|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| Grafo | agregarTest() | stage1()  [ | Nodo n1= new Nodo(25, “Josh”)  Nodo n2= new Nodo(21, “Nelson”)  Nodo n3= new Nodo(32, “Chaux”)  Nodo n4= new Nodo(15, “Charles”)  Nodo n5= new Nodo(2, “John”)  Nodo n6= new Nodo(3, “Karl”)  Nodo n7= new Nodo(4, “Julian”)  Nodo n8= new Nodo(9, “Steven”)  Nodo n9= new Nodo(1, “Andrew”) | Verdadero. Luego de ejecutar el método agregar 9 veces, el tamaño de la lista de adyacencia es efectivamente 9. |
| Grafo | agregarTest() | stage2() | Nodo n1= new Nodo(25, “Josh”) | Verdadero. El tamaño de la lista de adyacencia aumentó en una unidad. |
| Grafo | eliminarTest() | stage1() | Nodo n1= new Nodo(25, “Josh”)  Nodo n2= new Nodo(21, “Nelson”)  Nodo n3= new Nodo(32, “Chaux”)  Nodo n4= new Nodo(15, “Charles”)  Nodo n5= new Nodo(2, “John”)  Nodo n6= new Nodo(3, “Karl”)  Nodo n7= new Nodo(4, “Julian”)  Nodo n8= new Nodo(9, “Steven”)  Nodo n9= new Nodo(1, “Andrew”) | Verdadero. Luego de ejecutar el método agregar 9 veces, se eliminaron tres nodos y efectivamente, el tamaño de la lista de adyacencia es 6. |
| Grafo | eliminarTest() | stage3() | Nodo n1= new Nodo(25, “Josh”)  Nodo n2= new Nodo(21, “Nelson”)  Nodo n3= new Nodo(32, “Chaux”)  Nodo n4= new Nodo(15, “Charles”)  Nodo n5= new Nodo(2, “John”)  Nodo n6= new Nodo(3, “Karl”)  Nodo n7= new Nodo(4, “Julian”) | Verdadero. Se agregan 7 nodos, se eliminan dos, luego se compara el número de nodos antes y después y esta diferencia es igual a 2. |
| Grafo | consultarTest() | stage1() | Nodo n1= new Nodo(25, “Josh”)  Nodo n2= new Nodo(21, “Nelson”)  Nodo n3= new Nodo(32, “Chaux”)  Nodo n4= new Nodo(15, “Chasqui”)  Nodo n5= new Nodo(2, “John”)  Nodo n6= new Nodo(3, “Karl”)  Nodo n7= new Nodo(4, “Julian”)  Nodo n8= new Nodo(9, “Steven”)  Nodo n9= new Nodo(1, “Andrew”) | Verdadero. Al buscar un nodo correspondiente al nombre “Chasqui”, este fue encontrado dado que coincidían el nombre buscado y el del nodo hallado. |
| Grafo | consultarTest() | stage1() | Nodo n1= new Nodo(25, “Josh”)  Nodo n2= new Nodo(21, “Nelson”)  Nodo n3= new Nodo(32, “Chaux”)  Nodo n4= new Nodo(15, “Charles”)  Nodo n5= new Nodo(2, “John”)  Nodo n6= new Nodo(3, “Karl”)  Nodo n7= new Nodo(4, “Julian”) | Verdadero. El nodo con el valor “Josh” tiene asociado al nodo con el valor “Charles”. |
| Grafo | dijkstra() | stage4() | Nodo raiz =new Nodo(20, “John”)  Nodo n1= new Nodo(25, “Josh”)  Nodo n2= new Nodo(21, “Nelson”)  Nodo n3= new Nodo(32, “Chaux”)  Nodo n4= new Nodo(15, “Charles”) | Verdadero. El camino más corto entre el nodo raíz y el nodo n1 está dado por una linkedList igual a la linkedList ingresada como parámetro, en el que se describe el camino a seguir mediante el orden de los nodos dentro de la estructura. |
| Grafo | Bellman-Ford() | stage4() | Nodo raiz =new Nodo(20, “John”)  Nodo n1= new Nodo(25, “Josh”)  Nodo n2= new Nodo(21, “Nelson”)  Nodo n3= new Nodo(32, “Chaux”)  Nodo n4= new Nodo(15, “Charles”) | Verdadero. El camino mínimo entre el nodo raíz y el nodo n4 está dado por una lista igual a la lista ingresada como parámetro, en la que se describe el camino a seguir mediante el orden de los nodos dentro de la estructura. |
| Grafo | Floyd-Warshall() | stage4() | Nodo raiz =new Nodo(20, “John”)  Nodo n1= new Nodo(25, “Josh”)  Nodo n2= new Nodo(21, “Nelson”)  Nodo n3= new Nodo(32, “Chaux”)  Nodo n4= new Nodo(15, “Charles”)  Nodo n5= new Nodo(2, “Michael”)  Nodo n6= new Nodo(3, “Karl”) | Verdadero. El camino mínimo entre el nodo raíz y el nodo n4 está dado por una lista igual a la lista ingresada como parámetro, en la que se describe el camino a seguir mediante el orden de los nodos dentro de la estructura. |