COMP10001 Foundations of Computing
Semester 1, 2022

Tutorial Questions: Week 10

— VERSION: 1477, DATE: MAY 9, 2022 —

Discussion

1. What is an "iterator"? What are some helpful methods in the `itertools` library?

   Now try Exercise 1

Exercises

1. What output does the following code print?

```
import itertools
beatboxer = itertools.cycle(['boots', 'and', 'cats', 'and'])

for count in range(39):
    print(next(beatboxer))
```

Problems

1. Write a function which takes two strings as input and uses an `itertools` iterator to find whether the first word is an anagram of the second word. This might not be a very efficient way to find an anagram but it will help us work with iterators! `anagram('astronomer', 'moonstarer')` should return `True`

2. Write a function which takes a lowercase string as input and prints the frequency of each vowel in the string using a dictionary. `vowel_counts('i love python')` should print:

   ```
   i 1
   e 1
   o 2
   ```

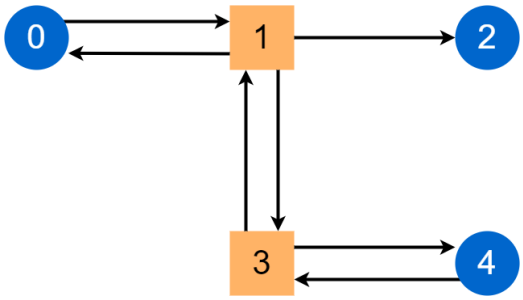
3. Write a function which takes two lists of integers and returns the average of the numbers which they both have in common. `in_common_average([1, 2, 3, 4, 5], [0, 2, 4, 6])` should return `3.0`

Project 2

Project 2 involves building and simulating a road network model. There will be a particular focus on tracking vehicle movement and modelling intersection behaviour. Below you will find three representations of the same road network; please take note of how they relate to each other.

1. Graph Visualisation

The conceptual model that we are working with is a graph network. The network is made up of "location" nodes (circles) and "intersection" nodes (squares). Directed roads are then represented as arrows connecting nodes. Note that the one-way road between intersection 1 and location 2 is represented by a single arrow.



2. Text File

The first project question asks you to read in a text file of the following form. Each intersection has a number of traffic signals that it will cycle through. A traffic signal is composed of `(source, destination)` pairs that represent a permitted direction of travel. The file also lists the roads in the network. These are formatted as `(source, destination): road_length`.

`sample.txt`

```
#Intersection:1
(0,2)
(0,3);(3,0);(3,2)

#Intersection:3
(1,4);(4,1)

#Roads
(0,1):1
(1,0):1
(1,2):2
(1,3):2
(3,1):1
(3,4):1
(4,3):1
```

3. Dictionaries

The first project question then requires you to convert the above file format into two Python dictionaries like the ones below. Similar dictionary representations are then used for the remainder of the project.

```
intersections = {1: [[(0, 2)], [(0, 3), (3, 0), (3, 2)]],
                 3: [[(1, 4), (4, 1)]]}
road_times = {(0, 1): 1, (1, 0): 1, (1, 2): 2, (1, 3): 2,
              (3, 1): 1, (3, 4): 1, (4, 3): 1}
```

The following questions and descriptions serve as an introduction to the project. Please see the project on Grok for full details and requirements.

Path Cost

A path through the road network describes the nodes that a car will travel through. The `path_cost` function takes a car's `path` with the previously defined `intersections` and `road_times` dictionaries. The function should return the total number of timesteps that the car travelled on its journey.

The car begins driving at timestep 0 from the first node. If it arrives at an intersection where the signal allows it to continue, it can move on to the next road immediately, (no additional timestep). However, traffic signals change every timestep; if the car has to wait at any intersection, `path_cost` should return `None`.

Given the previously defined `intersections` and `road_times`, evaluate the following function calls:

1. `path_cost([0,1,3,4], intersections, road_times)`
2. `path_cost([4,3,1,0], intersections, road_times)`

Intersection Step

We are interested in modelling what happens when multiple cars arrive at the same intersection. Each car is represented as `(car_id, path, arrival_time)`. The `intersection_step` function takes the `intersections` and `road_times` dictionaries to simulate a given `intersection_id`. It takes a list of `cars_at_intersection` and the current `timestep` to return a list of the `car_id`s of cars that will move through the intersection.

There are a few key rules:

- Only one car can travel through in each permitted direction
- If multiple cars need to move in the same direction, the car that arrived first is given priority
- If two cars travelling in the same direction arrived at the same timestep, the smallest `car_id` moves

Given the previously defined `intersections` and `road_times`, evaluate the following function call:

```
cars_at_intersection = [(3, [4,3,1,0], 3), (4, [4,3,1,2], 3),
                        (7, [4,3,1,0], 2), (6, [4,3,1,2], 3),
                        (5, [0,1,2], 2)]
intersection_id, timestep = 1, 3
intersection_step(intersections, road_times, intersection_id,
                  cars_at_intersection, timestep)
```

Full Simulation

We now wish to simulate the running of the entire road network. The `simulate` function will model cars entering the network at different times and proceeding to their destinations, interacting at intersections. The function takes in `intersections` and `road_times` with a list of `cars_to_add`. Each car in the list is represented as `(car_id, path, timestep)` to define which `timestep` to add the car to the network.

The function returns a list of actions describing what each car was doing at each timestep. The actions are represented with the following strings:

- `drive(timestep, car_id, source_node, destination_node)`
- `wait(timestep, car_id, intersection_id)`
- `arrive(timestep, car_id, destination)`

Given the previously defined `intersections` and `road_times`, evaluate the following function call:

```
cars_to_add = [(0, [4,3,1,0], 0), (1, [4,3,1,2], 0), (2, [0,1,2], 2)]
simulate(intersections, roads_cost, cars_to_add)
```