Breadth-first Search

Python

Array Queue

```python
# python3

from collections import deque

class Queue:
  # constructor creates a list
  def __init__(self):
    self.queue = list()

  # adding elements to queue
  def enqueue(self, data):
    if data not in self.queue:
      self.queue.insert(0, data)
      print("%s %s" %(data, 'queued!',))
    return False

  def dequeue(self):
    if len(self.queue) > 0:
      item = self.queue.pop()
      print("%s %s" %(item, 'dequeued!',))
    return ("Queue Empty!")

  # getting the size of the queue
  def size(self):
    return len(self.queue)

myQueue = Queue()

for i in range(1, 11):
  myQueue.enqueue(i)

while myQueue.size() > 0:
  myQueue.dequeue()
```

List Queue

```python
# python3

from collections import deque

graph = {}
graph["you"] = ["alice", "bob", "claire"]
graph["bob"] = ["anuj", "peggy"]
graph["alice"] = ["peggy"]
graph["claire"] = ["thom", "jonny"]
graph["anuj"] = []
graph["peggy"] = []
graph["thom"] = []
graph["jonny"] = []

def search(name):
  search_queue = deque() # create a queue
  search_queue += graph[name] # add all your neighbours (alice, bob and claire) to the queue.
  searched = []

  while search_queue: # While the queue isn't empty
    person = search_queue.popleft() # dequeue the first person from queue.
    if not person in searched:
      if person_is_seller(person):
        print(person + " is a mango seller!")
        return True
      else:
        search_queue += graph[person] # Add all of this person's friends to the search queue.
        searched.append(person)
  return False
```

```
# This function checks whether the person's name ends with the letter m. If it does, they're a mango seller.
def person_is_seller(name):
  return name[-1] == 'm'

search("you")
```

Ruby

Array Queue

```
# encoding: UTF-8

require 'thread'

@queue = Queue.new

def produce
  (1..10).each do |i|
    sleep 0.2
    @queue << i
    puts "#{i} enqueued!"
  end
end

def consume
  @queue.close if @queue.empty?

  until @queue.empty?
    sleep 0.3
    value = @queue.shift
    puts "#{value} dequeued!"
  end
end

produce
consume
```

List Queue

```
# python3

from collections import deque

graph = {}
graph["you"] = ["alice", "bob", "claire"]
graph["bob"] = ["anuj", "peggy"]
graph["alice"] = ["peggy"]
graph["claire"] = ["thom", "jonny"]
graph["anuj"] = []
graph["peggy"] = []
graph["thom"] = []
graph["jonny"] = []

def search(name):
  search_queue = deque() # create a queue
  search_queue += graph[name] # add all your neighbours (alice, bob and claire) to the queue.
  searched = []

  while search_queue: # While the queue isn't empty
    person = search_queue.popleft() # dequeue the first person from queue.
    if not person in searched:
      if person_is_seller(person):
        print(person + " is a mango seller!")
        return True
      else:
        search_queue += graph[person] # Add all of this person's friends to the search queue.
        searched.append(person)
  return False

# This function checks whether the person's name ends with the letter m. If it does, they're a mango seller.
def person_is_seller(name):
    return name[-1] == 'm'

search("you")
```