```
using System.Collections;
using System.Collections.Generic;
using NUnit.Framework;
using UnityEngine;
using UnityEngine.TestTools;

namespace Tests
{
    public class UnitTest
    {


        [Test]
        public void TestPQ() {
            // create a priority queue
            PriorityQueue testQueue = new PriorityQueue();
            // create two nodes
            Node q1 = new Node("I am number 1");
            Node q2 = new Node("I am number 2");
            // insert a node with priority 2
            testQueue.enqueue(q2,2);
            testQueue.enqueue(q1,1);
            // call dequeue and save the result into a variable
            int result = testQueue.dequeue().Item2;
            // even though we inserted q2 first, because q1 has higher priority,
            // it should still be returned to us
            // now assert result equals to 1
            Assert.AreEqual(1,result);
        }
        /*
        This unit test is very similiar to the test for distance
        Since we are doing unit testing, its good idea to test for different
        things in a separate function
        */
        [Test]
        public void TestPathEqualToTarget() {
            Graph g = new Graph();
            g.addVertex("Start");
            g.addVertex("Finish");
            g.addVertex("Intermediate");
            g.addEdge(g.getNodeByValue("Start"), g.getNodeByValue("Finish"), 10);
            g.addEdge(g.getNodeByValue("Start"), g.getNodeByValue("Intermediate"),
5);
            g.addEdge(g.getNodeByValue("Intermediate"), g.getNodeByValue("Finish"),
4);
            // Now since we are after the shortest path - which is a collection of
Nodes, we are going to use
            // shortestPathBetween function and check if our list is equals to
start-intermediate-finish
            List<Node> shortestPath = Algorithm.findShortestPath(g,
g.getNodeByValue("Start"), g.getNodeByValue("Finish"));
            List<Node> target = new List<Node>() {g.getNodeByValue("Start"),
g.getNodeByValue("Intermediate"), g.getNodeByValue("Finish")};
            CollectionAssert.AreEqual(shortestPath, target);
        }

        /*
        This unit test's aim is just to test the functionality
        of Algorithm we are using for shortest path calculations.
        The test should pass given a graph with collection of nodes
```

```
        Note this test does not use the actual graph in the game since
        we are going to test that separately in Integration testing when
        we want to see if Model-View-Controller are working correctly
        */
        [Test]
        public void TestDistanceEqualToTarget()
        {
            // the truth asserts if final calculated distance
            // should equals to distance returned by algorithm
            // thus proving algorithm return the correct distance
            // initialise a graph
            Graph g = new Graph();
            // add three vertex into the graph, since we only wants to find out
            // if algorithm return correct shortest path distance
            g.addVertex("Start");
            g.addVertex("Finish");
            g.addVertex("Intermediate");
            // now we are going to add edges, to test the algorithm, we test the
case
            // when path going through intermediate is shorter than the direct path
            // between start and finish
            g.addEdge(g.getNodeByValue("Start"), g.getNodeByValue("Finish"), 10);
            g.addEdge(g.getNodeByValue("Start"), g.getNodeByValue("Intermediate"),
5);
            g.addEdge(g.getNodeByValue("Intermediate"), g.getNodeByValue("Finish"),
4);
            // Now we can calculate the shortest distance between
            // start and finish which we know prior hand is 9
            Dictionary<Node, int> shortestDistanceDictionary =
Algorithm.dijkstra(g, g.getNodeByValue("Start")).shortestDistanceEstimate;
            int shortestDistance =
shortestDistanceDictionary[g.getNodeByValue("Finish")];
            int expectedResult = 9;
            // use assert function to check if shortestDistane equals to 9
            Assert.AreEqual(shortestDistance, expectedResult);
        }



    }
}

```