**Pruebas Árbol Binario de Búsqueda**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba No. 1 | Objetivo de la prueba: Mostrar que se agregan elementos al árbol  Firma del método: public void insert(t) | | | |
| Clase para probar | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ABBTest | insert(T) | setupStage1, se crea un Arbol ABB vacío | 3 | Arbol ABB ¡= vacío |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba No. 2 | Objetivo de la prueba: Mostrar que se encuentran los valores buscando en el árbol  Firma del método: public Node<T> find(t) | | | |
| Clase para probar | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ABBTest | find(T) | setupStage2, se crea un Arbol ABB y se inserta los nodos con valores 5, 4, 6, 3, 7 | 7 | 7 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba No. 3 | Objetivo de la prueba: Mostrar que se elimina un elemento del árbol  Firma del método: public void delete(T) | | | |
| Clase para probar | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ABBTest | delete(T) | setupStage2 | 7 | Find(7) == null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba No. 4 | Objetivo de la prueba: Mostrar que dado un nodo se obtiene su sucesor  Firma del método: public Node<T> getSucesor(T) | | | |
| Clase para probar | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| ABBTest | getSucessor(T) | setupStage2 | 5 | 6 |

**Pruebas Árbol Roji-Negro**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba No. 1 | Objetivo de la prueba: Mostrar que se agregan elementos al árbol y este queda aproximadamente balanceado haciendo las debidas rotaciones y recoloraciones  Firma del método: public void insert(t) | | | |
| Clase para probar | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| RBTTest | insert(T) | setupStage1, se crea un Arbol RBT y se inserta los nodos con valores 5, 4, 6, 3, 7 | 8 | Find(7).color == BLACK && find(7).left == 6 && find(7).right == 8 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba No. 2 | Objetivo de la prueba: Mostrar que se encuentran los valores buscando en el árbol  Firma del método: public Node<T> find(t) | | | |
| Clase para probar | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| RBTTest | find(T) | setupStage1 | 7 | 7 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba No. 3 | Objetivo de la prueba: Mostrar que el árbol queda aproximadamente balanceado luego de eliminar un nodo negro, haciendo las debidas rotaciones y recoloraciones  Firma del método: public void delete(T) | | | |
| Clase para probar | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| RBTTest | delete(T) | setupStage2, se crea un Arbol RBT y se inserta los nodos con valores 5, 4, 6, 3, 7, 8 | 7 | Find(7) == null, root.left == 4 && root.left == 8 && find(8).color == BLACK |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba No. 4 | Objetivo de la prueba: Mostrar que dado un valor se obtiene su sucesor  Firma del método: public Node<T> getSucesor(T) | | | |
| Clase para probar | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| RBTTest | getSucessor(T) | setupStage2 | 5 | 6 |

**Pruebas AVL**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba No. 1 | Objetivo de la prueba: Mostrar que se agregan elementos al árbol y este queda balanceado haciendo las debidas rotaciones  Firma del método: public void insert(t) | | | |
| Clase para probar | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| AVLTest | insert(T) | setupStage1, se crea un Arbol RBT y se inserta los nodos con valores 5, 4, 6, 3, 7 | 8 | Find(7).balance == 0 && root.balance ==0 && find(7).right.key == 8 && find(7).left.key == 6 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba No. 2 | Objetivo de la prueba: Mostrar que se encuentran los valores buscando en el árbol  Firma del método: public Node<T> find(t) | | | |
| Clase para probar | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| AVLTest | find(T) | setupStage1 | 7 | 7 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba No. 3 | Objetivo de la prueba: Mostrar que el árbol queda balanceado luego de eliminar un nodo haciendo las debidas rotaciones  Firma del método: public void delete(T) | | | |
| Clase para probar | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| AVLTest | delete(T) | setupStage2, se crea un Arbol RBT y se inserta los nodos con valores 5, 4, 6, 3, 7, 8 | 7 | Find(7) == null && root.balance == 0 && root.right.balance == 1 && root.left.balance == -1 |