Verificação e Validação de Software
2021/2022
Mestrado em Engenharia Informática

Assignment 1 - Ternary Search Trie

Tomás Dias nº57582

Estrutura do projeto

Na diretoria *src/test/java* encontram-se os packages:

- **sut.lb**: contém as classes e testes unitários aos métodos públicos da TST segundo o critério *Line and Branch Coverage*.

- **sut.eppp**: contém as classes e testes unitários ao método público *longestPrefixOf* da TST segundo o critério *Edge-Pair Coverage and Prime Path Coverage*.

- **sut.adup**: contém as classes e testes unitários ao método público *longestPrefixOf* da TST segundo o critério *All-Du-Paths Coverage*.

- **sut.acup**: contém as classes e testes unitários ao ao método privado *put* da TST segundo o critério *All-Coupling-Use-Paths Coverage*.

- **sut.logic**: contém as classes e testes unitários ao método público *longestPrefixOf* da TST segundo o critério *Logic-based Test Coverage*.

- **sut.bc**: contém as classes e testes unitários aos método público *put* da TST segundo o critério *Base Choice Coverage*.

- **sut.util**: contém classes auxiliares, nomeadamente a classe *TSTKeysResolver* para injeção de dependências.

Análise dos critérios de cobertura

Line and Branch Coverage

A implementação da SUT com este critério de cobertura foi realizada com o auxílio da ferramenta *Eclmma,* responsável por detalhar a cobertura dos testes unitários ao nível de LC, IC e BC. Aqui é apresentado, como exemplo, esses dados para a classe *TSTContainsTest*:

- nullKeyTest: LC - (2/3); IC - (7/15); BC - (1/4)
- tableDoesNotContainKeyTest: LC - (2/3); IC - (8/15); BC - (2/4)
- tableContainsKeyTest: LC - (2/3); IC - (10/15); BC - (3/4)

Edge-Pair Coverage and Prime Path Coverage

Foi construído o *Control Flow Graph (CFG)* do método *longestPrefixOf*. Para tal, foram definidos os *basic blocks*, o *entry node*, os *decision nodes*, os *exit nodes* e os *edges*.

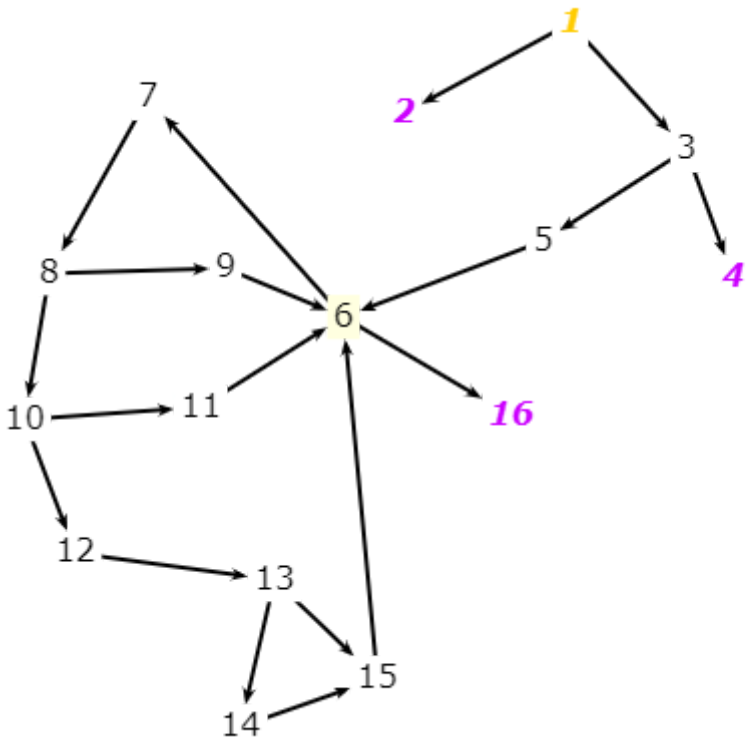
Basic blocks (nodes):
1: if (query == null)
2: throw new IllegalArgumentException("calls longestPrefixOf() with null argument");
3: if (query.length() == 0)
4: return null;
5: int length = 0; Node<T> x = root; int i = 0;
6: if (x != null && i < query.length())
7: char c = query.charAt(i);
8: if (c < x.c)
9: x = x.left;
10: else if (c > x.c)
11: x = x.right;
12: i++;
13: if (x.val != null)
14: length = i;
15: x = x.mid;
16: return query.substring(0, length);

Entry node: 1
Decision nodes: 1, 3, 6, 8, 10, 13
Exit nodes: 2, 4, 16

Control flow (edges):

- 1 → 2; 1 → 3
- 3 → 4; 3 → 5
- 5 → 6
- 6 → 7; 6 → 16
- 7 → 8
- 8 → 9; 8 → 10
- 9 → 6
- 10 → 11; 10 → 12
- 11 → 6
- 12 → 13
- 13 → 14; 13 → 15
- 14 → 15
- 15 → 6

Através da ferramenta *Graph Coverage Tool*, o grafo obtido foi:



Avançou-se para a implementação da SUT do critério *Edge-Pair Coverage*, sendo definidos os requisitos para a mesma, novamente utilizando a ferramenta *Graph Coverage Tool*:

25 requirements are needed for Edge-Pairs
1. [1,2]
2. [1,3,4]
3. [1,3,5]
4. [3,5,6]
5. [5,6,7]
6. [5,6,16]
7. [6,7,8]
8. [7,8,9]
9. [7,8,10]
10. [8,9,6]
11. [8,10,11]
12. [8,10,12]
13. [9,6,7]
14. [9,6,16]
15. [10,11,6]
16. [10,12,13]
17. [11,6,7]
18. [11,6,16]
19. [12,13,14]
20. [12,13,15]
21. [13,14,15]
22. [13,15,6]
23. [14,15,6]
24. [15,6,7]
25. [15,6,16]

Foi ignorado o teste 6. por ser inviável, sendo que a cobertura ficou distribuída do seguinte modo:

t	Requirements Covered
nullQueryTest	[1,2]
emptyQueryTest	[1,3,4]
queryEqualsKeyTest	[1,3,5] [3,5,6] [5,6,7] [6,7,8] [7,8,9] [8,9,6] [9,6,7] [7,8,10] [8,10,12] [10,12,13] [12,13,15] [13,15,6] [15,6,7] [12,13,14] [13,14,15] [14,15,6] [15,6,16]
queryDoesNotFindKeyTest	[1,3,5] [3,5,6] [5,6,7] [6,7,8] [7,8,10] [8,10,11] [10,11,6] [11,6,7] [11,6,16]
queryDoesNotFindKey2Test	[1,3,5] [3,5,6] [5,6,7] [6,7,8] [7,8,9] [8,9,6] [9,6,7] [9,6,16]

Em relação à SUT do critério *Prime Path Coverage*, foram definidos os seguintes requisitos:

45 requirements are needed for Prime Paths
1. [1,3,5,6,7,8,10,12,13,14,15]
2. [1,3,5,6,7,8,10,12,13,15]
3. [7,8,10,12,13,14,15,6,7]
4. [6,7,8,10,12,13,14,15,6]
5. [7,8,10,12,13,14,15,6,16]
6. [8,10,12,13,14,15,6,7,8]
7. [9,6,7,8,10,12,13,14,15]
8. [13,14,15,6,7,8,10,12,13]
9. [12,13,14,15,6,7,8,10,12]
10. [14,15,6,7,8,10,12,13,14]
11. [15,6,7,8,10,12,13,14,15]
12. [12,13,14,15,6,7,8,10,11]
13. [10,12,13,14,15,6,7,8,9]
14. [10,12,13,14,15,6,7,8,10]
15. [11,6,7,8,10,12,13,14,15]
16. [7,8,10,12,13,15,6,16]
17. [7,8,10,12,13,15,6,7]
18. [9,6,7,8,10,12,13,15]
19. [8,10,12,13,15,6,7,8]
20. [1,3,5,6,7,8,10,11]
21. [6,7,8,10,12,13,15,6]
22. [12,13,15,6,7,8,10,12]
23. [13,15,6,7,8,10,12,13]
24. [15,6,7,8,10,12,13,15]
25. [12,13,15,6,7,8,10,11]
26. [10,12,13,15,6,7,8,10]
27. [10,12,13,15,6,7,8,9]
28. [11,6,7,8,10,12,13,15]
29. [1,3,5,6,7,8,9]
30. [8,10,11,6,7,8]
31. [7,8,10,11,6,16]
32. [9,6,7,8,10,11]
33. [7,8,10,11,6,7]
34. [6,7,8,10,11,6]
35. [10,11,6,7,8,9]
36. [10,11,6,7,8,10]
37. [11,6,7,8,10,11]
38. [7,8,9,6,16]
39. [7,8,9,6,7]
40. [8,9,6,7,8]
41. [1,3,5,6,16]
42. [6,7,8,9,6]
43. [9,6,7,8,9]
44. [1,3,4]
45. [1,2]

t	Test Path	Requirements Covered
nullQueryTest	[1,2]	45
emptyQueryTest	[1,3,4]	44
queryResultIsEmptyTest	[1,3,5,6,7,8,9,6,7,8,10,11,6,16]	29; 31; 32; 34; 39; 40; 42
retrieveLongestKeyWithNotNullValueTest	[1,3,5,6,7,8,10,12,13,15,6,7,8,9,6,7,8,10,12,13,15,6,7,8,9,6,7,8,10,12,13,14,15,6,16]	42; 40; 39; 27; 21; 19; 18; 17; 7; 5; 4; 2;
retrieveLongestKeyWithNotNullValue2Test	[1,3,5,6,7,8,10,11,6,7,8,10,12,13,15,6,7,10,12,13,15,6,7,8,10,12,13,14,15,6,16]	36; 34; 33; 30; 28; 26; 23; 22; 21; 20; 17; 11; 5; 4
retrieveLongestKeyWithNotNullValue3Test	[1,3,5,6,7,8,9,6,7,8,10,11,6,7,8,10,12,13,14,15,6,16]	4; 5; 15; 29; 30; 32; 33; 34; 36; 39; 40; 42
retrieveLongestKeyWithNotNullValue4Test	[1,3,5,6,7,8,10,12,13,15,6,7,8,10,12,13,15,6,7,8,10,12,13,14,15,6,16]	2; 4; 5; 11; 17; 19; 21; 22; 23; 24; 26

All-Du-Paths Coverage

Sendo que o método público *longestPrefixOf* contém as seguintes variáveis:

- query: *defs*(query) = {1}; *uses*(query) = {1,3,6,7,16}
- length: *defs*(length) = {5,14}; *uses*(length) = {16}
- x: *defs*(x) = {5,9,11,15}; *uses*(x) = {6,8,9,10,11,13,15}
- i: defs(i) = {5,12}; uses(i) = {7,12,14}
- c: defs(c) = {7}; uses(c) = {8,10}

através da ferramenta *Data Flow Graph Coverage Tool,* foram definidos os requisitos de teste, bem como os du-paths:

DU Paths for all variables are:

Variable	DU Paths
query	[1,3] [1,3,5,6] [1,3,5,6,16] [1,3,5,6,7]
length	[5,6,16] [14,15,6,16]
x	[5,6] [5,6,7,8] [5,6,7,8,10] [5,6,7,8,9] [5,6,7,8,10,11] [5,6,7,8,10,12,13] [5,6,7,8,10,12,13,15] [5,6,7,8,10,12,13,14,15] [9,6] [9,6,7,8] [9,6,7,8,10] [9,6,7,8,9] [9,6,7,8,10,11] [9,6,7,8,10,12,13] [9,6,7,8,10,12,13,15] [9,6,7,8,10,12,13,14,15] [11,6] [11,6,7,8] [11,6,7,8,10] [11,6,7,8,9] [11,6,7,8,10,11] [11,6,7,8,10,12,13] [11,6,7,8,10,12,13,15] [11,6,7,8,10,12,13,14,15] [15,6] [15,6,7,8] [15,6,7,8,10] [15,6,7,8,9] [15,6,7,8,10,11] [15,6,7,8,10,12,13] [15,6,7,8,10,12,13,15] [15,6,7,8,10,12,13,14,15]
i	[5,6,7] [5,6,7,8,10,12] [12,13,14] [12,13,15,6,7] [12,13,14,15,6,7] [12,13,15,6,7,8,10,12] [12,13,14,15,6,7,8,10,12]
c	[7,8] [7,8,10]

All DU Path Coverage for all variables are:

Variable	All DU Path Coverage
query	[1,3,4] [1,3,5,6,16] [1,3,5,6,7,8,9,6,16]
length	[1,3,5,6,16] [1,3,5,6,7,8,10,12,13,14,15,6,16]
x	[1,3,5,6,16] [1,3,5,6,7,8,9,6,16] [1,3,5,6,7,8,10,11,6,16] [1,3,5,6,7,8,10,12,13,15,6,16] [1,3,5,6,7,8,10,12,13,14,15,6,16] [1,3,5,6,7,8,9,6,16] [1,3,5,6,7,8,9,6,7,8,9,6,16] [1,3,5,6,7,8,9,6,7,8,10,11,6,16] [1,3,5,6,7,8,9,6,7,8,10,12,13,15,6,16] [1,3,5,6,7,8,9,6,7,8,10,12,13,14,15,6,16] [1,3,5,6,7,8,10,11,6,16] [1,3,5,6,7,8,10,11,6,7,8,9,6,16] [1,3,5,6,7,8,10,11,6,7,8,10,11,6,16] [1,3,5,6,7,8,10,11,6,7,8,10,12,13,15,6,16] [1,3,5,6,7,8,10,11,6,7,8,10,12,13,14,15,6,16] [1,3,5,6,7,8,10,12,13,15,6,16] [1,3,5,6,7,8,10,12,13,15,6,7,8,9,6,16] [1,3,5,6,7,8,10,12,13,15,6,7,8,10,11,6,16] [1,3,5,6,7,8,10,12,13,15,6,7,8,10,12,13,15,6,16] [1,3,5,6,7,8,10,12,13,15,6,7,8,10,12,13,14,15,6,16]
i	[1,3,5,6,7,8,9,6,16] [1,3,5,6,7,8,10,12,13,15,6,16] [1,3,5,6,7,8,10,12,13,14,15,6,16] [1,3,5,6,7,8,10,12,13,15,6,7,8,9,6,16] [1,3,5,6,7,8,10,12,13,14,15,6,7,8,9,6,16] [1,3,5,6,7,8,10,12,13,15,6,7,8,10,12,13,15,6,16] [1,3,5,6,7,8,10,12,13,14,15,6,7,8,10,12,13,15,6,16]
c	[1,3,5,6,7,8,9,6,16] [1,3,5,6,7,8,10,11,6,16]

t	Test Path	Requirements Covered
emptyQueryTest	[1,3,4]	[1,3,4]
queryResultIsEmptyTest	[1,3,5,6,7,8,9,6,7,8,10,11,6,16]	[1, 3] [1, 3, 5, 6] [1, 3, 5, 6, 7] [5, 6] [5, 6, 7, 8] [5, 6, 7, 8, 9] [9, 6] [9, 6, 7, 8] [9, 6, 7, 8, 10] [9, 6, 7, 8, 10, 11] [11, 6] [5, 6, 7] [7, 8] [7, 8, 10]
retrieveLongestKeyWithNotNullValueTest	[1,3,5,6,7,8,10,12,13,15,6,7,8,10,12,13,15,6,7,8,10,12,13,14,15,6,16]	[1, 3] [1, 3, 5, 6] [1, 3, 5, 6, 7] [14, 15, 6, 16] [5, 6] [5, 6, 7, 8] [5, 6, 7, 8, 10] [5, 6, 7, 8, 10, 12, 13] [5, 6, 7, 8, 10, 12, 13, 15] [15, 6] [15, 6, 7, 8] [15, 6, 7, 8, 10] [15, 6, 7, 8, 10, 12, 13] [15, 6, 7, 8, 10, 12, 13, 15] [15, 6, 7, 8, 10, 12, 13, 14, 15] [5, 6, 7] [5, 6, 7, 8, 10, 12] [12, 13, 14] [12, 13, 15, 6, 7] [12, 13, 15, 6, 7, 8, 10, 12] [7, 8] [7, 8, 10]

retrieveLongestKeyWithNotNullValue2Test	[1,3,5,6,7,8,10,11,6,7,8,10,12,13,15,6,7,8,10,12,13,15,6,7,8,10,12,13,14,15,6,16]	[1, 3] [1, 3, 5, 6] [1, 3, 5, 6, 7] [14, 15, 6, 16] [5, 6] [5, 6, 7, 8] [5, 6, 7, 8, 10] [5, 6, 7, 8, 10, 11] [11, 6] [11, 6, 7, 8] [11, 6, 7, 8, 10] [11, 6, 7, 8, 10, 12, 13] [11, 6, 7, 8, 10, 12, 13, 15] [15, 6] [15, 6, 7, 8] [15, 6, 7, 8, 10] [15, 6, 7, 8, 10, 12, 13] [15, 6, 7, 8, 10, 12, 13, 15] [15, 6, 7, 8, 10, 12, 13, 14, 15] [5, 6, 7] [12, 13, 14] [12, 13, 15, 6, 7] [12, 13, 15, 6, 7, 8, 10, 12] [7, 8] [7, 8, 10]
retrieveLongestKeyWithNullValueTest	[1,3,5,6,7,8,10,12,13,15,6,7,8,10,12,13,15,6,7,8,10,12,13,14,15,6,7,8,10,12,13,15,6, 7,8,10,12,13,15,6,16]	[1, 3] [1, 3, 5, 6] [1, 3, 5, 6, 7] [5, 6] [5, 6, 7, 8] [5, 6, 7, 8, 10] [5, 6, 7, 8, 10, 12, 13] [5, 6, 7, 8, 10, 12, 13, 15] [15, 6] [15, 6, 7, 8] [15, 6, 7, 8, 10] [15, 6, 7, 8, 10, 12, 13] [15, 6, 7, 8, 10, 12, 13, 15] [15, 6, 7, 8, 10, 12, 13, 14, 15] [5, 6, 7] [5, 6, 7, 8, 10, 12] [12, 13, 14]

		[12, 13, 15, 6, 7] [12, 13, 14, 15, 6, 7] [12, 13, 15, 6, 7, 8, 10, 12] [12, 13, 14, 15, 6, 7, 8, 10, 12] [7, 8] [7, 8, 10]

Base Choice Coverage

Baseado em *Input State Partitioning*, testou-se o método público put, definindo-se as características, as partições e a *base choice*.

Partitions	Base Choice	Tests
[new key, existent key] [new key prefix, existent key prefix] [empty, not empty] [smallest key, largest key, typical key]	(new key, new key prefix, empty, typical key)	(new key, new key prefix, empty, typical key) (**existent key**, new key prefix, empty, typical key) (new key, **existent key prefix**, empty, typical key) (new key, new key prefix, **not empty**, typical key) (new key, new key prefix, empty, **smallest key**) (new key, new key prefix, empty, **largest key**)