

Candidate: Quan-Ha Le.

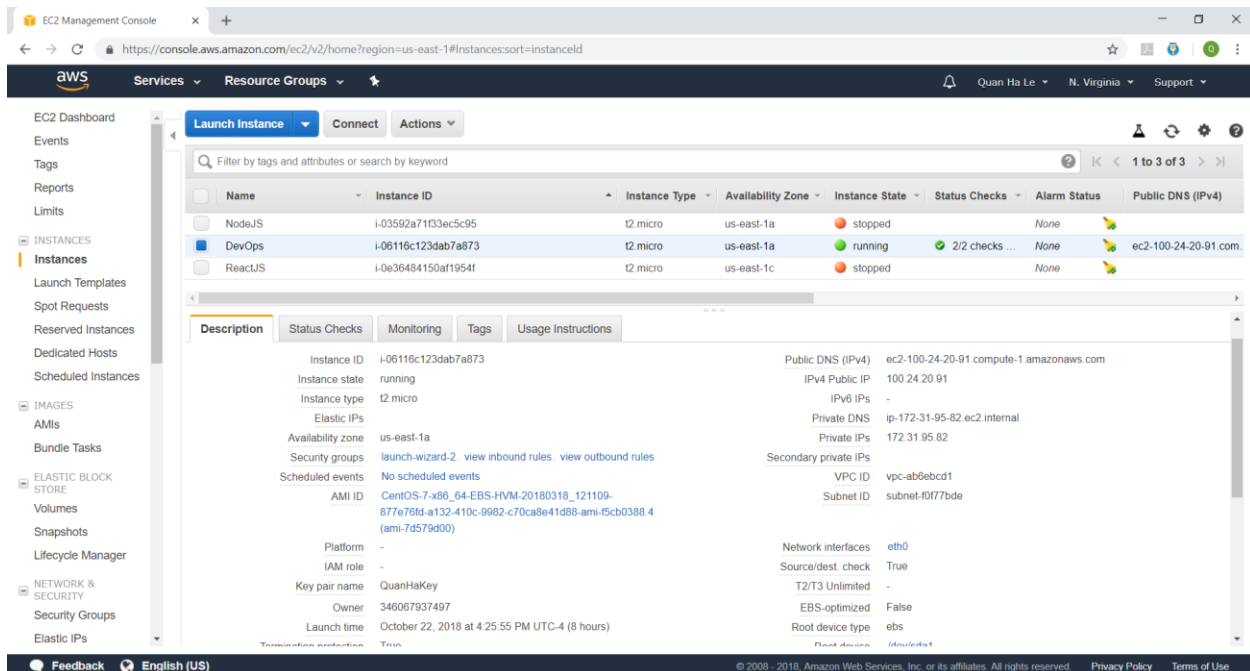
A. How to setup

- Python 3.7 for CentOS 7.5 on Amazon Web Services (AWS)
- I am using the Python language version of JUnit by Kent Beck and Erich Gamma - that is the Python unit testing framework.
- My GITHUB link for the Kiwiland Railroad is <https://github.com/lequanha/Kiwiland>

The AMI that I used to deploy my CentOS ec2 instance is
CentOS-7-x86_64-EBS-HVM-20180318_121109-877e76fd-a132-410c-9982-c70ca8e41d88-ami-f5cb0388.4 (ami-7d579d00)

From the above AMI, I installed Python 3.7 for CentOS 7.5, the steps please refer to the file inside my GITHUB: [Python3forCentOS.txt](#)

This is my screenshot for the ec2 instance on AWS



B. How to deploy

- From inside your putty session, please make a GIT CLONE

git clone <https://github.com/lequanha/Kiwiland.git>

This is the screenshot

```
root@ip-172-31-95-82:~/django
[root@ip-172-31-95-82 django]# git clone https://github.com/lequanha/Kiwiland.git
Cloning into 'Kiwiland'...
remote: Enumerating objects: 58, done.
remote: Counting objects: 100% (58/58), done.
remote: Compressing objects: 100% (45/45), done.
remote: Total 58 (delta 26), reused 40 (delta 12), pack-reused 0
Unpacking objects: 100% (58/58), done.
[root@ip-172-31-95-82 django]#
```

- Please use this statement to list the content

ls -lRl

This is the screenshot

```
root@ip-172-31-95-82:~/django
[root@ip-172-31-95-82 django]# ls -lRl
.:
total 0
drwxr-xr-x. 5 root root 166 Oct 23 05:40 Kiwiland

./Kiwiland:
total 732
-rw-r--r--. 1 root root 2541 Oct 23 05:40 Kiwiland Problem.txt
-rw-r--r--. 1 root root 512 Oct 23 05:40 Python3forCentOS.txt
-rw-r--r--. 1 root root 475136 Oct 23 05:40 QuanHaLe_documentation.doc
-rw-r--r--. 1 root root 265336 Oct 23 05:40 QuanHaLe_documentation.pdf
drwxr-xr-x. 3 root root 83 Oct 23 05:40 src
drwxr-xr-x. 3 root root 156 Oct 23 05:40 tests

./Kiwiland/src:
total 16
-rw-r--r--. 1 root root 3121 Oct 23 05:40 dijkstras.py
-rw-r--r--. 1 root root 1 Oct 23 05:40 __init__.py
drwxr-xr-x. 2 root root 100 Oct 23 05:40 __pycache__
-rw-r--r--. 1 root root 5598 Oct 23 05:40 railroad.py

./Kiwiland/src/__pycache__:
total 12
-rw-r--r--. 1 root root 1676 Oct 23 05:40 dijkstras.cpython-37.pyc
-rw-r--r--. 1 root root 124 Oct 23 05:40 __init__.cpython-37.pyc
-rw-r--r--. 1 root root 3863 Oct 23 05:40 railroad.cpython-37.pyc

./Kiwiland/tests:
total 20
-rw-r--r--. 1 root root 1417 Oct 23 05:40 dijkstrasmodule.py
-rw-r--r--. 1 root root 1362 Oct 23 05:40 getdistancemodule.py
-rw-r--r--. 1 root root 2465 Oct 23 05:40 gettripsnumbermodule.py
-rw-r--r--. 1 root root 101 Oct 23 05:40 __init__.py
drwxr-xr-x. 2 root root 238 Oct 23 05:40 __pycache__
-rw-r--r--. 1 root root 2614 Oct 23 05:40 test_receptiviti.py

./Kiwiland/tests/__pycache__:
total 24
-rw-r--r--. 1 root root 2338 Oct 23 05:40 dijkstrasmodule.cpython-37.pyc
-rw-r--r--. 1 root root 2296 Oct 23 05:40 getdistancemodule.cpython-37.pyc
-rw-r--r--. 1 root root 3792 Oct 23 05:40 gettripsnumbermodule.cpython-37.pyc
-rw-r--r--. 1 root root 236 Oct 23 05:40 __init__.cpython-37.pyc
-rw-r--r--. 1 root root 2768 Oct 23 05:40 test_insightglobal.cpython-37.pyc
-rw-r--r--. 1 root root 2742 Oct 23 05:40 test_receptiviti.cpython-37.pyc
[root@ip-172-31-95-82 django]#
```

- Please use the available MS Word and/or PDF files for your guidelines

QuanHaLe_documentation.doc

QuanHaLe_documentation.pdf

C. How to execute

To run the 10 tests provided by Receptiviti, please execute

```
# cd Kiwiland
```

```
# python3.7 -m unittest
```

```
#####
```

```
The route A-B-C has the distance of 9.0
```

```
.
The route A-D has the distance of 5.0
```

```
.
The route A-D-C has the distance of 13.0
```

```
.
The route A-E-B-C-D has the distance of 22.0
```

```
.
A-E-D: NO SUCH ROUTE
```

```
.
The number of possible trips from C to C with maximum of 3 stops are: 2
```

```
.
The number of possible trips from A to C with exact 4 stops are: 3
```

```
.
The shortest route from A to C is: ABC with distance 9.0
```

```
.
The shortest route from B to B is: BCEB with distance 9.0
```

```
.
The number of possible trips from C to C with maximum distance of 30 are: 7
```

```
.
```

```
-----
Ran 10 tests in 0.002s
```

```
OK
```

```
#####
```

This is the screenshot

```
root@ip-172-31-95-82:~/django/Kiwiland
[root@ip-172-31-95-82 django]# cd Kiwiland
[root@ip-172-31-95-82 Kiwiland]# python3.7 -m unittest

The route A-B-C has the distance of 9.0
.
The route A-D has the distance of 5.0
.
The route A-D-C has the distance of 13.0
.
The route A-E-B-C-D has the distance of 22.0
.
A-E-D: NO SUCH ROUTE
.
The number of possible trips from C to C with maximum of 3 stops are: 2
.
The number of possible trips from A to C with exact 4 stops are: 3
.
The shortest route from A to C is: ABC with distance 9.0
.
The shortest route from B to B is: BCEB with distance 9.0
.
The number of possible trips from C to C with maximum distance of 30 are: 7
.
-----
Ran 10 tests in 0.002s

OK
[root@ip-172-31-95-82 Kiwiland]#
```

The default test set's outputs above are the tests required by Receptiviti company.

To run interactively, execute `python3.7 -i` in the 'Kiwiland' directory. You can then enter below commands

```
import src.railroad as rr

my_routes = rr.Railroad("AB5, BC4, CD8, DC8, DE6, AD5, CE2, EB3, AE7")

my_routes.get_distance("A-B-C")

my_routes.get_distance("A-D")

my_routes.get_distance("A-D-C")

my_routes.get_distance("A-E-B-C-D")

my_routes.get_distance("A-E-D")

my_routes.get_number_of_possible_trips("C", "C", 3, my_routes.TripType.max_stops)

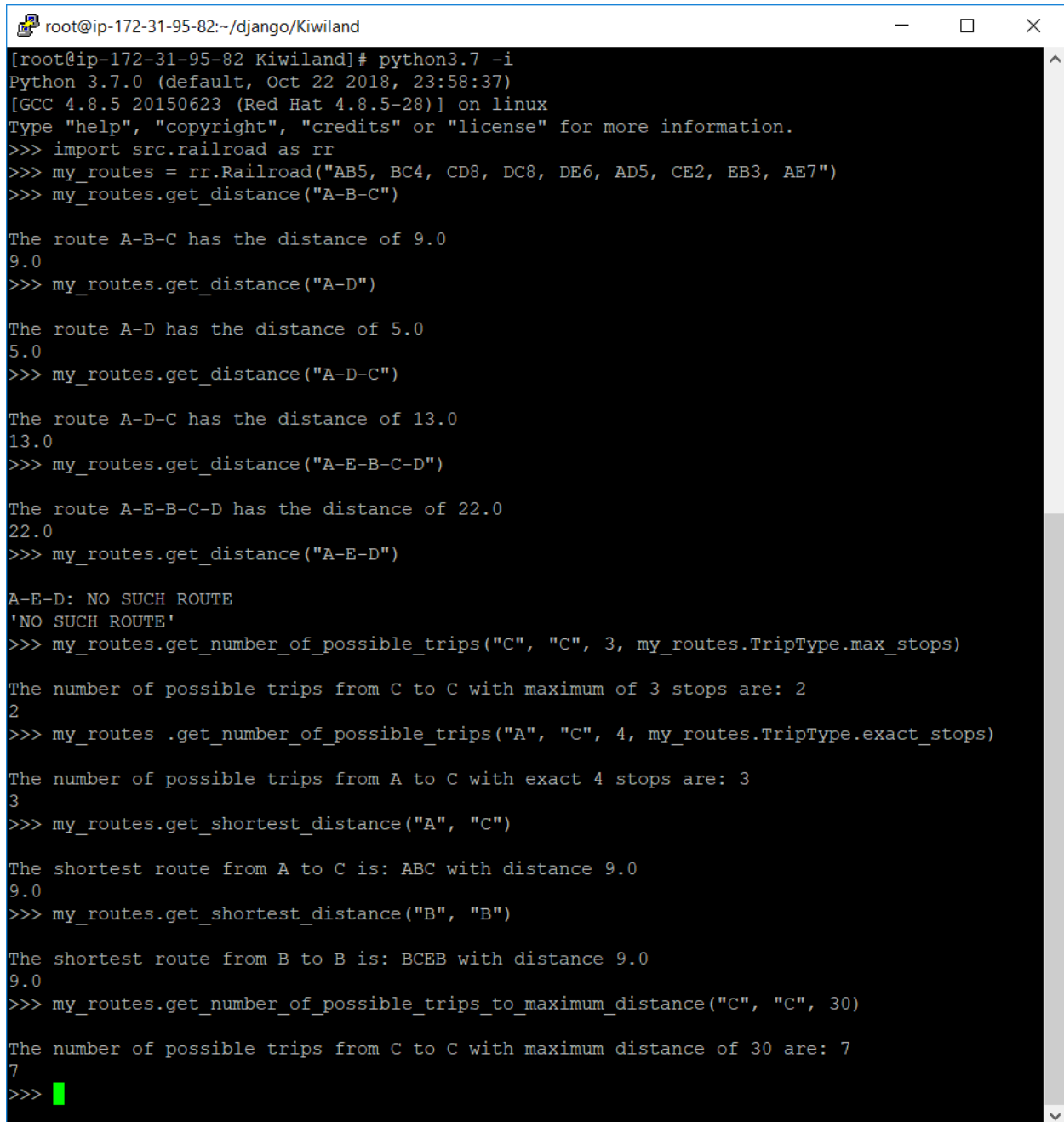
my_routes.get_number_of_possible_trips("A", "C", 4, my_routes.TripType.exact_stops)
```

```
my_routes.get_shortest_distance("A", "C")
```

```
my_routes.get_shortest_distance("B", "B")
```

```
my_routes.get_number_of_possible_trips_to_maximum_distance("C", "C", 30)
```

please look at this screenshot



```
root@ip-172-31-95-82:~/django/Kiwiland
[root@ip-172-31-95-82 Kiwiland]# python3.7 -i
Python 3.7.0 (default, Oct 22 2018, 23:58:37)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import src.railroad as rr
>>> my_routes = rr.Railroad("AB5, BC4, CD8, DC8, DE6, AD5, CE2, EB3, AE7")
>>> my_routes.get_distance("A-B-C")

The route A-B-C has the distance of 9.0
9.0
>>> my_routes.get_distance("A-D")

The route A-D has the distance of 5.0
5.0
>>> my_routes.get_distance("A-D-C")

The route A-D-C has the distance of 13.0
13.0
>>> my_routes.get_distance("A-E-B-C-D")

The route A-E-B-C-D has the distance of 22.0
22.0
>>> my_routes.get_distance("A-E-D")

A-E-D: NO SUCH ROUTE
'NO SUCH ROUTE'
>>> my_routes.get_number_of_possible_trips("C", "C", 3, my_routes.TripType.max_stops)

The number of possible trips from C to C with maximum of 3 stops are: 2
2
>>> my_routes.get_number_of_possible_trips("A", "C", 4, my_routes.TripType.exact_stops)

The number of possible trips from A to C with exact 4 stops are: 3
3
>>> my_routes.get_shortest_distance("A", "C")

The shortest route from A to C is: ABC with distance 9.0
9.0
>>> my_routes.get_shortest_distance("B", "B")

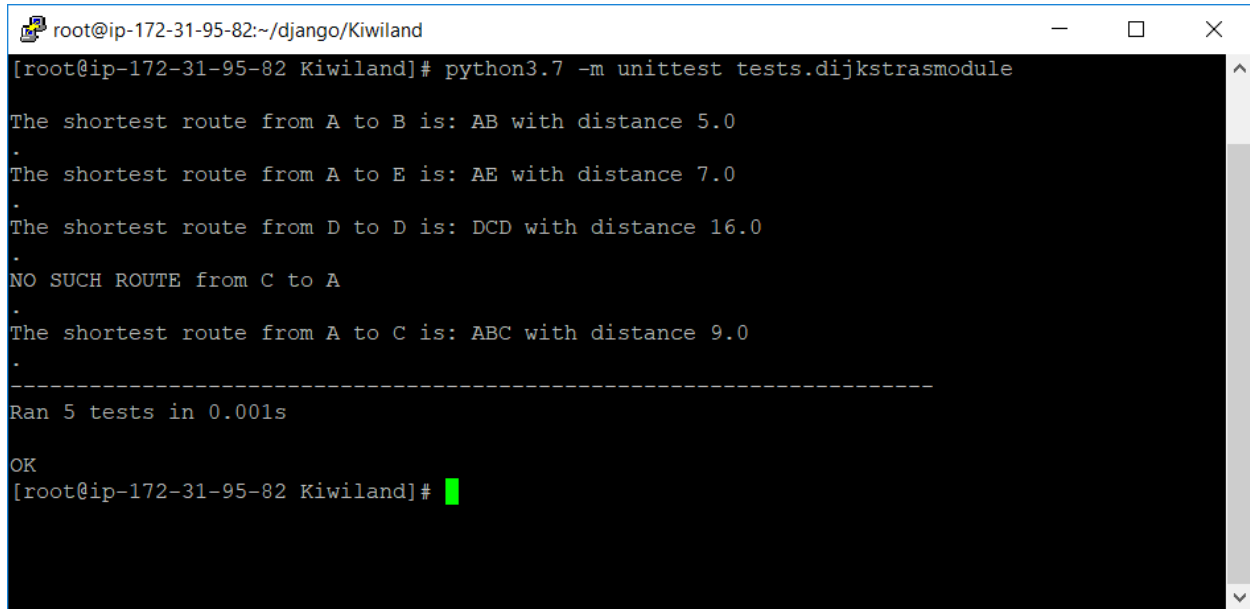
The shortest route from B to B is: BCEB with distance 9.0
9.0
>>> my_routes.get_number_of_possible_trips_to_maximum_distance("C", "C", 30)

The number of possible trips from C to C with maximum distance of 30 are: 7
7
>>> █
```

Other unit tests that I have also set up

- This is to test Dijkstras' Algorithm

```
# python3.7 -m unittest tests.dijkstrasmodule
```

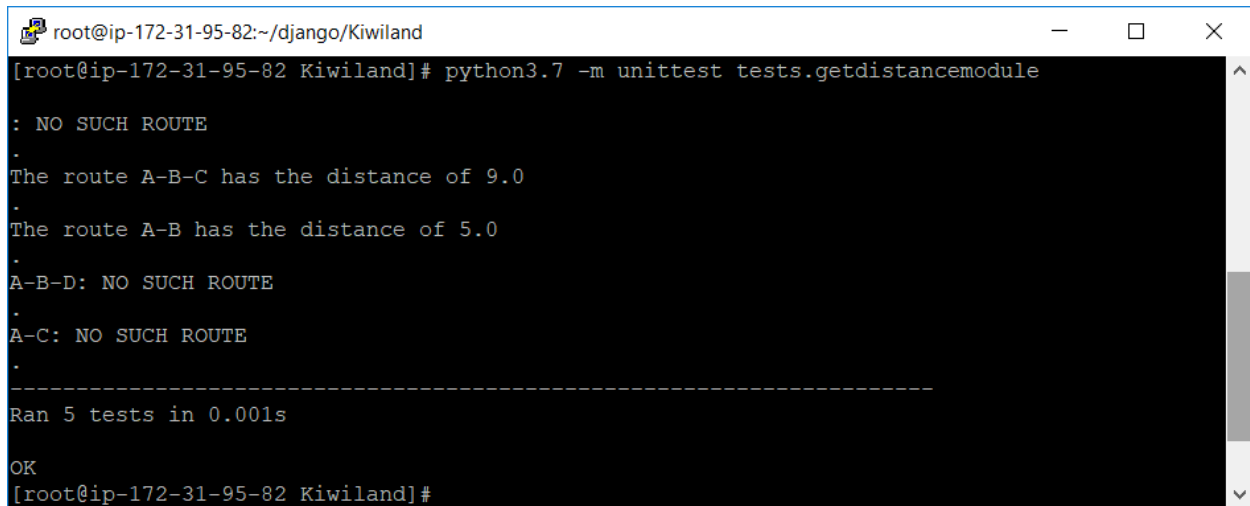


```
root@ip-172-31-95-82:~/django/Kiwiland
[root@ip-172-31-95-82 Kiwiland]# python3.7 -m unittest tests.dijkstrasmodule
The shortest route from A to B is: AB with distance 5.0
.
The shortest route from A to E is: AE with distance 7.0
.
The shortest route from D to D is: DCD with distance 16.0
.
NO SUCH ROUTE from C to A
.
The shortest route from A to C is: ABC with distance 9.0
.
-----
Ran 5 tests in 0.001s

OK
[root@ip-172-31-95-82 Kiwiland]#
```

- This is to test the get_distance method

```
# python3.7 -m unittest tests.getdistancemodule
```



```
root@ip-172-31-95-82:~/django/Kiwiland
[root@ip-172-31-95-82 Kiwiland]# python3.7 -m unittest tests.getdistancemodule
: NO SUCH ROUTE
.
The route A-B-C has the distance of 9.0
.
The route A-B has the distance of 5.0
.
A-B-D: NO SUCH ROUTE
.
A-C: NO SUCH ROUTE
.
-----
Ran 5 tests in 0.001s

OK
[root@ip-172-31-95-82 Kiwiland]#
```

- This is to test the get_number_of_possible_trips method and the get_number_of_possible_trips_to_maximum_distance method

```
# python3.7 -m unittest tests.gettripsnumbermodule
```

```
root@ip-172-31-95-82:~/django/Kiwiland
[root@ip-172-31-95-82 Kiwiland]# python3.7 -m unittest tests.gettripsnumbermodule

The number of possible trips from A to C with maximum distance of 19 are: 5
.
The number of possible trips from E to B with maximum distance of 0 are: 0
.
The number of possible trips from E to A with maximum distance of 100 are: 0
.
The number of possible trips from A to A with exact 3 stops are: 0
.
The number of possible trips from E to D with exact 4 stops are: 0
.
The number of possible trips from A to C with exact 2 stops are: 2
.
The number of possible trips from A to A with maximum of 3 stops are: 0
.
The number of possible trips from E to D with maximum of 2 stops are: 0
.
The number of possible trips from E to D with maximum of 5 stops are: 2
.
-----
Ran 9 tests in 0.004s

OK
[root@ip-172-31-95-82 Kiwiland]#
```

D. Technical implementation

1. I implement on Python the Dijkstra's Algorithm

Dijkstra's Algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, railroad networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.

The algorithm exists in many variants; Dijkstra's original variant found the shortest path between two nodes, but I would like to apply the most common variant fixes each single node as the "start" node and finds shortest paths from the start node to all other reachable nodes in the graph, producing a shortest-path dictionary.

The source code for Dijkstra's Algorithm is \Kiwiland\src\dijkstras.py

Let the node at which we are starting be called the start node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

- Step 1. Create a set named visit_nodes and loop through all the nodes, initialize the visit_nodes with the start node only.
- Step 2. Loop through the unvisited nodes inside the visit_nodes set
- Step 3. Take a next_node
- Step 4. Append all neighbour nodes of the above next_node into the visit_nodes set and calculate or re- update the distances from the start node to each neighbour node.

- Step 5. Store the minimum route inside the previous_nodes list
- Step 6. Store the minimum distance inside the shortest_routes list
- Step 7. End Loop when you already visited all the visit_nodes set and you have no more reachable nodes to add into the visit_nodes set.

2. The Railroad Graph

The source code is \Kiwiland\src\railroad.py

3. The Receptiviti Test.

The source code is \Kiwiland\tests\test_receptiviti.py

I apply TestSuite and this is the default 10 tests required by Receptiviti company as belows

1. The distance of the route A-B-C.
2. The distance of the route A-D.
3. The distance of the route A-D-C.
4. The distance of the route A-E-B-C-D.
5. The distance of the route A-E-D.
6. The number of trips starting at C and ending at C with a maximum of 3 stops.
7. The number of trips starting at A and ending at C with exactly 4 stops.
8. The length of the shortest route (in terms of distance to travel) from A to C.
9. The length of the shortest route (in terms of distance to travel) from B to B.
10. The number of different routes from C to C with a distance of less than 30.

Test Input: Graph: AB5, BC4, CD8, DC8, DE6, AD5, CE2, EB3, AE7

This input is stored inside this file \Kiwiland\tests__init__.py, if you want to change the graph input of all tests, please change this line inside \Kiwiland\tests__init__.py

```
Kiwiland = Railroad("AB5, BC4, CD8, DC8, DE6, AD5, CE2, EB3, AE7")
```

4. The Dijkstra's Algorithm Test

The source code is \Kiwiland\tests\dijkstrasmodule.py

I apply TestSuite and there are 5 tests to find the shortest routes by Dijkstra's Algorithm.

5. The Get Distance Test

The source code is \Kiwiland\tests\getdistancemodule.py

I apply TestSuite and there are 5 tests to find the distance of specific routes.

6. The Get Number of Possible Trip Test

The source code is \Kiwiland\tests\gettripsnumbermodule.py

I apply TestSuite and there are 9 tests to find the number of possible trips from a departure_city to a destination_city.