2019 CAB320 - Week 02 Prac - Solutions

Last modified on 2019-02-26 by f.maire@qut.edu.au

Exercise 1

- Consider the generator expression   g = (i*i for i in xrange(5,10))
- What are the first 3 values generated?   hint: try g.next()

```
>>> g = (i*i for i in xrange(5,10))
>>> next(g)
25
>>> next(g)
36
>>> next(g)
49
```

Exercise 2

- Walk through the ***slow_zebra_puzzle.py*** file
- What is returned by the function *zebra_puzzle()* ?

  a generator
- How do you use the object returned by the function *zebra_puzzle()* ?
  hint: see Exercise 1

```
g = zebra_puzzle()
(w,z) = next(g)
```

Note that you can comment out some of the constraints to test your code.

With these constraints commented out, the search takes less than one minute on my computer.

The problem with this initial program is that it waits until all variables are assigned to check whether the assignment satisfies the constraints.

:

See file solution_zebra_puzzle.py

Exercise 3

Most current navigation systems generate global, metric representation of the environment with either obstacle-based grid maps or feature-based metric maps . While suitable for small areas, global metric maps have inefficiencies of scale. Arguably, topological mapping is more efficient as it concisely represents a partial view of the world as a graph. In this Exercise, you will show that when the graph is without cycles, a map is superfluous for navigation purpose.

When following the instructions of a navigation GPS device, the driver of a car receives instructions of the form *“at the next roundabout, take the third exit”*. This command format is well suited for logging the itinerary followed by a mobile agent in an environment that has a graphical topology like a road-network or the corridors of a building.

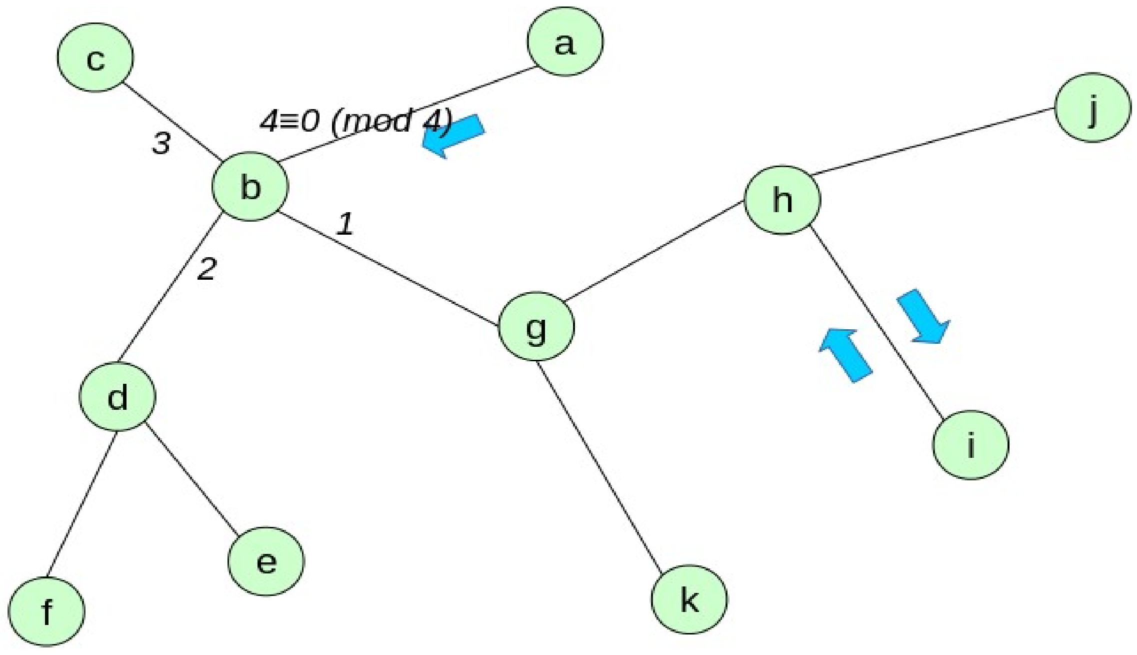
The figure below illustrates such an environment. The agent/robot traverses the graph following its edges and using its vertices as roundabouts. In order to indicate the direction the robot is facing on an edge, we specify the location of the robot by an arc. We adhere to the standard terminology of Graph Theory, and reserve the term **arc** for an edge that has been given an orientation.

The blue arrow on the left of arc **ab** represents the position of the robot going from Node **a** to Node **b**.

The navigational route instructions from the arc $a \rightarrow b$ as the starting position, to the arc labeled $h \rightarrow i$ as the destination, would sound as follows if told by a GPS device:

1. “drive to the next the intersection”
2. “turn first left”
3. “drive to the next the intersection”
4. “turn first left”
5. “drive to the next the intersection”
6. “turn second left”
7. “drive to the next the intersection”
8. “arrive at your destination”

A less verbose representation would code the route into the integer sequence (1, 1, 2)

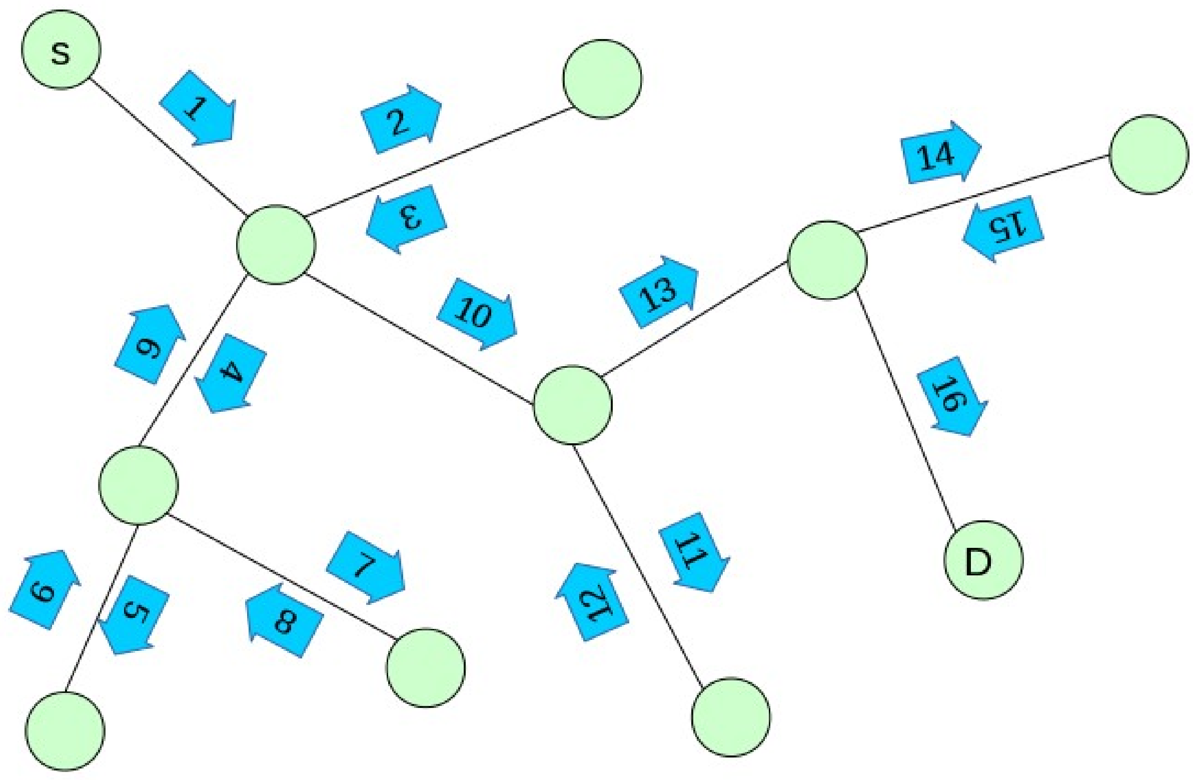


- Observe that when arriving at a node with n edges, the commands c and c' have the same effect if and only if the difference c-c' is a multiple of n.
- What is the effect of the command c=0 ? make a U-turn
- Compare the effect of the commands -c and n-c same effect
- What are the possible commands for a U-turn ? multiple of n where n is the degree of the node
- What happens at a leaf node (node with exactly one incident edge), does the value of c matters at a leaf? value of c is irrelevant as any value triggers a U-turn
- Given a forward route (c1 , c2 , . . . , ck) from arc alpha to arc beta, provide a formula for a return route from beta to alpha. (-ck , . . . , -c2, -c1) = (nk-ck , . . . , n2-c2, n1-c1)

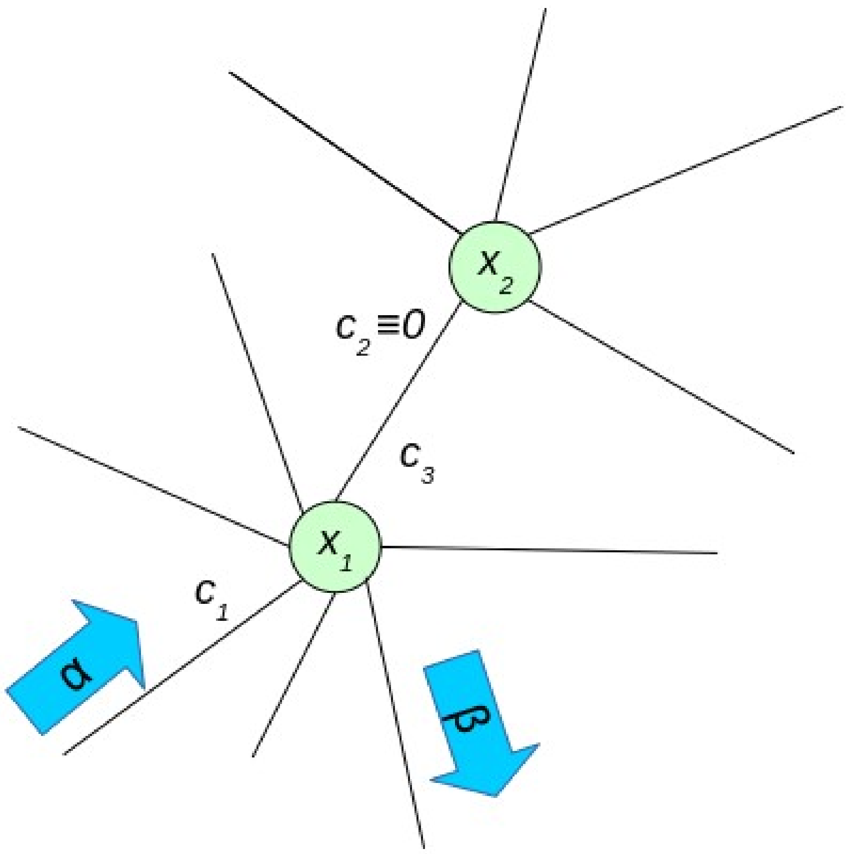
Exercise 4

This exercise continues Exercise 3 but is much harder. Consider it as a brain-teaser! Skip it if you are short on time.

When two routes c and c' start with the same arc α and end with the same destination arc β, we say that the routes are ***equivalent***. For example, in the Figure below, the route from Node S to Node D (1, 0, 2, 2, 0, 2, 0, 2, 3, 2, 0, 2, 1, 0, 1) is equivalent to the direct route (2,1,2).



- Design an algorithm to reduce a route to its shortest equivalent route. Hint: Look at the figure next page. Observe that whenever c2 is a multiple of n2 where n2 is the degree of the node x2 , the sequence (c1 , c2 , c3) has the same effect as the singleton sequence (c1 + c3).



```
input :
a route with the degrees of the traversed nodes
$\mathcal{C}$ : a command sequence $(c_1, c_2, \ldots, c_n)$ where the $c_i$'s are integer values (possibly negative)
$\mathcal{D}$ : a degree sequence $(d_1, d_2, \ldots, d_n)$ where $d_i$ is the number of edges incident to the $i^{th}$ traversed node
output:
the shortest route $\mathcal{C}'$ equivalent to $\mathcal{C}$
begin
    $\mathcal{C}' = \mathcal{C}$                                   /* initialize the reduced route */
    $\mathcal{D}' = \mathcal{D}$                                   /* initialize the associated degree sequence */
    repeat
        $k = \text{length}(\mathcal{C}')$                          /* recompute the length of the reduced route */
        for $i \in [2, k-1]$ do
            if $c_i \equiv 0 \ (mod\, d_i)$ then
                /* apply Proposition 7 to contract $(c_{i-1}, c_i, c_{i+1})$ into $(c_{i-1} + c_{i+1})$ in $\mathcal{C}'$ */
                $\mathcal{C}' \longleftarrow (c_1, c_2, \ldots, c_{i-2}, c_{i-1} + c_{i+1}, c_{i+2}, \ldots, c_k)$
                /* remove the degrees of the $i^{th}$ and $(i+1)^{th}$ nodes from $\mathcal{D}'$ */
                $\mathcal{D}' \longleftarrow (d_1, d_2, \ldots, d_{i-2}, d_{i-1}, d_{i+2}, \ldots, d_k)$
                break
            end
        end
    until no index $i \in [2, k-1]$ such that $c_i \equiv 0 \ (mod\, d_i)$ can be found
end
```

Algorithm 1: Route Minimization