COMP10001 Foundations of Computing
Semester 1, 2022

Tutorial Solutions: Week 10

— VERSION: 1477, DATE: MAY 9, 2022 —

Discussion

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

   A: *An iterator is an object that keeps track of the traversal of a container. It is used by loops to keep track of iteration through a list, set, dictionary, tuple or string (these are objects we say are "iterable"). Iterators allow use of the* `next(<iterator>)` *function to progress to the next item in the iterator, and will raise a* `StopIteration` *exception if the end is reached. Note that iterators, unlike any container types, can be infinite in length.*
   `itertools` *provides many methods to construct iterators. They include* `cycle` *which produces an iterator to cycle through a container, looping from the end back to the beginning infinitely.* `product` *will combine two containers into one tuple, with each element from the first one combined with each element from the second.* `combinations` *will produce a sequence of every possible combination of elements from a container, while* `permutations` *will include combinations with different pair orderings too.* `groupby` *will group elements of a container together in particular categories based on a function parameter.*

   Now try Exercise 1

Exercises

1. What output does the following code print?

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

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

   A: *This code will print ten iterations of* `boots and cats and` *which will end with* `cats`*:*

```
boots
and
cats
and
boots
...
and
boots
and
cats
```

   Try changing the `for` *loop to* `while True:` *(an infinite loop) to see this cycle print infinitely!*

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`

   A:

```python
from itertools import permutations

def anagram(word1, word2):
    for ordering in permutations(word1, len(word1)):
        if "".join(ordering) == word2:
            return True
    return False
```

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
```

A:

```python
def vowel_count(text):
    vowel_counts = {}
    # Count vowel frequencies
    for letter in text:
        if letter in 'aeiou':
            if letter in vowel_counts:
                vowel_counts[letter] += 1
            else:
                vowel_counts[letter] = 1

    for vowel, count in vowel_counts.items():
        print(vowel, count)
```

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`

A:

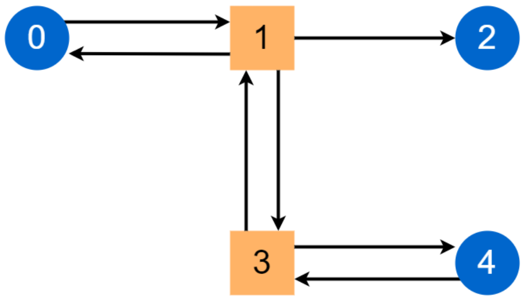
```python
def in_common_average(list1, list2):
    common = set(list1) & set(list2)
    return sum(common) / len(common)
```

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)`

A: *1. Returns `4`. The car travels from 0→1 at which point it is timestep 1. This means that it can immediately travel through the intersection since (0, 3) is included in the traffic signal. Then it travels 1→3 in 2 timesteps and can immediately pass through the intersection since (1, 4) is always in the traffic signal there. Then the car travels 3→4 in 1 timestep for a total of 4 timesteps.*
2. Returns `None`. The car travels from 4→3 in 1 timestep and can immediately travel through the intersection. The car then travels 3→1 in 1 timestep to arrive at node 1 at timestep 2. At timestep 2 the traffic signal is `[(0, 2)]` at intersection 1, meaning that the car must wait at the intersection. Because the car must wait, the function returns `None`.

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)
```

A: *Returns `[4, 7]`. Cars 3 and 7 need to travel in the same direction, as do cars 4 and 6. Out of 3 and 7, car 7 arrived first so has priority to move. Out of 4 and 6, they arrived at the same time so car 4 has priority since it has a lower id number. The timestep is 3, so the traffic signal is `[(0, 3), (3, 0), (3, 2)]`. This means that car 5 cannot move, but cars 4 and 7 can. Because 4 and 7 are the only cars that can move through the intersection, the function returns `[4, 7]`.*

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)
```

A: *Returns:*

```
['drive(0,␣0,␣4,␣3)', 'drive(0,␣1,␣4,␣3)', 'drive(1,␣0,␣3,␣1)',
 'wait(1,␣1,␣3)', 'wait(2,␣0,␣1)', 'drive(2,␣1,␣3,␣1)',
 'drive(2,␣2,␣0,␣1)', 'drive(3,␣0,␣1,␣0)', 'drive(3,␣1,␣1,␣2)',
 'wait(3,␣2,␣1)', 'arrive(4,␣0,␣0)', 'drive(4,␣1,␣1,␣2)',
 'drive(4,␣2,␣1,␣2)', 'arrive(5,␣1,␣2)', 'drive(5,␣2,␣1,␣2)',
 'arrive(6,␣2,␣2)']
```

Step 0: The simulation adds cars 0 and 1 in immediately and they both drive from node 4→3.
Step 1: Car 0 and 1 reach intersection 3 at the same time, so car 0 gets priority and moves through to drive towards node 1. Car 1 must wait at intersection 3.
Step 2: Car 0 must wait at intersection 1 due to the traffic signal. Car 1 now moves through intersection 3 and drives towards intersection 1. Car 2 enters the network and drives from node 0→1.
Step 3: Car 0 makes it through intersection 1 and drives towards node 0. Car 1 arrives at intersection 1 and also makes it through towards node 2. (Both cars could go through because they were going in different directions). Car 2 must wait at intersection 1 because the traffic signal does not allow it to go.
Step 4: Car 0 arrives at node 1. Car 1 continues driving towards node 2, (the road takes 2 timesteps to travel). Car 2 makes it through intersection 1 and drives towards node 2.
Step 5: Car 1 arrives at node 2. Car 2 continues driving towards node 2.
Step 6: Car 2 arrives at node 2.