Lab 3 Report

Name 陳宇宏
Student ID 110598067
Date

1 Test Plan

1.1 Test requirements

The Lab 3 requires to (1) select 6 methods from 6 classes of the SUT (GeoProject), (2) design Unit test cases by using **basis path or graph coverage** technique for the selected methods, (3) develop test scripts to implement the test cases, (4) execute the test scripts on the selected methods, (5) report the test results, and (6) specify your experiences of designing test cases systematically using the graph coverage technique.

In particular, based on the target coverage criteria (i.e., statement, branch, or others), the **test requirements** for Lab 3 are to design test cases *with* **graph coverage technique** for each selected method so that "*each statement and branch (or path) of the method under test will be covered by at least one test case* and *the both minimum* ***statement*** *(node) and* ***branch*** *(edge) coverage are greater than those of Lab 2 and 90%, respectively*."

1.2 Test Strategy

To satisfy the test requirements listed in Section 1, a proposed strategy is to

(1) select **3 methods that were chosen in Lab1 or Lab2** and **3 new methods** that are NOT selected previously. The selected methods MUST contain **predicate** and/or **loop** structures (as many as possible).

(2) set the objective of the minimum statement or branch (or path) coverage to be greater than that of Lab 2 and adjust the test objective (e.g., 90%, 95% or 100%) based on the time available (if necessary).

(3) design the test cases for those selected methods by using the **basis path or graph coverage** testing technique.

1.3 Test activities

To implement the proposed strategy, the following activities are planned to perform.

No.	Activity Name	Plan hours	Schedule Date
1	Study GeoProject	1	5/10
2	Learn **basis path and**	2	5/10

	graph coverage		
3	Design test cases for the selected methods	5	5/11
4	Implement test cases	3	5/13
5	Perform tests and check code coverage. If not satisfy, design more test cases…	1	5/13
…			
n	Complete Lab3 report	1	5/13

1.4 Design Approach

The **basis path and graph coverage** technique will be used to design the test cases. Specifically, the control flow graph (CFG) of each selected method shall be drawn first, and the possible test paths that satisfy the test requirements (i.e., **statement (node), branch (edge), or path coverage**) shall be derived from the CFG. The possible **inputs** and **expected outputs** for the derived test paths shall be computed from the specification of SUT for each method under test. *Add more test cases by considering to satisfy other coverage criteria, such as edge-pair, all-use, or prime-path coverage criteria.*

1.5 Success criteria

All test cases designed for the selected methods must pass (or 90% of all test cases must pass) and *both statement and branch (or path) coverage should have achieved at least 90%, respectively.*

2 Test Design

To fulfill the test requirements listed in section 1.1, the following methods are selected and corresponding test cases are designed.

No.	Class	Method	Source Code Links	CFG Links	Test Paths	Inputs	Expected Outputs
1	Geomem	find(double topLeftLat, double topLeftLon, double bottomRightLat, double bottomRightLon, long start, long finish)	In Excel	In Excel	P2 : {n1, n2, n3, n2, n4}	T1 :{0, 0, 0, 0, 0, 0}	Iterable<Info<String,String>> 在轉成 List 後，使用 isEmpty()會回傳 True
2	Coverage	getHashLength()	In Excel	In Excel	"P1 : {n1, n2} P2 : {n1, n3} "	T1 :{Sets.<String> newHashSet()} T2 :{Sets.<String>	T1 : {0} T2: {5}

						newHashSet("56789")}	
3	CoverageLongs	getHashLength()	In Excel	In Excel	P1 : {n1, n2} P2 : {n1, n3}	T1 :{new long[] {}, 0, 1.0} T2 :{new long[] {100,10000}, 2, 1.0}	T1 : {0} T2: {4}
4	GeoHash	fromLongToString(long hash)	In Excel	In Excel	P1 : {n1, n2, n4} P2 : {n1, n2, n3, n4} P3 : {n1,n2,n3,n5,n6,n7,n6,n8}	"T1:{0} T2:{13} T3:{1}"	T1 : { IllegalArgumentException} T2 : { IllegalArgumentException} T3 : {“0”}
5	GeoHash	"public static String gridAsString(String hash, int fromRight, int fromBottom, int toRight, int toBottom, Set<String> highlightThese)"	In Excel	In Excel	"P1 : {n1, n2, n3, n13 }, P2: {n1, n2, n3, n5, n6, n12 ,n4, n3, n13 }, P3: {n1, n2, n3, n5, n6, n8, n9, n11 ,n7, n6, n12 ,n4,	"T1:{ ""dr"",0, 1,0,0, Sets.newHashSet(""dr"")} T2:{""dr"",1,0,0,0, Sets.newHashSet(""dr"")} T3:{""dr"",0,0,0,0, Sets.newHashSet(""dr"")} T4:{""dr"",0,0,0,0, Sets.newHashSet(""dr"")}"	T1 : “” T2: “\n” T3: “dr \n” T4 “DR \n”

					n3, n13 }, P4: {n1,n2,n3,n5,n6, n8,n9,n10,n11,n7,n6,n12, n4,n3,n13}"		
6	GeoHash	heightDegrees(int)	In Excel	In Excel	"P1 : {n1, n2} P2 : {n1, n3} "	"T1 : {13} T2 : {12}"	T1 :{ 0.0} T2 : {0.0}

The details of the design are given below:

The Excel file of test cases…

3 Test Implementation

The design of test cases specified in Section 2 was implemented using JUnit 4. The test scripts of 3 selected test cases are given below. The rest of the test script implementations can be found in the link (or JUnit files).

No.	Test method	Source test code

1	GeoHash heightDegrees ()	@Test
		public void heightDegreesT1(){
		GeoHash.heightDegrees(13);
		
		assertEquals(0.0,GeoHash.heightDegrees(13),0.01);
		}
		@Test
		public void heightDegreesT2(){
		GeoHash.heightDegrees(12);
		
		assertEquals(0.0,GeoHash.heightDegrees(12),0.01);
		}

2	GeoHashLong	@Test

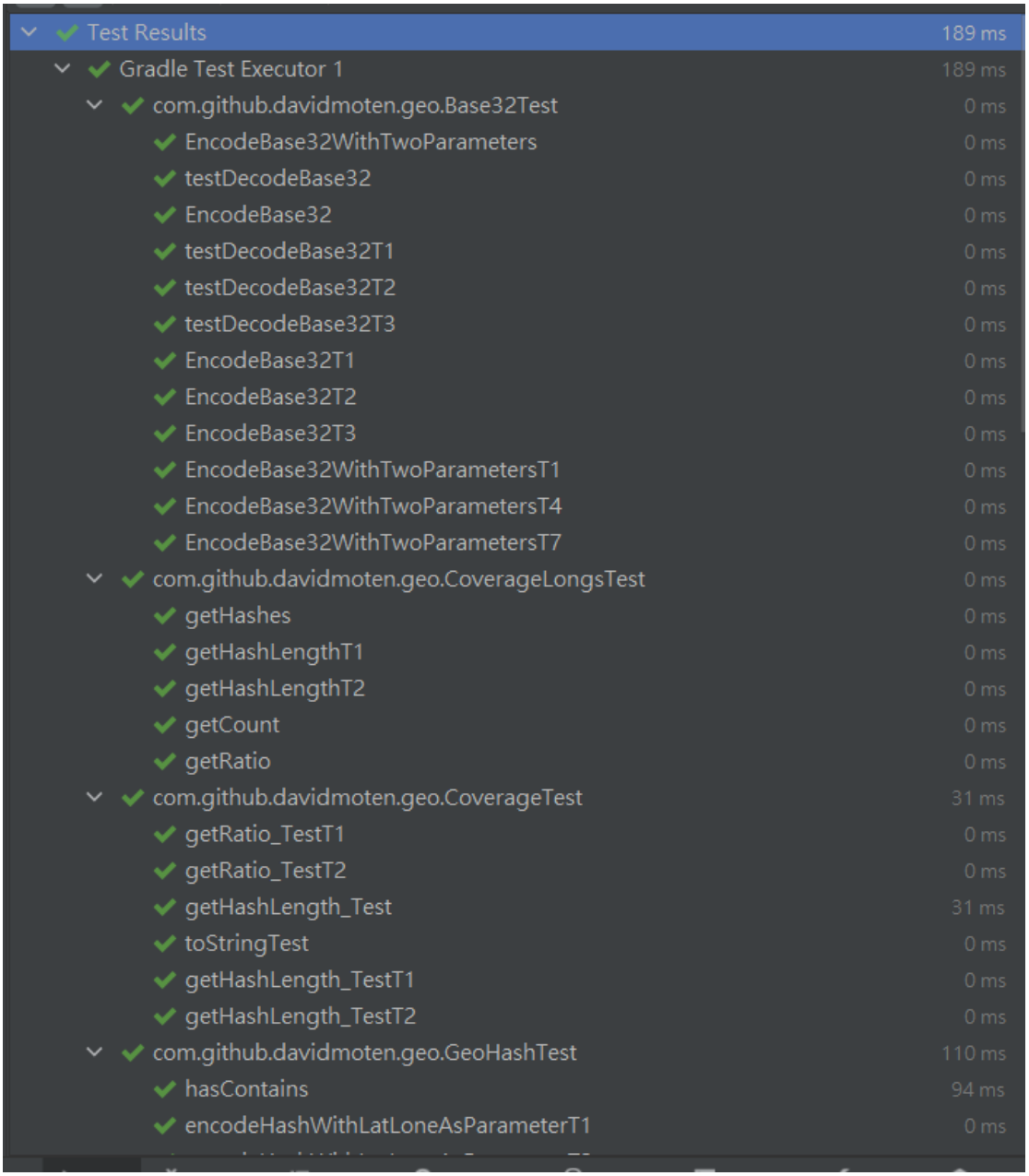
	getHashLength ()	public void getHashLengthT1() {
		coverageLongs = new CoverageLongs(new long[] {}, 0, 1.0);
		coverageLongs.getHashLength();
		
		assertEquals(0,coverageLongs.getHashLength());
		
		System.out.println(coverageLongs);
		}
		@Test
		public void getHashLengthT2() {
		coverageLongs = new CoverageLongs(new long[] {100,10000}, 2, 1.0);
		coverageLongs.getHashLength();
		
		assertEquals(4,coverageLongs.getHashLength());
		
		System.out.println(coverageLongs);
		}

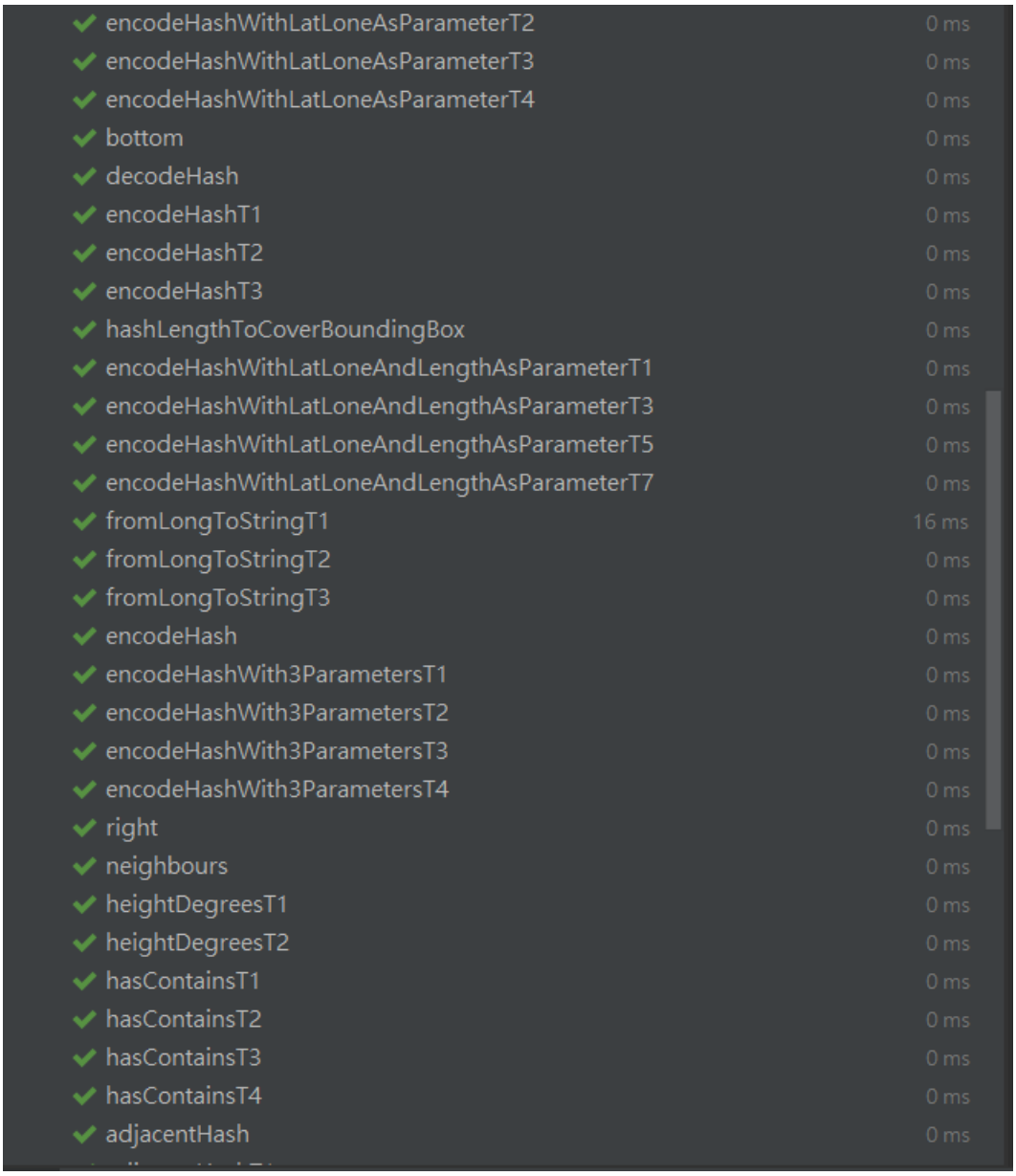
3	gridAsString(String hash,
	
	int fromRight,
	
	int fromBottom,
	
	int toRight,
	
	int toBottom,
	
	Set<String> highlightThese)
		@Test
		public void getHashLengthT1() {
		coverageLongs = new CoverageLongs(new long[] {}, 0, 1.0);
		coverageLongs.getHashLength();
		
		assertEquals(0,coverageLongs.getHashLength());
		
		System.out.println(coverageLongs);
		}
		@Test
		public void getHashLengthT2() {
		coverageLongs = new CoverageLongs(new long[] {100,10000}, 2, 1.0);

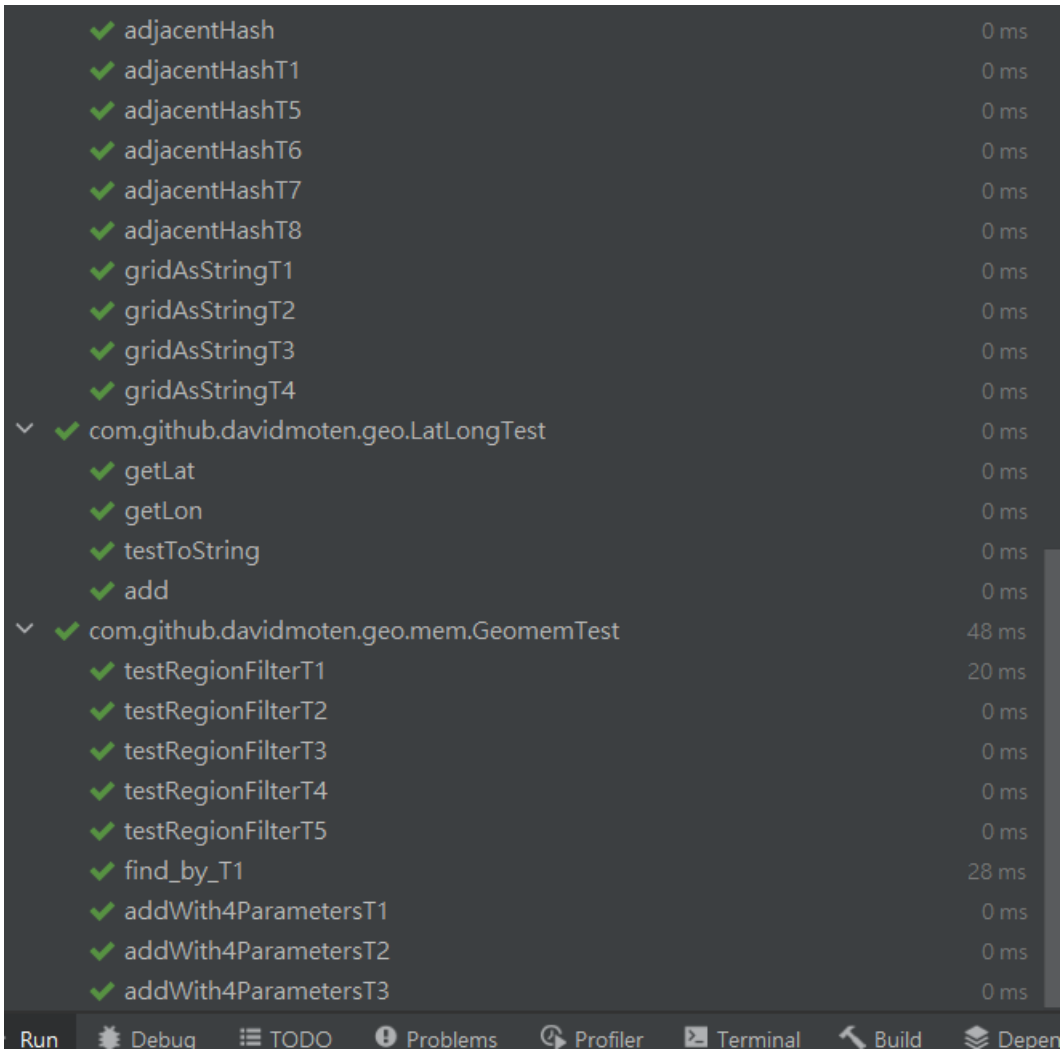
		coverageLongs.getHashLength();
		
		assertEquals(4,coverageLongs.getHashLength());
		
		System.out.println(coverageLongs);
		}

4 Test Results

4.1 JUnit test result snapshot


Test Results 189 ms
Gradle Test Executor 1 189 ms
com.github.davidmoten.geo.Base32Test 0 ms
EncodeBase32WithTwoParameters 0 ms
testDecodeBase32 0 ms
EncodeBase32 0 ms
testDecodeBase32T1 0 ms
testDecodeBase32T2 0 ms
testDecodeBase32T3 0 ms
EncodeBase32T1 0 ms
EncodeBase32T2 0 ms
EncodeBase32T3 0 ms
EncodeBase32WithTwoParametersT1 0 ms
EncodeBase32WithTwoParametersT4 0 ms
EncodeBase32WithTwoParametersT7 0 ms
com.github.davidmoten.geo.CoverageLongsTest 0 ms
getHashes 0 ms
getHashLengthT1 0 ms
getHashLengthT2 0 ms
getCount 0 ms
getRatio 0 ms
com.github.davidmoten.geo.CoverageTest 31 ms
getRatio_TestT1 0 ms
getRatio_TestT2 0 ms
getHashLength_Test 31 ms
toStringTest 0 ms
getHashLength_TestT1 0 ms
getHashLength_TestT2 0 ms
com.github.davidmoten.geo.GeoHashTest 110 ms
hasContains 94 ms
encodeHashWithLatLoneAsParameterT1 0 ms


encodeHashWithLatLoneAsParameterT2 0 ms
encodeHashWithLatLoneAsParameterT3 0 ms
encodeHashWithLatLoneAsParameterT4 0 ms
bottom 0 ms
decodeHash 0 ms
encodeHashT1 0 ms
encodeHashT2 0 ms
encodeHashT3 0 ms
hashLengthToCoverBoundingBox 0 ms
encodeHashWithLatLoneAndLengthAsParameterT1 0 ms
encodeHashWithLatLoneAndLengthAsParameterT3 0 ms
encodeHashWithLatLoneAndLengthAsParameterT5 0 ms
encodeHashWithLatLoneAndLengthAsParameterT7 0 ms
fromLongToStringT1 16 ms
fromLongToStringT2 0 ms
fromLongToStringT3 0 ms
encodeHash 0 ms
encodeHashWith3ParametersT1 0 ms
encodeHashWith3ParametersT2 0 ms
encodeHashWith3ParametersT3 0 ms
encodeHashWith3ParametersT4 0 ms
right 0 ms
neighbours 0 ms
heightDegreesT1 0 ms
heightDegreesT2 0 ms
hasContainsT1 0 ms
hasContainsT2 0 ms
hasContainsT3 0 ms
hasContainsT4 0 ms
adjacentHash 0 ms



Test Summary

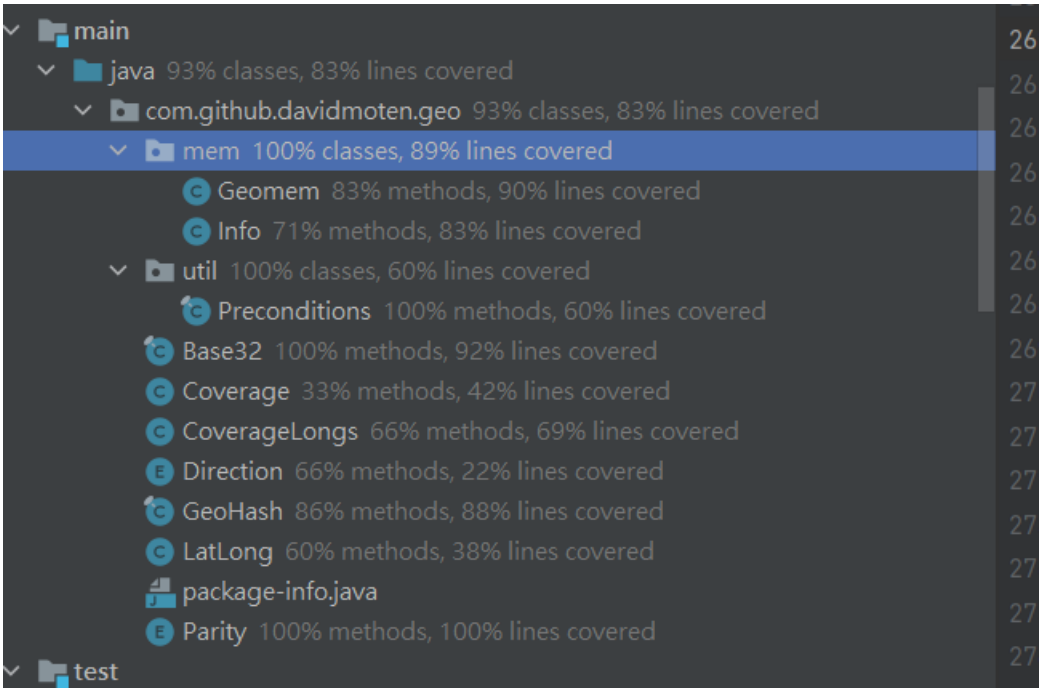
77	0	0	1.116s	100%
tests	failures	ignored	duration	successful

Packages Classes

Package	Tests	Failures	Ignored	Duration	Success rate
com.github.davidmoten.geo	68	0	0	1.024s	100%
com.github.davidmoten.geo.mem	9	0	0	0.092s	100%

4.2 Code coverage snapshot

- Coverage of each selected method under test



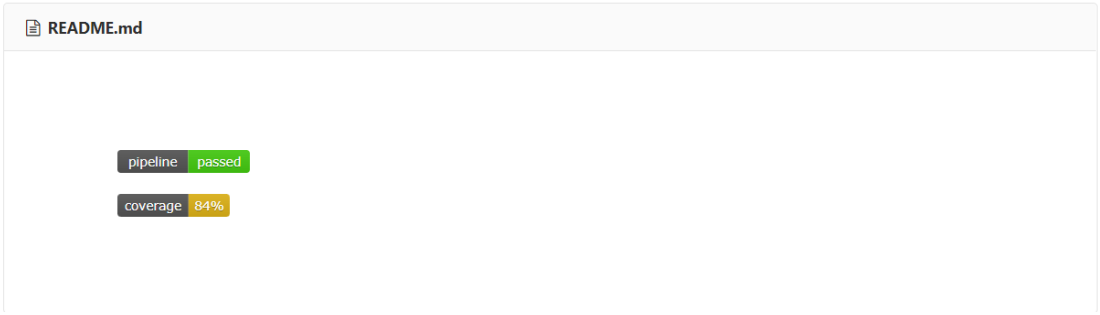
- Total coverage

geo

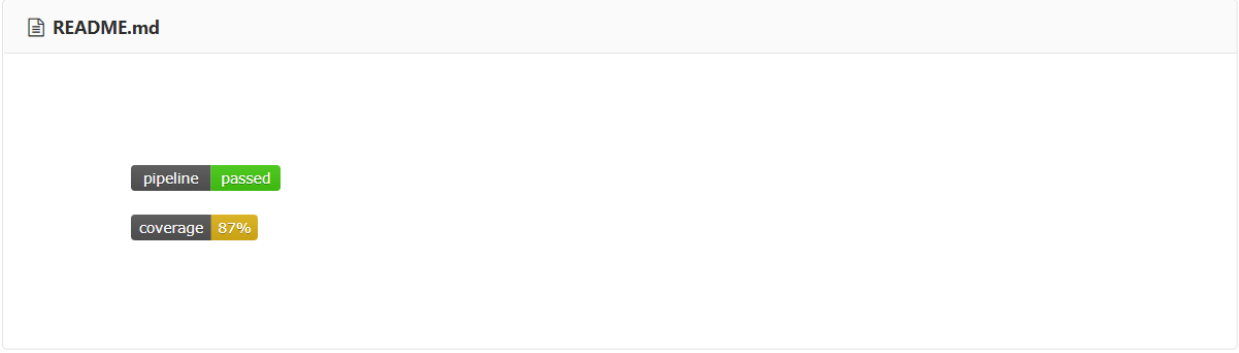
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.github.davidmoten.geo		92%		91%	21	149	23	348	7	68	0	10
com.github.davidmoten.geo.mem		87%		80%	6	30	3	61	2	20	0	3
com.github.davidmoten.geo.util		100%		100%	0	4	0	6	0	2	0	1
Total	183 of 2,326	92%	18 of 186	90%	27	183	26	415	9	90	0	14

4.3 CI result snapshot (3 iterations for CI)

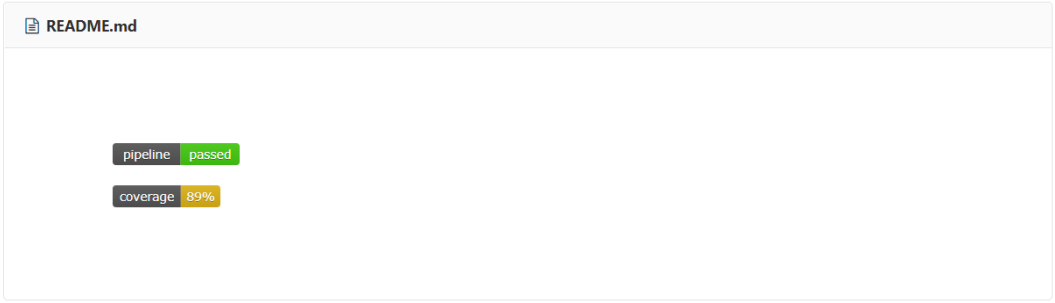
- CI#1



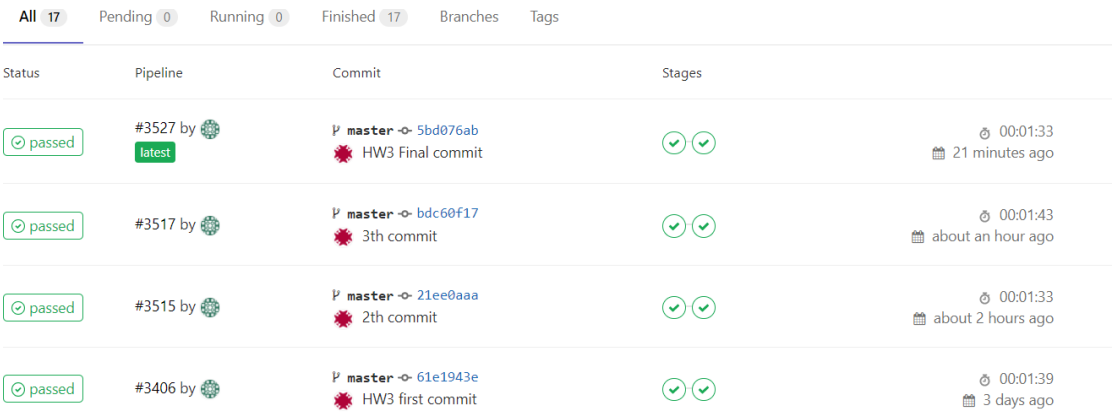
- CI#2



- CI#3



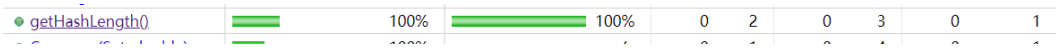
- CI Pipeline



5 The Coverage Comparison

The code coverage of Lab1 (and/or Lab2) and Lab3 are listed in the below Table. The results show that the statement and branch coverage are increased from 100% to 100% in Lab3.(這次挑的 method 只有一個是前兩次 Lab 測過的，以滿足 ***statement*** *(node) and* ***branch*** *(edge) coverage 大於 90%*)

Lab2:



Lab3:

No.	Test method	Lab1 (or Lab2)		Lab3	
		statement coverage	branch coverage	statement coverage	branch coverage
1	Coverage getHashLength ()	都是 100%		都是 100%	

6 Summary

In Lab 3, **6 test cases have been designed and implemented using JUnit and the basis path/graph coverage technique**. The test is conducted in 3 CI and **the execution results of the 6 test methods are all passed**. **The total statement and branch coverage of the test are 92% and 90%, respectively.** Thus, the test requirements described in Section 1 are satisfied. Some lessons learned in this Lab are 這次深入的實作和學習了 path/grahp coverge 的測試方式，透過畫 CFG 圖可以更容易的設計測試，也因為 GeoHash 測試蠻複雜的，為了確認有正確的測試到想要測的路徑，也使用了 intellij 的 debug run 的方式來驗證。