Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Санкт-Петербургский национальный исследовательский университет

ИТМО»

**Лабораторная работа №1**

Тестирование программного обеспечения

Вариант: 9109

Выполнили:

Кравец Р. Д.

Проверил: Егошин Алексей Васильевич

Санкт-Петербург

2023

**Задание:**

1. Для указанной функции провести модульное тестирование разложения функции в степенной ряд. Выбрать достаточное тестовое покрытие.  
2. Провести модульное тестирование указанного алгоритма. Для этого выбрать характерные точки внутри алгоритма, и для предложенных самостоятельно наборов исходных данных записать последовательность попадания в характерные точки. Сравнить последовательность попадания с эталонной.  
3. Сформировать доменную модель для заданного текста.  Разработать тестовое покрытие для данной доменной модели

**Вариант задания:**

1. Функция arcsin(x)
2. Программный модуль для работы с бинарным деревом (<http://www.cs.usfca.edu/~galles/visualization/BST.html>)
3. Описание предметной области:

Охранник обхватил их обоих за шеи и, почтительно поклонившись спине капитана, выволок с мостика, не обращая внимания на их сопротивление. Стальная дверь закрылась, и капитан снова остался один. Он задумчиво промурлыкал что-то и полистал свою записную книжку со стихами.

**Цель:**

Проверить корректность приблизительных вычислений функции arcsin(x) на промежутке с точностью 0.37.

**Методика:**

Изображение выглядит как линия, График, диаграмма, Параллельный

Автоматически созданное описание

Значения, в которых необходимо проверить функцию: – граничные значения, где функция меняет знак, а также переходит в неопределённость. Для достоверности возьмём также пару значений из промежутка . Итоговая таблица для проверки:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | -1.001 | -1 | -0.999 | -0.5 | -0.001 | 0 | 0.001 | 0.5 | 0.999 | 1 | 1.001 |
| Arcsin(x) | NaN | -pi/2 | -1.526 | -pi/6 | -0.001 | 0 | 0.001 | Pi/6 | 1.526 | Pi/2 | NaN |

Написанная функция использует разложение в степенной ряд для нахождения значений.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

**Вывод:**

За счет аппроксимации вычислений результаты функции немного отличаются от истинных, однако все они находятся в пределах заданной погрешности.

**Цель:**

Проверить корректность работы алгоритма поиска, добавления, удаления элемента в бинарном дереве. А также проверка правильного распределения узлов в бинарном дереве.

**Методика:**

В тестах происходит:

* Проверка, что элементы содержаться в бинарном дереве после их добавления
* @Test  
  void testAddElement(){  
   Tree tree = new Tree();  
   tree.insertNode(6);  
   tree.insertNode(12);  
    
   *assertTrue*(tree.containsNodeByValue(6));  
   *assertTrue*(tree.containsNodeByValue(12));  
  }
* Проверка размера дерева
* @Test  
  void testSizeOfTree() {  
   Tree tree = new Tree();  
   tree.insertNode(1);  
   tree.insertNode(2);  
   tree.insertNode(3);  
   tree.insertNode(4);  
   *assertEquals*(4, tree.size());  
  }
* проверка на правильность удаления элементов
* @Test  
  void testRemoveElement(){  
   Tree tree = new Tree();  
   tree.insertNode(12);  
   tree.insertNode(6);  
    
   *assertTrue*(tree.containsNodeByValue(6));  
   tree.deleteNode(6);  
   *assertFalse*(tree.containsNodeByValue(6));  
  }
* Проверка на поиск элемента.
* @Test  
  void testSearchElement() {  
   Tree tree = new Tree();  
   tree.insertNode(5);  
   tree.insertNode(8);  
    
   *assertEquals*(5, tree.findNodeByValue(5).getValue());  
   *assertEquals*(8, tree.findNodeByValue(8).getValue());  
   *assertFalse*(tree.containsNodeByValue(20));  
  }
* Проверка на правильное размещение левого листа дерева, которое соответствует минимальному значению
* @Test  
  public void testLeftChild() {  
   Node nodeRoot;  
   Node node;  
   Tree tree = new Tree();  
   tree.insertNode(4);  
   tree.insertNode(3);  
   tree.insertNode(5);  
   nodeRoot = tree.findNodeByValue(4);  
   node = tree.findNodeByValue(3);  
    
   *assertEquals*(node.getValue(), nodeRoot.getLeftChild().getValue());  
   *assertEquals*(null, node.getLeftChild());  
   *assertEquals*(null, node.getRightChild());  
  }
* Проверка на правильное размещение правого листа дерева, которое соответствует максимальному значению
* @Test  
  public void testRightChild() {  
   Node nodeRoot;  
   Node node;  
   Tree tree = new Tree();  
   tree.insertNode(4);  
   tree.insertNode(3);  
   tree.insertNode(5);  
   nodeRoot = tree.findNodeByValue(4);  
   node = tree.findNodeByValue(5);  
    
   *assertEquals*(node.getValue(), nodeRoot.getRightChild().getValue());  
   *assertEquals*(null, node.getLeftChild());  
   *assertEquals*(null, node.getRightChild());  
  }
* Проверка на правильно размещение корня дерева
* @Test  
  public void testRoot() {  
   Node nodeRoot;  
   Node nodeLeft;  
   Node nodeRight;  
   Tree tree = new Tree();  
   tree.insertNode(4);  
   tree.insertNode(3);  
   tree.insertNode(5);  
   nodeRoot = tree.findNodeByValue(4);  
   nodeLeft = nodeRoot.getLeftChild();  
   nodeRight = nodeRoot.getRightChild();  
    
   *assertEquals*(4, nodeRoot.getValue());  
   *assertEquals*(nodeLeft, nodeRoot.getLeftChild());  
   *assertEquals*(nodeRight, nodeRoot.getRightChild());  
  }

**Вывод:**

Алгоритм бинарного дерева работает корректно, т.к. все тестовые случаи выдают ожидаемое поведение.

**Цель:**

Проверить корректность поведения доменной модели, созданной из описания.

**Методика:**

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Тестовое покрытие для данной доменной модели:

* Проверка на возможность/невозможность тащить 2 людей. Если сумма их сил будет меньше, чем у человека, который тащит.
* Проверка на возможность/невозможность схватить 2 людей. Если половина сумма их сил будет меньше, чем у человека, который стащил.
* Проверка на максимальное и минимальное количество сил у человека
* Проверка на нормальное количество сил у человека, которое находится в в диапазоне от 0 до 100.
* Проверка закрытия открытой/закрытой двери

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

* Проверка на добавление новых/существующих людей в комнату
* Проверка на удаление людей из комнаты. Когда человек находится в комнате и когда его не существует там.
* Проверка на возможность чтения капитаном записок

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

**Вывод:**

Тесты не выявили некорректную работу доменной модели.

**Код:**

https://github.com/ArdQsi/ITMO\_6\_sem/tree/main/tpo/lab1

**Выводы:**

В ходе выполнения данной лабораторной работы я углубил свои знания Junit5 и написал юнит-тесты для разработанных классов. Также хочется отметить, что очень сложно добиться 100% покрытия тестами.