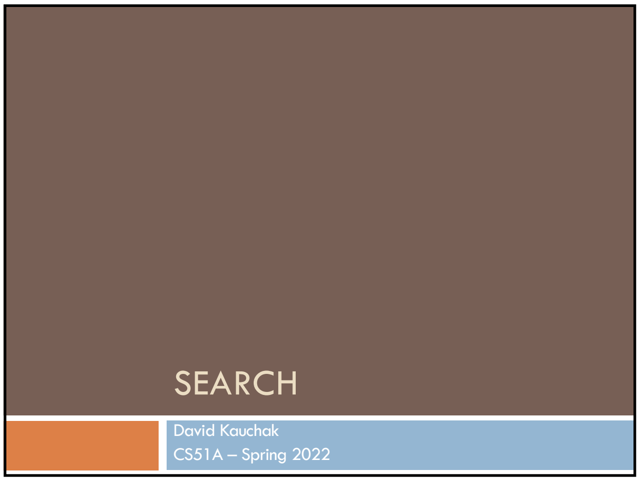
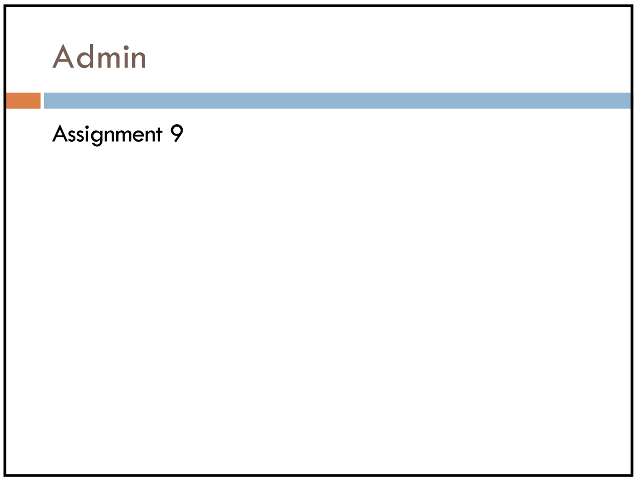


SEARCH

David Kauchak
CS51A – Spring 2022

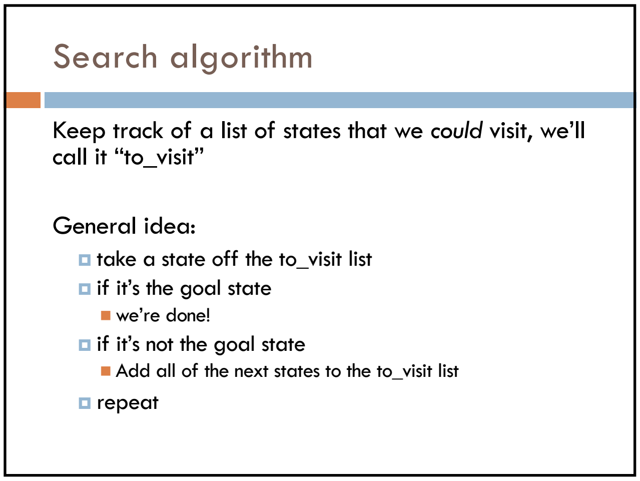
1



Admin

Assignment 9

2



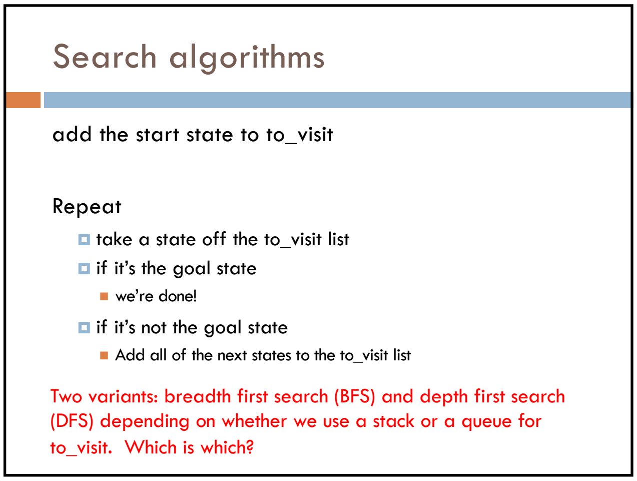
Search algorithm

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

3



Search algorithms

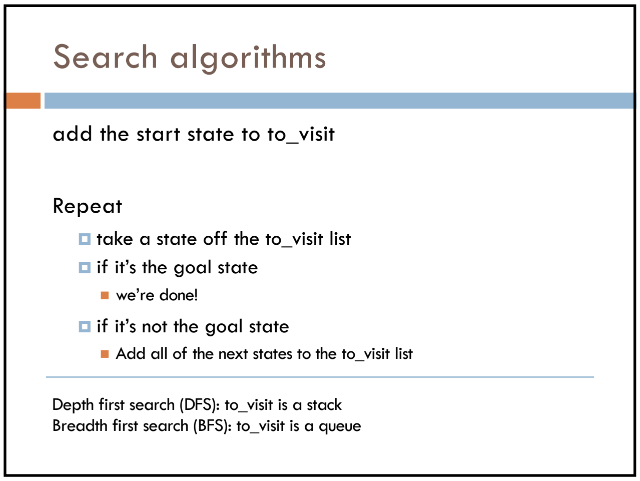
add the start state to to_visit

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

Two variants: breadth first search (BFS) and depth first search (DFS) depending on whether we use a stack or a queue for to_visit. Which is which?

4



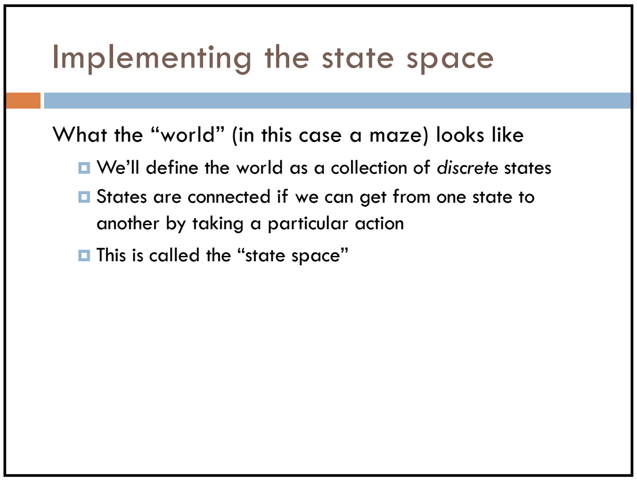
Search algorithms

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

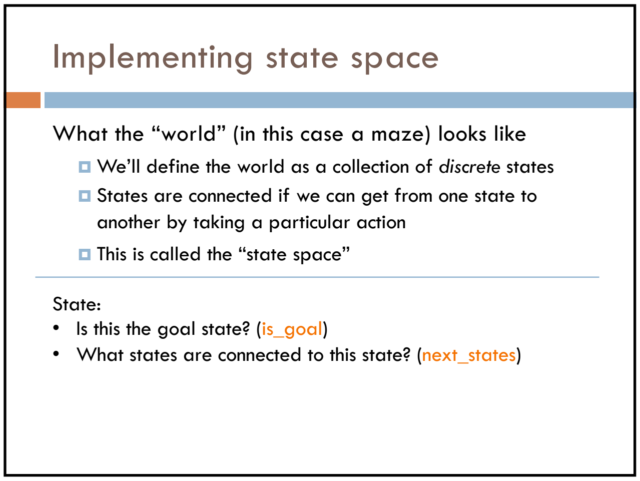
5



Implementing the 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"

6

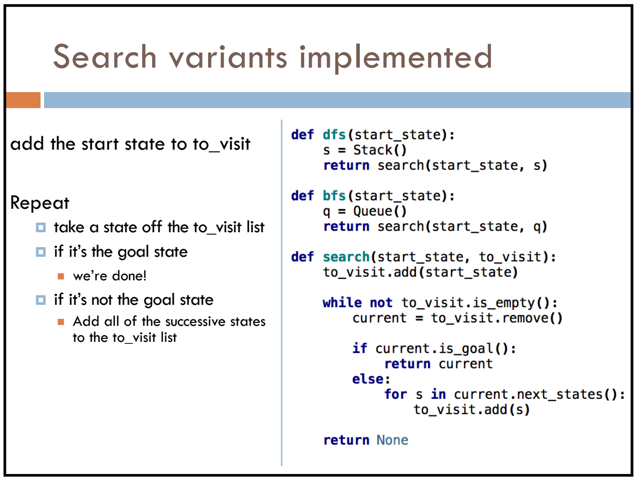


Implementing state space

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)

7



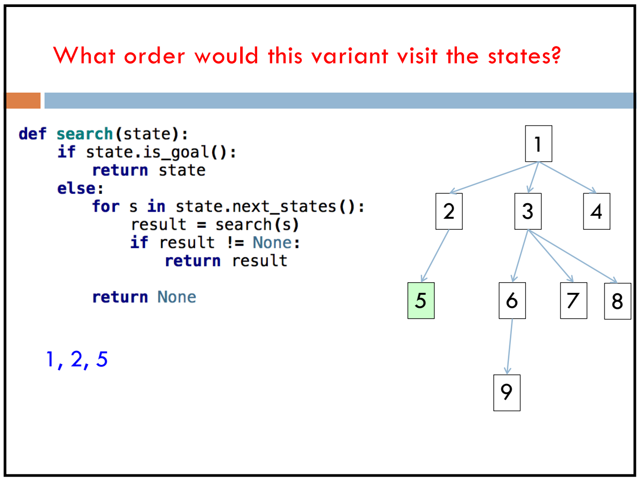
Search variants implemented

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
```

8

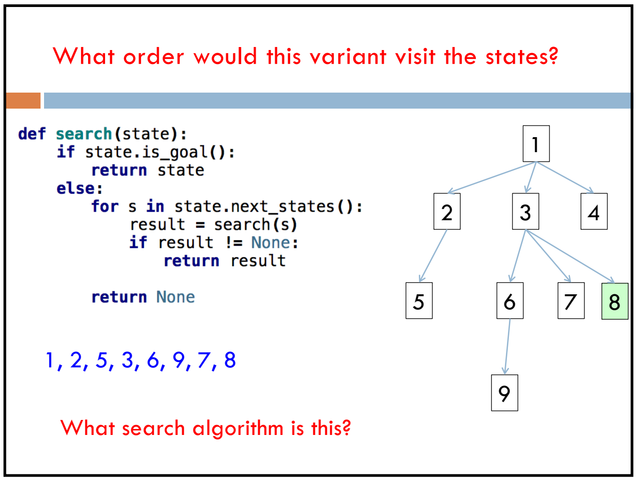


What order would this variant visit the states?

```
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

9



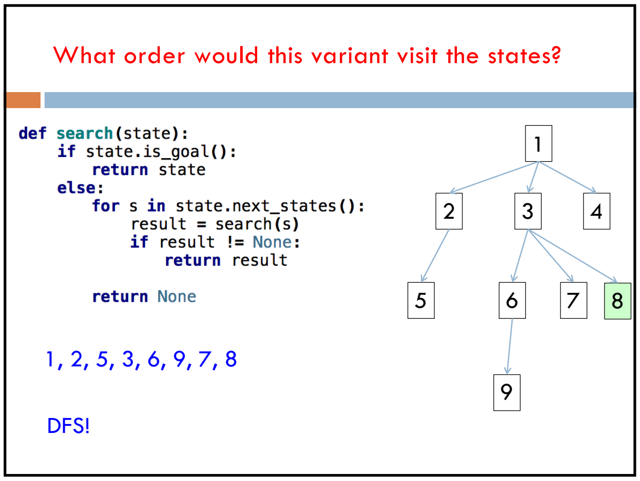
What order would this variant visit the states?

```
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?

10



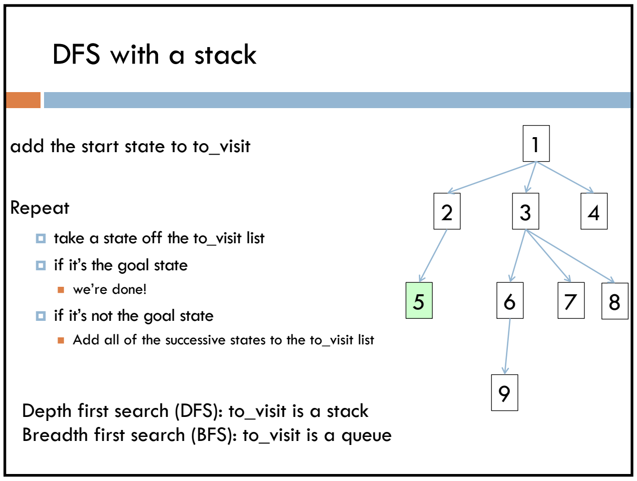
What order would this variant visit the states?

```
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!

11



DFS with a stack

add the start state to to_visit

Repeat

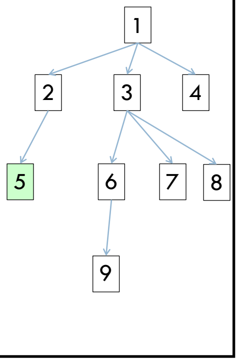
- take a state off the to_visit list
- if it's the goal state
    - we're done!
- if it's not the goal state
    - Add all of the successive states to the to_visit list

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

12

DFS with a stack

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

13

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?

14

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
```

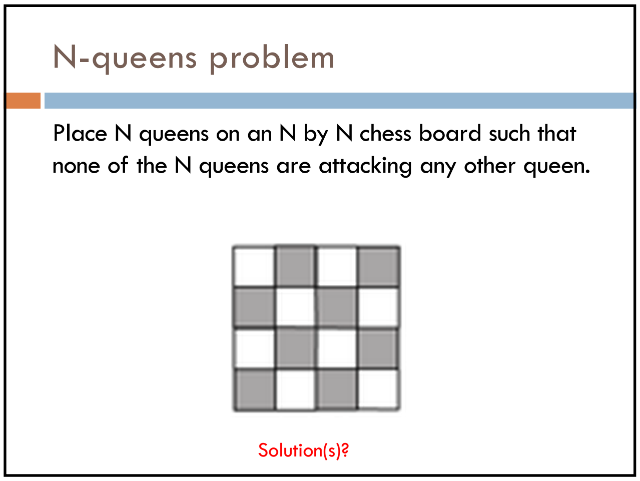
```
def dfs(state):
    if state.is_goal():
        return [state]
    else:
        result = []

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

        return result
```

Returns ALL solutions found, not just one

15

Matrices!

16

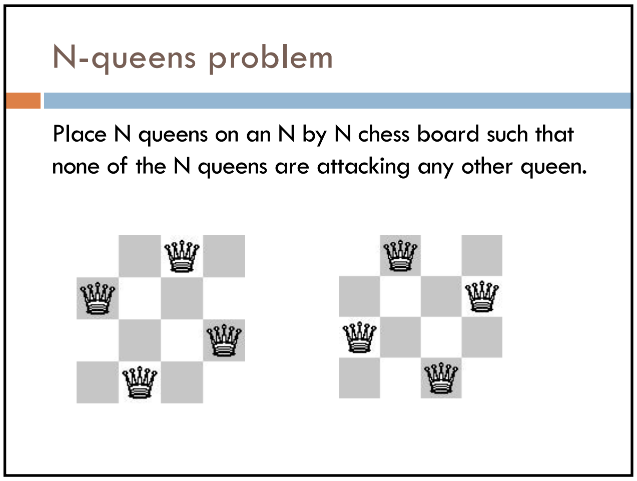


N-queens problem

Place N queens on an N by N chess board such that none of the N queens are attacking any other queen.

Solution(s)?

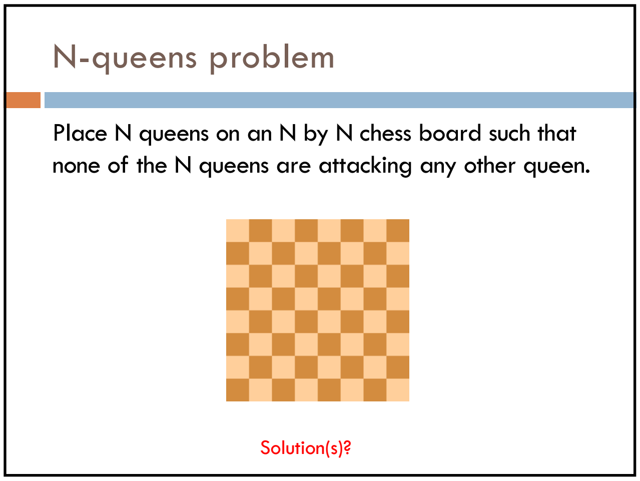
17



N-queens problem

Place N queens on an N by N chess board such that none of the N queens are attacking any other queen.

18

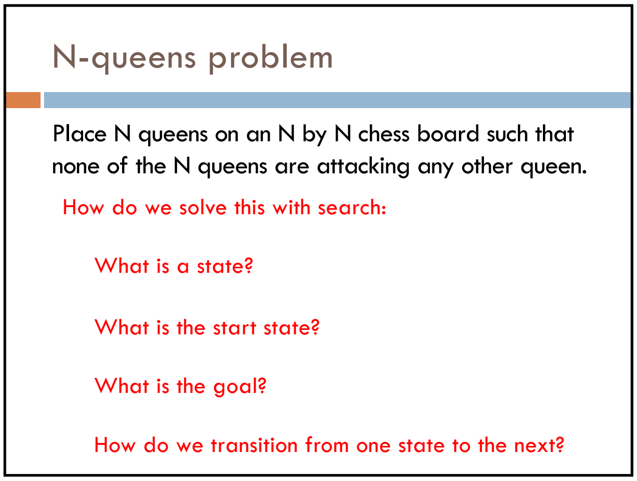


N-queens problem

Place N queens on an N by N chess board such that none of the N queens are attacking any other queen.

Solution(s)?

19



N-queens problem

Place N queens on an N by N chess board such that none of the N queens are attacking any other queen.

How do we solve this with search:

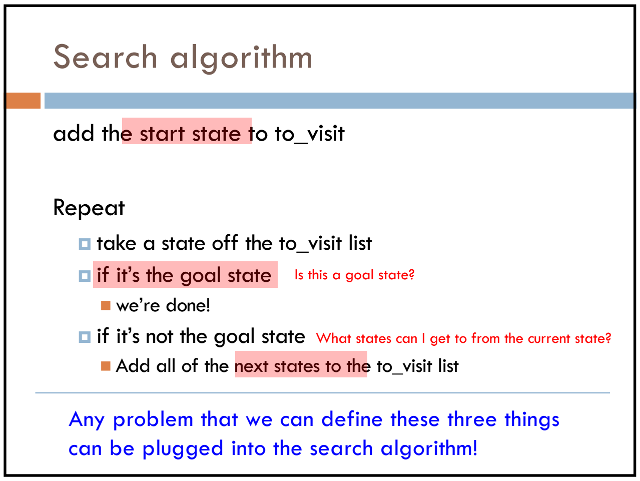
What is a state?

What is the start state?

What is the goal?

How do we transition from one state to the next?

20



Search algorithm

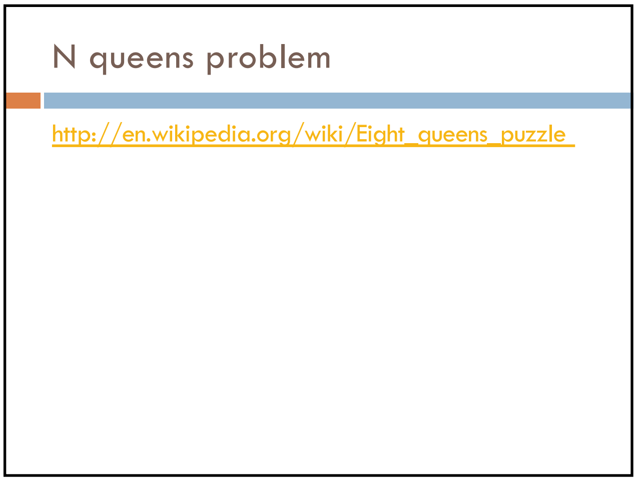
add the start state to to_visit

Repeat

- take a state off the to_visit list
- if it's the goal state Is this a goal state?
  - we're done!
- if it's not the goal state What states can I get to from the current state?
  - Add all of the next states to the to_visit list

Any problem that we can define these three things can be plugged into the search algorithm!

21



N queens problem

http://en.wikipedia.org/wiki/Eight_queens_puzzle

22