DISEÑO DE PRUEBAS UNITARIAS:

Crearemos un objeto de la clase grafo para poder testear sus respectivos métodos. Posteriormente crearemos los setUp1,2,3 y 4 para tener escenarios variados a la hora de probar los diferentes métodos. En el setUp1 inicializamos el grafo con lista de adyacencia y agregamos un nodo con valor 19. En el setUp2 solo inicializamos el grafo con lista de adyacencia. En el setUp3 inicializamos el grafo con lista de adyacencia y agregamos 5 nodos con valores 19, 3, 1, 25 y 6, también agregamos un edge entre el 25 y 6.

En el setUp4 inicializamos el grafo con lista de adyacencia y agregamos 3 nodos con valores 1, 2, 3, tambien agregamos un edge entre 1 y 2, 2 y 3 y 3 y 1.

Tests de añadir vertex:

addVertexTest1(): En este test haremos uso del setUp1 y agregaremos un nodo con nombre nodo1 y valor 10000 al grafo. Realizaremos un assertEquals, buscamos el nodo con valor 10000 y nos debe devolver el nodo1.

addVertexTest2(): En este test haremos uso del setUp2, agregamos un nodo y al buscarlo con el assertEquals este debería devolvernos el mismo.

addVertexTest3(): En este test usaremos el setUp3, agregamos un nodo con nombre nodo3, usamos el assertEquals y este debe devolvernos el ultimo nodo que añadimos. Con estos tres test en tres diferentes escenarios probamos que el metodo addVertex funciona correctamente.

Tests de añadir edge:

addEdgeTest1(): En este test usaremos el setUp3, usaremos el assertNull y buscaremos un edge que no exista en dicho escenario, el test debería pasar correctamente.

addEdgeTest2(): En este test usaremos el setUp3, agregamos un edge entre los nodos con valores 1 y 6, usaremos un assertEquals que debería dar igual al nodo 6, dado que cuando buscamos el edge nos devuelve el nodo con valor 6 que es el que esta “conectado” al nodo con valor 1.

addEdgeTest3(): En este test usaremos el setUp3, agregaremos un edge entre dos nodos con valores 19 y 25 al buscar el edge entre estos dos debería darnos el valor del nodo 25 por tanto con un assertEquals nos aseguramos de que sea así y el test debería pasar.

Tests de buscar vertex:

searchVertexTest1(): En este test usaremos el setUp3, con un assertNull buscaremos un nodo con un valor inexistente en el escenario cómo 99, a pesar de ser el escenario con más nodos agregados el test de nulo debería pasar correctamente dado que no existe ningún nodo con dicho valor

searchVertexTest2(): En este test utilizaremos el setUp2, agregaremos un nodo con valor 1 y lo buscaremos en un assertEquals, el resultado debería ser el nodo con valor 1 y por tanto el test funciona correctamente.

searchVertexTest3(): En este test utilizaremos el setUp3, agregamos un nodo con un valor y una posición altos como 87 y 100 los buscaremos en uno de los escenarios con mas nodos agregados y el resultado del assertEquals debería ser el nodo con valor 87, por tanto el test funciona correctamente.

Tests search edge:

searchEdgeTest1():En este test utilizaremos el setUp2 y buscaremos un edge inexistente en el escenario, dado que trabajaremos este test con un assertNull, el test debería funcionar correctamente.

searchEdgeTest1(): En este test utilizaremos el setUp3, agregaremos un edge entre los nodos con valores 1 y 19, con un assertNull buscaremos un edge entre los nodos con valores 1 y 9999, dado que no existe ese edge el test funciona correctamente.

searchEdgeTest1():En este test haremos uso del setUp3, agregaremos un edge entre los nodos con valores 1 y 25 y realizaremos la búsqueda en el assertEquals, que deberá ser igual al nodo con valor 25 ya que es el nodo conectado al 1.

Tests BFS:

bfsTest1(): En este test haremos uso del escenario setUp4 que contiene un grafo fuertemente conexo, cuando realizamos un assertTrue del metodo bfs en el grafo deberia darnos como correcto dado a las conexiones del grafo en el escenario setUp4 y el test deberia pasar.

bfsTest2(): En este test utilizaremos el escenarios setUp1, dado que este escenario cuenta con un único nodo con valor 19, al utilizar el método bfs debería arrojarnos un true, en el test utilizaremos un asserTtrue y el test debería pasar correctamente.

bfsTest3(): En este test usaremos el escenario setUp3, como este escenario no tiene un grafo fuertemente conexo, utilizaremos un assertFalse y al realizar el bfs debería soltarnos un false, por lo que el test funciona correctamente.

Tests Floyd-Warshall:

floydWarshallTest1():En este test usaremos el escenario setUp4, este escenario cuenta con 3 nodos distintos, al realizar el floyd Warshall debería darnos un camino con length igual a 3, nos aseguramos de esto con un assertEquals y el test debería funcionar.

floydWarshallTest2(): En este test usaremos el escenario setUp1, este escenario tiene un unico nodo con valor de 19, cuando hacemos el floyd Warshall el programa debería devolvernos un camino con length igual a 1, con un assertEquals a 1 el test debería pasar.

floydWarshallTest3(): En este test usaremos el escenario setUp3, en este escenario agregamos un total de 5 nodos al inicio de los tests, por tanto el length del camino al implementar el algoritmos de floyd Warshall debería ser de 5, con un assertEquals nos aseguramos de esto y el test pasaría.

Tests de dfs:

dfsTest1(): En este test usamos el setUp4, y con el algoritmo de dfs nos debería devolver 1, utilizamos un assertEquals y el test pasa correctamente.

dfsTest2(): En este test utilizamos el setUp2, cuando aplicamos el algoritmo del bfs, este nos deberia dar como resultado 0, por tanto utilizaremos un assertEquals para comprobarlo, el test funciona correctamente.

dfsTest3(): En este test usaremos el setUp1, y con el algoritmo de dfs nos debería devolver 1, con assertEquals nos aseguramos que eso suceda, y el test debería pasar sin problemas.

Tests de delete graph

deleteGraphTest1(): En este test usaremos el setUp3, borraremos todo el grafo, cuando utilizamos el método deleteGraph y el grafo no está vacío, borra el arreglo y nos devuelve un true. Utilizamos un assertTrue para verificar que lo anterior es cierto y el test pasa correctamente.

deleteGraphTest2(): En este test utilizaremos el setUp2, en este escenario no hemos agregado ningún nodo por lo que el grafo se encuentra vacío, para verificar que el delete funciona no únicamente con los grafos llenos sino también con los vacios, utilizaremos un assertFalse dado que el método nos debería devolver un false a la hora de intentar borrar un grafo vacío.

deleteGraphTest3(): En este test usaremos el setUp1, en este escenario tenemos solo un nodo, lo cual es suficiente para que cuando usemos el método de deleteGraph, nos borre el grafo y nos devuelve un true. Utilizaremos un assertTrue y el test pasa correctamente.