src/Tests/PartB_SMAStarTests.java

```
  1 packageimportstaticimportimportimportimportimportpublicclass
PartB_SMAStarTestsprivatefinalByteArrayOutputStreamoutContent=new
ByteArrayOutputStream@Testpublicvoidtest BasicPathfinding()newPrintStreamNode
goal=newNode190null0nullNodestart=newNode10null0intplanetSize=2intmemorySize=2
Stringfrontier_result="[(1:0)1.414]\n""[(1:45)10000.000,(1:315)10000.000]\n"
"fail\n""2\n"@Testpublicvoidtest BasicPathfinding2()newPrintStreamNodegoal=new
Node190null0nullNodestart=newNode10null0intplanetSize=2intmemorySize=4String
frontier_result="[(1:0)1.414]\n""[(1:45)1.551,(1:315)2.633]\n""[(1:90)1.571,
(1:315)2.633,(1:0)2.985]\n""(1:0)(1:45)(1:90)\n""1.571\n""3\n"@Testpublicvoid
testAdvancedPathfinding()newPrintStreamNodegoal=newNode290null0nullNodestart=
newNode70null0intplanetSize=8intmemorySize=9Stringfrontier_result=
"[(7:0)7.280]\n""[(6:0)7.325,(7:45)11.260,(7:315)14.030]\n""[(5:0)7.385,
(7:0)9.280,(6:45)10.511,(7:45)11.260,(6:315)13.260,(7:315)14.030]\n"
"[(4:0)7.472,(7:0)9.280,(6:0)9.325,(5:45)9.782,(6:45)10.511,(7:45)11.260,
(5:315)12.495,(6:315)13.260,(7:315)14.030]\n""[(3:0)7.606,(4:45)9.089,
(7:0)9.280,(6:0)9.325,(5:0)9.385,(5:45)9.782,(6:45)10.511,(7:45)11.260,
(4:315)11.737]\n""[(2:0)7.828,(3:45)8.481,(4:45)9.089,(7:0)9.280,(6:0)9.325,
(5:0)9.385,(4:0)9.472,(5:45)9.782,(6:45)10.511]\n""[(2:45)8.102,(1:0)8.236,
(3:45)8.481,(4:45)9.089,(7:0)9.280,(6:0)9.325,(5:0)9.385,(4:0)9.472,
(3:0)9.606]\n""[(2:90)8.142,(1:0)8.236,(1:45)9.044,(4:45)9.089,(7:0)9.280,
(6:0)9.325,(5:0)9.385,(4:0)9.472,(3:45)9.696]\n""(7:0)(6:0)(5:0)(4:0)(3:0)
(2:0)(2:45)(2:90)\n""8.142\n""8\n"@Testpublicvoidtest AdvancedPathfinding2()new
PrintStreamNodegoal=newNode10null0nullNodestart=newNode3180null0intplanetSize=
4intmemorySize=8Stringfrontier_result="[(3:180)4.000]\n""[(2:180)4.000,
(3:135)6.130,(3:225)6.130]\n""[(1:180)4.000,(2:135)5.369,(2:225)5.369,
(3:180)6.000,(3:135)6.130,(3:225)6.130]\n""[(1:135)4.633,(1:225)4.633,
(2:135)5.369,(2:225)5.369,(2:180)6.000,(3:180)6.000,(3:135)6.130,
(3:225)6.130]\n""[(1:225)4.633,(1:90)4.985,(2:225)5.369,(1:180)5.571,
(2:180)6.000,(3:180)6.000,(3:225)6.130,(2:135)6.583]\n""[(1:90)4.985,
(1:270)4.985,(1:180)5.571,(1:180)5.571,(2:180)6.000,(3:180)6.000,(2:135)6.583,
(2:225)6.583]\n""[(1:270)4.985,(1:45)5.122,(1:180)5.571,(1:180)5.571,
(2:180)6.000,(3:180)6.000,(2:135)6.583,(2:225)6.583]\n""[(1:45)5.122,
(1:315)5.122,(1:180)5.571,(1:180)5.571,(2:180)6.000,(3:180)6.000,(2:135)6.583,
(2:225)6.583]\n""[(1:315)5.122,(1:0)5.142,(1:180)5.571,(1:180)5.571,
(2:180)6.000,(3:180)6.000,(2:135)6.583,(2:225)6.583]\n""[(1:0)5.142,
(1:0)5.142,(1:180)5.571,(1:180)5.571,(2:180)6.000,(3:180)6.000,(2:135)6.583,
(2:225)6.583]\n""(3:180)(2:180)(1:180)(1:135)(1:90)(1:45)(1:0)\n""5.142\n"
"10\n"@Testpublicvoidtest GoalOfZero()newPrintStreamNodegoal=newNode00null0null
Nodestart=newNode10null0intplanetSize=2intmemorySize=2Stringfrontier_result=
"[(1:0)1.000]\n""[(1:45)10000.000,(1:315)10000.000]\n""fail\n""2\n"@Testpublic
voidtestPathOutOfBounds()newPrintStreamNodegoal=newNode190null0nullNodestart=
newNode30null0intplanetSize=2intmemorySize=2Stringfrontier_result=
"[(3:0)3.162]\n""fail\n""1\n"@Testpublicvoidtest PathOutOfBounds2()new
PrintStreamNodegoal=newNode490null0nullNodestart=newNode10null0intplanetSize=2
intmemorySize=2Stringfrontier_result="[(1:0)4.123]\n""[(1:45)10000.000,
(1:315)10000.000]\n""fail\n""2\n"@Testpublicvoidtest PathOutOfBounds3()new
PrintStreamNodegoal=newNode490null0nullNodestart=newNode40null0intplanetSize=2
intmemorySize=2Stringfrontier_result="[(4:0)5.657]\n""fail\n""1\n"@Testpublic
voidtestMemoryConstraint()newPrintStreamNodegoal=newNode2180null0nullNodestart
=newNode10null0intplanetSize=3intmemorySize=2Stringfrontier_result=
"[(1:0)3.000]\n""[(2:0)10000.000,(1:315)10000.000]\n""fail\n""2\n" Tests;
  2
  3   org.junit.Assert.assertEquals;
  4
  5  java.io.ByteArrayOutputStream;
  6  java.io.PrintStream;
  7
  8  org.junit.Test;
```

```
 9
10  Algorithms.PartB_SMAStar;
11  General.Node;
12
13    {
14
15           ();
16
17
18          {
19           System.setOut( (outContent));
20
21               (, , , , );
22               (, , , , goal);
23              ;
24              ;
25
26                   +
27                   +
28                   + ;
29
30           PartB_SMAStar.smaStar(start, goal, planetSize, memorySize);
31           assertEquals(frontier_result, outContent.toString());
32      }
33
34
35          {
36           System.setOut( (outContent));
37
38               (, , , , );
39               (, , , , goal);
40              ;
41              ;
42
43                   +
44                   +
45                   +
46                   +
47                   + ;
48
49           PartB_SMAStar.smaStar(start, goal, planetSize, memorySize);
50           assertEquals(frontier_result, outContent.toString());
51      }
52
53
54          {
55           System.setOut( (outContent));
56
57               (, , , , );
58               (, , , , goal);
59              ;
60              ;
61
62                   +
63                   +
64                   +
65                   +
66                   +
67                   +
68                   +
69                   +
```

```
                +
                + ;

        PartB_SMAStar.smaStar(start, goal, planetSize, memorySize);
        assertEquals(frontier_result, outContent.toString());
    }


       {
        System.setOut( (outContent));

            (, , , , );
            (, , , , goal);
           ;
           ;

                +
                +
                +
                +
                +
                +
                +
                +
                +
                +
                + ;
        PartB_SMAStar.smaStar(start, goal, planetSize, memorySize);
        assertEquals(frontier_result, outContent.toString());
    }


       {
        System.setOut( (outContent));

            (, , , , );
            (, , , , goal);
           ;
           ;

                +
                +
                + ;

        PartB_SMAStar.smaStar(start, goal, planetSize, memorySize);
        assertEquals(frontier_result, outContent.toString());
    }


       {
        System.setOut( (outContent));

            (, , , , );
            (, , , , goal);
           ;
           ;

                +
                + ;

```

```
131         PartB_SMAStar.smaStar(start, goal, planetSize, memorySize);
132         assertEquals(frontier_result, outContent.toString());
133     }
134 
135 
136        {
137         System.setOut( (outContent));
138 
139             (, , , , );
140             (, , , , goal);
141            ;
142            ;
143 
144                 +
145                 +
146                 + ;
147 
148         PartB_SMAStar.smaStar(start, goal, planetSize, memorySize);
149         assertEquals(frontier_result, outContent.toString());
150     }
151 
152 
153        {
154         System.setOut( (outContent));
155 
156             (, , , , );
157             (, , , , goal);
158            ;
159            ;
160 
161                 +
162                 + ;
163 
164         PartB_SMAStar.smaStar(start, goal, planetSize, memorySize);
165         assertEquals(frontier_result, outContent.toString());
166     }
167 
168 
169        {
170         System.setOut( (outContent));
171 
172             (, , , , );
173             (, , , , goal);
174            ;
175            ;
176 
177                 +
178                 +
179                 + ;
180 
181         PartB_SMAStar.smaStar(start, goal, planetSize, memorySize);
182         assertEquals(frontier_result, outContent.toString());
183     }
184 }
```