



Wrapping up

CS51A
David Kauchak
Fall 2025

1



Admin

No normal mentor hours this week

Will announce additional mentor sessions soon

My office hours:

- Normal hours this week and on Monday next week
- No office hours after that (including Tuesday morning)

2



Final exam

Sample problems posted

Three hours, comprehensive

Can bring 4 pages of notes (4 single-sided or two double-sided)

Tuesday afternoon, 2-5pm

3




What we covered

Python!

- variables
- functions
 - writing functions
 - calling functions
 - optional parameters
- loops
 - for
 - while
- conditionals
 - if, elif, else
- aliasing
 - lists
 - matrices

4



What we covered

Python!

- strings
 - string methods
- recursion
- classes
 - self
 - writing your own methods
 - local variables vs. instance variables
- file I/O
 - reading/writing from files

5



What we covered

Python!

- built-in modules
 - random
 - turtle
 - deepcopy
- exceptions
 - raising
 - handling (try/except)
- data structures
 - lists
 - tuples
 - dictionaries
 - sets

6



What we covered

Machine learning

- Naive Bayes model

Neural Networks

Search


- algorithms — BFS, DFS (recursive and using stack), best-first
- problem solving
- adversarial search and game playing

Webpages

- html basics
- Reading from webpages

AI ethics

7



Where we started

```
# This program figures out the number of hot dogs
# needed for a BBQ
teran = 1
jasmin = 2
chris = 2 * jasmin
brenda = chris - 1
grace = (brenda+1)//2 + 1 # add 1 to brenda using truncated division to round up
total_hotdogs = teran + jasmin + chris + brenda + grace
print(total_hotdogs)
```

8

Where we ended

```
def dfs(state):
    """Recursive search for a path from start to goal"""
    # Base case: if state is goal, return the path
    if is_goal(state):
        return [state]

    # If the current state is a goal state, then return it as a list
    if is_goal(state):
        return [state]

    # Else, return an empty list
    result = []

    # For each action in the actions list
    for a in actions:
        # Compute the next state
        next_state = apply_action(state, a)

        # If next state is not in the visited set, then
        if next_state not in visited:
            # Add it to the visited set
            visited.add(next_state)

            # Recursively search for a path from next state
            path = dfs(next_state)

            # If path is not empty, then
            if path:
                # Add the current action to the path
                result.append([a] + path)

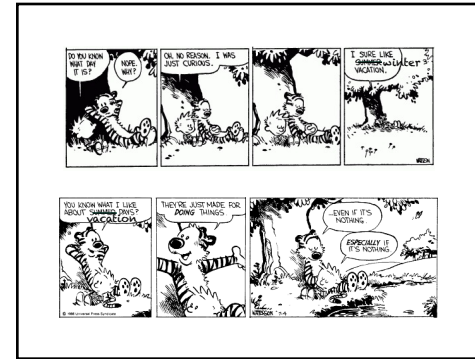
    return result

# Initialize the search
start_state = S0
goal_state = G0
visited = set()

# Call the dfs function
path = dfs(start_state)

# Print the path
print(path)
```

9



10