# SEARCH

David Kauchak
CS51A – Spring 2025

1

## Admin

Assignment 8

2

## What is AI?

| | |
|---|---|
| **Think like a human** Cognitive Modeling | **Think rationally** Logic-based Systems |
| **Act like a human** Turing Test | **Act rationally** Rational Agents |

3

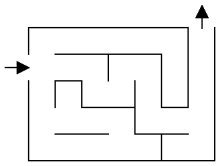
## What is AI?

| | |
|---|---|
| **Think like a human** Cognitive Modeling | **Think rationally** Logic-based Systems |
| **Act like a human** Turing Test | **Act rationally** Rational Agents |

Next couple of weeks

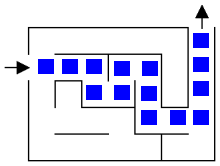
4

## Solve the maze!



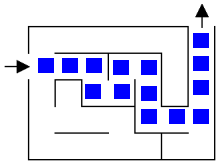
5

## Solve the maze!



6

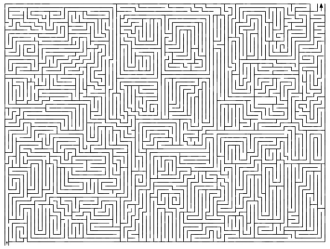
## Solve the maze!



How did you figure it out?

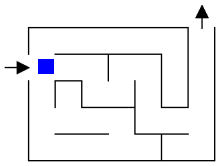
7

## Solve the maze!



8

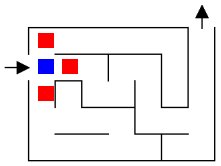
## One approach



What now?

9

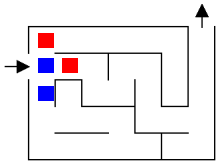
## One approach



Three choices

10

## One approach

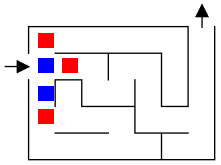


Pick one!

What now?

11

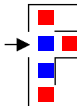
## One approach



Still three options!

12

## One approach

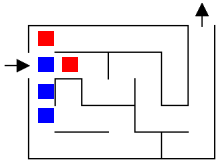


Still three options!

Which would you explore/pick?

13

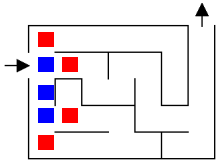
## One approach



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

14

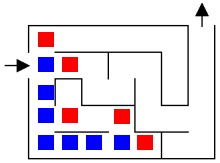
## One approach



Keep exploring

15

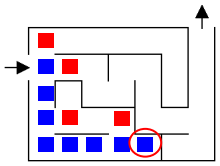
## One approach



Keep exploring

16

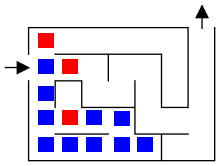
## One approach



What now?

17

## One approach

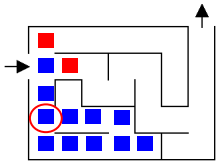


Are we stuck?

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

18

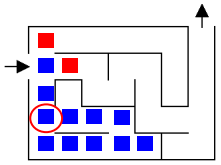
## One approach



How do we know
not to go straight?

19

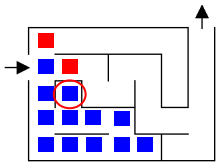
## One approach



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

20

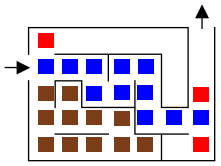
## One approach



Now what?

21

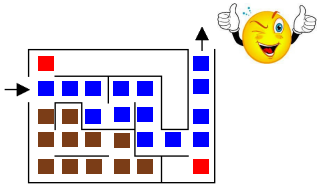
## One approach



Now what?

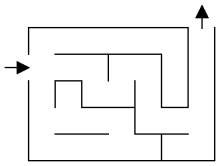
22

## One approach



23

## Search problems



What information do we need to figure out a solution?

24

## Search problems

Where to start

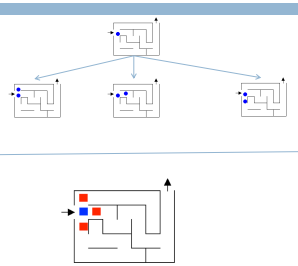
Where to finish (goal)

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"

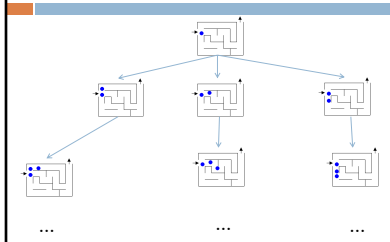
25

## State space example



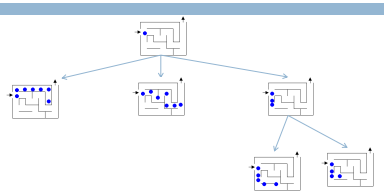
26

## State space example




27

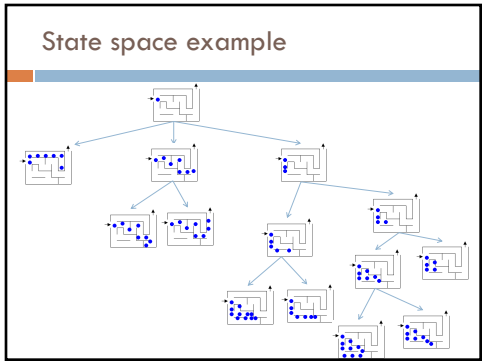
## State space example



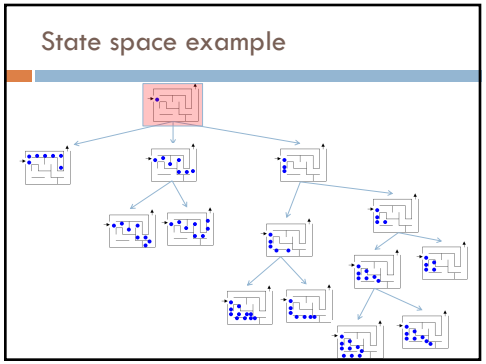
For a given problem, still could have different state-spaces

How many more states are there?

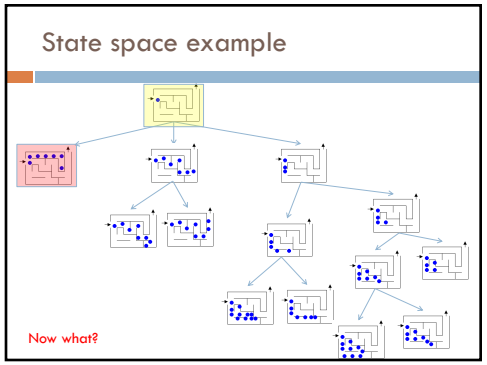
28


State space example

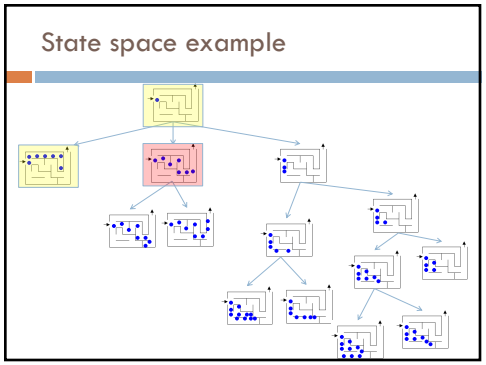
29


State space example

30

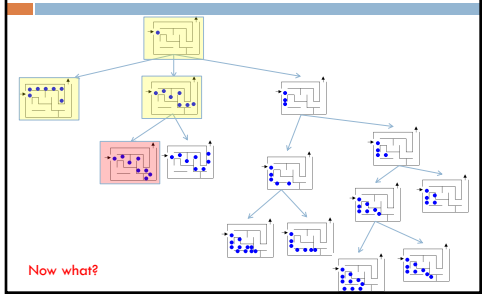

State space example
Now what?

31


State space example

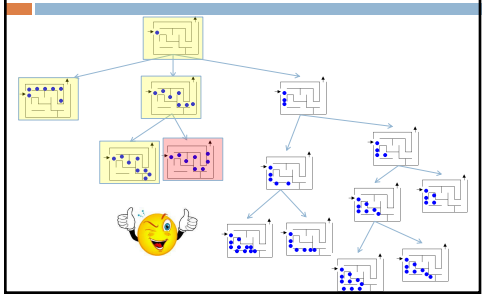
32

## State space example



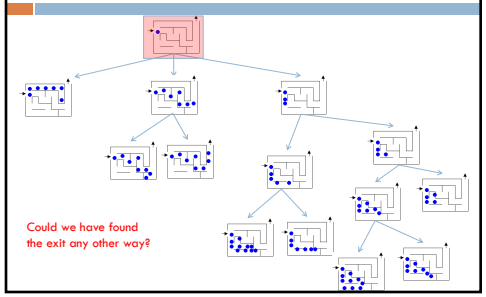

33

## State space example



34

## State space example




35

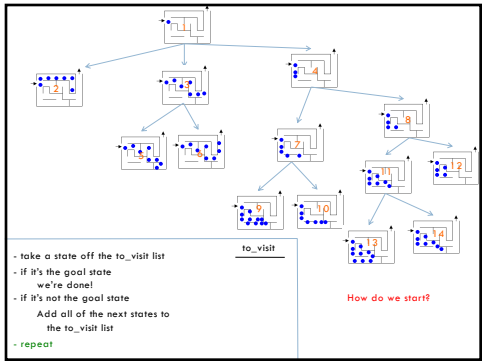
## Search algorithm

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

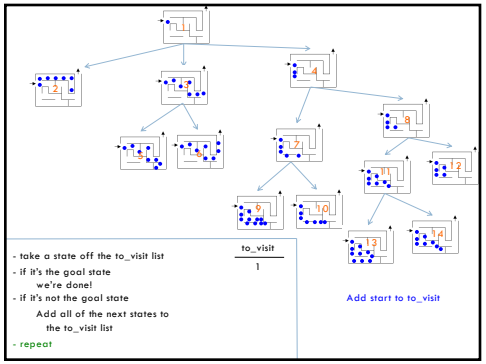
General idea:

- 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

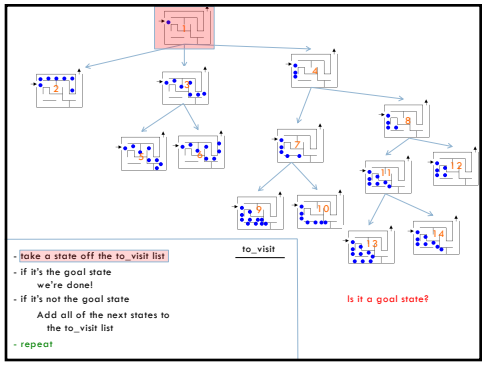
36


- 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
to_visit
How do we start?

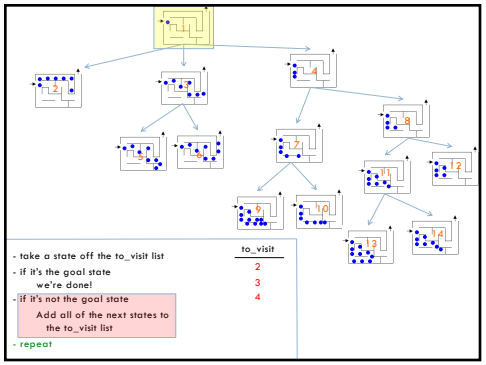
37


- 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
to_visit
1
Add start to to_visit

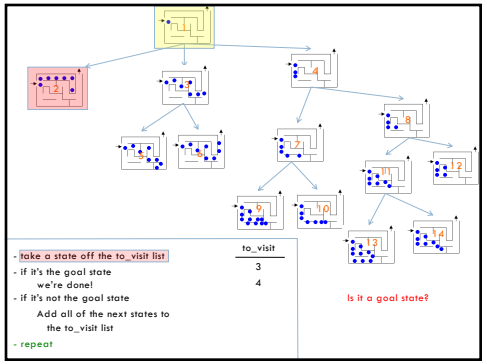
38


- 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
to_visit
Is it a goal state?

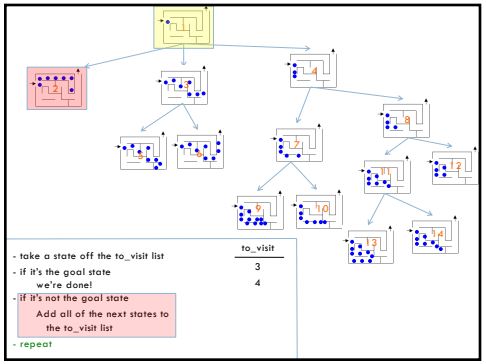
39


- 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
to_visit
2
3
4

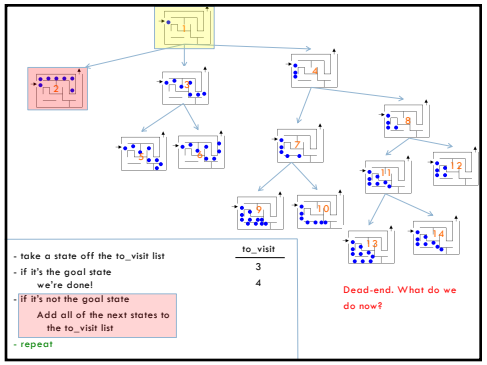
40


- 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
to_visit
3
4
Is it a goal state?

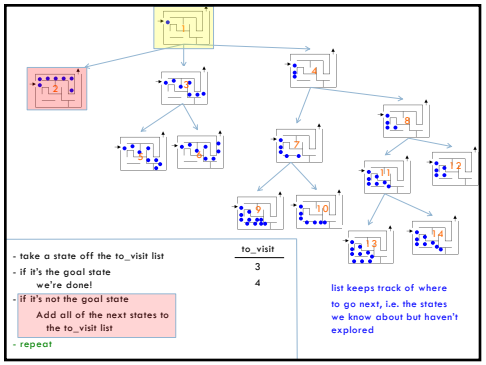
41


- 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
to_visit
3
4

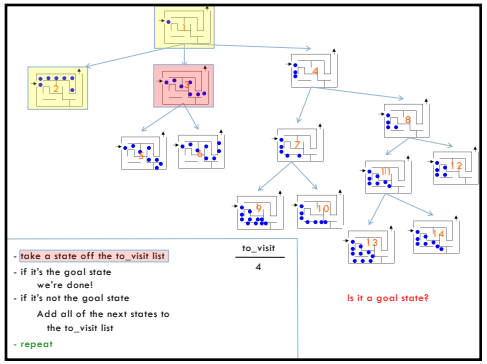
42


- 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
to_visit
3
4
Dead-end. What do we
do now?

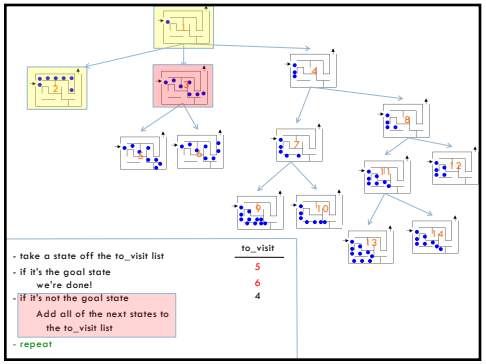
43


- 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
to_visit
3
4
list keeps track of where
to go next, i.e. the states
we know about but haven't
explored

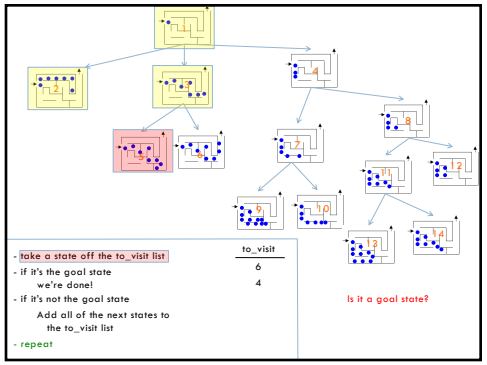
44


- 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
to_visit
4
Is it a goal state?

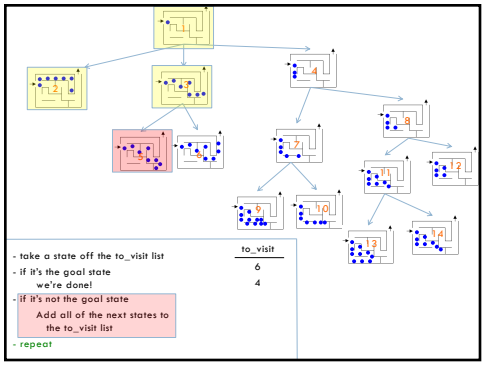
45


- 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
to_visit
5
6
4

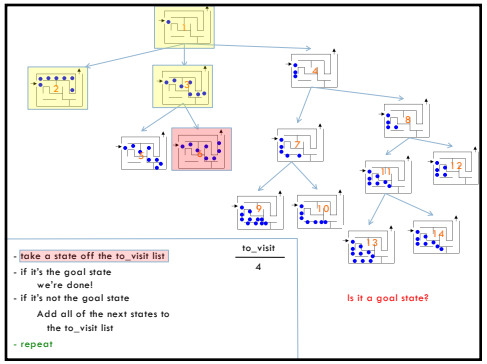
46


- 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
to_visit
6
4
Is it a goal state?

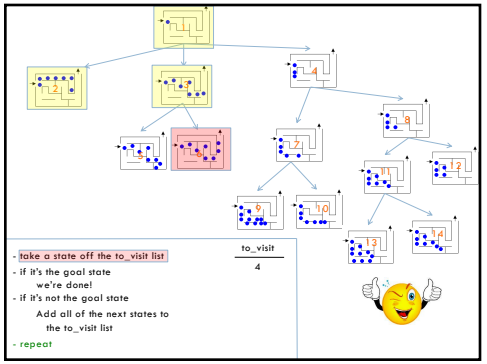
47


- 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
to_visit
6
4

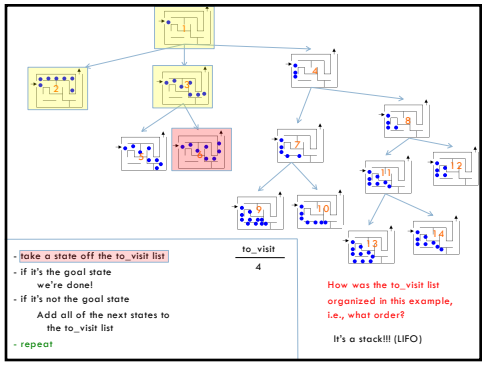
48


- 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
to_visit
4
Is it a goal state?

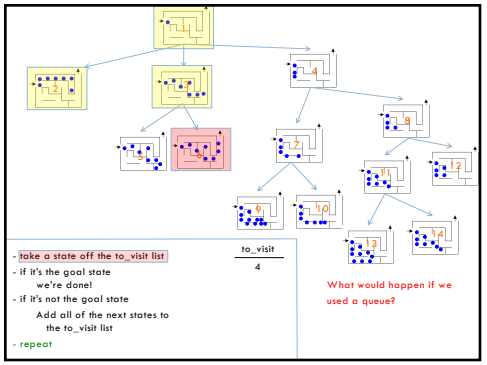
49


- 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
to_visit
4

50


- 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
to_visit
4
How was the to_visit list
organized in this example,
i.e., what order?
It's a stack!!! (LIFO)

51


- 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
to_visit
4
What would happen if we
used a queue?

52

## 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

53

## 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

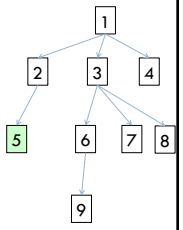
Depth first search (DFS): to_visit is a stack
Breadth first search (BFS): to_visit is a queue

54

## 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



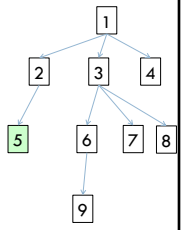

Depth first search (DFS): to_visit is a stack
Breadth first search (BFS): to_visit is a queue

55

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

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

Why not 1, 2, 5?

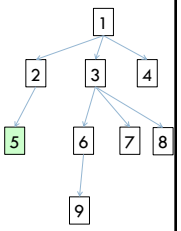



Depth first search (DFS): to_visit is a stack
Breadth first search (BFS): to_visit is a queue

56

What order will BFS and DFS visit the states?

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



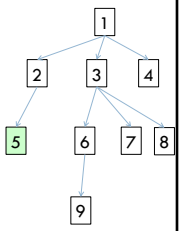


Depth first search (DFS): to_visit is a stack
Breadth first search (BFS): to_visit is a queue

57

What order will BFS and DFS visit the states?

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



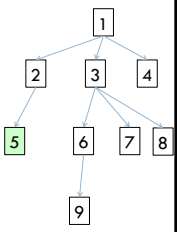


Depth first search (DFS): to_visit is a stack
Breadth first search (BFS): to_visit is a queue

58

What order will BFS and DFS visit the states?

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





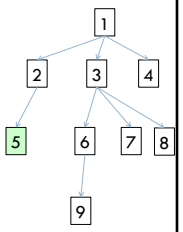
Depth first search (DFS): to_visit is a stack
Breadth first search (BFS): to_visit is a queue

59

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




Depth first search (DFS): to_visit is a stack
Breadth first search (BFS): to_visit is a queue

60

## Implementing state space

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"

61

## Implementing state space

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:

- Is this the goal state? (is_goal)
- What states are connected to this state? (next_states)

62

## Search variants implemented

add the start state to to_visit

Repeat

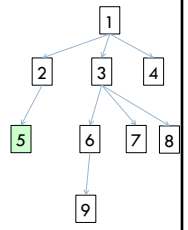
- take a state off the to_visit list
- if it's the goal state
  - we're done!
- if it's not the goal state
  - Add all of the successive states to the to_visit list

```
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
```

63

## 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
```

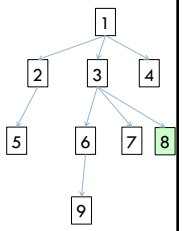



1, 2, 5

64

## 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
```



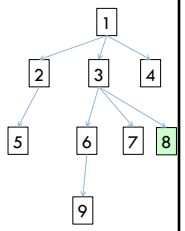

1, 2, 5, 3, 6, 9, 7, 8

What search algorithm is this?

65

## 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!

66

## One last DFS variant

```
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
```

```
def dfs(state):
    if state.is_goal():
        return [state]
    else:
        result = []

        for s in state.next_states():
            result += dfs(s)

        return result
```

How is this different?

67

## One last DFS variant

```
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
```

```
def dfs(state):
    if state.is_goal():
        return [state]
    else:
        result = []

        for s in state.next_states():
            result += dfs(s)

        return result
```

Returns ALL solutions found, not just one

68

## Matrices!


69