# Contains Instruction coverage

  public boolean contains(String key) {

if (key == null) //I1

throw new IllegalArgumentException("argument to contains() is null"); //I2

return get(key) != null; //I3

}

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Test case values | Expected values | Instruction Coverage |
| containsNullKey | null | IllegalArgumentException | I1 , I2 |
| containsNonNullKey | “someKey” | false | I1, I3 |

# Size instruction coverage

public int size() {

return n; //I1

}

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Test case values | Expected values | Instruction Coverage |
| sizeZeroTest | Not applied | 0 | I1 |

# Get instruction coverage

public T get(String key) {

if (key == null) //I1

throw new IllegalArgumentException("calls get() with null argument"); //I2

if (key.length() == 0) //I3

throw new IllegalArgumentException("key must have length >= 1"); //I4

Node<T> x = get(root, key, 0); //I5

if (x == null) //I6

return null; //I7

return x.val; //I8

}

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Test case values | Expected values | Instruction Coverage |
| getNullKey | null | IllegalArgumentException | I1, I2 |
| getEmptyStringKey | “” | IllegalArgumentException | I1, I3, I4 |
| getNonExistentKey | “someKey” | null | I1, I3, I5, I6, I7 |
| getExistentKey | “key” | “value” | I1, I3, I5, I6, I8 |

# Put instruction coverage

public void put(String key, T val) {

if (key == null) //I1

throw new IllegalArgumentException("calls put() with null key"); //I2

if (!contains(key)) //I3

n++; //I4

root = put(root, key, val, 0); //I5

}

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Test case values | Expected values | Instruction Coverage |
| putNullKey | null, 1 | IllegalArgumentException | I1, I2 |
| putValidNewKey | “someKey”, 1 | No exception | I1, I3, I4, I5 |

# LongestPrefixOf coverage

public String longestPrefixOf(String query) {

if (query == null) //I1

throw new IllegalArgumentException("calls longestPrefixOf() with null argument"); //I2

if (query.length() == 0) //I3

return null; //I4

int length = 0; //I5

Node<T> x = root; //I6

int i = 0; //I7

while (x != null /\*I8\*/ && i < query.length() /\*I9\*/) {

char c = query.charAt(i); //I10

if (c < x.c)/\*I11\*/ x = x.left; //I12

else if (c > x.c)/\*I13\*/ x = x.right; //I14

else {

i++; //I15

if (x.val != null) //I16

length = i; //I17

x = x.mid; //I18

}

}

return query.substring(0, length); //I19

}

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Test case values | Expected values | Instruction Coverage |
| longestPrefixOfNull | null | IllegalArgumentException | I1, I2 |
| longestPrefixOfEmpty | “” | null | I1, I3, I4 |
| longestPrefixOfAllInstructions | “c” | “c” | I1, I3, I5, I6, I7, I8, I9, I10, I11, I12, I13, I14, I15, I16, I17, I18, I19 |

# Keys coverage

public Iterable<String> keys() {

Queue<String> queue = new LinkedList<>(); //I1

collect(root, new StringBuilder(), queue); //I2

return queue; //I3

}

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Test case values | Expected values | Instruction Coverage |
| keysTest | None | Empty Iterator | I1, I2, I3 |

# KeysWithPrefix coverage

public Iterable<String> keysWithPrefix(String prefix) {

if (prefix == null) //I1

throw new IllegalArgumentException("calls keysWithPrefix() with null argument"); //I2

Queue<String> queue = new LinkedList<>(); //I3

Node<T> x = get(root, prefix, 0); //I4

if (x == null) //I5

return queue; //I6

if (x.val != null) //I7

queue.add(prefix); //I8

collect(x.mid, new StringBuilder(prefix), queue); //I9

return queue; //I10

}

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Test case values | Expected values | Instruction Coverage |
| keysWithPrefixNull | null | IllegalArgumentException | I1, I2 |
| keysWithPrefixEmpty | “somePrefix” | Empty Iterator | I1, I3, I4, I5, I6 |
| keysWithPrefixContains | “c” | Iterator with size 1 | I1, I3, I4, I5, I6, 7, I8, I9, I10 |

# KeysThatMatch coverage

public Iterable<String> keysThatMatch(String pattern) {

Queue<String> queue = new LinkedList<>(); //I1

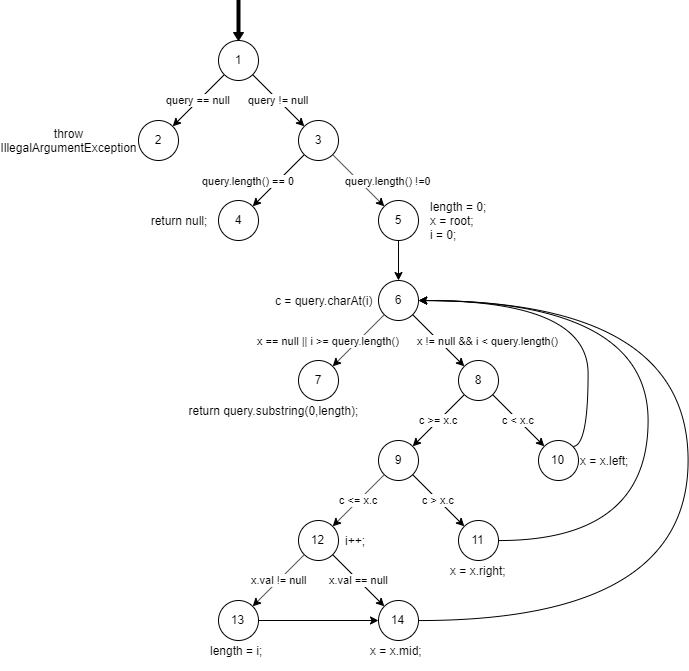
collect(root, new StringBuilder(), 0, pattern, queue); //I2

return queue; //I3

}

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Test case values | Expected values | Instruction Coverage |
| keysThatMatchTest | “pattern” | Empty Iterator | I1, I2, I3 |

# Edge coverage



|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Test case values | Expected values | Edges  Covered |
| lpoEdgeCoverage1 | null | IllegalArgumentException | [1,2] |
| lpoEdgeCoverage2 | “” | null | [1,3],[3,4] |
| lpoEdgeCoverage3 | “a” | “” | [1,3],[3,5],[5,6], [6,8], [8,10], [10,6],[6,7] |
| lpoEdgeCoverage4 | “d” | “” | [1,3],[3,5],[5,6], [6,8],[8,9], [9,11], [11,6],  [6,7] |
| lpoEdgeCoverage5 | “c” | “” | [1,3],[3,5],[5,6], [6,8],[8,9],[9,12], [12,14], [14,6],  [6,7] |
| lpoEdgeCoverage6 | “c” | “c” | [1,3],[3,5],[5,6], [6,8],[8,9],[9,12], [12,13], [13,14], [14,6], [6,7] |

# Logic based coverage

p1 : c1, where c1: query == null

p2: c2, where c2: query.length() == 0

p3: c3 && c4, where c3: x != null; c4: i < query.length()

p4: c5, where c5: c < x.c

p5: c6, where c6: c > x.c

p6: c7, where c7: x.val != null

Reachability

P1 r(p1) = true

P2 r(p2) = r(p1) && !p1 = true && query != null

P3 r(p3) = r(p2) && !p2 = true && query != null && query.length != 0

P4 r(p4) = r(p3) && p3 = true && query != null && query.length != 0 && x != null && i < query.length()

P5 r(p5) = r(p4) && !p4 = true && query != null && query.length != 0 && x != null && i < query.length() && c >= x.c

P6 r(p6) = r(p5) && !p5 = true && query != null && query.length != 0 && x != null && i < query.length() && c <= x.c

# Put method Base Choice Coverage

Parâmetros que influenciem a execução do método:

* Parâmetros de entrada da função: String key, T value
* Campos da instância: Node<T> node

Key characteristics:

* Null : true | false
* Empty : true | false

Value characteristics:

* Null : true | false

Node characteristics:

* Null : true | false

“Trie already includes the new key”

* Base choice (false,false,false,false)

“Trie already includes some new key prefix”

* Base choice (false,false,false,false)

“Trie is empty”

* Base choice (false,false,false,true)