**Recipe:** find\_path

**Inputs:**

*graph*, an instance of the Graph class with edges representing the connections between people

*start\_person*, a vertex in graph representing the starting node

*end\_person*, a vertex in graph representing the ending node

*parents*, a mapping between vertices and their parents in the graph from the perspective of a breadth-first search algorithm starting at start\_person

**Outputs:**

*path*, a sequence of tuples of the form (actor1, a set of the attributes of the edge from actor1 to actor2)

**Steps:**

1. Initialize *path* to an empty sequence
2. Initialize *current\_node* to *end\_person*
3. Initialize *previous\_node* to *end\_person*
4. *edge* 🡨 an empty set
5. *start\_to\_end* 🡨 false
6. While *current\_node* is not null, do the following:
   1. If *current\_node* is not equal to *previous\_node*, then:
      1. *edge* 🡨 the result of calling the *get\_attrs* method of *graph* with the arguments *current\_node* and *previous\_node*
   2. Insert the tuple consisting of *current\_node* and *edge* to the start of *path*
   3. *previous\_node* 🡨 *current\_node*
   4. *current\_node* 🡨 the value of *parents* that corresponds to the key *current\_node*
7. For each *node*, *edge* in the tuples of *path*:
   1. If *node* is *start\_person*, then set *start\_to\_end* to True.
8. If *start\_to\_end* is false,
   1. Set path to an empty sequence
9. Return *path*

**Discussion**

1. Finding the diameter is simple, though rather time consuming. We can run a breadth first search on an arbitrarily chosen node, then run it again on the node with the highest value in the distances dictionary returned from the function. We can continue this until the highest value in the current iteration’s distances dictionary equals the highest value in the previous iteration’s distances dictionary.
2. While time consuming, one can run a bfs on every node in a graph and then plug those into a distance histogram. By multiplying each value of the histogram by some point value assigned to the keys of the histogram, say 10 points for every direct connection being represented by the value at keys[1] being multiplied by 10, and so on for the other keys of the histogram, hubs can be found by how many points each node has. If direct connections are substantially more important than even secondary connections, maybe the point values could even go up by a whole magnitude.
3. If a movie appears a significant amount of times in a graph, then clearly that movie connects a substantial amount of actors and is thus much more relevant to the structure of the graph as opposed to a movie that connects 2 actors that are already in the graph. Though if a movie connects a new actor to the graph, it might be considered important too.
4. Kevin Bacon has a substantial amount of 2s and 3s, with a noticeable amount of 1s and nothing past 4. Stephanie Fratus in turn has an unnoticeable amount of 1s, barely any 2s, a significant amount of 3s and 4s and a few 5s and 6s. Clearly, Kevin Bacon is substantially more well-connected to other actors and actresses and thus must star in significantly more movies with greater casts.