

SEARCH

David Kauchak  
CS51A – Fall 2019

Admin

Assignment 8

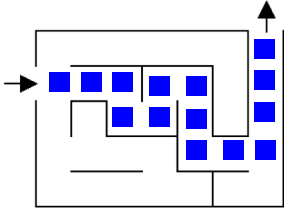
What is AI?

|   |  |
|---|--|
| <b>Think like a human</b><br>Cognitive Modeling | <b>Think rationally</b><br>Logic-based Systems |
| <b>Act like a human</b><br>Turing Test          | <b>Act rationally</b><br>Rational Agents       |

Next couple of weeks

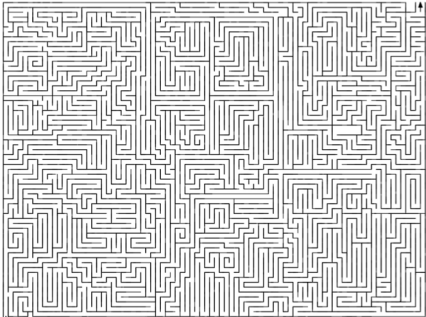
Solve the maze!

### Solve the maze!



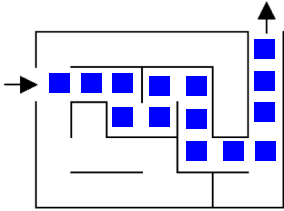
A maze with a path highlighted in blue squares. The path starts from an arrow pointing right on the left side, moves right, then down, right, down, right, down, right, up, right, up, right, up, right, and finally up to an arrow pointing up at the top right corner.

### Solve the maze!



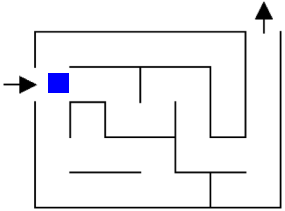
A complex, dense maze consisting of many paths and dead ends, arranged in a square grid pattern.

### Solve the maze!



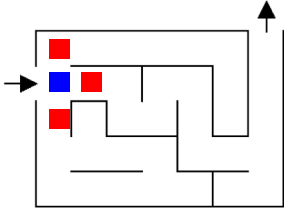
How did you figure it out?

### One approach



What now?

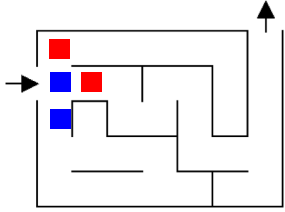
One approach



Three choices

The diagram shows a maze with a blue square at the start (left) and three red squares representing choices. An arrow points right from the blue square. The maze has a vertical wall on the right and a horizontal wall at the bottom. The path starts from the blue square and branches into three paths leading to the red squares.

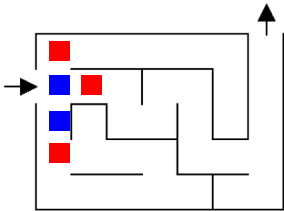
One approach



Pick one!  
What now?

The diagram shows the same maze as the previous slide. The blue square is at the start, and one red square is highlighted, indicating a choice. An arrow points right from the blue square. The maze has a vertical wall on the right and a horizontal wall at the bottom. The path starts from the blue square and branches into three paths leading to the red squares.

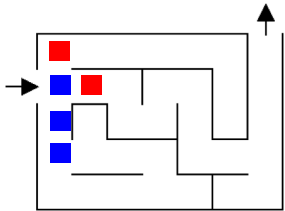
One approach



Still three options!  
Which would you explore/pick?

The diagram shows the same maze as the previous slides. The blue square is at the start, and three red squares are highlighted, indicating three options. An arrow points right from the blue square. The maze has a vertical wall on the right and a horizontal wall at the bottom. The path starts from the blue square and branches into three paths leading to the red squares.

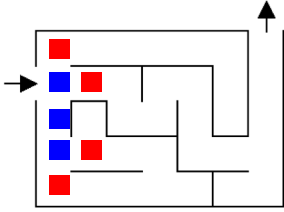
One approach



Most people go down a single path until they realize that it's wrong

The diagram shows the same maze as the previous slides. The blue square is at the start, and one red square is highlighted, indicating a choice. An arrow points right from the blue square. The maze has a vertical wall on the right and a horizontal wall at the bottom. The path starts from the blue square and branches into three paths leading to the red squares.

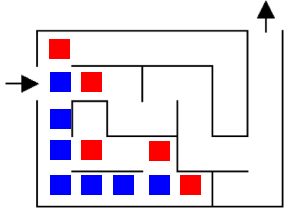
One approach



Keep exploring

The diagram shows a maze with a starting point on the left indicated by a black arrow. The maze contains several blue squares and red squares. The text "Keep exploring" is written in blue below the maze.

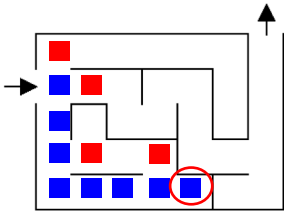
One approach



Keep exploring

The diagram shows the same maze as the previous slide, but with more blue squares filled in, indicating further exploration. The text "Keep exploring" is written in blue below the maze.

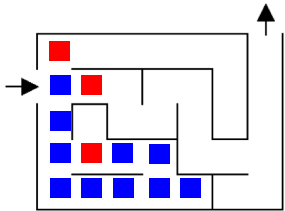
One approach



What now?

The diagram shows the maze with the same exploration progress as the second slide. A red circle highlights a blue square at the bottom right of the maze. The text "What now?" is written in red below the maze.

One approach



Are we stuck?

No. Red positions are just possible options we haven't explored

The diagram shows the maze with the same exploration progress as the second slide. The text "Are we stuck?" is written in red, followed by "No. Red positions are just possible options we haven't explored" in blue below the maze.

One approach

How do we know not to go left?

One approach

Have to be careful and keep track of where we've been if we can loop

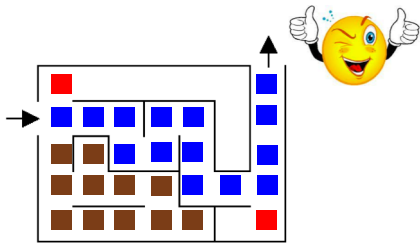
One approach

Now what?

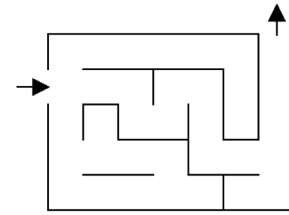
One approach

Now what?

## One approach



## Search problems



What information do we need to figure out a solution?

## Search problems

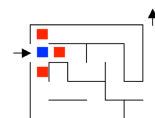
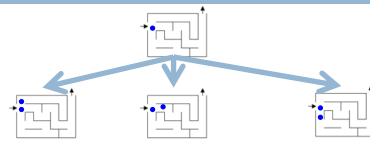
Where to start

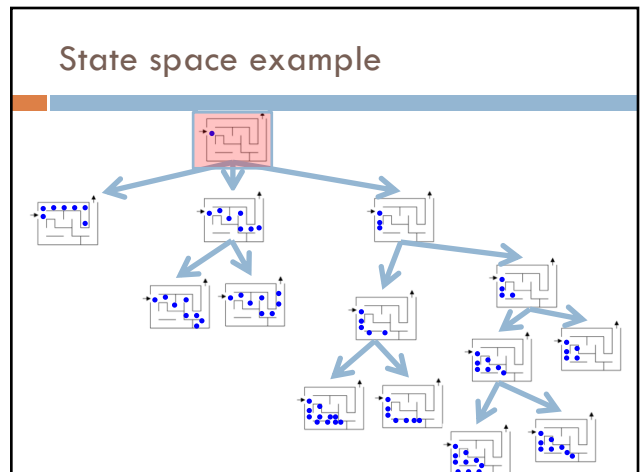
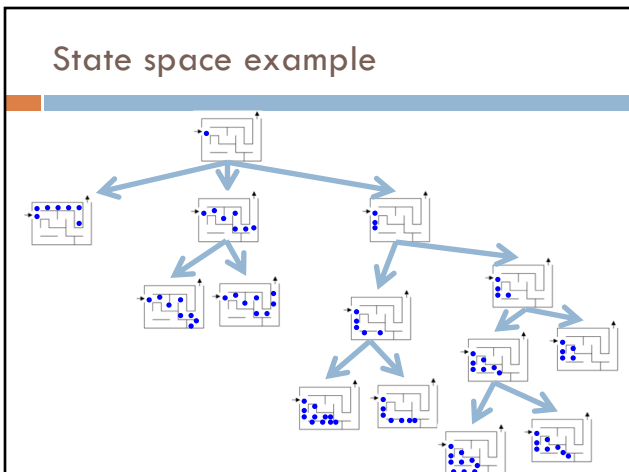
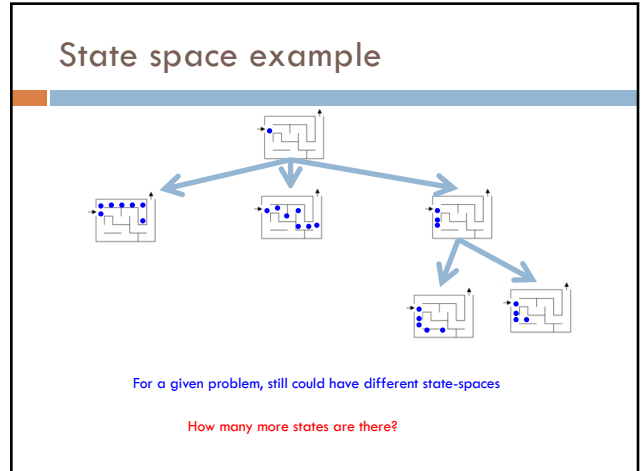
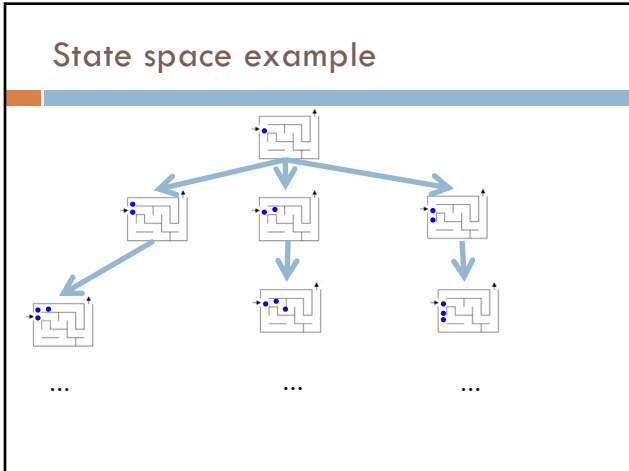
Where to finish (goal)

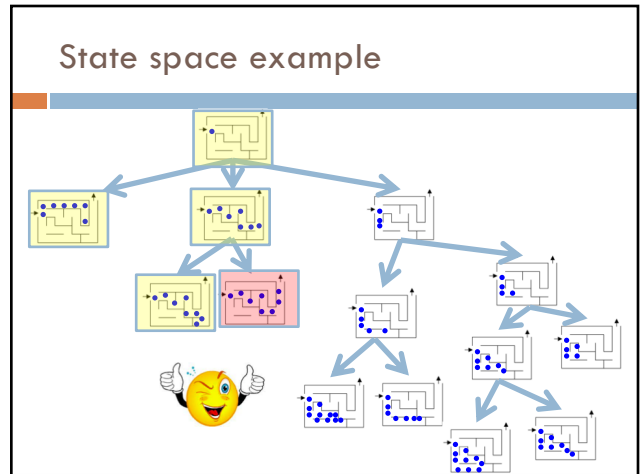
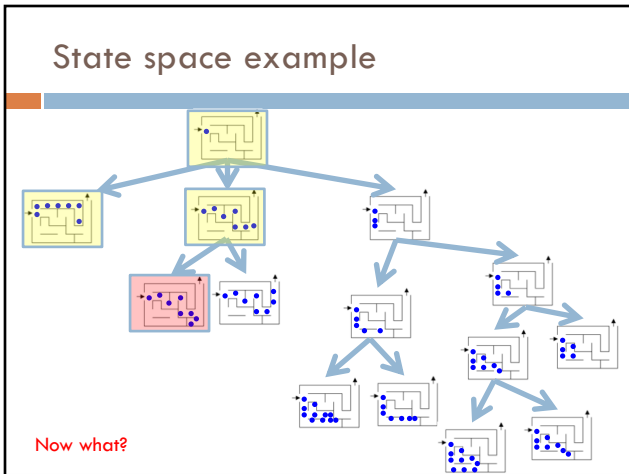
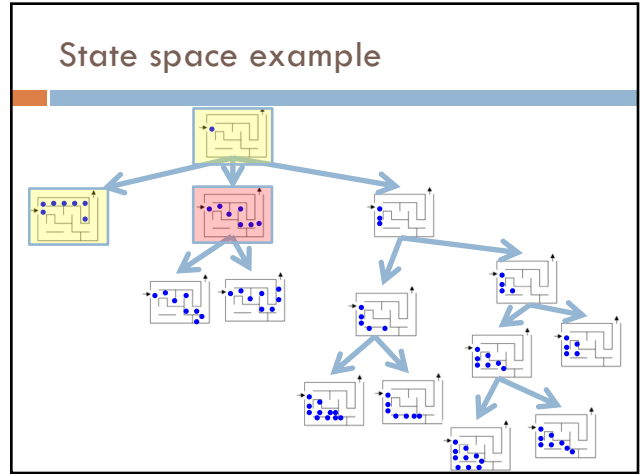
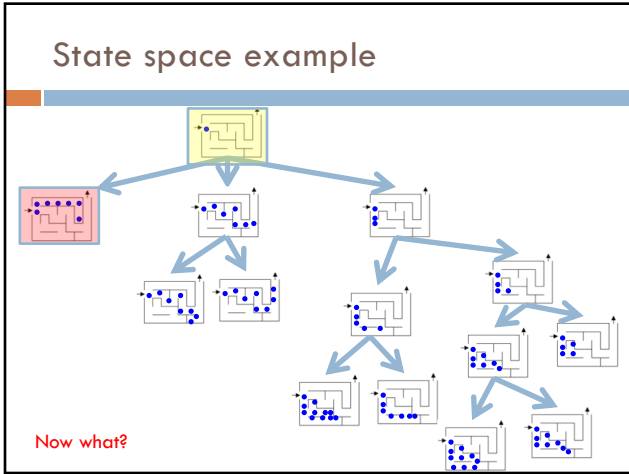
What the "world" (in this case a maze) looks like

- ▣ We'll define the world as a collection of discrete states
- ▣ States are connected if we can get from one state to another by taking a particular action
- ▣ This is called the "state space"

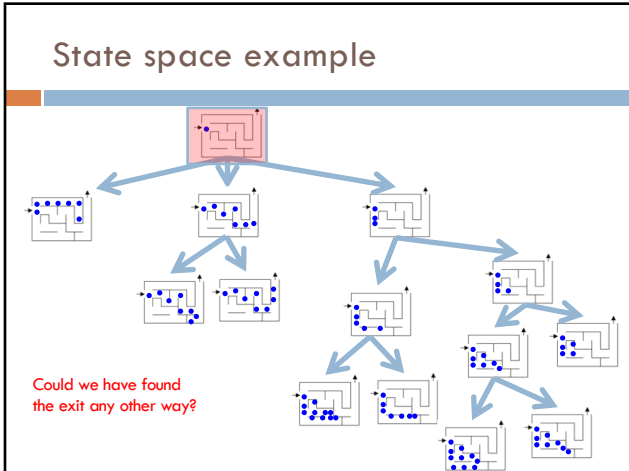
## State space example









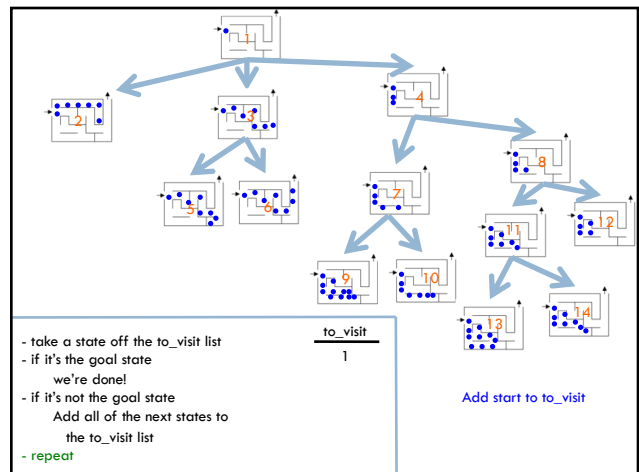
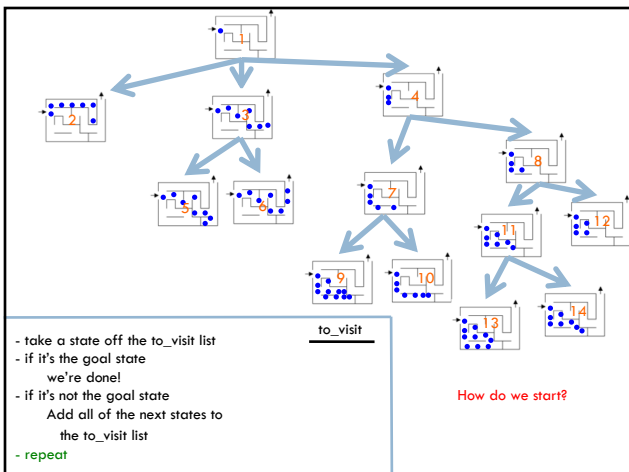


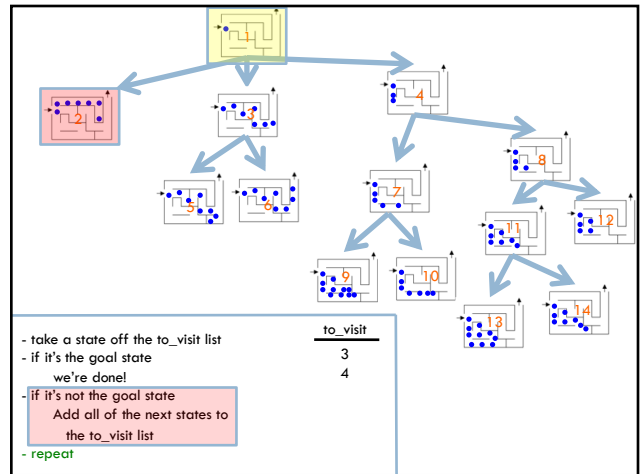
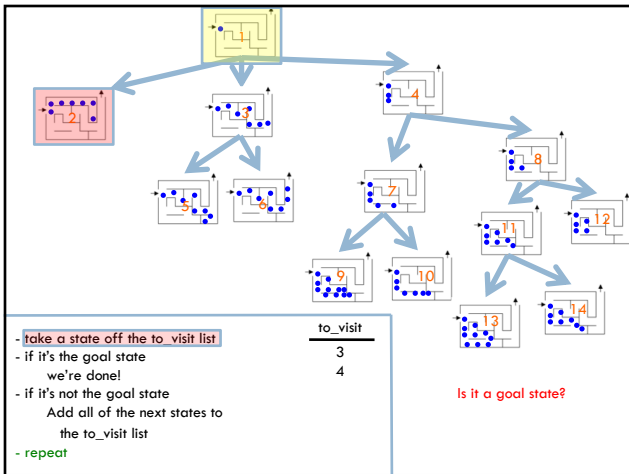
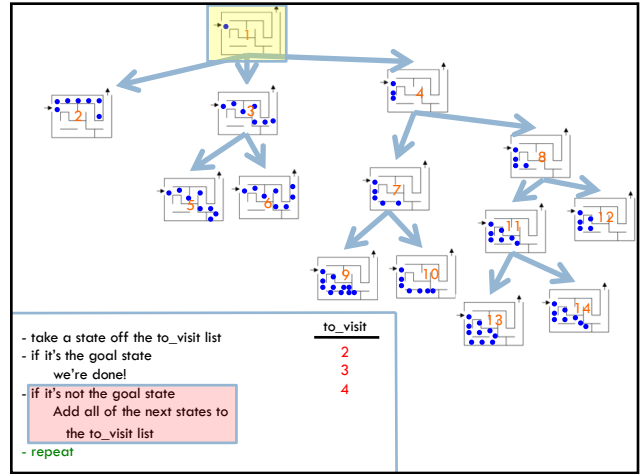
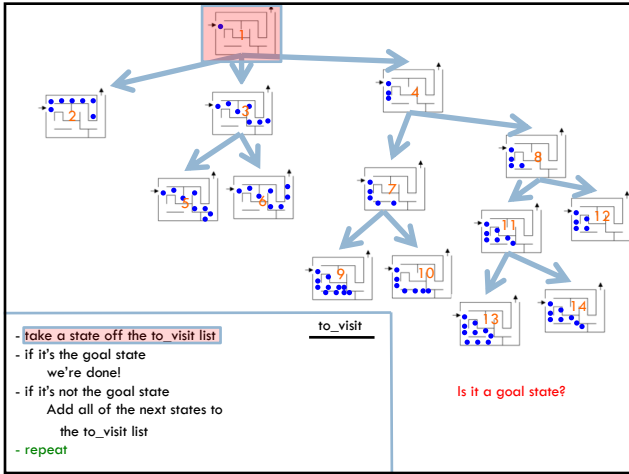
### Search algorithm

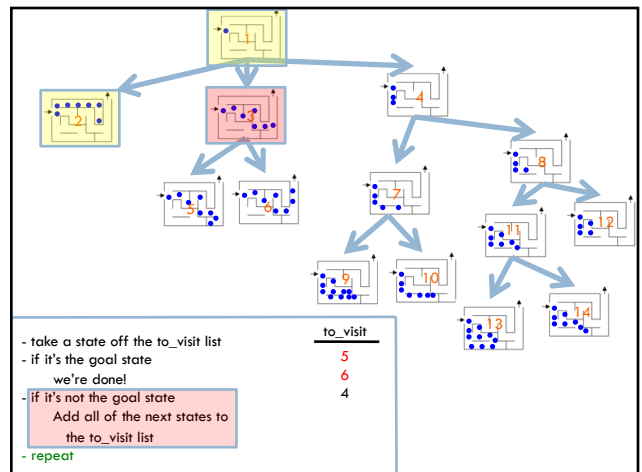
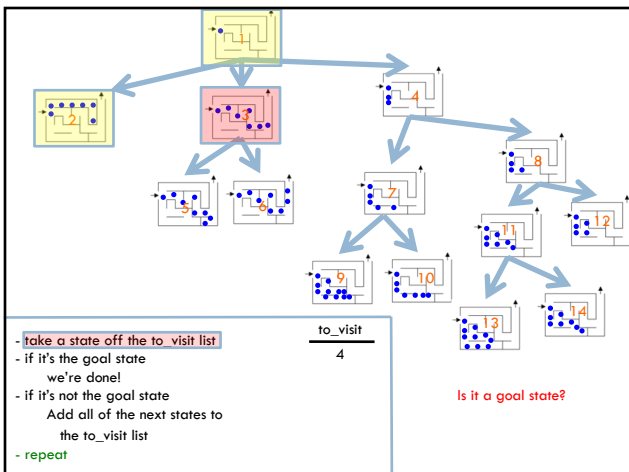
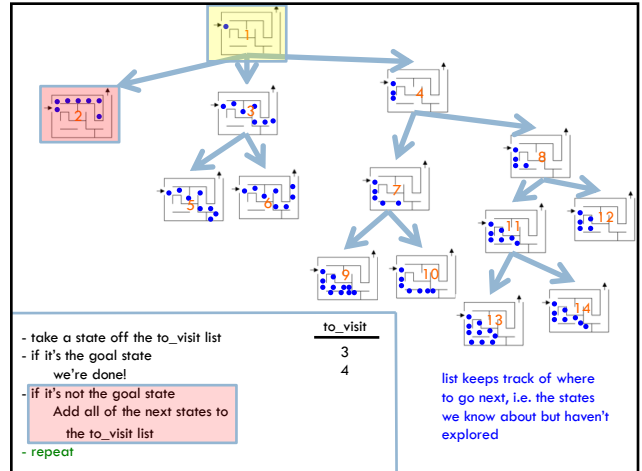
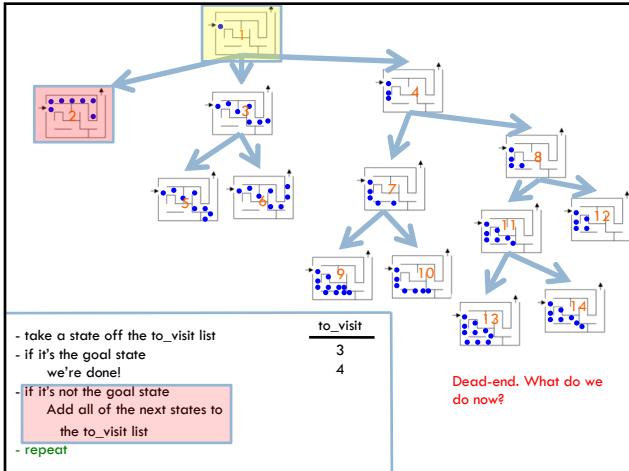
Keep track of a list of states that we *could* visit, we'll call it "to\_visit"

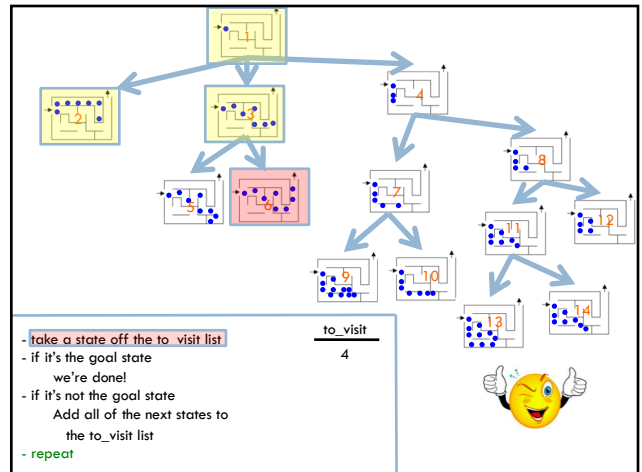
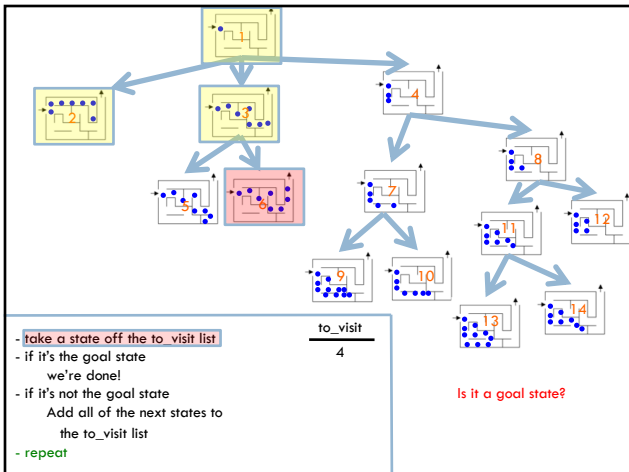
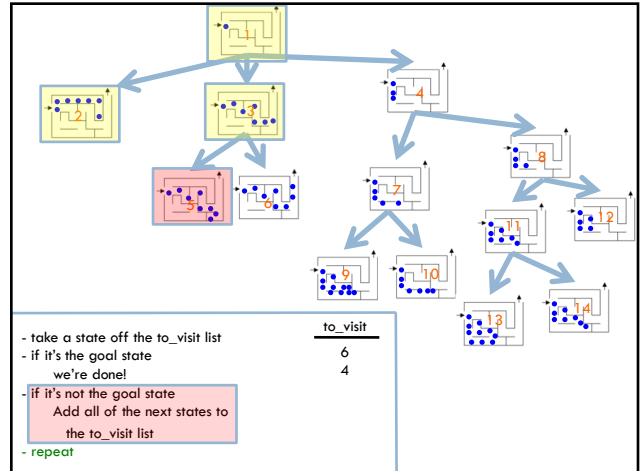
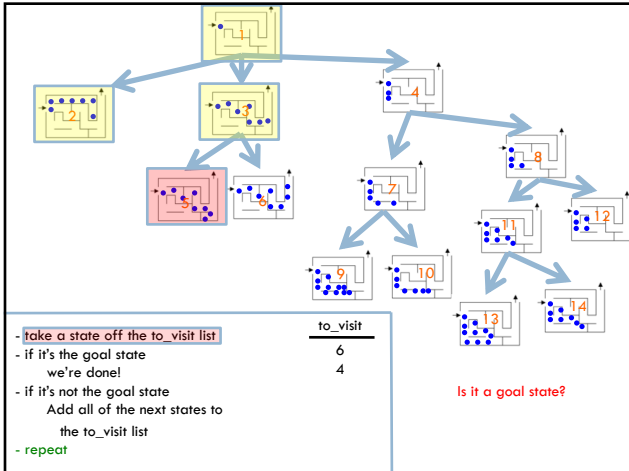
General idea:

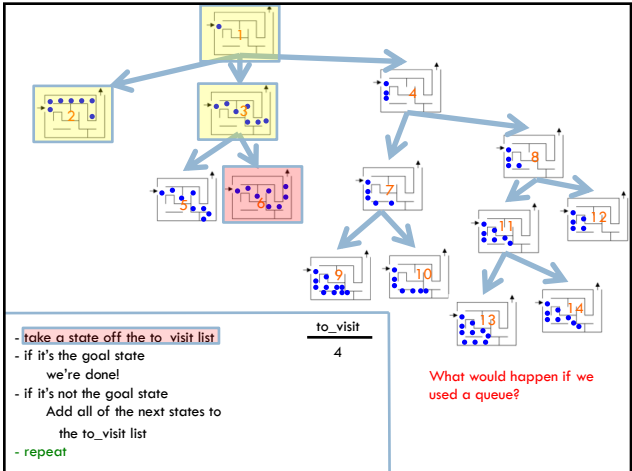
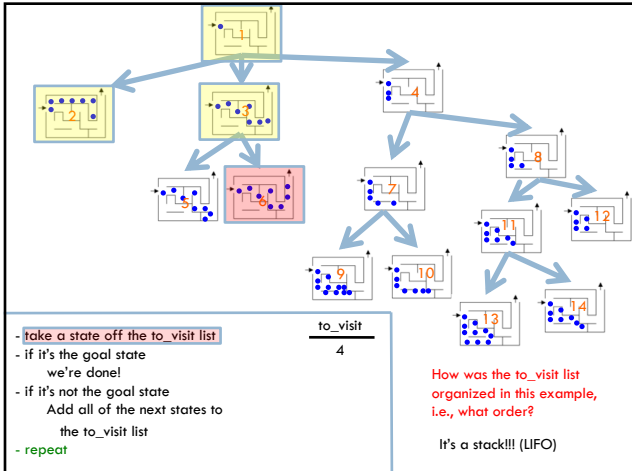
- take a state off the to\_visit list
- if it's the goal state
  - we're done!
- if it's not the goal state
  - Add all of the next states to the to\_visit list
- repeat











### Search algorithms

add the start state to to\_visit

Repeat

- take a state off the to\_visit list
- if it's the goal state
  - we're done!
- if it's not the goal state
  - Add all of the next states to the to\_visit list

### Search algorithms

add the start state to to\_visit

Repeat

- take a state off the to\_visit list
- if it's the goal state
  - we're done!
- if it's not the goal state
  - Add all of the next states to the to\_visit list

---

Depth first search (DFS): to\_visit is a stack  
 Breadth first search (BFS): to\_visit is a queue

**What order will BFS and DFS visit the states assuming states are added to to\_visit left to right?**

add the start state to to\_visit

Repeat

- take a state off the to\_visit list
- if it's the goal state
  - we're done!
- if it's not the goal state
  - Add all of the successive states to the to\_visit list

Depth first search (DFS): to\_visit is a stack  
Breadth first search (BFS): to\_visit is a queue

**What order will BFS and DFS visit the states?**

DFS: 1, 4, 3, 8, 7, 6, 9, 2, 5

Why not 1, 2, 5?

Depth first search (DFS): to\_visit is a stack  
Breadth first search (BFS): to\_visit is a queue

**What order will BFS and DFS visit the states?**

DFS: 1, 4, 3, 8, 7, 6, 9, 2, 5

Depth first search (DFS): to\_visit is a stack  
Breadth first search (BFS): to\_visit is a queue

**What order will BFS and DFS visit the states?**

DFS: 1, 4, 3, 8, 7, 6, 9, 2, 5

Depth first search (DFS): to\_visit is a stack  
Breadth first search (BFS): to\_visit is a queue

### What order will BFS and DFS visit the states?

DFS: 1, 4, 3, 8, 7, 6, 9, 2, 5

```

graph TD
    1[1] --> 2[2]
    1 --> 3[3]
    1 --> 4[4]
    2 --> 5[5]
    3 --> 6[6]
    3 --> 7[7]
    3 --> 8[8]
    6 --> 9[9]
    style 5 fill:#90EE90
  
```

3  
2  
**STACK**

Depth first search (DFS): to\_visit is a stack  
Breadth first search (BFS): to\_visit is a queue

### What order will BFS and DFS visit the states?

DFS: 1, 4, 3, 8, 7, 6, 9, 2, 5

BFS: 1, 2, 3, 4, 5

```

graph TD
    1[1] --> 2[2]
    1 --> 3[3]
    1 --> 4[4]
    2 --> 5[5]
    3 --> 6[6]
    3 --> 7[7]
    3 --> 8[8]
    6 --> 9[9]
    style 5 fill:#90EE90
  
```

Depth first search (DFS): to\_visit is a stack  
Breadth first search (BFS): to\_visit is a queue

### Search variants implemented

add the start state to to\_visit

```

def dfs(start_state):
    s = Stack()
    return search(start_state, s)

def bfs(start_state):
    q = Queue()
    return search(start_state, q)

def search(start_state, to_visit):
    to_visit.add(start_state)

    while not to_visit.is_empty():
        current = to_visit.remove()

        if current.is_goal():
            return current
        else:
            for s in current.next_states():
                to_visit.add(s)

    return None
  
```

Repeat

- take a state off the to\_visit list
- if it's the goal state
  - we're done!
- if it's not the goal state
  - Add all of the successive states to the to\_visit list

### What order would this variant visit the states?

```

def search(state):
    if state.is_goal():
        return state
    else:
        for s in state.next_states():
            result = search(s)
            if result != None:
                return result

        return None
  
```

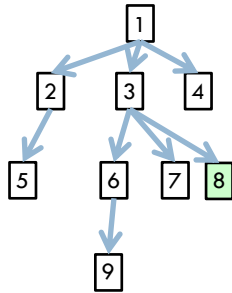
1, 2, 5

```

graph TD
    1[1] --> 2[2]
    1 --> 3[3]
    1 --> 4[4]
    2 --> 5[5]
    3 --> 6[6]
    3 --> 7[7]
    3 --> 8[8]
    6 --> 9[9]
    style 5 fill:#90EE90
  
```

What order would this variant visit the states?

```
def search(state):
    if state.is_goal():
        return state
    else:
        for s in state.next_states():
            result = search(s)
            if result != None:
                return result
        return None
```

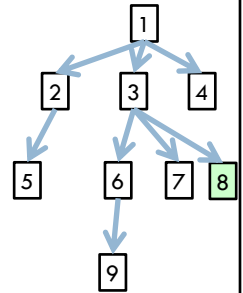


1, 2, 5, 3, 6, 9, 7, 8

What search algorithm is this?

What order would this variant visit the states?

```
def search(state):
    if state.is_goal():
        return state
    else:
        for s in state.next_states():
            result = search(s)
            if result != None:
                return result
        return None
```



1, 2, 5, 3, 6, 9, 7, 8

DFS!