THE UNIVERSITY of ADELAIDE

Path Finding Test Report

NXT Archeology Robot Test Report

Yunyao Yao

ID: 1203525

Chapter 1

Test Plan

1.1 Tested Methods

In these tests, I tested the methods in the PathFinder class including DFS and setPath and the sub-method they call. These class is used for robot's path finding. When a robot starts a auto-navigation, It will call DFS method in this class to search the map. If the robot goes into a dead end and there are still unexplored area, it will call setPath method to find a path from current location to closest unexplored area.As these part of code is not written by me, it would be better to let me to test it rather than the author.

Tested Method Link:

sep2012-13\Code\Final Version\src\Navigation\PathFinder.java

1.1.1 DFS

This is a pathFinding method of the robot.It takes a starting point as parameter. It will find a unexplored (not wall, not no-go zone) Pixel following this order: North, South, West, East. It store the path it travels by saving a integer into the variable PrevMark of each pixel in the path. For example, "PrevMark=1" means the the previous pixel of current one is at north (see the table below for more definition). If robot goes into a dead end which mean no more available moves this method will return. Return value "1" means no more unexplored pixels left in the map, the survey is complete perfectly. Return value "0" means the robot went into a dead end and unfortunately there are still unexplored pixel in the map.

PrevMark	Meaning
1	Previous: North
2	Previous: East
3	Previous: South
4	Previous: West

1.1.2 SetPath

setPath is also a path finding method. However, rather than a single parameter like DFS, it takes two parameters, starting point and destination point. It will return a path from the starting point to destination point as a LinkedList of Pixels. The method will add all pixels on the path in order. Then robot can follow this path to reach the unexplored area.

1.2 Test Strategy

I used JUnit test tools to test the code. For DFS I examine the PrevMark variable of Pixel Objects which marked previous Pixel by direction. For setPath I examine the LinkedList it returns with the LinkedList a build manually.

1.3 Pass/Fail Criteria

1.3.1 DFS

For each Pixel if the value of PrevMark are equal to what I expect in the graph(showed below in each test case).

1.3.2 setPath

The LinkedList this method returns is equal to the one a manually build which contains all the pixels in the expected path.

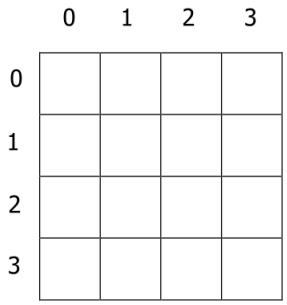
Chapter 2

Test Cases

I have 5 test cases in total in my test script. For each test case I create a 4 by 4 small map and tested both two methods. The map I created come with walls to block the way of the robot.

2.1 Test 1

2.1.1 Map Used In This Test:



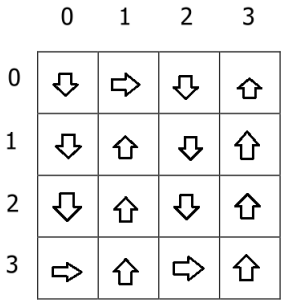
No wall is set in this test case.

2.1.2 Reason For Choosing This Case

This map has no walls, it is designed in order to test that whether the robot can go through the whole map if there is no obstacle.

2.1.3 DFS

Start at (0,0):



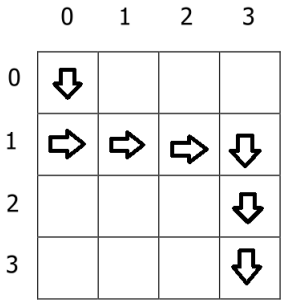
DFS are expected to search the map like this. So the value of PrevMark is checked by every pixel:

PrevMark Value Diagram:

	0	1	2	3
0		3	4	3
1	1	3	1	3
2	1	3	1	3
3	1	4	1	4

2.1.4 SetPath

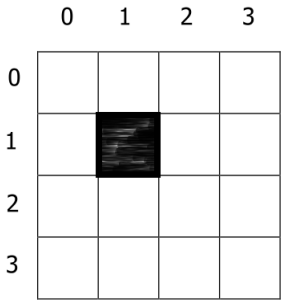
Set a Path From (0,0) to (3,3), the expected result is showed below:



All the pixels on this path is added to a LinkedList, and it will be compared with the return value of SetPath method.

2.2 Test 2

2.2.1 Map Used In This Test:



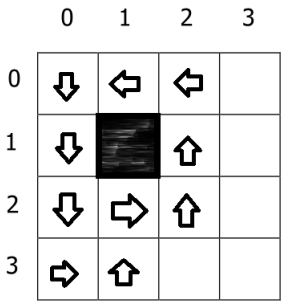
(1, 1) is set as wall in this test case.

2.2.2 Reason For Choosing This Case

This map has only one wall. It is designed in order to test that whether it will reached expected location at the end, if robot' path is blocked at the beginning.

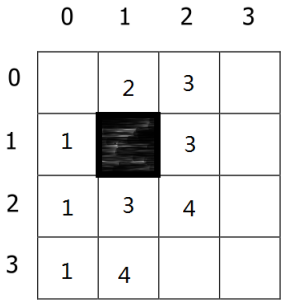
2.2.3 DFS

Start at (0,0):



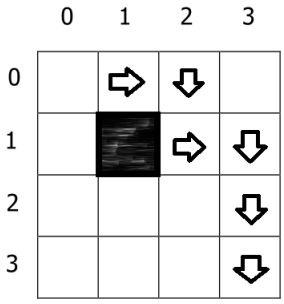
DFS are expected to search the map like this. So the value of PrevMark is checked by every pixel:

PrevMark Value Diagram:



2.2.4 SetPath

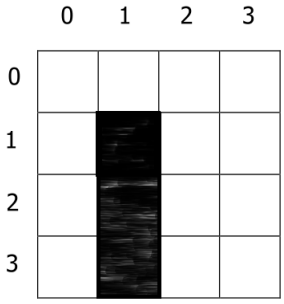
Set a Path From (1,0) to (3,3), the expected result is showed below:



All the pixels on this path is added to a LinkedList, and it will be compared with the return value of SetPath method.

2.3 Test 3

2.3.1 Map Used In This Test:



(1, 1) is set as wall in this test case.
(1, 2) is set as wall in this test case.

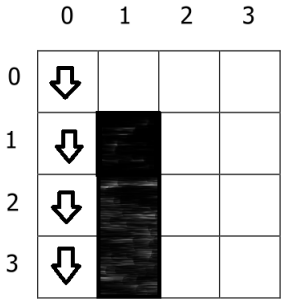
(1, 3) is set as wall in this test case.

2.3.2 Reason For Choosing This Case

This map has a dead-end, and it is designed to test that whether the method can correctly return after the robot go into a dead-end.

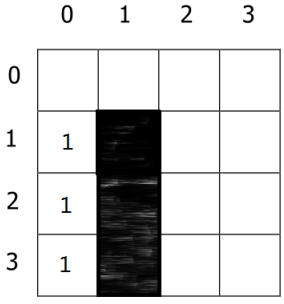
2.3.3 DFS

Start at (0,0):



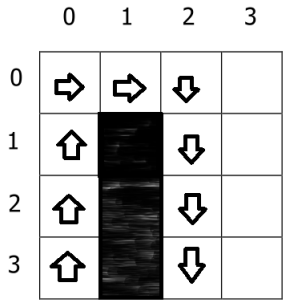
DFS are expected to search the map like this. So the value of PrevMark is checked by every pixel:

PrevMark Value Diagram:



2.3.4 SetPath

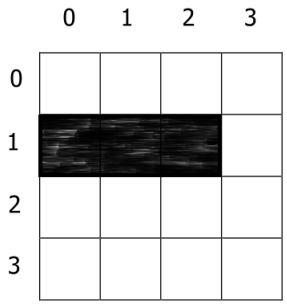
Set a Path From (0,3) to (2,3), the expected result is showed below:



All the pixels on this path is added to a LinkedList, and it will be compared with the return value of SetPath method.

2.4 Test 4

2.4.1 Map Used In This Test:



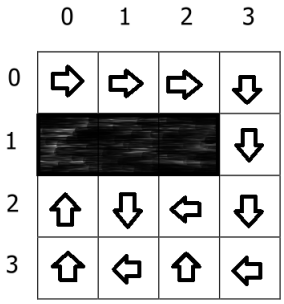
(0, 1) is set as wall in this test case.
(1, 1) is set as wall in this test case.
(2, 1) is set as wall in this test case.

2.4.2 Reason For Choosing This Case

This map has walls, but the robot should be able to go through all the pixels in this case, this case is to test that whether it can go through this map.

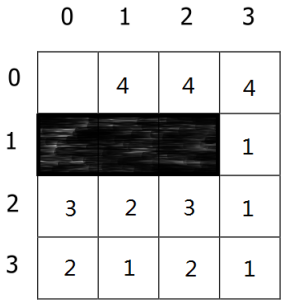
2.4.3 DFS

Start at (0,0):



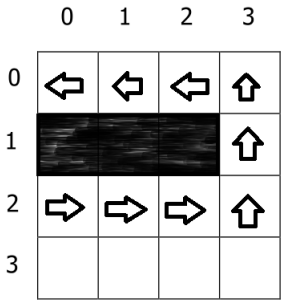
DFS are expected to search the map like this. So the value of PrevMark is checked by every pixel:

PrevMark Value Diagram:



2.4.4 SetPath

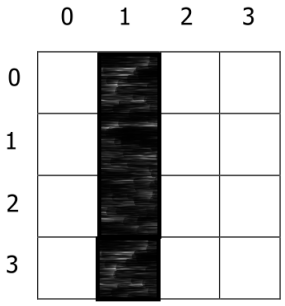
Set a Path From (0,2) to (0,0), the expected result is showed below:



All the pixels on this path is added to a LinkedList, and it will be compared with the return value of SetPath method.

2.5 Test 5

2.5.1 Map Used In This Test:



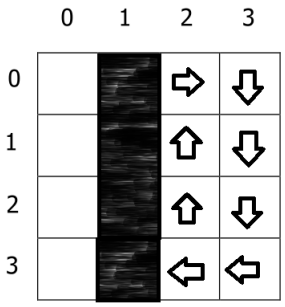
(1, 0) is set as wall in this test case.
(1, 1) is set as wall in this test case.
(1, 2) is set as wall in this test case.
(1, 3) is set as wall in this test case.

2.5.2 Reason For Choosing This Case

This map is map is separated by walls. This case is designed to test that whether the robot can correctly go through part of the map with DFS and find out that the path is blocked.

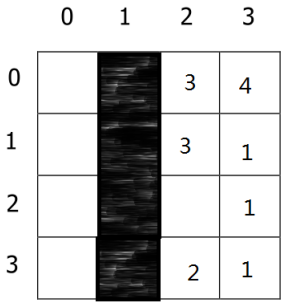
2.5.3 DFS

Start at (0,0):



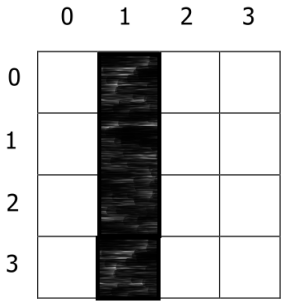
DFS are expected to search the map like this. So the value of PrevMark is checked by every pixel:

PrevMark Value Diagram:



2.5.4 SetPath

Set a Path From (0,2) to (2,2), the expected result is showed below:



As there is no path from (0,2) to (2,2) the method should return null.