SEARCH 2

David Kauchak
CS51A – Spring 2025

1

Admin

Assignment 8

Assignment 9

2

Copying a matrix

```
>>> m = [[0,0,0], [0,0,0], [0,0,0]]
>>> m2 = m[:]
>>> m2[0][0] = 1
```

Any problems?

3

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

5

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?

6

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

7

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"

8

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)

9

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
```

10

Tic tac toe

https://cs.pomona.edu/classes/cs51a/examples/tic_tac_toe.txt

Representing the board

- Three pieces of information

```
def __init__(self, size):
    self.size = size
    self.current_mark = "X"

    # construct a new board that is size by size
    self.board = []

    for i in range(self.size):
        self.board.append(["_"] * size)
```

11

Tic tac toe

https://cs.pomona.edu/classes/cs51a/examples/tic_tac_toe.txt

Adding a move

- Returns a **new** TicTacToe state
- Need to update all information for the **new** state

```
def add_mark(self, row, col):    1 usage
    new_board = copy.deepcopy(self)
    new_board.board[row][col] = new_board.current_mark

    if new_board.current_mark == "X":
        new_board.current_mark = "O"
    else:
        new_board.current_mark = "X"

    return new_board
```

12

Tic tac toe

https://cs.pomona.edu/classes/cs51a/examples/tic_tac_toe.txt

Checking for a win (diagonal only)

1. Upper left to lower right?

```
def is_diagonal_win(self):  1 usage
    if self.board[0][0] == "_":
        return False
    else:
        mark = self.board[0][0]
```

13

Tic tac toe

https://cs.pomona.edu/classes/cs51a/examples/tic_tac_toe.txt

Checking for a win (diagonal only)
1. Upper left to lower right

```
def is_diagonal_win(self):  1 usage
    if self.board[0][0] == "_":
        return False
    else:
        mark = self.board[0][0]

        for i in range(self.size):
            if self.board[i][i] != mark:
                return False

        return True
```

14

Tic tac toe

https://cs.pomona.edu/classes/cs51a/examples/tic_tac_toe.txt

Checking for a win (diagonal only)
2. Upper right to lower left?

```
def is_other_diagonal_win(self):  1 usage
    if self.board[0][self.size - 1] == "_":
        return False
    else:
        mark = self.board[0][self.size - 1]
```

15

Tic tac toe

https://cs.pomona.edu/classes/cs51a/examples/tic_tac_toe.txt

Checking for a win (diagonal only)
2. Upper right to lower left

```
def is_other_diagonal_win(self):  1 usage
    if self.board[0][self.size - 1] == "_":
        return False
    else:
        mark = self.board[0][self.size - 1]

        for i in range(self.size):
            if self.board[i][self.size - 1 - i] != mark:
                return False

        return True
```

16

Tic tac toe

https://cs.pomona.edu/classes/cs51a/examples/tic_tac_toe.txt

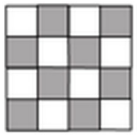
The rest of the code:

- is_goal
- __str__
- Running the code

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.

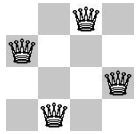


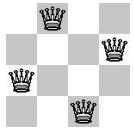
Solution(s)?

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.

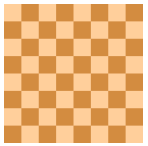




19

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.



Solution(s)?

20

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?

21

Search algorithm

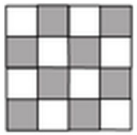
add the start state to to_visit

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!

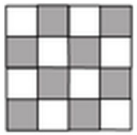
22

Start state



23

next_states?



24

next_states

Many options
- Add a queen anywhere
- Add a queen anywhere that doesn't cause a conflict
- Add a queen in the next row that doesn't cause a conflict

25

next_states

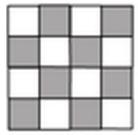
Many options

- Add a queen anywhere
- Add a queen anywhere that doesn't cause a conflict
- Add a queen in the next row that doesn't cause a conflict

26

next_states

Add a queen in the next row that doesn't cause a conflict



Where are the options?

27

N queens problem

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

28

Foxes and Chickens

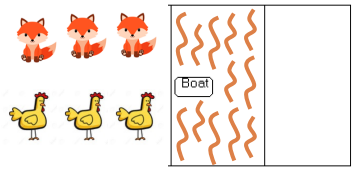
Three foxes and three chickens wish to cross the river. They have a small boat that will carry up to two animals. Everyone can navigate the boat. If at any time the foxes outnumber the chickens on either bank of the river, they will eat the chickens. Find the smallest number of crossings that will allow everyone to cross the river safely.

What is the "state" of this problem (it should capture all possible valid configurations)?

29

Foxes and Chickens

Three foxes and three chickens wish to cross the river. They have a small boat that will carry up to two animals. Everyone can navigate the boat. If at any time the foxes outnumber the chickens on either bank of the river, they will eat the chickens. Find the smallest number of crossings that will allow everyone to cross the river safely.



30

Foxes and Chickens

Three foxes and three chickens wish to cross the river. They have a small boat that will carry up to two animals. Everyone can navigate the boat. If at any time the foxes outnumber the chickens on either bank of the river, they will eat the chickens. Find the smallest number of crossings that will allow everyone to cross the river safely.

FFFCCC B

FFCC B FC

FC B FFCC

…

31

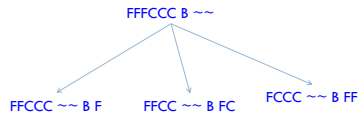
Searching for a solution

FFFCCC B ~~

What states can we get to from here?

32

Searching for a solution



Next states?

33

Fox and Chickens Solution

```
FFFCCC B|~~~~~~|
FFCC    |~~~~~~|B FC
FFCCC  B|~~~~~~| F
CCC     |~~~~~~|B FFF
FCCC   B|~~~~~~| FF
FC      |~~~~~~|B FFCC
FFCC   B|~~~~~~| FC
FF      |~~~~~~|B FCCC
FFF    B|~~~~~~| CCC
F       |~~~~~~|B FFCCC
FC     B|~~~~~~| FFCC
        |~~~~~~|B FFFCCC
```

How is this solution different than the n-queens problem?

34

Fox and Chickens Solution

```
FFFCCC B|~~~~~~|
FFCC    |~~~~~~|B FC
FFCCC  B|~~~~~~| F
CCC     |~~~~~~|B FFF
FCCC   B|~~~~~~| FF
FC      |~~~~~~|B FFCC
FFCC   B|~~~~~~| FC
FF      |~~~~~~|B FCCC
FFF    B|~~~~~~| CCC
F       |~~~~~~|B FFCCC
FC     B|~~~~~~| FFCC
        |~~~~~~|B FFFCCC
```

Solution is not a state, but a sequence of actions (or a sequence of states)

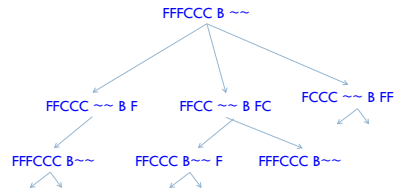
35

Code!

https://cs.pomona.edu/classes/cs51a/examples/chickens.txt

36

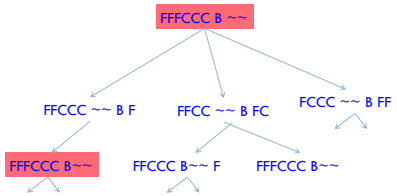
One other problem



What would happen if we ran DFS here?

37

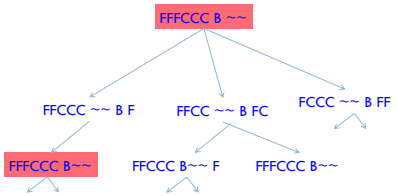
One other problem



If we always go left first, will continue forever!

38

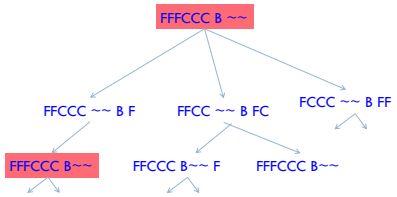
One other problem



Does BFS have this problem?

39

One other problem



Does BFS have this problem? No!

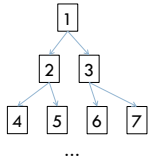
40

DFS vs. BFS

Why do we use DFS then, and not BFS?

41

DFS vs. BFS



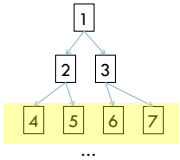
Consider a search problem where each state has two states you can reach

Assume the goal state involves 20 actions, i.e. moving between ~20 states

How big can the queue get for BFS?

42

DFS vs. BFS



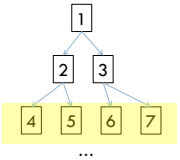
Consider a search problem where each state has two states you can reach

Assume the goal state involves 20 actions, i.e. moving between ~20 states

At any point, need to remember roughly a "row"

43

DFS vs. BFS



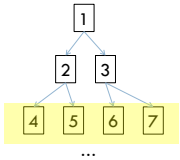
Consider a search problem where each state has two states you can reach

Assume the goal state involves 20 actions, i.e. moving between ~20 states

How big does this get?

44

DFS vs. BFS



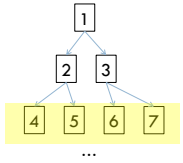
Consider a search problem where each state has two states you can reach

Assume the goal state involves 20 actions, i.e. moving between ~20 states

Doubles every level we have to go deeper.
For 20 actions that is 2^{20} = ~1 million states!

45

DFS vs. BFS



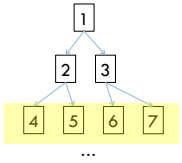
Consider a search problem where each state has two states you can reach

Assume the goal state involves 20 actions, i.e. moving between ~20 states

How many states would DFS keep on the stack?

46

DFS vs. BFS



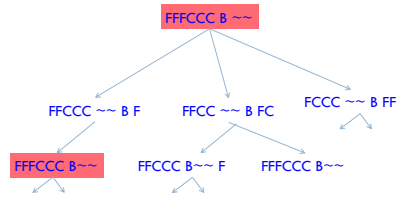
Consider a search problem where each state has two states you can reach

Assume the goal state involves 20 actions, i.e. moving between ~20 states

Only one path through the tree, roughly 2*20 states

47

One other problem



If we always go left first, will continue forever!

Solution?

48

DFS avoiding repeats

```
def dfs(state, visited):
    # note that we've visited this state
    visited[str(state)] = True

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

        for s in state.next_states():
            # check if we've visited a state already
            if not(str(s) in visited):
                result += dfs(s, visited)

        return result
```

49

Other search problems

What problems have you seen that could be posed as search problems?

What is the state?

Start state

Goal state

State-space/transition between states

50