6.101 Recitation 8: Week 4 Bacon Number Wrap-up **3/4/24**

This sheet is yours to keep!

Question 1: Discuss with someone near you: what are some examples of each of the following terms?

- **graph -** consists of a set of vertices and a set of edges

- **vertices (also known as nodes, states) -** a single unique point or state in the graph

- **(undirected and directed) edges -** a pair of vertices that are neighbors (connected together.) Edges can either be directed (you can go from A → B but NOT from B → A) or undirected (you can go from A → B and from B → A)

- **path -** sequence of connected vertices.

- **graph search -** finding a path from some starting vertex to a goal vertex

- **breadth-first search -** searches all length n paths before length n + 1 paths, guaranteed to find shortest path in terms of number of nodes. Add /remove nodes from opposite sides of the agenda.

- **depth-first search –** travels down one particular path as far as possible until it reaches the end, then backtracks up the path to find the closest unexplored neighbor. Add/remove nodes from the same side of the agenda. Shortest path not guaranteed, but can be more memory efficient depending on the type of graph being explored.

Question 5: Given the following outlines of different path finding functions below, discuss:
- What is good about these functions?
- What could be improved?
- Is there anything potentially buggy in these solutions?

```
def bacon_path(transformed_data, actor_id):
    """
    bacon path
    """
    agenda = [(4724,)]
    visited = [(4724,)]
    # 20 lines of path-finding code from reading, using actor_id as the goal

def actor_to_actor_path(transformed_data, actor_id_1, actor_id_2):
    """
    Given a database from transform_data(),
    return a shortest path of actors from actor 1 to actor 2 as a list.
    """
    agenda = [(actor_id_1,)]
    visited = [(actor_id_1,)]
    # same 20 lines of code from reading, using actor_id_2 as the goal


def actor_path(transformed_data, actor_id_1, goal_test_function):
    """
    Given a database from transform_data(),
    return a shortest path of actors starting from actor 1 until reaching an actor who
    satisfies goal_test_function (which takes an actor and returns a truthy value).
    """
    # path-finding code using actor_id_1 as start and
    # goal_test_function(actor) as the stopping criterion
```

Question 6: How can we rewrite bacon_path in terms of actor_to_actor_path?

```
def bacon_path(transformed_data, actor_id):
```

Question 7: How can we rewrite actor_to_actor_path in terms of actor_path?

```
def actor_to_actor_path(transformed_data, actor_id_1, actor_id_2):
```

R8 Participation Credit **Kerberos : _____@mit.edu**

Hand this sheet in at the end of recitation to get participation credit for today.

Question 2: Assuming that transformed_data was a list of (actor_id_1, actor_id_2, film_id) tuples, write the body of the get_neighbors function below.

```
def get_neighbors(transformed_data, actor):
    """
    Gets a set of all of the actors that the provided actor id
    has acted with (not including the given actor).
    """
```

Question 3: Assuming that transformed_data was a dictionary where the keys were actor_ids and the values were sets of co-star actor_ids, write the body of the get_neighbors function below.

```
def get_neighbors(transformed_data, actor):
    """
    Gets a set of all of the actors that the provided actor id
    has acted with (not including the given actor).
    """
```

Question 4: How might such a get_neighbors function be useful in the bacon lab?

Question 8: How would you initialize the visited set and agenda for the function below?

```
def general_actor_path(transformed_data, start_actors, goal_test_function):
    # start actors is a set of actor ids representing valid start of path
    actor_map, _ = transformed_data
    visited =
    agenda =
```

Question 9: How would you initialize the visited set and agenda for the function below?

```
def super_general_actor_path(transformed_data, start_test_function,
                                                    goal_test_function):
    '''
    start_test_function(actor_id) returns true if actor_id
    is a valid actor to start path from, false otherwise
    '''
    actor_map, _ = transformed_data
    visited =
    agenda =
```