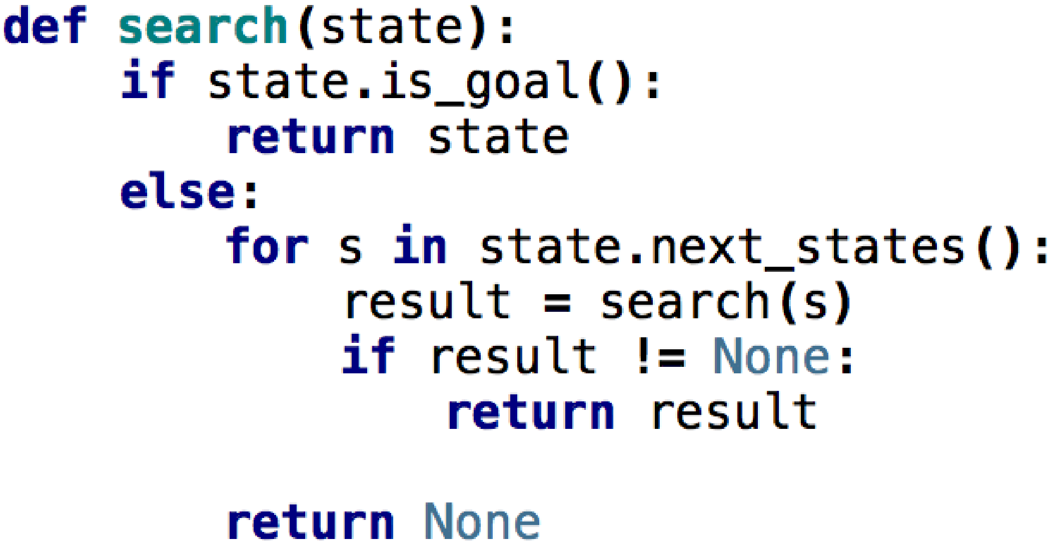
PROBLEM SOLVING VIA SEARCH

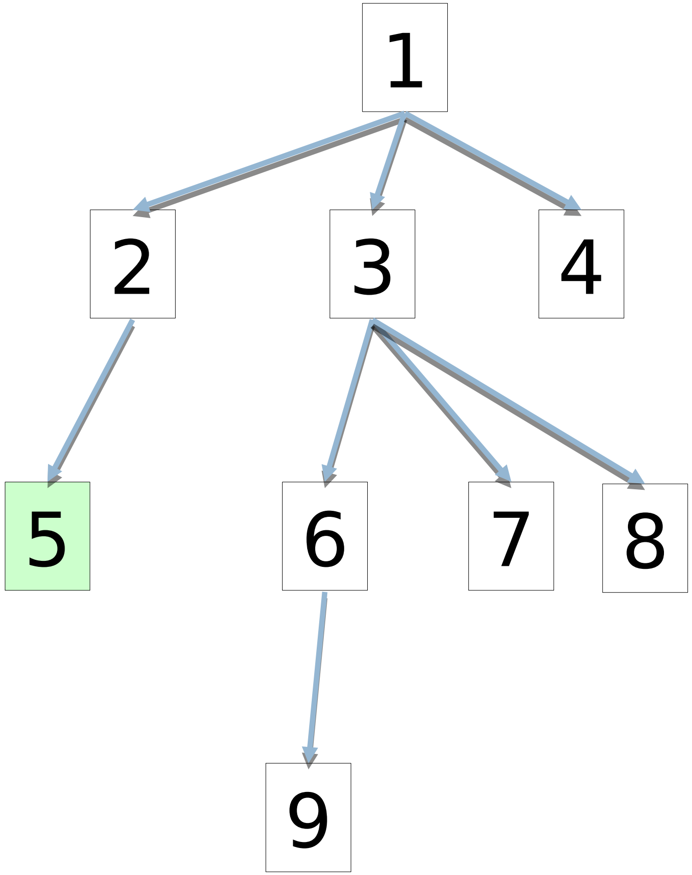
Joe Osborn

CS51A – Spring 2020

What order would this variant visit the states?

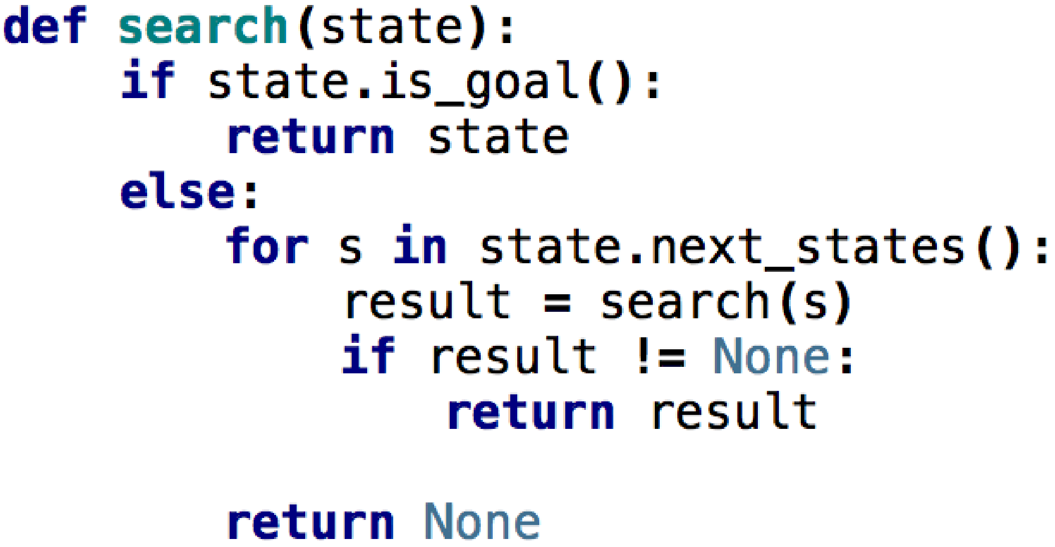


```
def search(state):
    if state.is_goal():
        return state
    else:
        for s in state.next_states():
            result = search(s)
            if result != None:
                return result

        return None
```

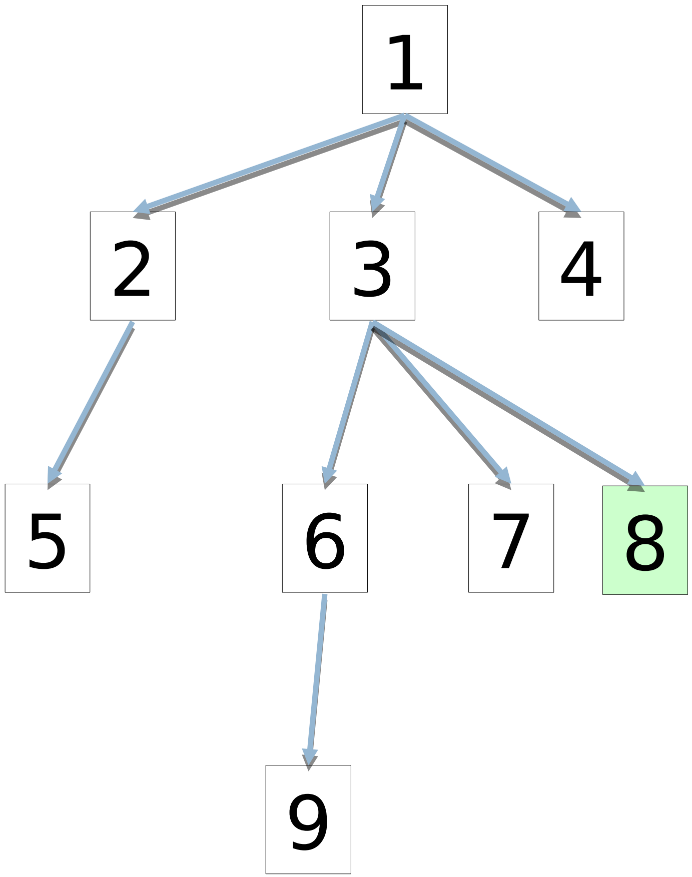


1, 2, 5

What order would this variant visit the states?



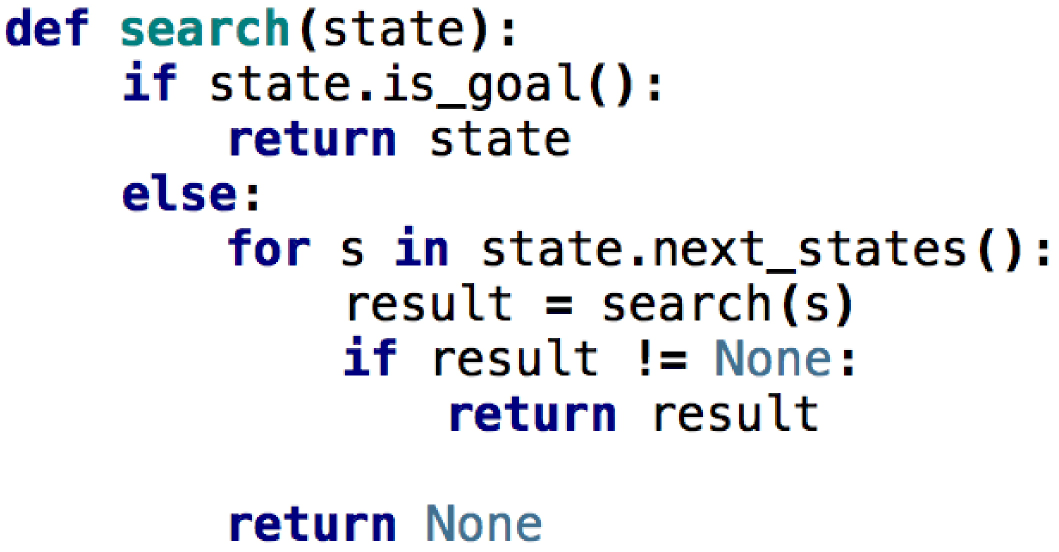
```
def search(state):
    if state.is_goal():
        return state
    else:
        for s in state.next_states():
            result = search(s)
            if result != None:
                return result

        return None
```



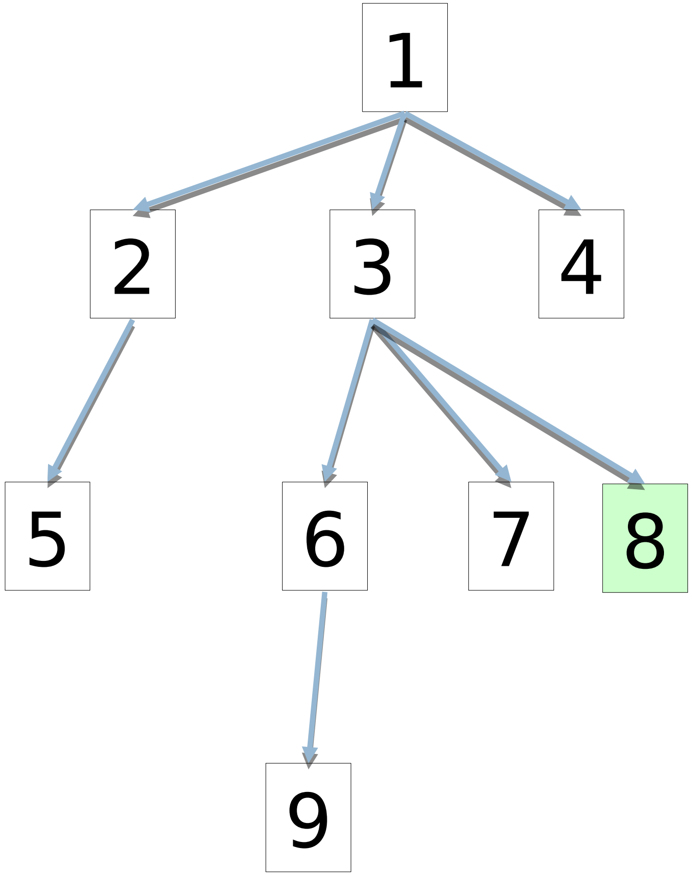
1, 2, 5, 3, 6, 9, 7, 8

What search algorithm is this?

What order would this variant visit the states?



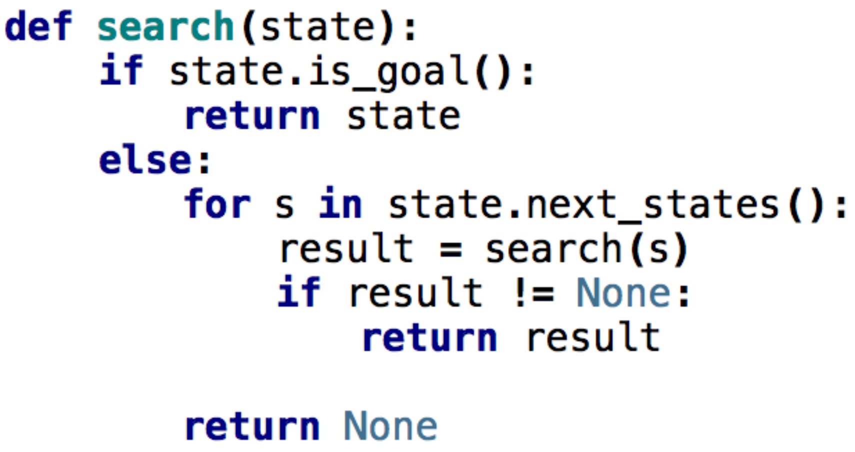
```
def search(state):
    if state.is_goal():
        return state
    else:
        for s in state.next_states():
            result = search(s)
            if result != None:
                return result

        return None
```



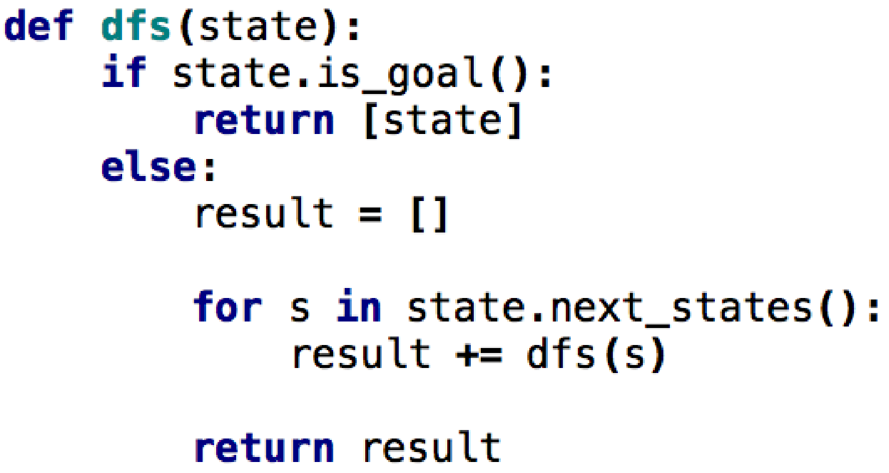
1, 2, 5, 3, 6, 9, 7, 8

DFS! Where’s the stack?

One last DFS variant



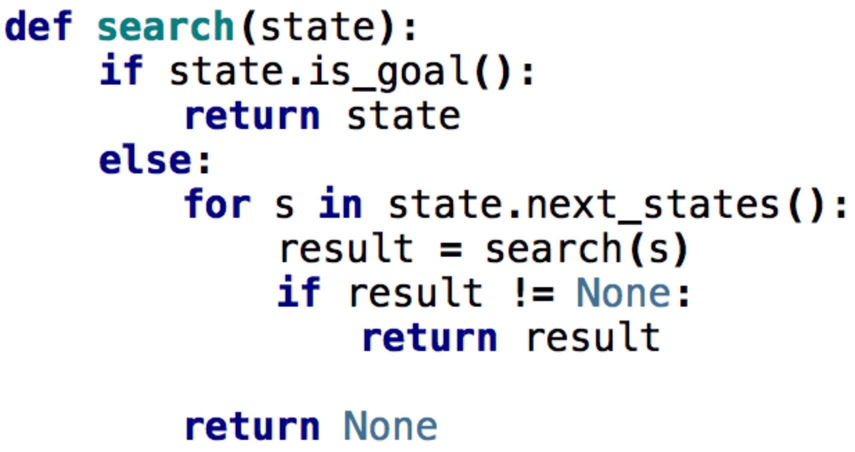
```
def search(state):
    if state.is_goal():
        return state
    else:
        for s in state.next_states():
            result = search(s)
            if result != None:
                return result

        return None
```



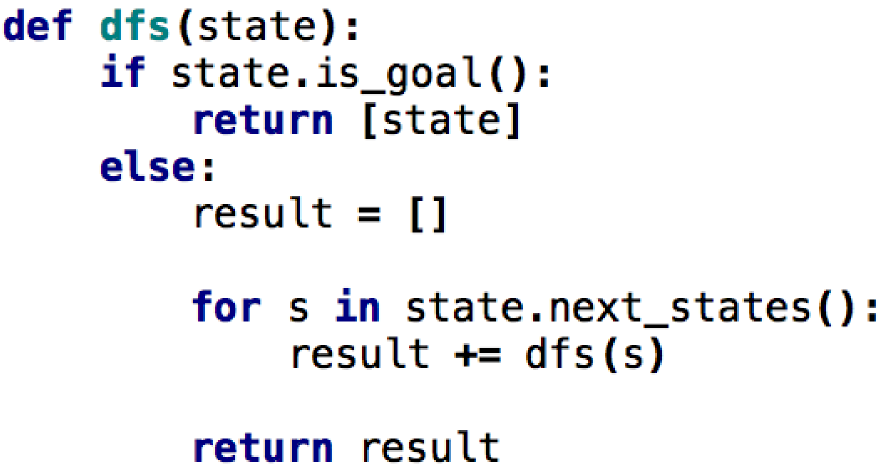
```
def dfs(state):
    if state.is_goal():
        return [state]
    else:
        result = []

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

        return result
```

How is this different?

One last DFS variant



```python
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
```

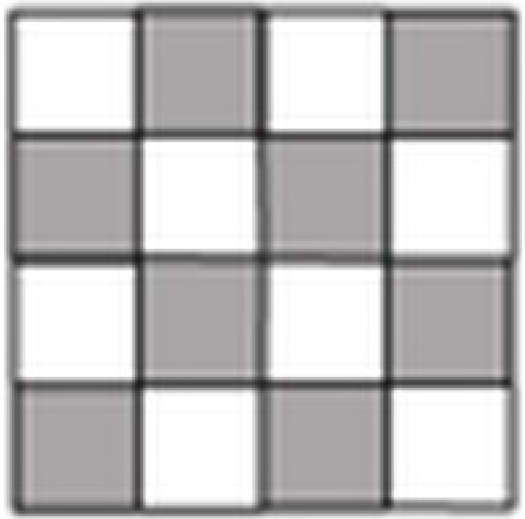


```python
def dfs(state):
    if state.is_goal():
        return [state]
    else:
        result = []

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

        return result
```

Returns ALL solutions found, not just one

N-queens problem

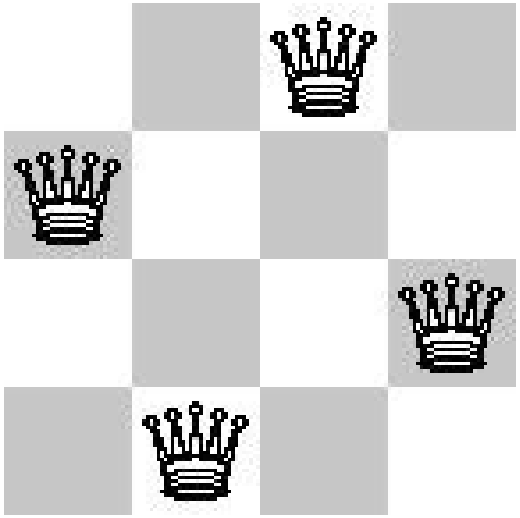
Place N queens on an N by N chess board such that none of the N queens are attacking any other queen.

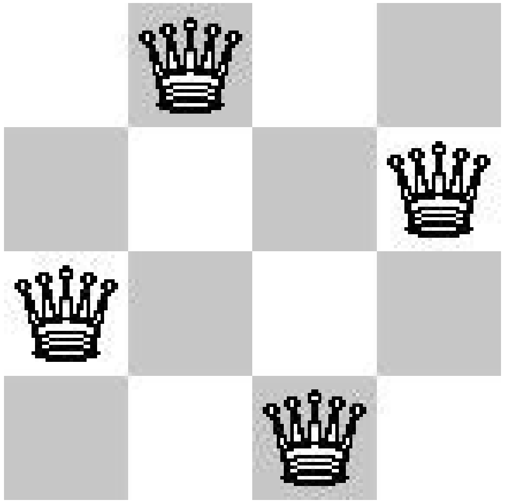


Solution(s)?

N-queens problem

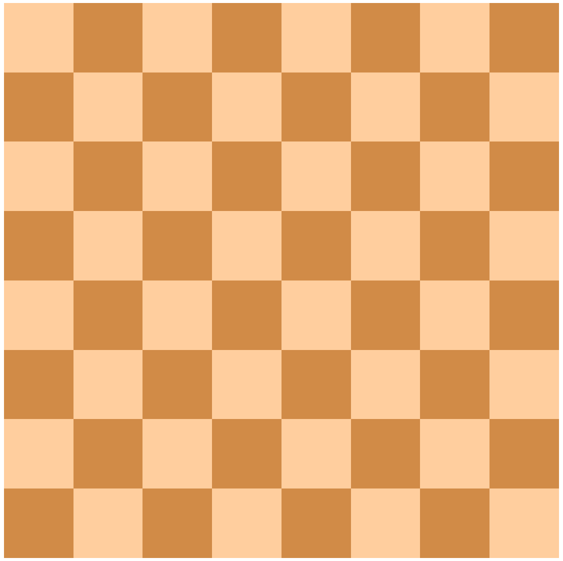
Place N queens on an N by N chess board such that none of the N queens are attacking any other queen.





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)?

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?

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
  - Add all of the next states to the to_visit list

What states can I get to from the current state?

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

N queens problem

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