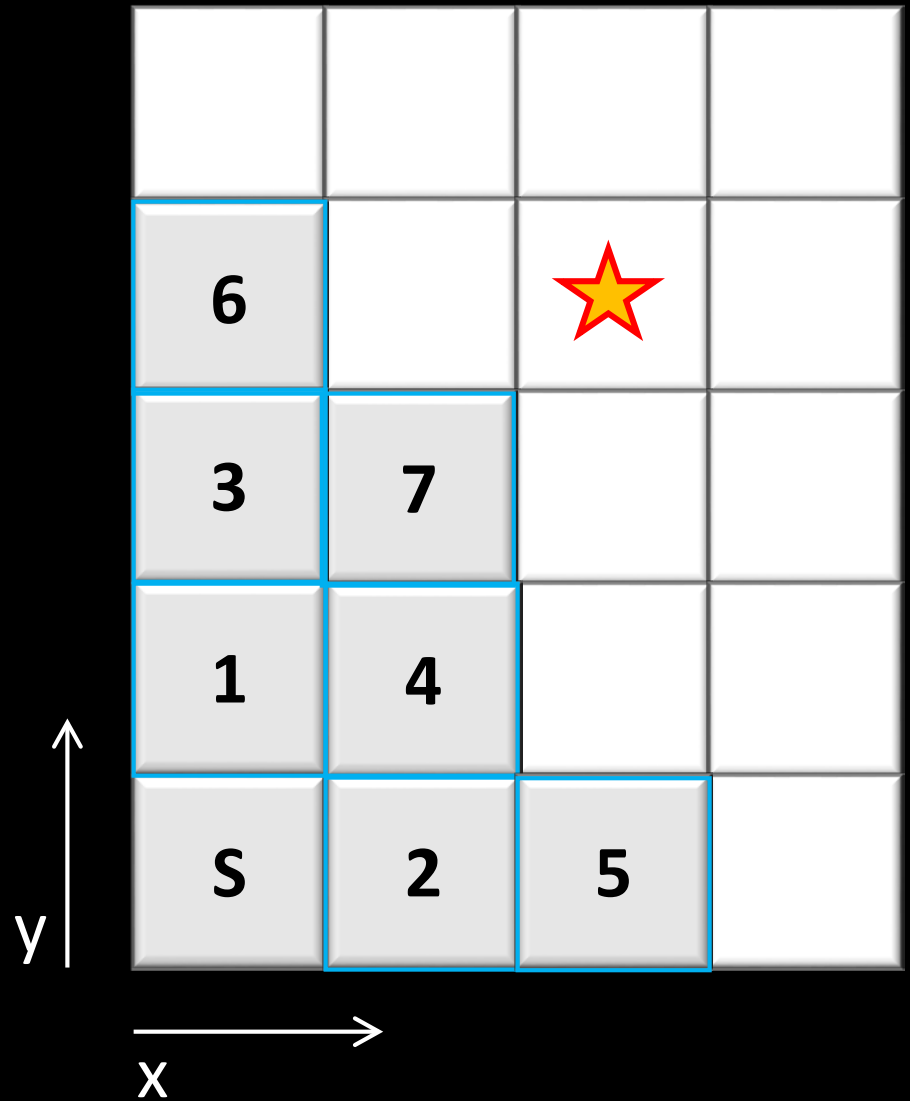
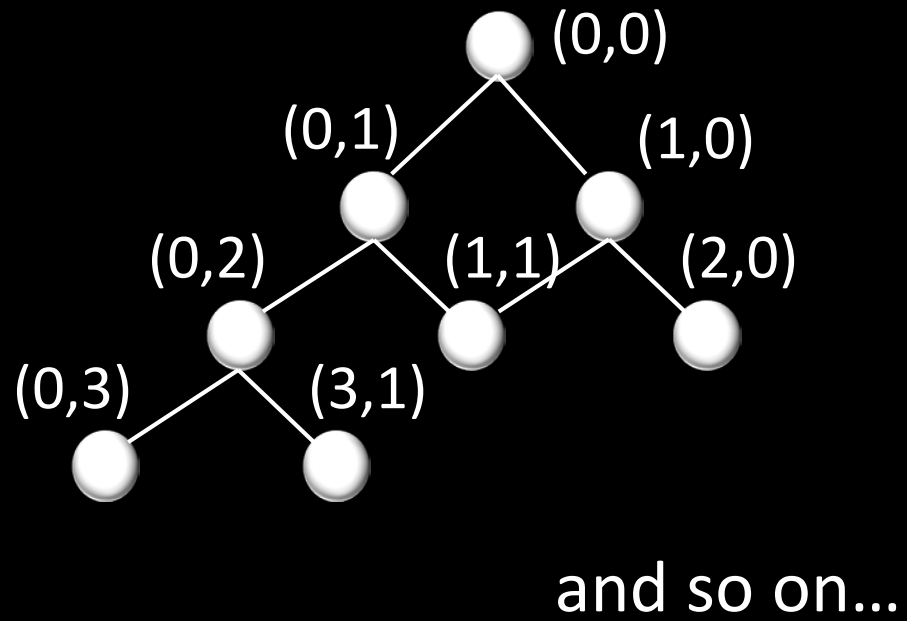


x

Breadth First Search (BFS)

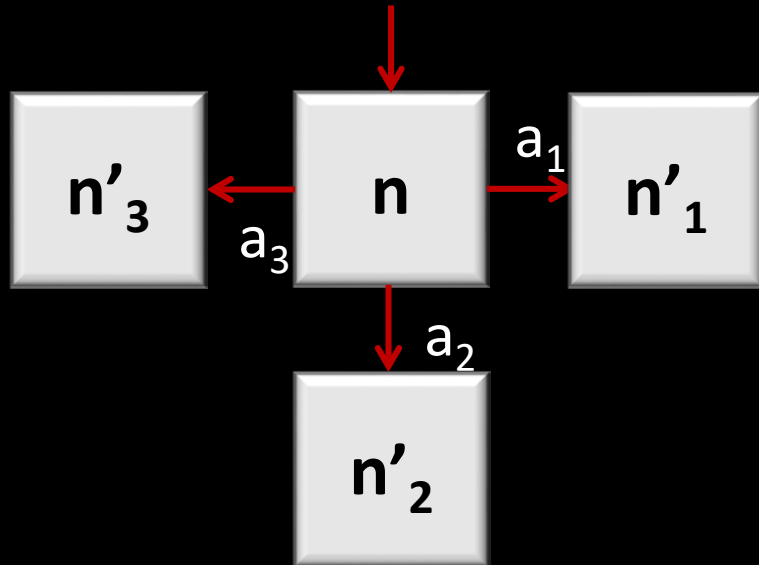
Search order: N, E, S, W

Find a goal

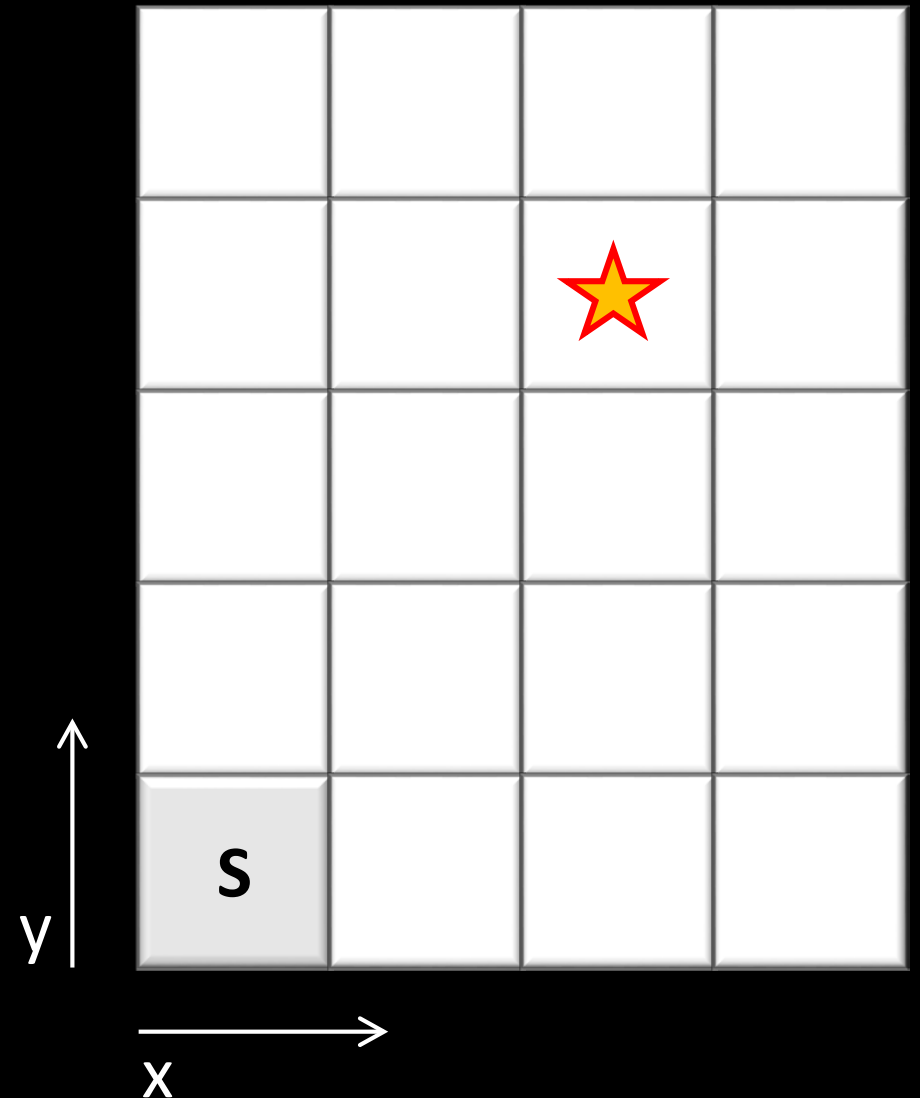


Search Algorithms, General

- Common graph structure
 - For every node, n
 - you have a set of actions, a
 - that moves you to a new node, n'



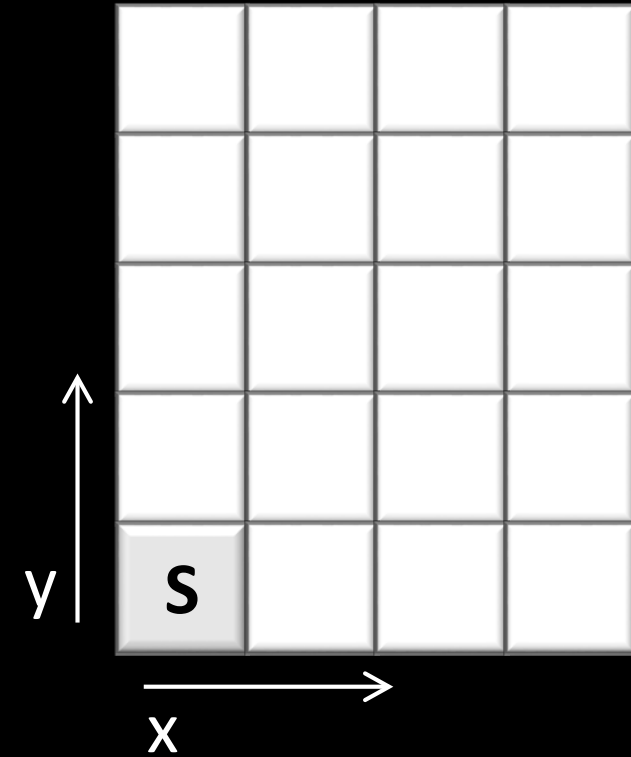
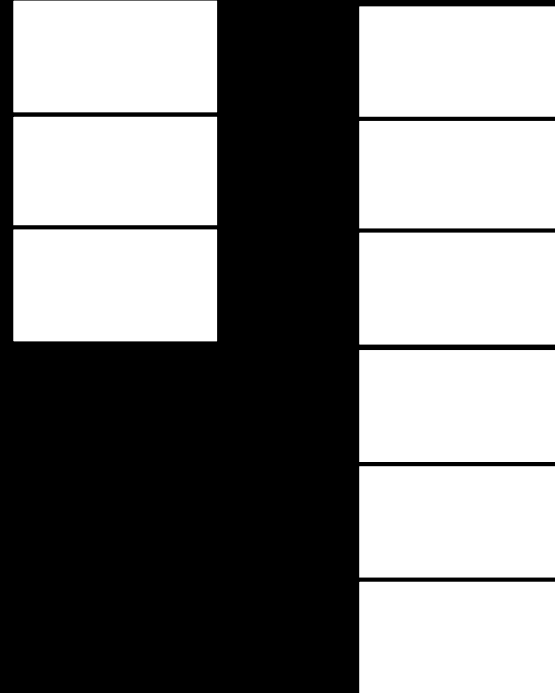
Find a goal



Search Algorithms, General

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

frontier visited



*How much space
to allocate to
visited?*

Depth First Search (DFS)

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

frontier visited

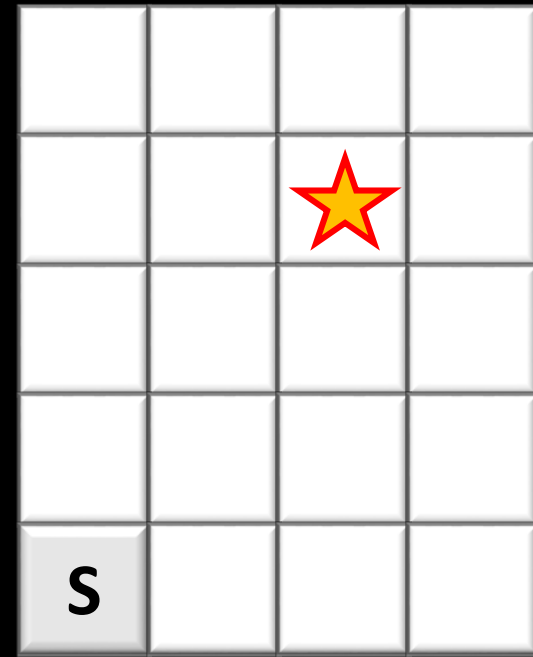
0,0

...

X*Y

y ↑

→ x

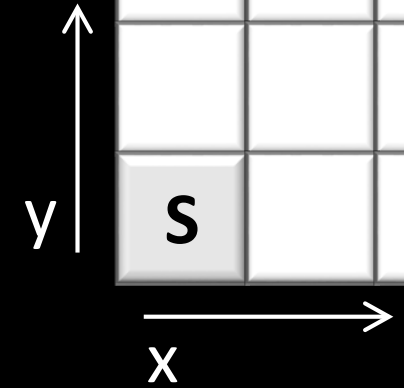


Depth First Search (DFS)

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

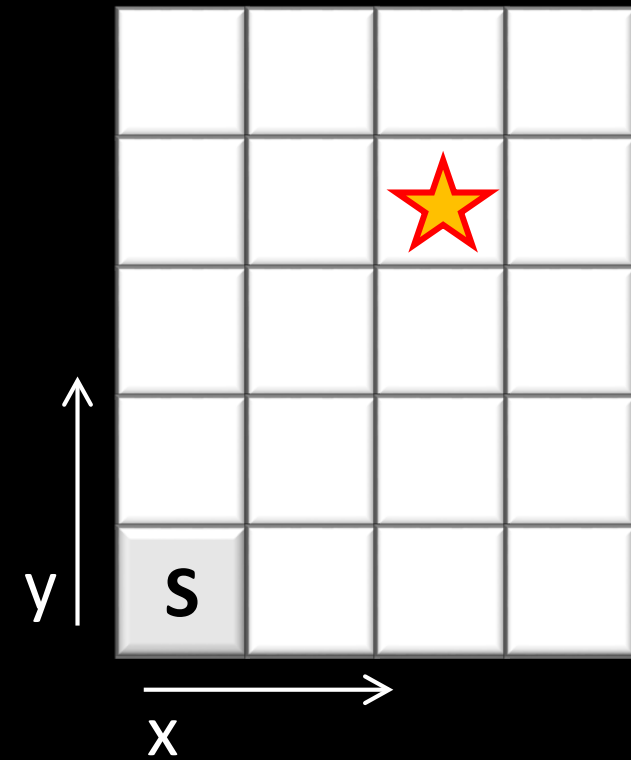
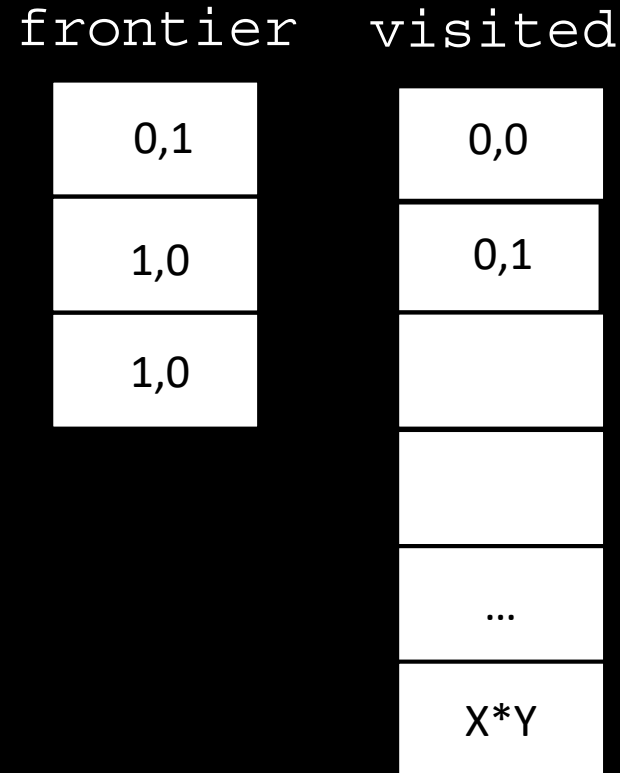
frontier visited

0,1	0,0
1,0	
	...
	X*Y



Depth First Search (DFS)

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

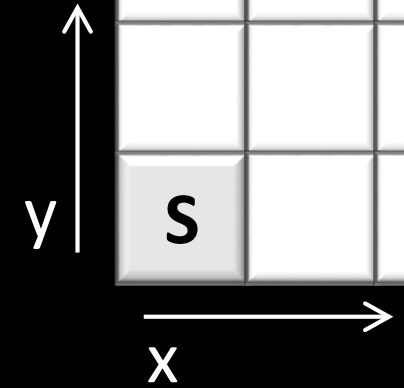


Depth First Search (DFS)

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

frontier visited

0,2	0,0
1,1	0,1
1,0	0,2
	...
	X*Y

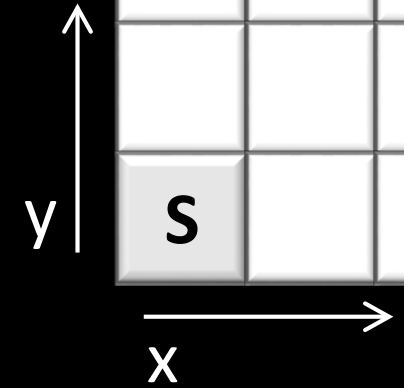


Depth First Search (DFS)

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

frontier visited

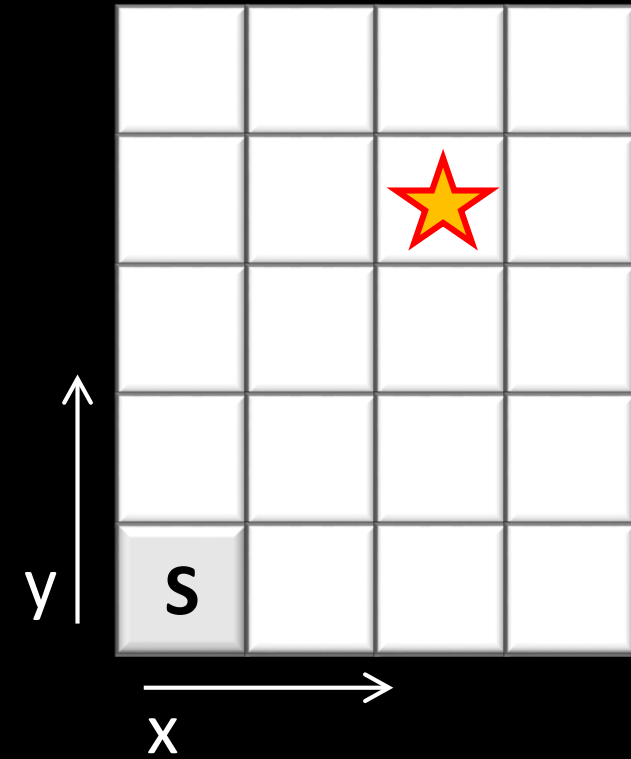
0,3	0,0
1,2	0,1
1,0	0,2
1,0	0,3



Depth First Search (DFS)

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

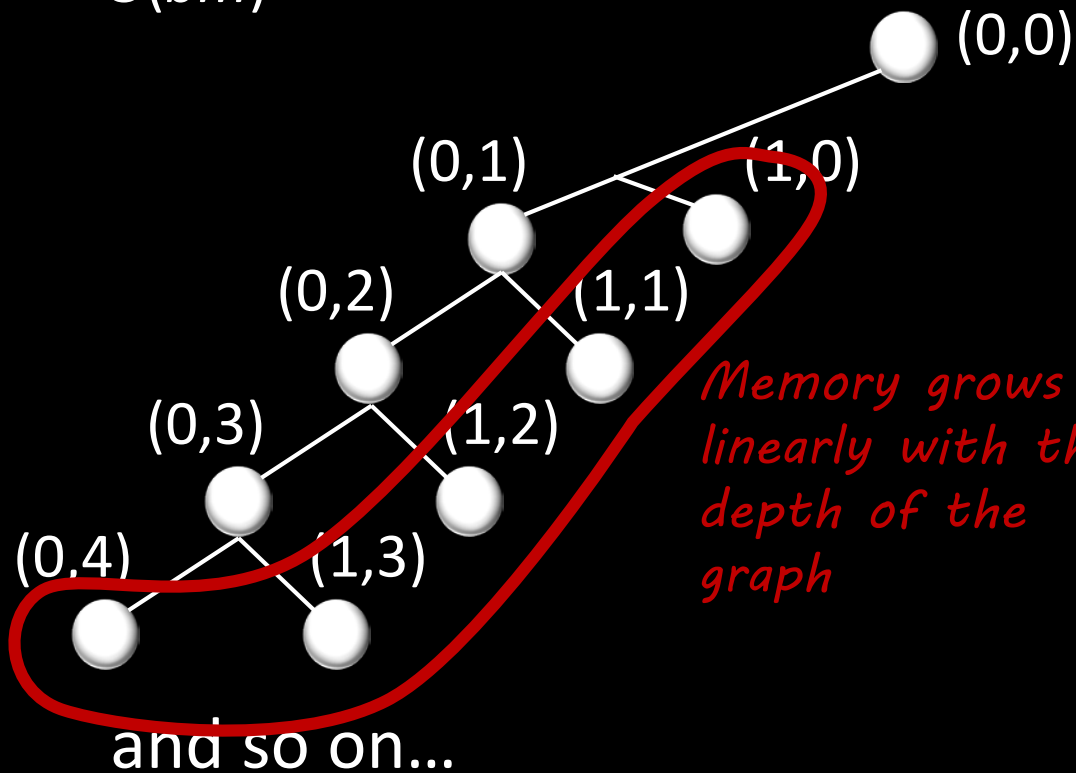
frontier	visited
0,4	0,0
1,3	0,1
1,2	0,2
1,1	0,3
1,0	
etc...	



Type of Buffer?
Last-In First-Out (LIFO) Buffer

Depth First Search (DFS)

- Is it complete?
 - Yes, but only on finite graphs
- What is the time complexity?
 - $O(b^m)$
- What is the space complexity?
 - $O(bm)$



Memory grows linearly with the depth of the graph

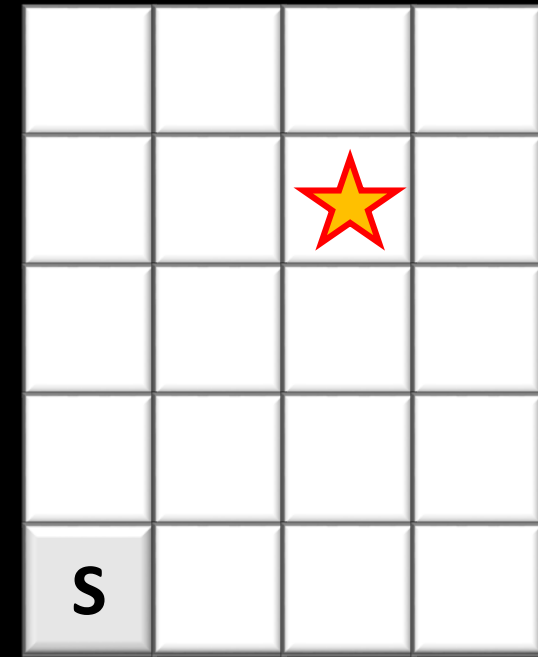
frontier

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

visited

0,0
0,1
0,2
0,3
...
X*Y

y ↑

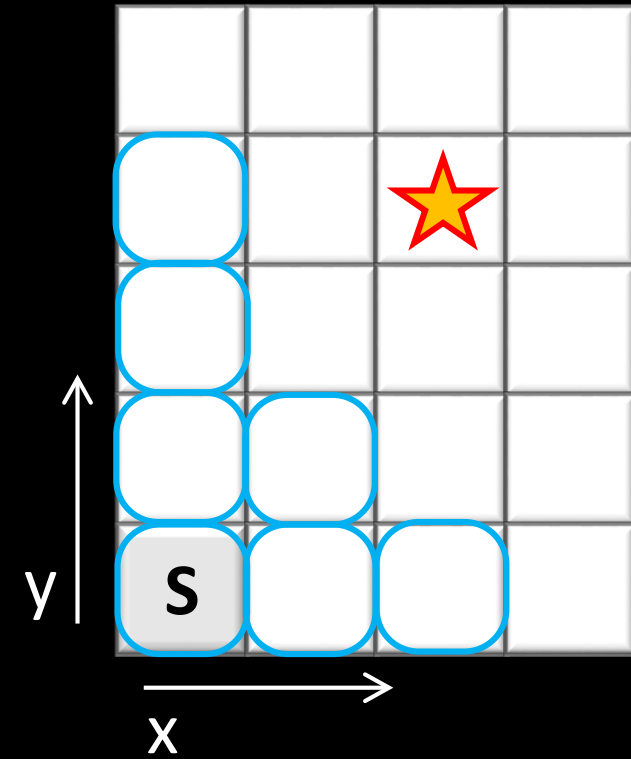


x →

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
```

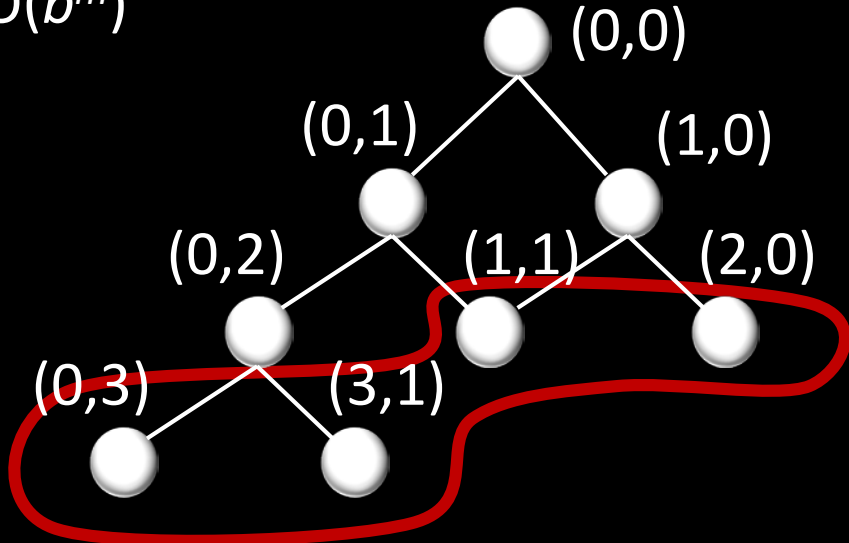
frontier	visited
0,0	0,0
0,1	0,1
1,0	1,0
0,2	0,2
1,1	
2,0	...
0,3	



Type of Buffer?
First-In First-Out (FIFO) Buffer

Breadth First Search (BFS)

- Is it complete?
 - Yes, as long as b is finite
 - Guaranteed to find the shortest path
- What is the time complexity?
 - $O(b^m)$
- What is the space complexity?
 - $O(b^m)$

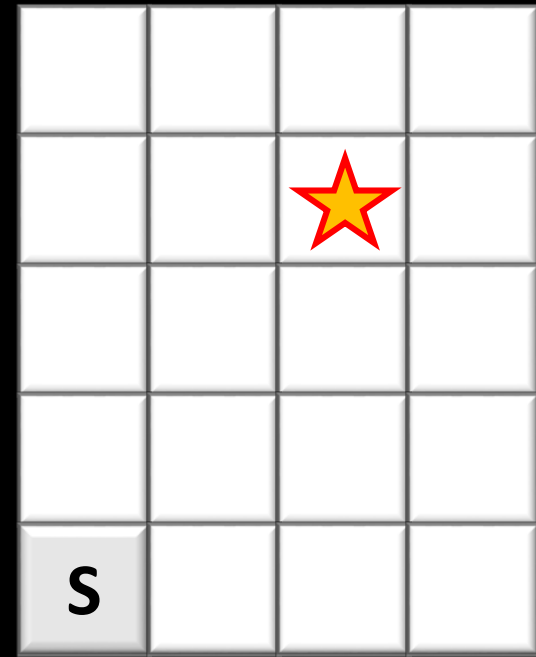


Memory grows exponentially with the depth of the graph

frontier visited

	0,0
	0,1
	1,0
	0,2
1,1	
2,0	
0,3	
	...

y ↑



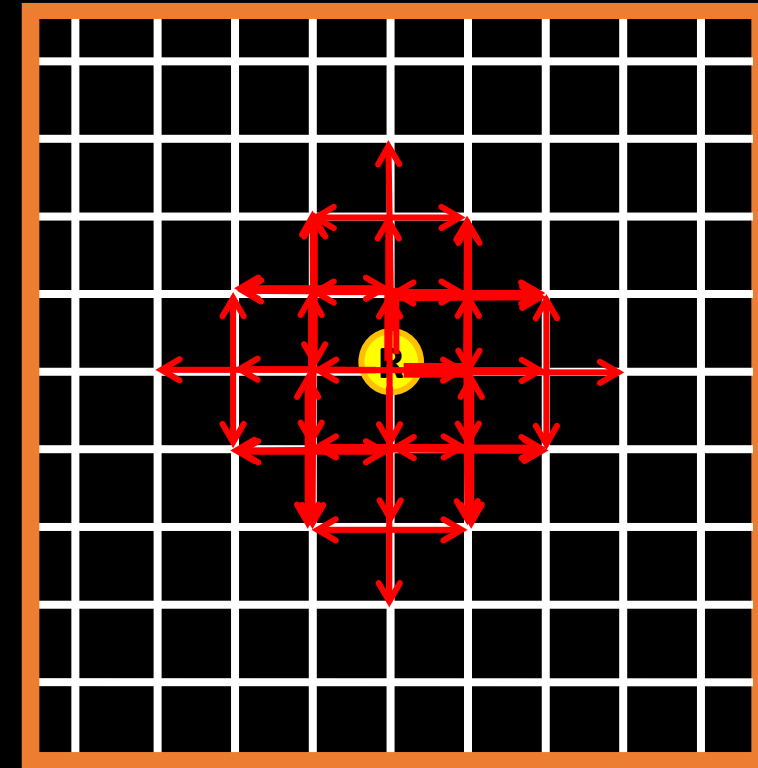
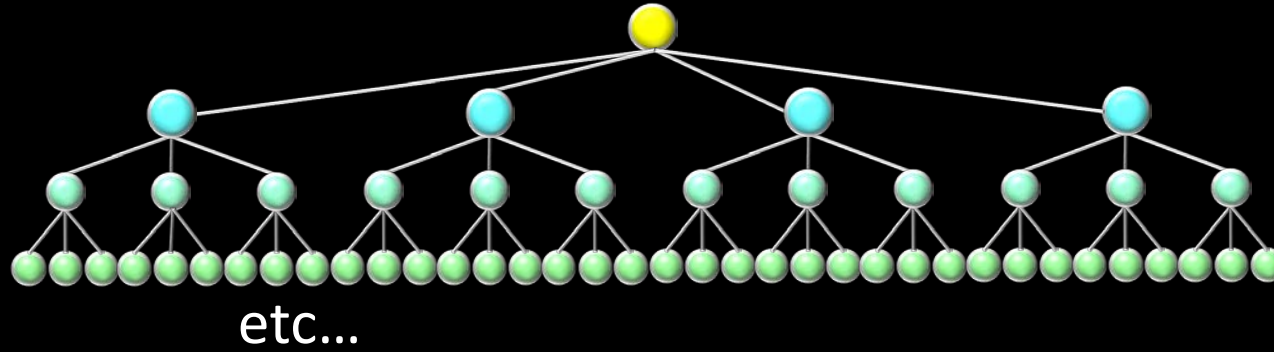
x →

*Type of Buffer?
First-In First-Out (FIFO) Buffer*

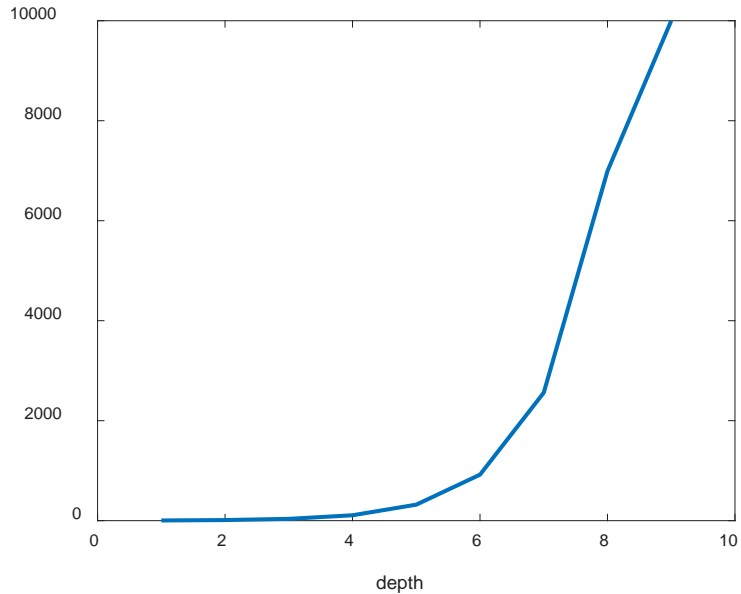
BFS: Memory and Computation

Frontier size:

- 4
- 12
- 36



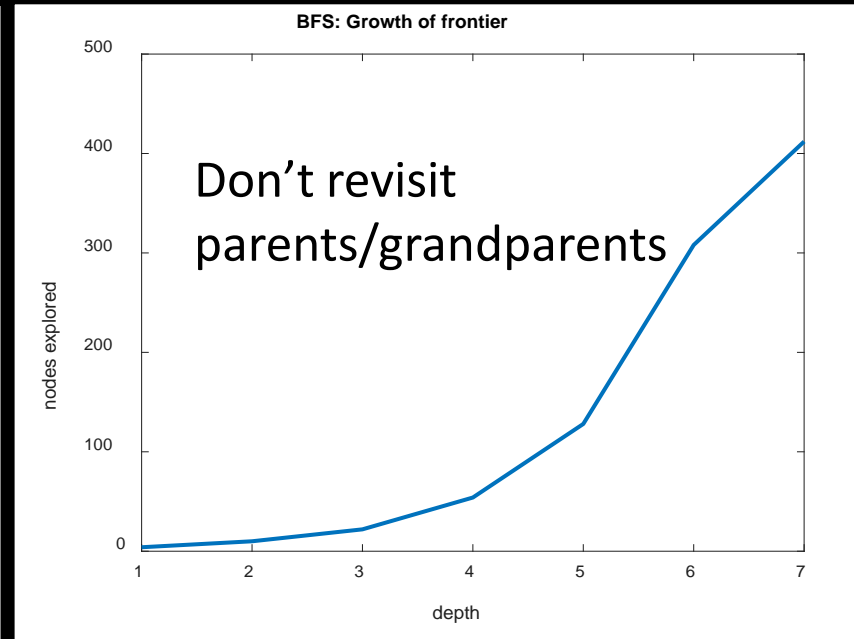
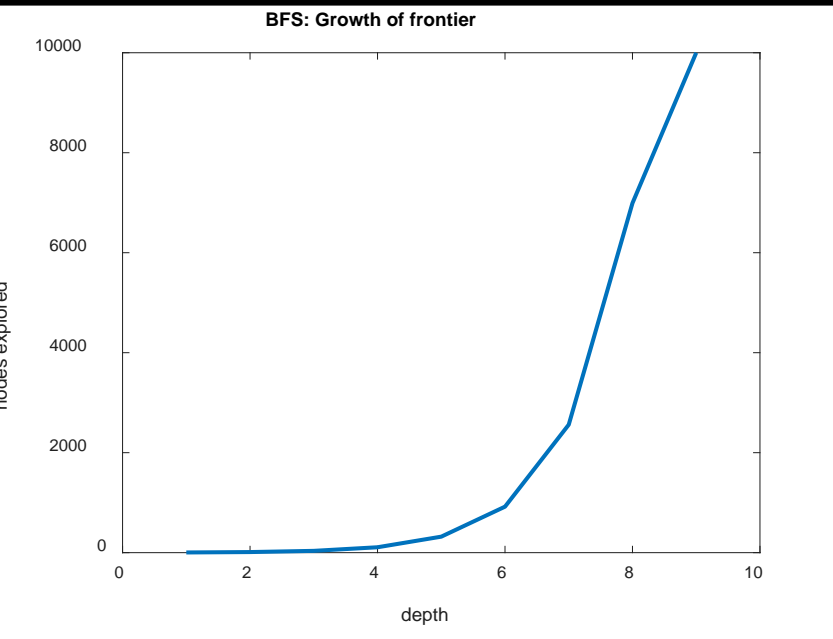
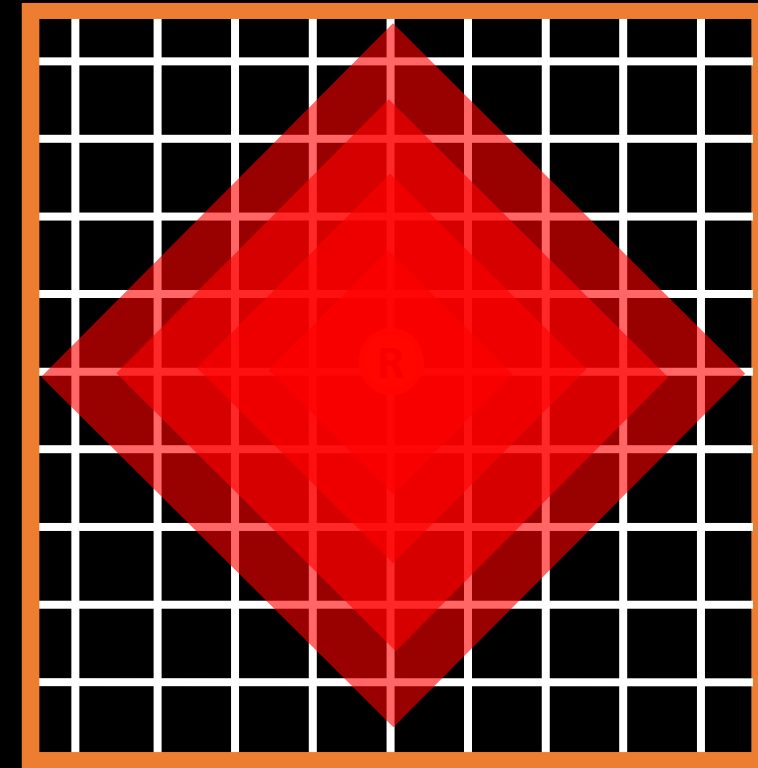
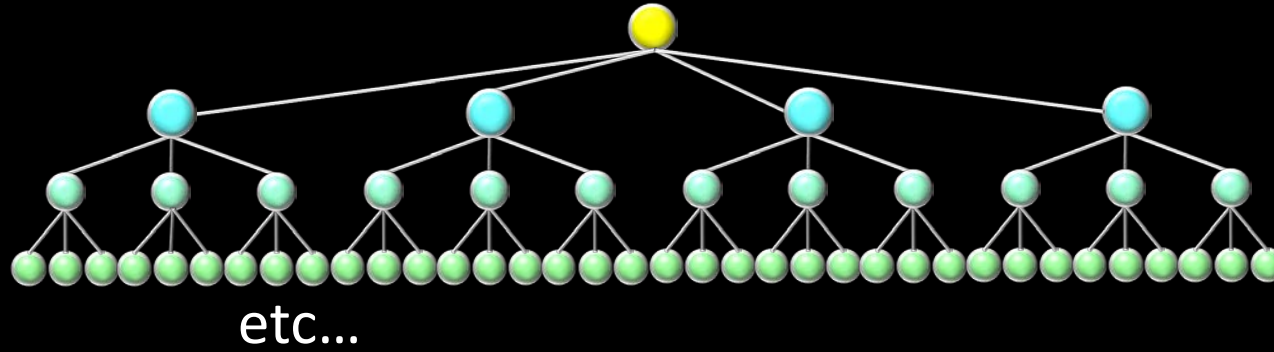
BFS: Growth of frontier



BFS: Memory and Computation

Frontier size:

- 4
- 12
- 36

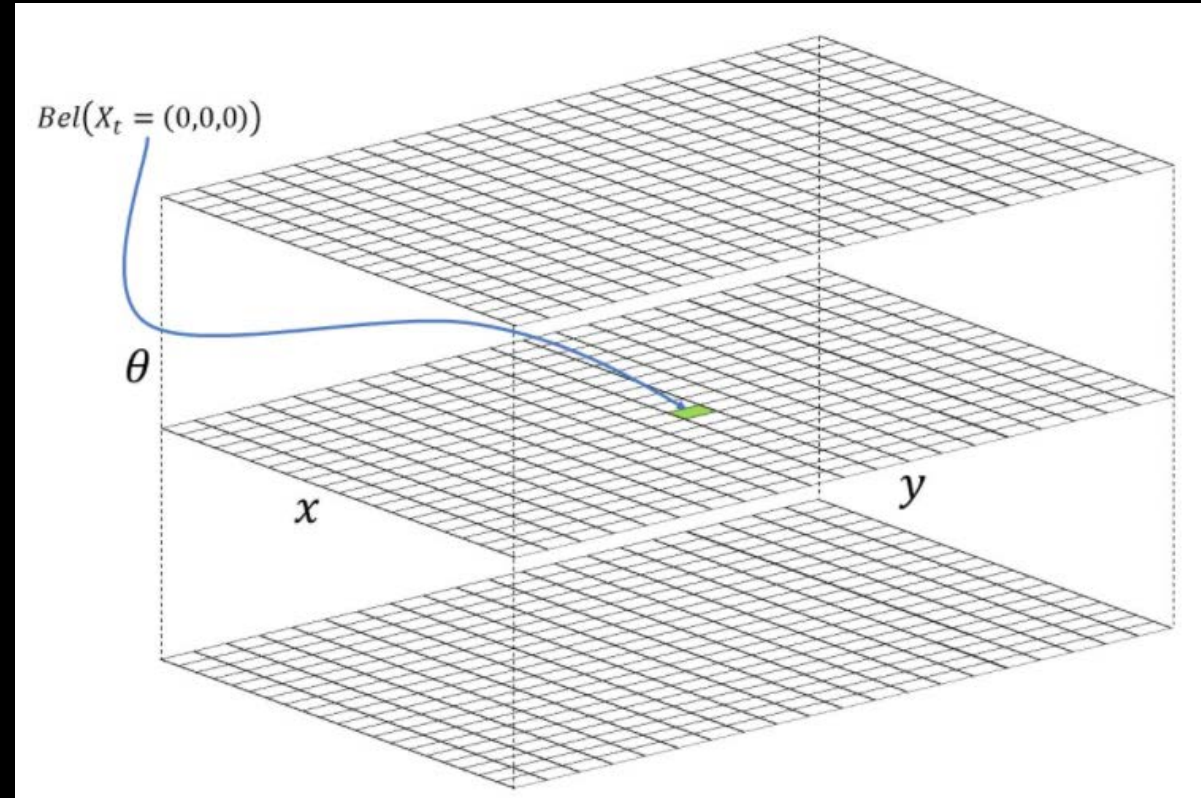


Uninformed Search Algorithms, General

- When is DFS appropriate?
 - If the space is restricted
 - If solutions tend to occur at the same depth in the tree
- When is DFS inappropriate?
 - If some paths have infinite length / if the graph contains cycles
 - If some solutions are very deep, while others are very shallow
- When is BFS appropriate?
 - If you need to find the shortest path
 - If space is not a problem
 - If some solutions are shallow
 - If there might be infinite paths
- When is BFS inappropriate?
 - If space is limited / if the branching factor is very large
 - If solutions tend to be located deep in the tree

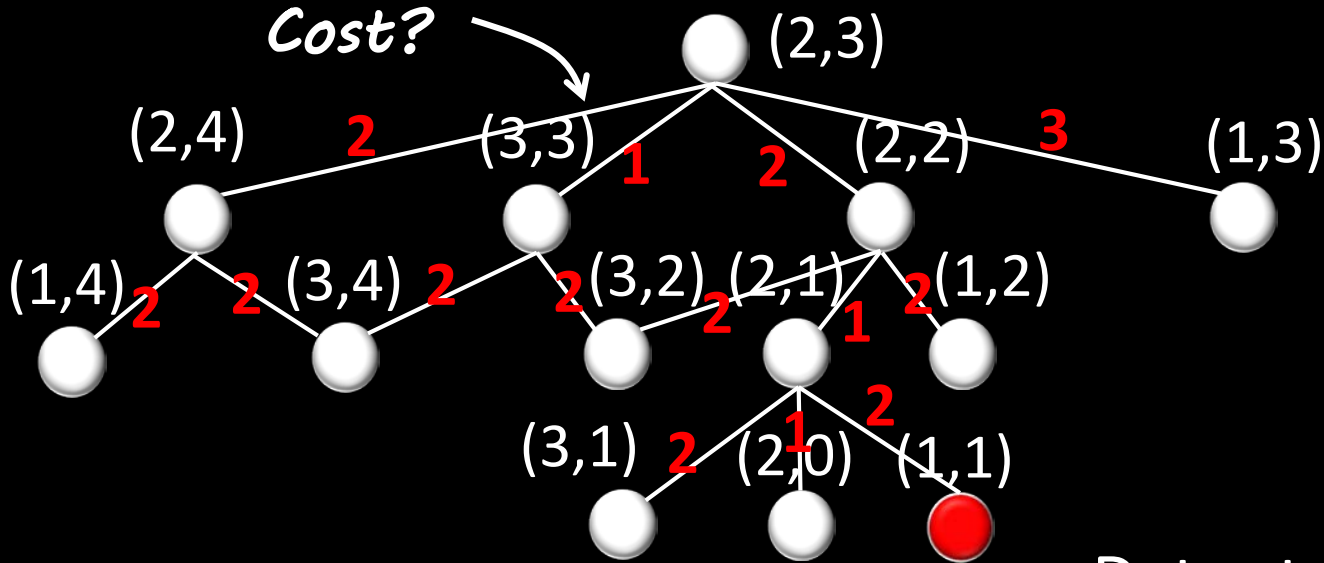
ECE4960 Robot

- Is BFS / DFS possible for your task on the Artemis?
 - What is the maximum branching factor?
 - $b = 6$
 - What is the longest path?
 - $m \sim 20 \cdot 20 \cdot 20 = 8,000$
 - Depth First Search
 - Frontier: $O(bm) = 48,000$ nodes
 - Float \rightarrow 192kB
 - Artemis memory?
 - 1MB flash and 384k RAM
 - Breadth First Search
 - Frontier: $O(b^m) = 6^{20 \cdot 20 \cdot 20}$
- *How do we find the optimal path, using less memory?*



Lowest-Cost First Search

- Consider parent cost!



What node to expand next?

Data structure

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

What cost heuristic could we add?

- Go straight, cost 1
- Turn one quadrant, cost 1

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

Lowest-Cost First Search

- Consider parent cost!

```
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)
```

What cost heuristic could we add?

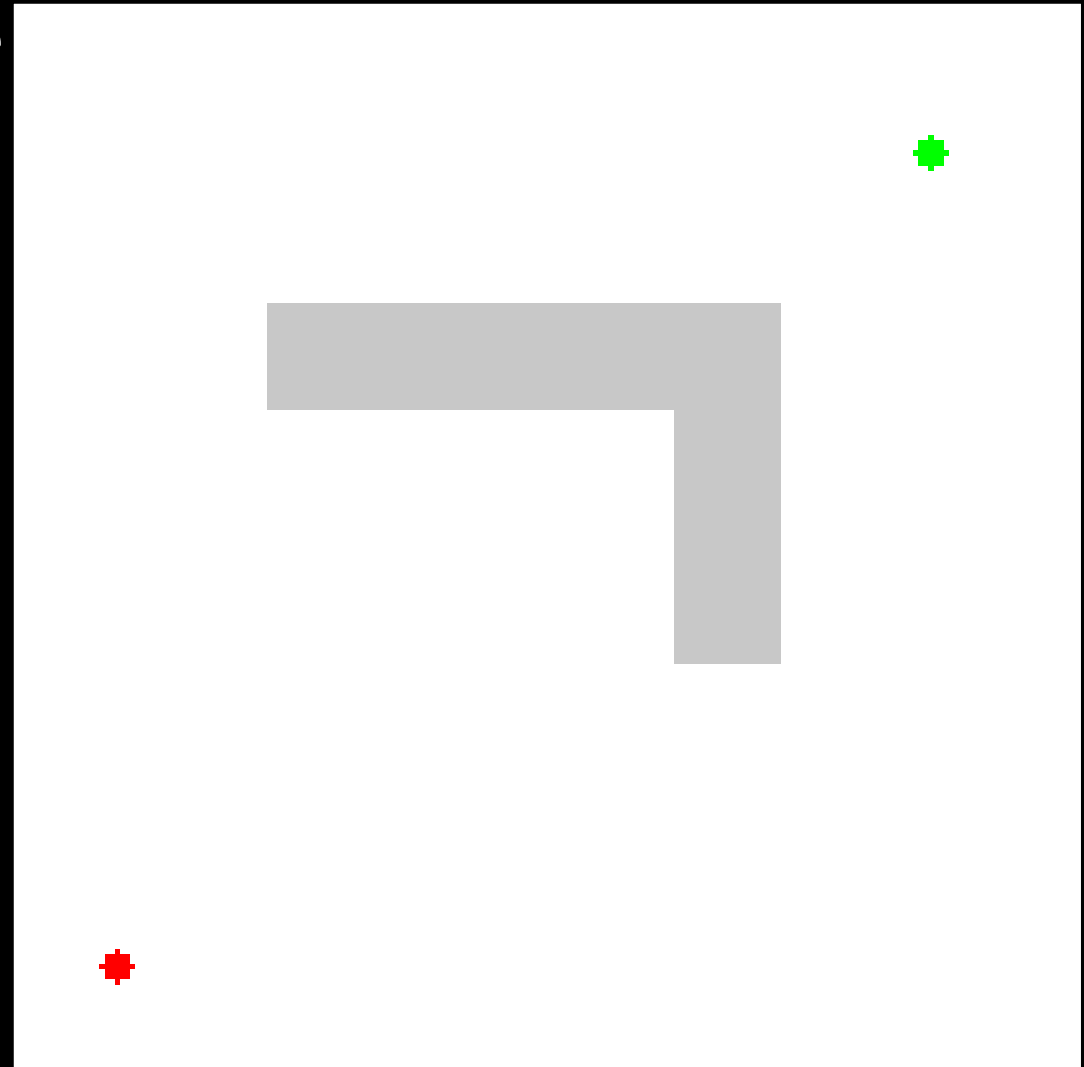
- Go straight, cost 1
- Turn one quadrant, cost 1

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

Lowest-Cost First Search

- Is it *complete*?
 - Yes, as long as path costs are positive
- What is the time complexity?
 - $O(b^m)$
- What is the space complexity?
 - $O(b^m)$

(Wikipedia)

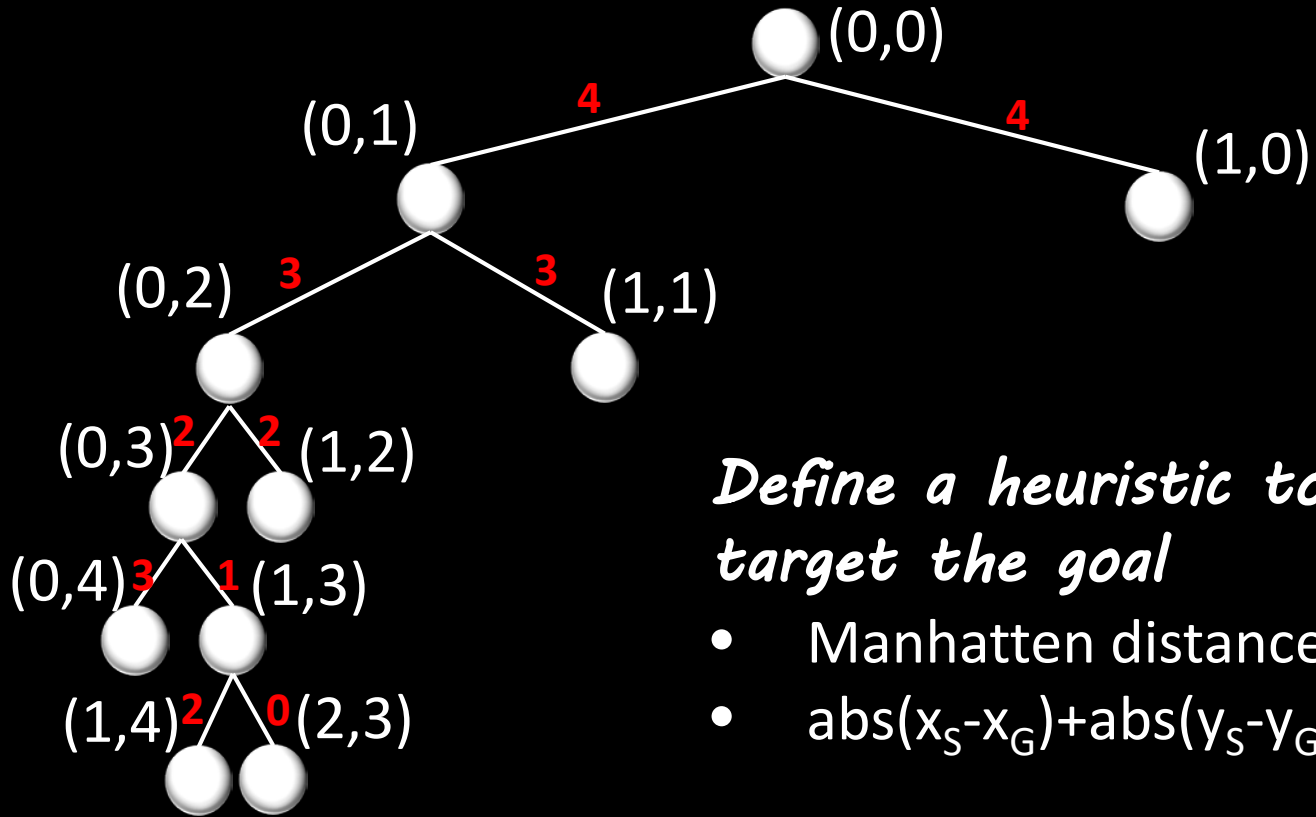


Could we be smarter?

- Sure, *you know where the goal is!*
- ...Informed search
 - Consider parent cost, and..
 - ..estimate the shortest path to the “goal”
- Assign a value to the frontier
 - Pick closest frontier (minimize distance)

Informed Search

- Greedy Search

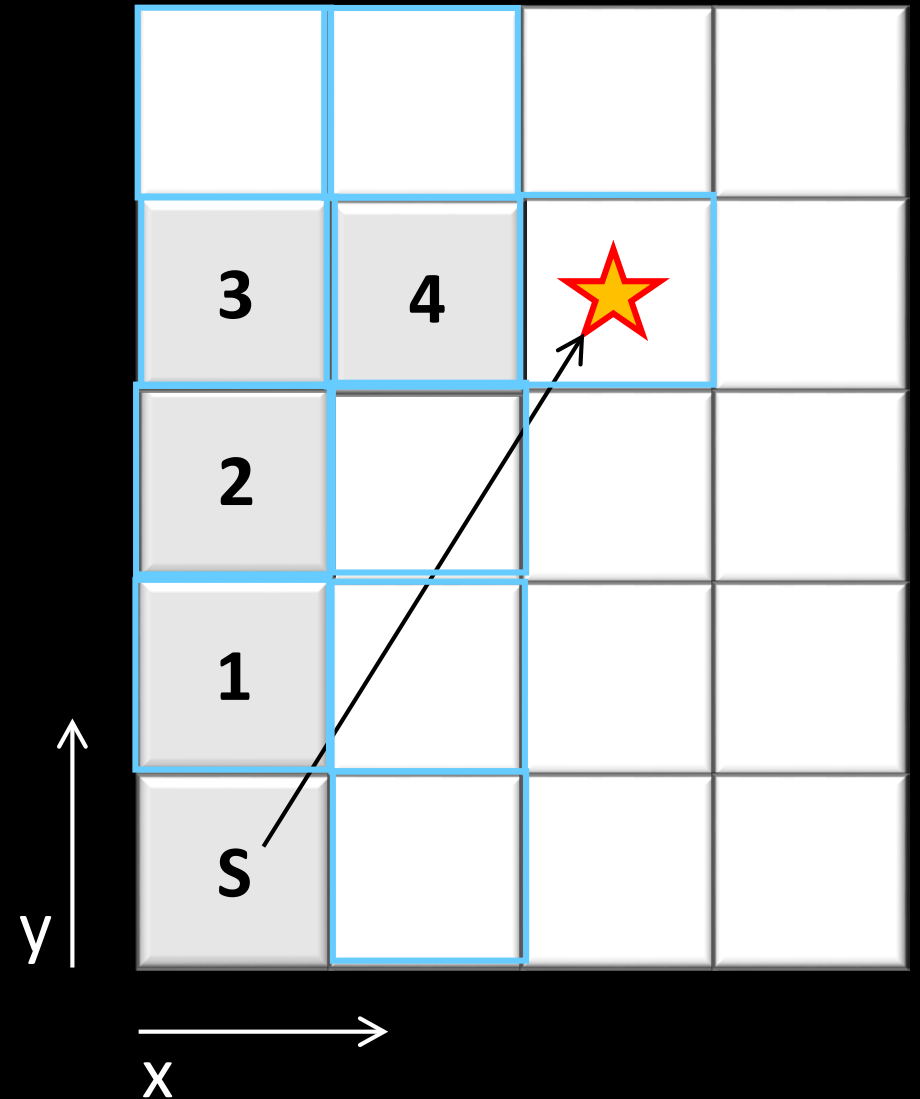


Define a heuristic to target the goal

- Manhattan distance
- $\text{abs}(x_S - x_G) + \text{abs}(y_S - y_G)$

Search order: N, E, S, W

Find a treasure



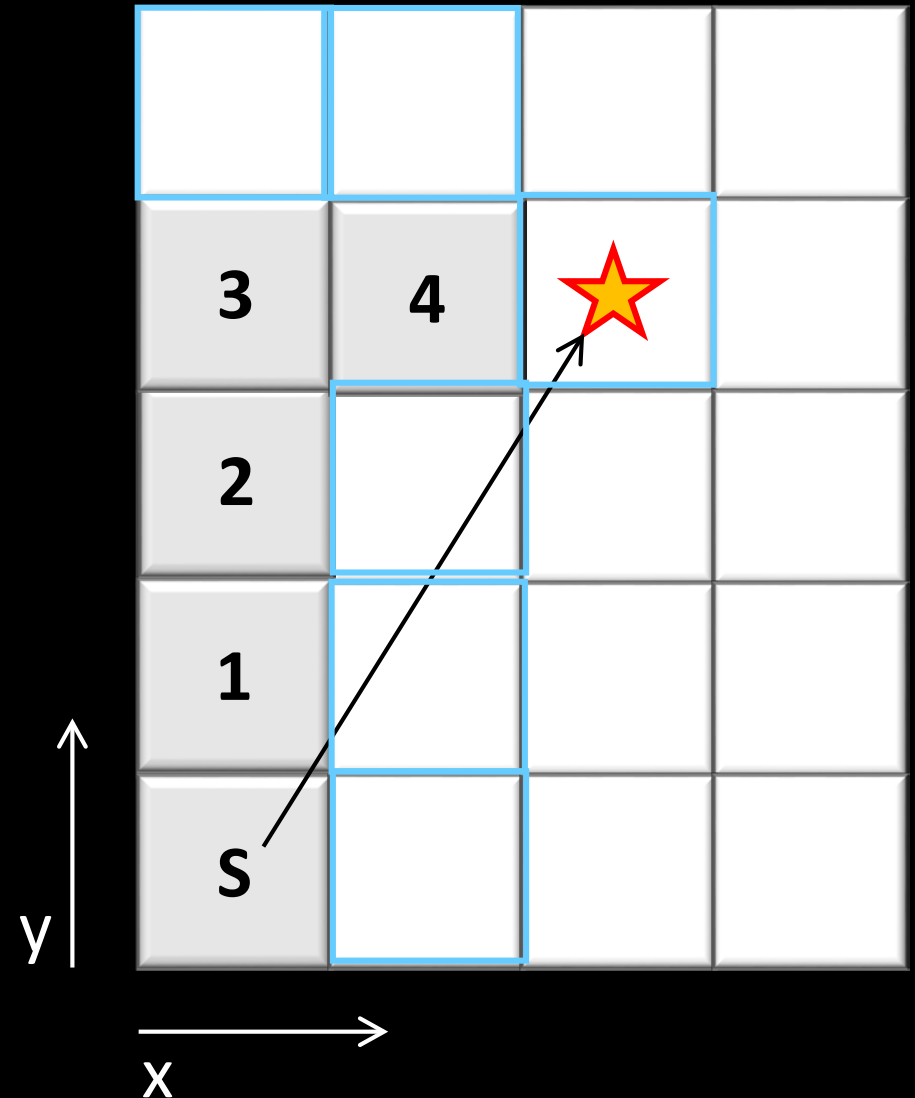
Informed Search

- Greedy Search

```
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)
```

Search order: N, E, S, W

Find a treasure

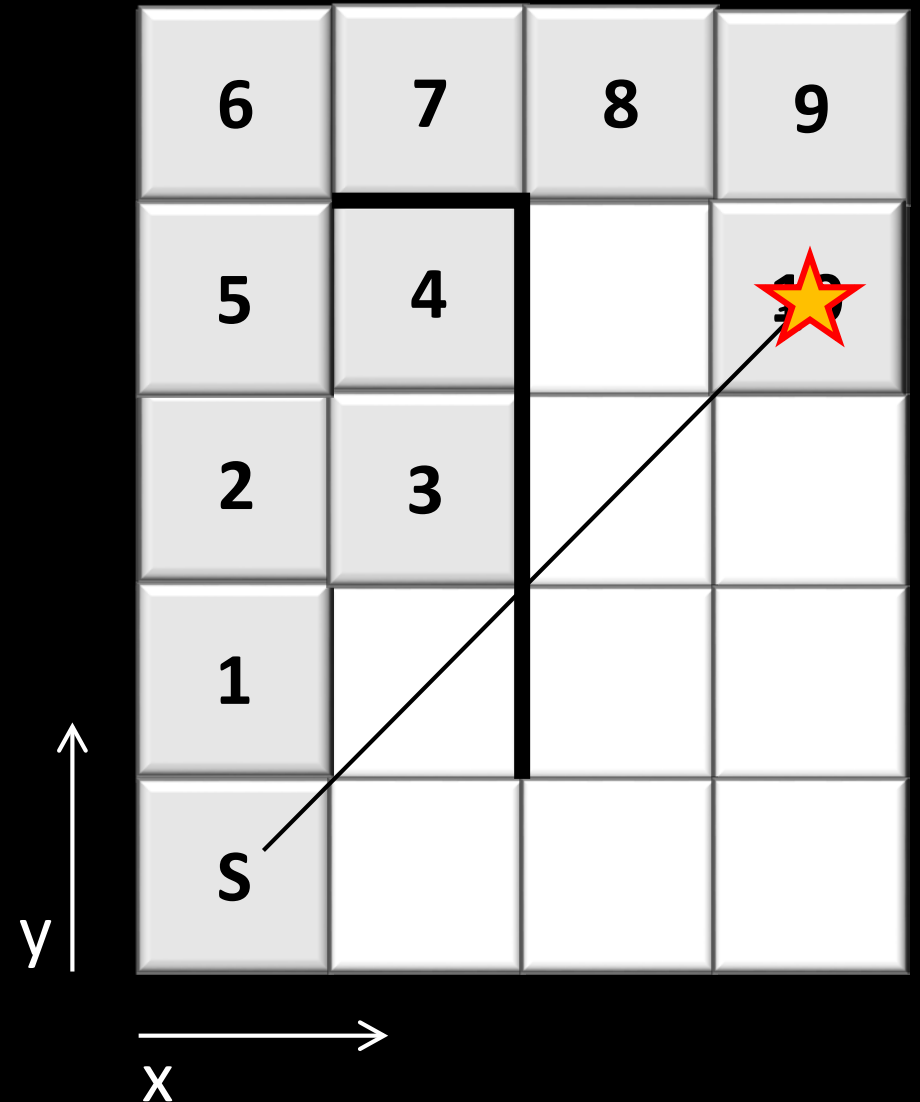


Informed Search

- Greedy Search
 - Complete?
 - No
 - Time complexity?
 - $O(b^m)$
 - Space complexity?
 - $O(b^m)$
 - Optimal?
 - no...

Search order: N, E, S, W

Find a treasure



Search Algorithms, General

- Breadth First Search
 - *Complete* and optimal
 - ...but searches *everything*
- Lowest-Cost First Algorithm *Considers parent cost*
 - *Complete and optimal*
 - ...but it wastes time exploring in directions that aren't promising
- Greedy Search *Considers goal*
 - *Complete*
 - ...only explores promising directions

Can we do better?

*A**

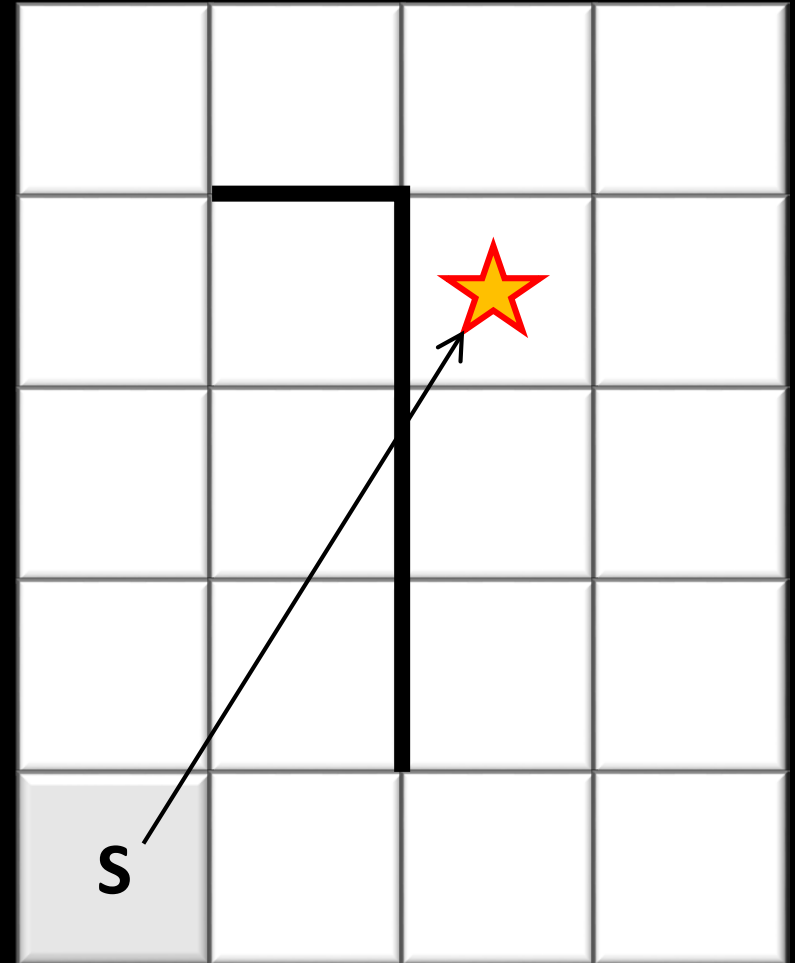
Informed Search

- 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')
```

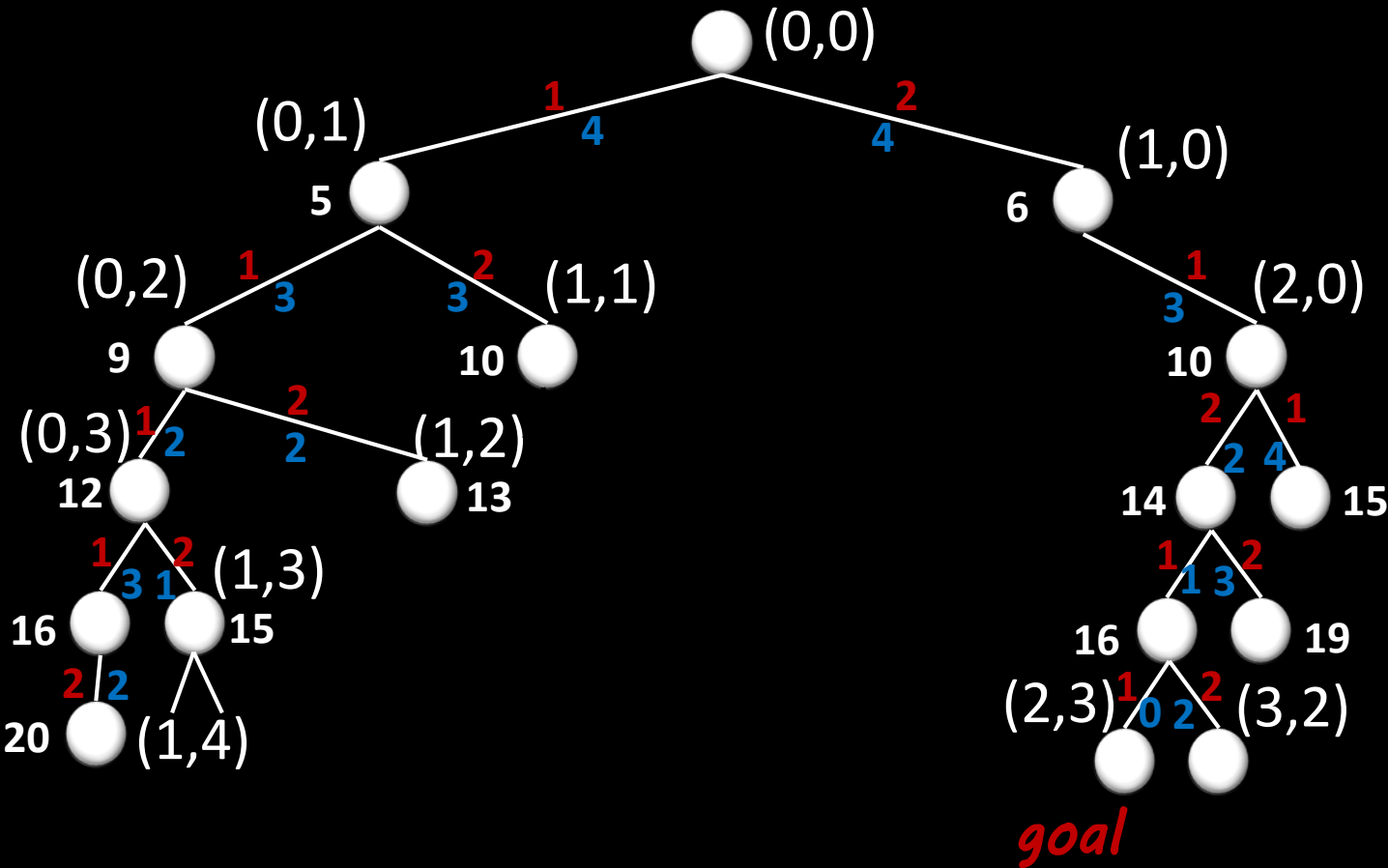
Search order: N, E, S, W

Find a treasure



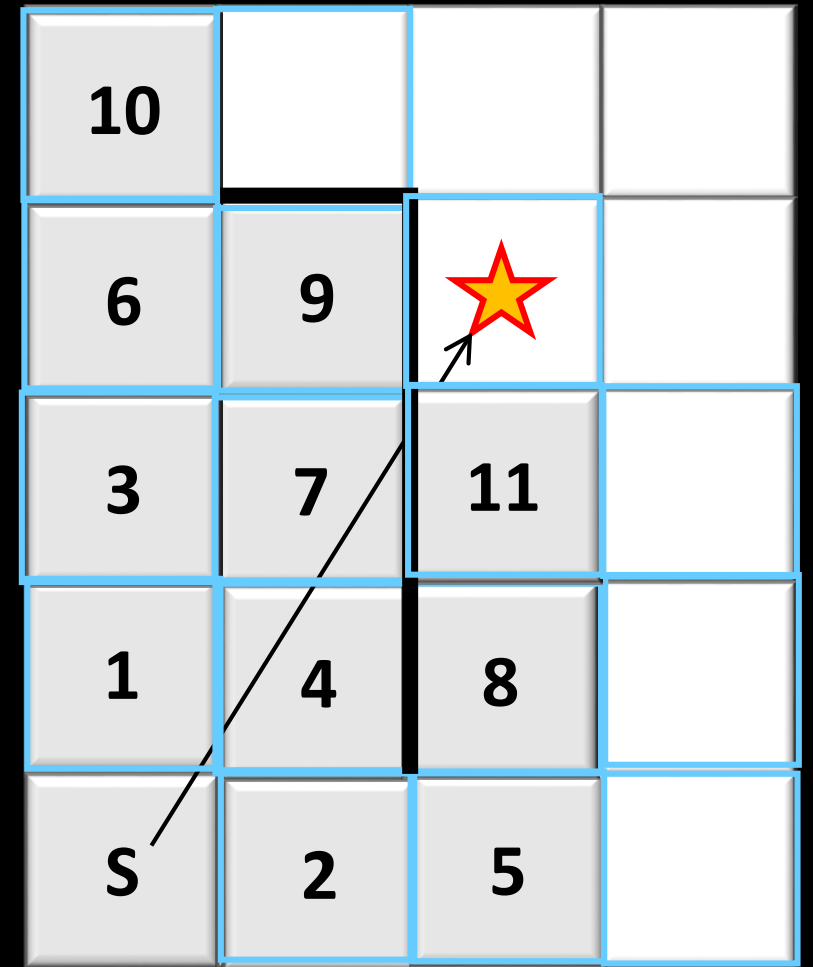
Informed Search

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



Search order: N, E, S, W

Find a goal



y ↑

→ x

A* Search

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

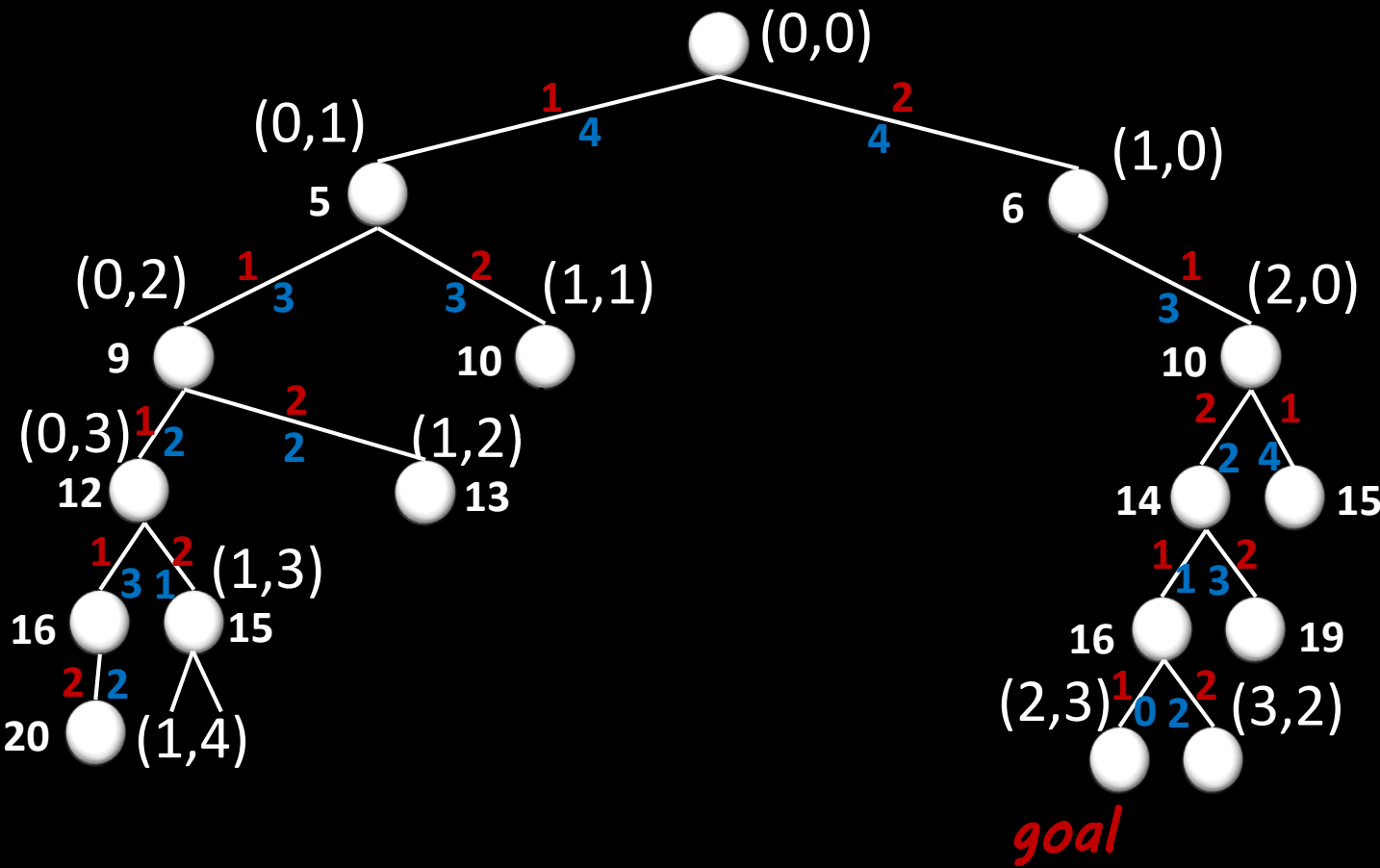
admissible heuristic

inadmissible heuristic



Informed Search

- A* (“A-star”)
 - Cost and goal heuristic



- Complete?
 - Yes!
- Time Complexity
 - $O(b^m)$
- Space Complexity
 - $O(b^m)$
- Optimal?
 - Yes, if the heuristic is admissible!

Summary

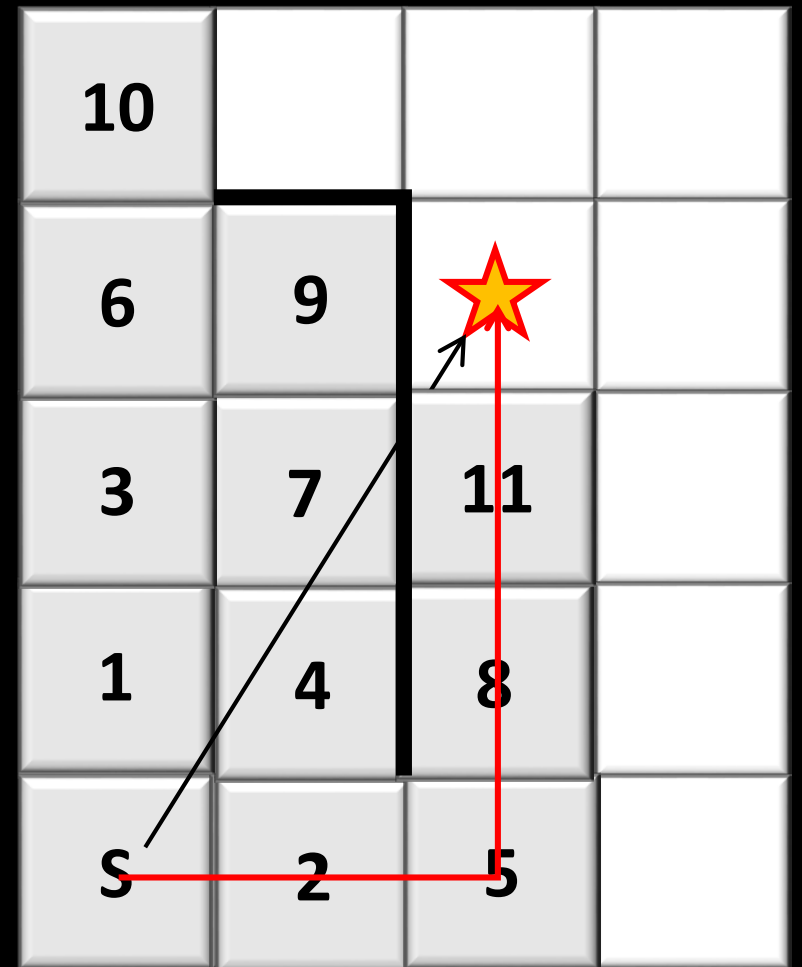
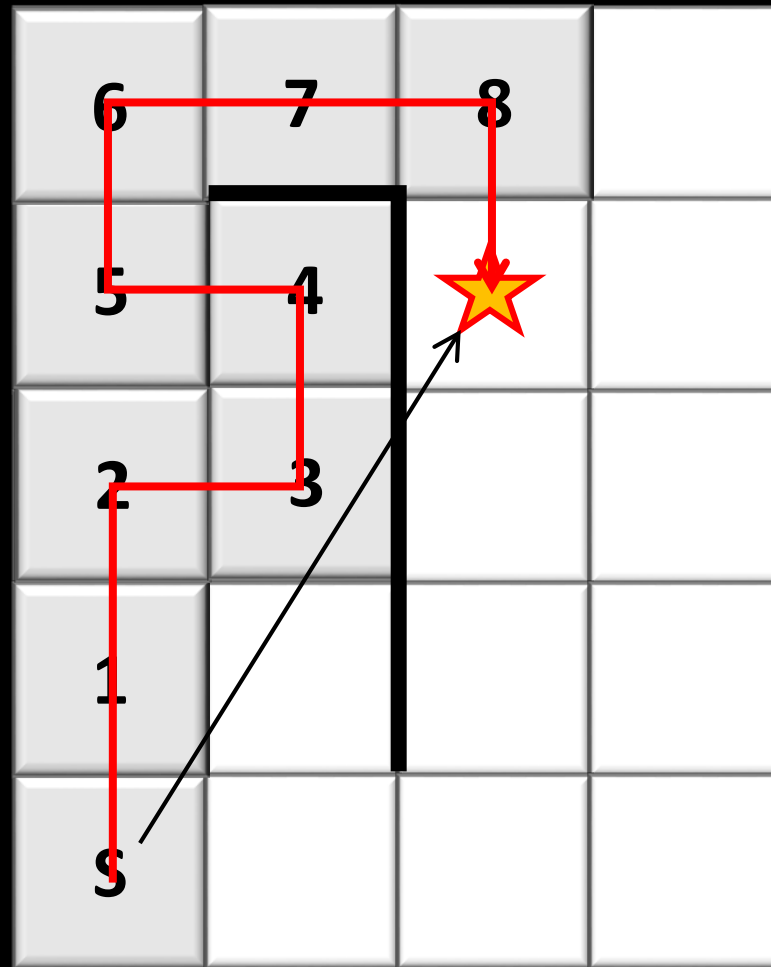
LCFS

minimum path

Greedy

A*

*minimum path
and efficient*



Navigation icons: Car, Bus, Walking, Bicycling, Airplane

Upson Hall, 124 Hoy Rd, Ithaca, NY 14850

Cornell Dairy Bar, 411 Tower Rd, Ithaca, NY 14850

Add destination

OPTIONS

- Send directions to your phone
- via Hoy Rd
 - 18 min
 - 0.8 mile
 - DETAILS
- 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

