


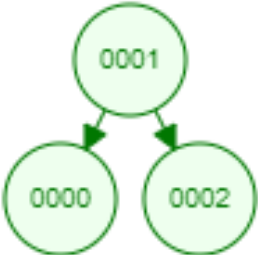
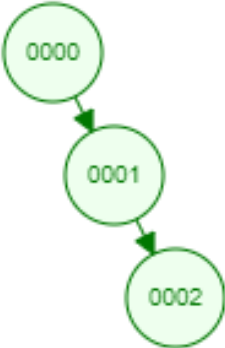
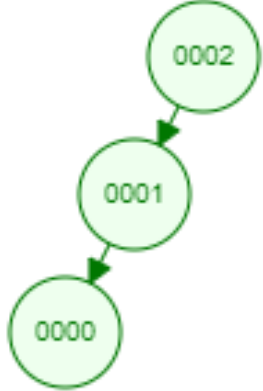
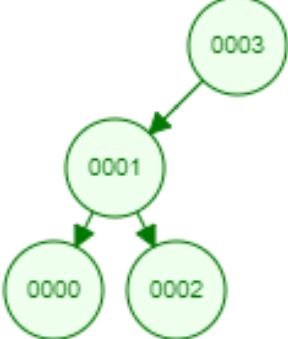
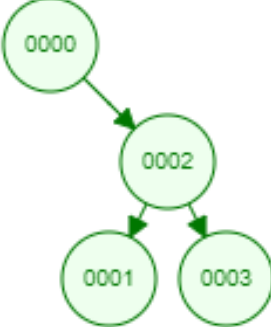
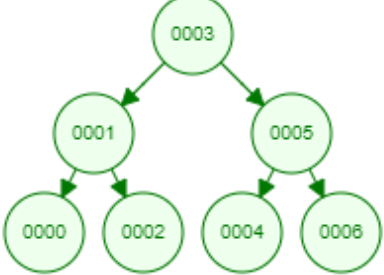
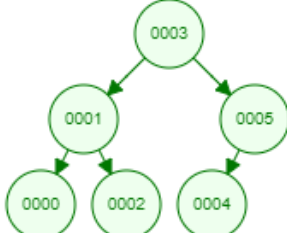



I ran all of my tests on these trees, using arrays populated with the necessary information.

Tree #	Pre-Order	Post-Order	In-Order	Symmetric	Max Imbalance	Visual Representation
0	0	0	0	true	0	
1	01	10	01	false	1	
2	10	01	01	false	1	
3	102	021	012	true	0	
4	012	210	012	false	2	

5	210	012	012	false	2	 <pre> graph TD 0002((0002)) --> 0001((0001)) 0001 --> 0000((0000)) </pre>
6	3102	0213	0123	false	2	 <pre> graph TD 0003((0003)) --> 0001((0001)) 0001 --> 0000((0000)) 0001 --> 0002((0002)) </pre>
7	0213	1320	0123	false	2	 <pre> graph TD 0000((0000)) --> 0002((0002)) 0002 --> 0001((0001)) 0002 --> 0003((0003)) </pre>
8	3102546	0214653	0123456	true	0	 <pre> graph TD 0003((0003)) --> 0001((0001)) 0003 --> 0005((0005)) 0001 --> 0000((0000)) 0001 --> 0002((0002)) 0005 --> 0004((0004)) 0005 --> 0006((0006)) </pre>
9	310254	021453	012345	false	1	 <pre> graph TD 0003((0003)) --> 0001((0001)) 0003 --> 0005((0005)) 0001 --> 0000((0000)) 0001 --> 0002((0002)) 0005 --> 0004((0004)) </pre>

Runs: 1/1 ✖ Errors: 0 ✖ Failures: 0



▶  BSTTester [Runner: JUnit 4] (0.000 s)

```
210
211 @Test
212 public void test()
213 {
214     testPreOrder();
215     testPostOrder();
216     testInOrder();
217     testIsSymmetric();
218     testMaxImbalance();
219     testReverse();
220 }
```

```
160 private void checkPre(int i)
161 {
162     Iterable<Integer> list = trees[i].preOrder();
163     int count = 0;
164     for (Integer j : list)
165     {
166         assertTrue(j.equals(pre[i][count++]));
167     }
168 }
169
170 private void checkPost(int i)
171 {
172     Iterable<Integer> list = trees[i].postOrder();
173     int count = 0;
174     for (Integer j : list)
175     {
176         assertTrue(j.equals(post[i][count++]));
177     }
178 }
179
180 private void checkIn(int i)
181 {
182     Iterable<Integer> list = trees[i].inOrder();
183     int count = 0;
184     for (Integer j : list)
185     {
186         assertTrue(j.equals(in[i][count++]));
187     }
188 }
189
190 private void testPreOrder()
191 {
192     for (int i = 0; i < 8; i++)
193     {
194         checkPre(i);
195     }
196 }
197
198 private void testPostOrder()
199 {
200     for (int i = 0; i < 8; i++)
201     {
202         checkPost(i);
203     }
204 }
205
206 private void testInOrder()
207 {
208     for (int i = 0; i < 8; i++)
209     {
210         checkIn(i);
211     }
212 }
```

```

215 private void testIsSymmetric()
216 {
217     for (int i = 0; i < 8; i++)
218     {
219         assertEquals(trees[i].isSymmetric(), symmetric[i]);
220     }
221 }
222
223
224 private void testMaxImbalance()
225 {
226     for (int i = 0; i < 8; i++)
227     {
228         assertEquals(trees[i].maxImbalance(), imbalance[i]);
229     }
230 }
231
232 private void testReverse()
233 {
234     for (int i = 0; i < 8; i++)
235     {
236         testReverse(trees[i]);
237     }
238 }
239
240 private void testReverse(BST tree)
241 {
242     Iterable<Integer> inOrder = tree.inOrder();
243
244     BST reversedTree = tree.reverse();
245
246     Iterable<Integer> reversed = reversedTree.inOrder();
247
248     Stack<Integer> reverser = new Stack<Integer>();
249     Queue<Integer> actualReversed = new Queue<Integer>();
250
251     for (Integer i : inOrder)
252     {
253         reverser.push(i);
254     }
255     while (!reverser.isEmpty())
256     {
257         actualReversed.enqueue(reverser.pop());
258     }
259
260     assertEquals(reversed.toString(), actualReversed.toString());
261 }
262
263 }

```