Filip Ioana Alexandra

Grupa 343

Proiect – Testarea sistemelor software

Problema nr. III – Licența <https://www.infoarena.ro/problema/licenta>

Cuprins:

**(I)** Generarea datelor de test folosind metode funcționale

**(a)** Partiționarea în clase de echivalență

**(b)** Analiza valorilor de frontieră

**(c)** Partiționarea in categorii

**(II)** Testarea structurală – Transformarea programului într-un graf orientat

1. Acoperirea la nivel de instrucțiune
2. Acoperirea la nivel de decizie
3. Acoperirea la nivel de condiție

**(III)** Testarea circuitelor independente

1. Formula lui McCabe
2. Circuite independente

**(IV)** Testarea la nivel de cale

1. Expresia regulată
2. Numărul de căi

**(V)** Mutanți

1. **Generarea datelor de test folosind metode funcționale**
2. **Partiționarea în clase de echivalență**

Domeniul de intrări:

Există 5 intrări:

* un întreg pozitiv k

Presupun că parametrul k trebuie să fie între 1 și 60, deci disting 3 clase de echivalență:

K\_1 = 1…60

K\_2 = {k | k < 1}

K\_3 = {k | k > 60}

* un întreg pozitiv n

Presupun că parametrul n trebuie să fie între 1 și 6, deci disting 3 clase de echivalență:

N\_1 = 1…6

N\_2 = {k | k < 1}

N\_3 = {k | k > 6}

* n perechi de numere întregi pozitive – S

Deoarece S reprezintă o mulțime de n intervale de minute disjuncte, voi presupune că sunt îndeplinite următoarele condiții :

* + 0 ≤ XS[i] < YS[i] ≤ 60, oricare ar fi 1 ≤ i ≤ n si (XS[i], YS[i]) pereche din S
  + |S| = n
* un întreg pozitiv m

Presupun că parametrul m trebuie să fie între 1 și 6, deci disting 3 clase de echivalență:

M\_1 = 1…6

M\_2 = {k | k < 1}

M\_3 = {k | k > 6}

* m perechi de numere întregi pozitive – T

Deoarece T reprezintă o mulțime de m intervale de minute, voi presupune că sunt indeplinite următoarele condiții :

* 0 ≤ XT[i] < YT[i] ≤ 60, oricare ar fi 1 ≤ i ≤ m si (XT[i], YT[i]) pereche din T
* |T| = m

Domeniul de ieșiri:

Constă în următoarele 2 posibile răspunsuri:

* Intervalul (X, Y), unde Y-X = k, care reprezintă intervalul de lungime k al intersecției unui interval din S cu un interval din T. Dacă există multe intervale de forma aceasta, se alege cel cu X minim.
* Un mesaj ce arată ca nu există nicio solutie : -1.

Acestea sunt folosite pentru a impărți domeniul de intrare în 2 clase :

* I\_1(T, k) = { S | există un interval din S care intersectat cu un interval din T să rezulte un interval de lungime ≥ k}
* I\_2(T, k) = { S | nu există niciun interval din S care intersectat cu un interval din T să rezulte un interval de lungime ≥ k }

Clasele de echivalență globale, pentru întregul program, se obțin prin combinarea claselor individuale, prezentate anterior :

C\_1111 = { (k, n, S, m, T) | k \in K\_1, n \in N\_1, |S| = n, m \in M\_1, |T| = m, S \in I\_1(T, k) }

C\_1112 = { (k, n, S, m, T) | k \in K\_1, n \in N\_1, |S| = n, m \in M\_1, |T| = m, S \in I\_2(T, k) }

C\_112 = { (k, n, S, m, T) | k \in K\_1, n \in N\_1, |S| = n, m \in M\_2 }

C\_113 = { (k, n, S, m, T) | k \in K\_1, n \in N\_1, |S| = n, m \in M\_3 }

C\_12 = { (k, n, S, m, T) | k \in K\_1, n \in N\_2 }

C\_13 = { (k, n, S, m, T) | k \in K\_1, n \in N\_3 }

C\_2 = { (k, n, S, m, T) | k \in K\_2 }

C\_3 = { (k, n, S, m, T) | k \in K\_3 }

Rezultă deci, 8 clase de echivalență globale.

Setul de date de test se alcătuiește în funcție de aceste clase de echivalență globale, alegându-se date de intrare specifice fiecăreia, precum:

c\_1111 : (4, 2, {(1 10), (11 13)}, 2,{(2 4), (5 12)})

c\_1112 : (7, 2, {(1 10), (11 13)}, 2,{(2 4), (5 12)})

c\_112 : (4, 2, {(1 10), (11 13)}, 0, \_ )

c\_113 : (4, 2, {(1 10), (11 13)}, 7, \_ )

c\_12 : (4, 0, \_ , \_ , \_ )

c\_13 : (4, 7, \_ , \_ , \_ )

c\_2 : (0, \_ , \_ , \_ , \_ )

c\_3 : (61, \_ , \_ , \_ , \_ )

8 seturi de date de test

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Intrări** | | | | | **Rezultatul afișat (expected)** |
| **k** | **n** | **S** | **m** | **T** |
| 4 | 2 | {(1 10), (11 13)} | 2 | {(2 4), (5 12)} | (5, 9) |
| 7 | 2 | {(1 10), (11 13)} | 2 | {(2 4), (5 12)} | -1 |
| 4 | 2 | {(1 10), (11 13)} | 0 | - | Cere introducerea valorii m mai mare decât 0 |
| 4 | 2 | {(1 10), (11 13)} | 0 | - | Cere introducerea valorii m mai mică decât 7 |
| 4 | 0 | - | - | - | Cere introducerea valorii n mai mare decât 0 |
| 4 | 0 | - | - | - | Cere introducerea valorii n mai mare decât 0 |
| 4 | 7 | - | - | - | Cere introducerea valorii n mai mică decât 7 |
| 0 | - | - | - | - | Cere introducerea valorii k mai mare decât 0 |
| 61 | - | - | - | - | Cere introducerea valorii k mai mică decât 61 |

|  |
| --- |
| @Test  public void PartitionareClaseEchivalenta() throws IOException {  // c\_1111 : (4, 2, {(1 10), (11 13)}, 2,{(2 4), (5 12)})  testClass.licenta("src/pachet1/PartitionareClaseEchivalenta/1.1.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareClaseEchivalenta/expected1.1.txt")),  FileUtils.readLines(new File("licenta.out")));  // c\_1112 : (7, 2, {(1 10), (11 13)}, 2,{(2 4), (5 12)})  testClass.licenta("src/pachet1/PartitionareClaseEchivalenta/1.2.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareClaseEchivalenta/expected1.2.txt")),  FileUtils.readLines(new File("licenta.out")));  // c\_112 : (4, 2, {(1 10), (11 13)}, 0, \_ )  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareClaseEchivalenta/1.3.in"));  // c\_113 : (4, 2, {(1 10), (11 13)}, 7, \_ )  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareClaseEchivalenta/1.4.in"));  // c\_12 : (4, 0, \_ , \_ , \_ )  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareClaseEchivalenta/1.5.in"));  // c\_13 : (4, 7, \_ , \_ , \_ )  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareClaseEchivalenta/1.6.in"));  // c\_2 : (0, \_ , \_ , \_ , \_ )  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareClaseEchivalenta/1.7.in"));  // c\_3 : (61, \_ , \_ , \_ , \_ )  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareClaseEchivalenta/1.8.in"));  } |

**b. Analiza valorilor de frontieră**

Odată ce am identificat clasele, valorile de frontieră sunt ușor de găsit:

* Valorile 0, 1, 60, 61 pentru k
* Valorile 0, 1, 6, 7 pentru n
* Valorile 0, 1, 6, 7 pentru m
* Primul interval din S cu primul interval din T sunt primele din mulțimi a căror intersecție rezultă un interval cu lungime ≥ k; sau ultimul interval din S cu ultimul interval din T sunt primele din mulțimi a căror intersectie rezulta un interval ≥ k.

Observație: Înainte de a verifica intersectiile, S si T vor fi mai intâi sortate crescător după capătul stâng.

Deci voi testa urmatoarele valori:

* K\_1: 1, 60
* K\_2: 0
* K\_3: 61
* N\_1: 1, 6
* N\_2: 0
* N\_3: 7
* M\_1: 1, 6
* M\_2: 0
* M\_3: 7
* I\_1 := i\_11 este primul intervalul de lungime ≥ k rezultat din intersecția primului interval din S cu primul interval din T;

i\_12 este primul intervalul de lungime ≥ k rezultat din intersecția ultimului interval din S cu ultimul interval din T.

Deci pentru C\_1111 voi alege 8 seturi de date de test, după cum urmează:

1. k = n = m = 1 și singurul interval din S se intersectează cu singurul interval din T

(1, 1, {(1 5)}, 1, {(2 9)}) - 2 3

1. k = n = 1, m = 6 și singurul interval din S se intersectează cu primul interval din T

(1, 1, {(1 3)}, 6, {(7 30), (2 4), (31 36), (38 39), (44 56), (5 6)}) - 2 3

1. k = n = 1, m = 6 și singurul interval din S se intersectează cu ultimul interval din T

(1, 1, {(50 60)}, 6, {(7 30), (2 4), (31 36), (38 39), (44 56), (5 6)}) - 50 51

1. k = m = 1, n = 6 și primul interval din S se intersectează cu singurul interval din T

(1, 6, {(1 3), (6 9), (4 5), (10 20), (30 56), (21 22)}, 1, {(2 4)}) - 2 3

1. k = m = 1, n = 6 și ultimul interval din S se intersectează cu singurul interval din T

(1, 6, {(1 3), (6 9), (4 5), (10 20), (30 56), (21 22)}, 1, {(50 60)}) - 50 51

1. k = 1, n = m = 6 și primul interval din S se intersectează cu primul interval din T

(1, 6, {(1 3), (6 9), (4 5), (10 20), (30 56), (21 22)}, 6, {(7 30), (2 4), (31 36), ( 38 39), (44 56), (5 6)}) - 2 3

1. k = 1, n = m = 6 și ultimul interval din S se intersectează cu ultimul interval din T

(1, 6, {(23 26), (16 20), (50 59), (7 8), (33 38), (1 3)}, 6, {(48 53), (4 5), (10 15), (22 23), (40 44), (29 32)}) - 50 51

1. k = 60, n = m = 1 și singurul interval din S să se intersecteze cu singurul interval din T și să rezulte un interval de lungime 60

(60, 1, {(0 60)}, 1, {(0 60)}) - 0 60

Pentru C\_1112 voi alege 8 seturi de date de test (toate cu outputul -1), astfel:

1. k = n = m = 1, dar singurul interval din S nu se intersectează cu singurul din T

(1, 1, {(1 5)}, 1, {(30 49)})

1. k = n = 1, m = 6, dar singurul interval din S nu se intersectează cu niciun interval din T

(1, 1, {(1 3)}, 6, {(7 30), (57 58), (31 36), ( 38 39), (44 56), (5 6)})

1. k = m = 1, n = 6 si niciun interval din S nu se intersectează cu singurul interval din T

(1, 6, {(1 3), (6 9), (4 5), (10 20), (30 56), (21 22)}, 1, {(50 54)})

1. k = 1, n = m = 6 și niciun interval din S nu se intersectează cu vreun interval din T

(1, 6, {(23 26), (16 20), (50 51), (7 8), (33 38), (1 3)}, 6, {(48 49), (4 5), (10 15), (22 23), (40 44), (29 32)})

1. k = 60, n = m = 1, dar intersecția singurului interval din S cu singurul din T să fie < k

(60, 1, {(0 60)}, 1, {(40 60)})

1. k = 60, n = 1, m = 6, dar intersecția singurului interval din S cu oricare din T sigur va fi < k

(60, 1, {(0 60)}, 6, {(23 26), (16 20), (50 51), (7 8), (33 38), (1 3)})

1. k = 60, n = 6, m = 1, dar intersecția singurului interval din T cu oricare din S sigur va fi < k

(60, 6, {(23 26), (16 20), (50 51), (7 8), (33 38), (1 3)}, 1, {(0 60)})

1. k = 60, n = m = 6, dar intersecția oricărui interval din S cu oricare interval din T va fi < k

(60, 6, {(23 26), (16 20), (50 51), (7 8), (33 38), (1 3)}, 6, {(48 49), (4 5), (10 15), (22 23), (40 44), (29 32)})

Iar pentru restul claselor voi lua:

C\_112 : (4, 2, {(1 10), (11 13)}, 0, \_ )

C\_113 : (4, 2, {(1 10), (11 13)}, 7, \_ )

C\_12 : (4, 0, \_ , \_ , \_ )

C\_13 : (4, 7, \_ , \_ , \_ )

C\_2 : (0, \_ , \_ , \_ , \_ )

C\_3 : (61, \_ , \_ , \_ , \_ )

De unde rezultă 22 seturi de date de test.

|  |
| --- |
| @Test  public void ValoriFrontiera() throws IOException {  testClass.licenta("src/pachet1/ValoriFrontiera/2.1.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.1.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/ValoriFrontiera/2.2.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.2.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/ValoriFrontiera/2.3.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.3.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/ValoriFrontiera/2.4.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.4.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/ValoriFrontiera/2.5.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.5.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/ValoriFrontiera/2.6.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.6.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/ValoriFrontiera/2.7.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.7.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/ValoriFrontiera/2.8.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.8.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/ValoriFrontiera/2.9.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.9.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/ValoriFrontiera/2.10.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.10.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/ValoriFrontiera/2.11.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.11.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/ValoriFrontiera/2.12.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.12.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/ValoriFrontiera/2.13.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.13.txt")),  FileUtils.readLines(new File("licenta.out")));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/ValoriFrontiera/2.14.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/ValoriFrontiera/2.15.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/ValoriFrontiera/2.16.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/ValoriFrontiera/2.17.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/ValoriFrontiera/2.18.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/ValoriFrontiera/2.19.in"));  testClass.licenta("src/pachet1/ValoriFrontiera/2.20.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.20.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/ValoriFrontiera/2.21.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.21.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/ValoriFrontiera/2.22.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/ValoriFrontiera/expected2.22.txt")),  FileUtils.readLines(new File("licenta.out")));  } |

**c. Partiționarea in categorii**

1. Descompunerea in unități: o singură unitate – funcția licenta.
2. Indentificarea parametrilor: k, n, S, m, T.
3. Găsirea categoriilor:
   * k: dacă este în intervalul valid 1…60
   * n: dacă este în intervalul valid 1…6
   * S: dacă este de lungime minimă, maximă sau intermediară
   * m: dacă este în intervalul valid 1…6
   * T: dacă este de lungime minimă, maximă sau intermediară
   * S si T: dacă există sau nu o intersecție a intervalelor de lungime ≥ k; iar dacă exista aceasta este între primele, ultimele intervale sau între acestea.
4. Partiționarea fiecărei categorii in alternative:
   * k: <0, 0, 1, 2, … , 59, 60, 61, >61
   * n: <0, 1, 2, … , 5, 6, >6
   * S: lungime minimă, maximă sau intermediară
   * m: <0, 1, 2, … , 5, 6, >6
   * T: lungime minimă, maximă sau intermediară
   * S si T: intersecția primului interval din S cu primul interval din T este ≥ k, intersecția ultimului interval din S cu ultimul interval din T este ≥ k, există o altă pereche de intervale din S și T a căror intersecție sa fie ≥ k, sau nu există nicio pereche de intervale din S și T a căror intersecție să fie ≥ k.
5. Specificația de testare

* k
  1. {k | k < 0}
  2. 0
  3. 1 [ok\_k, lungime1\_k]
  4. 2…59 [ok\_k, lungime\_medie\_k]
  5. 60 [ok\_k, lungime60\_k]
  6. 61
  7. {k | k > 61}
* n
  1. {n | n < 0}
  2. 0
  3. 1 [ok\_n, lungime1\_n]
  4. 2…5 [ok\_n, lungime\_medie\_n]
  5. 6 [ok\_n, lungime6\_n]
  6. 7
  7. {n | n > 7}
* S
  1. {S | |S| = 1} [if ok\_n and lungime1\_n]
  2. {S | 1 < |S| < 6} [if ok\_n and lungime\_medie\_n]
  3. {S | |S| = 6} [if ok\_n and lungime6\_n]
* m
  1. {m | m < 0}
  2. 0
  3. 1 [ok\_m, lungime1\_m]
  4. 2…5 [ok\_m, lungime\_medie\_m]
  5. 6 [ok\_m, lungime6\_m]
  6. 7
  7. {m | > 7}
* T
  1. {T | |T| = 1} [if ok\_m and lungime1\_m]
  2. {T | 1 < |T| < 6} [if ok\_m and lungime\_medie\_m]
  3. {T | |T| = 6} [if ok\_m and lungime6\_m]
* S și T
  1. {S, T | intersecția primului interval din S cu primul interval din T este de lungime ≥ k} [if ok\_k and ok\_n and ok\_m]
  2. {S, T | intersecția ultimului interval din S cu ultimul interval din T este de lungime ≥ k} [if ok\_k and ok\_n and not lungime1\_n and ok\_m and not lungime1\_m]
  3. {S, T | există o altă pereche de intervale din S ți T a căror intersecție să fie ≥ k} [if ok\_k and ok\_n and not lungime1\_n and ok\_m and not lungime1\_m]
  4. {S, T | nu există nicio pereche de intervale din S și T a căror intersecție să fie ≥ k } [if ok\_k and ok\_n and ok\_m]

Din specificația de testare ar trebui sa rezulte 7 \* 7 \* 3 \* 7 \* 3 \* 4 = 12348 de cazuri de testare. Pe de altă parte unele combinații de alternative nu au sens și pot fi eliminate. Acest lucru se poate realiza adăugând constrângeri acestor alternative. Acestea sunt date fie de proprietățile alternativelor fie de condițiile bazate pe ele. În acest caz, alternativele vor fi combinate doar dacă condițiile de selecție sunt satisfacute. Prin acest procedeu, cazurile de testare vor fi reduse la:

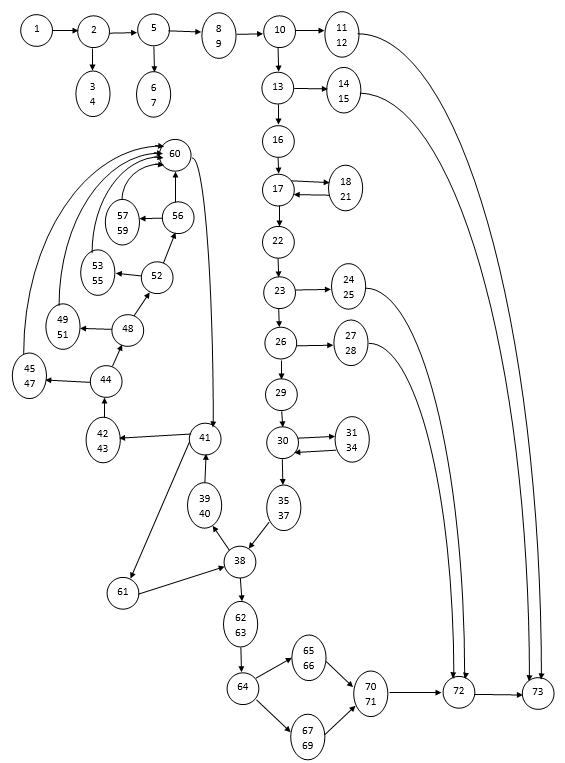
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Intrări** | | | | | **Rezultatul afișat (expected)** |
| **k** | **n** | **S** | **m** | **T** |
| -5 | - | - | - | - | Cere introducerea valorii k mai mare decât 0 |
| 0 | - | - | - | - | Cere introducerea valorii k mai mare decât 0 |
| 61 | - | - | - | - | Cere introducerea valorii k mai mică decât 61 |
| 65 | - | - | - | - | Cere introducerea valorii k mai mică decât 61 |
| 1 | -5 | - | - | - | Cere introducerea valorii n mai mare decât 0 |
| 1 | 0 | - | - | - | Cere introducerea valorii n mai mare decât 0 |
| 1 | 7 | - | - | - | Cere introducerea valorii n mai mică decât 7 |
| 1 | 11 | - | - | - | Cere introducerea valorii n mai mică decât 7 |
| 1 | 1 | {(1 10)} | -5 | - | Cere introducerea valorii m mai mare decât 0 |
| 1 | 1 | {(1 10)} | 0 | - | Cere introducerea valorii m mai mare decât 0 |
| 1 | 1 | {(1 10)} | 7 | - | Cere introducerea valorii m mai mică decât 7 |
| 1 | 1 | {(1 10)} | 11 | - | Cere introducerea valorii m mai mică decât 7 |
| 1 | 1 | {(1 10)} | 1 | {(2 8)} | (2, 3) |
| 1 | 1 | {(1 5)} | 1 | {(9 16)} | -1 |
| 1 | 1 | {(1 13)} | 2 | {(2 8), (11 12)} | (2, 3) |
| 1 | 1 | {(9 13)} | 2 | {(5 8), (9 16)} | (9, 10) |
| 1 | 1 | {(1 5)} | 2 | {(5 10), (20 26)} | -1 |
| 1 | 1 | {(1 3)} | 6 | {(8 10), (20 21), (2 4), (13 17), (30 40), (5 7)} | (2 3) |
| 1 | 1 | {(32 35)} | 6 | {(8 10), (20 21), (2 4), (13 17), (30 40), (5 7)} | (32, 33) |
| 1 | 1 | {(14 25)} | 6 | {(8 10), (20 21), (2 4),  (13 17),(30 40), (5 7)} | (14, 15) |
| 1 | 1 | {(21 25)} | 6 | {(8 10), (20 21), (2 4), (13 17), (30 40), (5 7)} | -1 |
| 1 | 2 | {(1 10),  (11 13)} | -5 | - | Cere introducerea valorii m mai mare decât 0 |
| 1 | 2 | {(1 5),  (9 13)} | 0 | - | Cere introducerea valorii m mai mare decât 0 |
| 1 | 2 | {(1 5),  (9 13)} | 7 | - | Cere introducerea valorii m mai mică decât 7 |
| 1 | 2 | {(11, 15),  (1 3)} | 11 | - | Cere introducerea valorii m mai mică decât 7 |
| 1 | 2 | {(1 10),  (11 13)} | 1 | {(2 8)} | (2, 6) |
| 1 | 2 | {(1 10),  (11 13)} | 1 | {(11 16)} | (11, 12) |
| 1 | 2 | {(1 10), (11 13)} | 1 | {(20 21)} | -1 |
| 1 | 2 | {(1 10),  (11 13)} | 2 | {(2 8), (11 12)} | (2, 3) |
| 1 | 2 | {(1 5),  (9 13)} | 2 | {(5 8), (9 16)} | (9, 10) |
| 1 | 2 | {(1 5),  (9 13)} | 2 | {(5 10), (20 26)} | (9, 10) |
| 1 | 2 | {(11, 15),  (1 3)} | 2 | {(8 10), (20 21)} | -1 |
| 1 | 2 | {(1 10), (11 13)} | 6 | {(8 10), (20 21), (2 4), (13 17), (30 40), (5 7)} | (2, 3) |
| 1 | 2 | {(1 5),  (32 35)} | 6 | {(8 10), (20 21), (2 4), (13 17), (30 40), (5 7)} | (32, 33) |
| 1 | 2 | {(41 42),  (14 16)} | 6 | {(8 10), (20 21), (2 4), (13 17), (30 40), (5 7)} | (14 15) |
| 1 | 2 | {(25 28),  (12 13)} | 6 | {(8 10), (20 21), (2 4), (13 17), (30 40), (5 7)} | -1 |
| 1 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | -5 | - | Cere introducerea valorii m mai mare decât 0 |
| 1 | 6 | {(8 10), (20 21), (2 4), (13 17), (30 40), (5 7)} | 0 | - | Cere introducerea valorii m mai mare decât 0 |
| 1 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 7 | - | Cere introducerea valorii m mai mică decât 7 |
| 1 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 11 | - | Cere introducerea valorii m mai mică decât 7 |
| 1 | 6 | {(8 10), (20 21), (2 4),  (13 17),(30 40), (5 7)} | 1 | {(1 3)} | (2, 3) |
| 1 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 1 | {(32 35)} | (32, 33) |
| 1 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 1 | {(14 25)} | (14, 15) |
| 1 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 1 | {(21 25)} | -1 |
| 1 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 2 | {(1 10), (11 13)} | (2, 3) |
| 1 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 2 | {(1 5), (32 35)} | (32, 33) |
| 1 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 2 | {(41 42), (14 16)} | (14 15) |
| 1 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 2 | {(25 28), (12 13)} | -1 |
| 1 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 6 | {(25 28), (5 12), (13 21),  (1 4), (30 33), (34 41)} | (2, 3) |
| 1 | 6 | {(8 10), (20 21), (4 5),  (13 17), (35 40), (5 7)} | 6 | {(25 28), (5 12), (13 21),  (1 4), (30 33), (34 41)} | (35, 36) |
| 1 | 6 | {(8 10), (20 21), (1 3),  (13 17), (30 32), (5 7)} | 6 | {(29 30), (25 28), (3 4),  (22 24), (33 40), (8 13)} | (8 9) |
| 1 | 6 | {(50 51), (20 21), (1 3),  (13 17), (30 32), (5 7)} | 6 | {(29 30), (25 28), (3 4),  (22 24), (33 40), (8 13)} | -1 |
| 4 | -5 | - | - | - | Cere introducerea valorii n mai mare decât 0 |
| 4 | 0 | - | - | - | Cere introducerea valorii n mai mare decât 0 |
| 4 | 7 | - | - | - | Cere introducerea valorii n mai mică decât 7 |
| 4 | 11 | - | - | - | Cere introducerea valorii n mai mică decât 7 |
| 4 | 1 | {(1 10)} | -5 | - | Cere introducerea valorii m mai mare decât 0 |
| 4 | 1 | {(1 10)} | 0 | - | Cere introducerea valorii m mai mare decât 0 |
| 4 | 1 | {(1 10)} | 7 | - | Cere introducerea valorii m mai mică decât 7 |
| 4 | 1 | {(1 10)} | 11 | - | Cere introducerea valorii m mai mică decât 7 |
| 4 | 1 | {(1 10)} | 1 | {(2 8)} | (2, 6) |
| 4 | 1 | {(1 5)} | 1 | {(9 16)} | -1 |
| 4 | 1 | {(1 13)} | 2 | {(2 8), (11 12)} | (2, 6) |
| 4 | 1 | {(9 13)} | 2 | {(5 8), (9 16)} | (9, 13) |
| 4 | 1 | {(1 5)} | 2 | {(5 10), (20 26)} | -1 |
| 4 | 1 | {(1 8)} | 6 | {(8 10), (20 21), (1 6),  (13 17), (30 40), (11 12)} | (1 5) |
| 4 | 1 | {(32 44)} | 6 | {(8 10), (20 21), (2 4), (13 17), (30 40), (5 7)} | (32, 36) |
| 4 | 1 | {(14 25)} | 6 | {(8 10), (20 21), (2 4),  (13 19), (30 40), (5 7)} | (14, 18) |
| 4 | 1 | {(21 25)} | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | -1 |
| 4 | 2 | {(1 10),  (11 13)} | -5 | - | Cere introducerea valorii m mai mare decât 0 |
| 4 | 2 | {(1 5),  (9 13)} | 0 | - | Cere introducerea valorii m mai mare decât 0 |
| 4 | 2 | {(1 5),  (9 13)} | 7 | - | Cere introducerea valorii m mai mică decât 7 |
| 4 | 2 | {(11, 15),  (1 3)} | 11 | - | Cere introducerea valorii m mai mică decât 7 |
| 4 | 2 | {(1 10),  (11 13)} | 1 | {(2 8)} | (2, 6) |
| 4 | 2 | {(1 10),  (11 19)} | 1 | {(11 16)} | (11, 15) |
| 4 | 2 | {(1 10), (11 13)} | 1 | {(10 12)} | -1 |
| 4 | 2 | {(1 10),  (11 13)} | 2 | {(2 8), (11 12)} | (2, 6) |
| 4 | 2 | {(1 5),  (9 13)} | 2 | {(5 8), (9 16)} | (9, 13) |
| 4 | 2 | {(1 5),  (9 13)} | 2 | {(5 19), (20 26)} | (9, 13) |
| 4 | 2 | {(11, 15),  (1 3)} | 2 | {(8 10), (20 21)} | -1 |
| 4 | 2 | {(1 10), (11 13)} | 6 | {(8 10), (20 21), (1 5), (13 17), (30 40), (6 7)} | (1, 5) |
| 4 | 2 | {(1 5),  (32 38)} | 6 | {(8 10), (20 21), (2 4), (13 17), (30 40), (5 7)} | (32, 36) |
| 4 | 2 | {(41 42),  (10 20)} | 6 | {(8 10), (20 21), (2 4), (13 17), (30 40), (5 7)} | (13 17) |
| 4 | 2 | {(25 28),  (12 13)} | 6 | {(8 10), (20 21), (2 4),  (13 17),(30 40), (5 7)} | -1 |
| 4 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | -5 | - | Cere introducerea valorii m mai mare decât 0 |
| 4 | 6 | {(8 10), (20 21), (2 4), (13 17), (30 40), (5 7)} | 0 | - | Cere introducerea valorii m mai mare decât 0 |
| 4 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 7 | - | Cere introducerea valorii m mai mică decât 7 |
| 4 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 11 | - | Cere introducerea valorii m mai mică decât 7 |
| 4 | 6 | {(8 10), (20 21), (1 6),  (13 17), (30 40), (11 12)} | 1 | {(1 8)} | (1 5) |
| 4 | 6 | {(8 10), (20 21), (2 4), (13 17), (30 40), (5 7)} | 1 | {(32 44)} | (32, 36) |
| 4 | 6 | {(8 10), (20 21), (2 4),  (13 19), (30 40), (5 7)} | 1 | {(14 25)} | (14, 18) |
| 4 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 1 | {(21 25)} | -1 |
| 4 | 6 | {(8 10), (20 21), (1 5),  (13 17), (30 40), (6 7)} | 2 | {(1 10), (11 13)} | (1, 5) |
| 4 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 2 | {(1 5), (32 38)} | (32, 36) |
| 4 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 2 | {(41 42), (10 20)} | (13 17) |
| 4 | 6 | {(8 10), (20 21), (2 4),  (13 17), (30 40), (5 7)} | 2 | {(25 28), (12 13)} | -1 |
| 4 | 6 | {(8 10), (20 21), (1 5),  (13 17), (30 40), (6 7)} | 6 | {(25 28), (8 12), (13 21),  (1 7), (30 33), (34 41)} | (1, 5) |
| 4 | 6 | {(8 10), (20 21), (4 5),  (13 17), (35 40), (5 7)} | 6 | {(25 28), (5 12), (13 21),  (1 4), (30 33), (34 41)} | (35, 39) |
| 4 | 6 | {(8 14), (20 21), (1 3),  (15 17), (30 32), (5 7)} | 6 | {(29 30), (25 28), (3 4),  (22 24), (33 40), (5 13)} | (8 12) |
| 4 | 6 | {(8 10), (20 21), (1 3),  (13 17), (30 32), (5 7)} | 6 | {(29 30), (25 28), (3 4),  (22 24), (33 40), (11 13)} | -1 |
| 60 | -5 | - | - | - | Cere introducerea valorii n mai mare decât 0 |
| 60 | 0 | - | - | - | Cere introducerea valorii n mai mare decât 0 |
| 60 | 7 | - | - | - | Cere introducerea valorii n mai mică decât 7 |
| 60 | 11 | - | - | - | Cere introducerea valorii n mai mică decât 7 |
| 60 | 1 | {(1 10)} | -5 | - | Cere introducerea valorii m mai mare decât 0 |
| 60 | 1 | {(1 10)} | 0 | - | Cere introducerea valorii m mai mare decât 0 |
| 60 | 1 | {(1 10)} | 7 | - | Cere introducerea valorii m mai mică decât 7 |
| 60 | 1 | {(1 10)} | 11 | - | Cere introducerea valorii m mai mică decât 7 |
| 60 | 1 | {(0 60)} | 1 | {(0 60)} | (0, 60) |
| 60 | 1 | {(0 60)} | 1 | {(10 60)} | -1 |
| 60 | 1 | {(0 60)} | 2 | {(1 5), (10 60)} | -1 |

|  |
| --- |
| @Test  public void PartitionareCategorii() throws IOException {  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.1.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.2.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.3.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.4.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.5.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.6.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.7.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.8.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.9.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.10.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.11.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.12.in"));  testClass.licenta("src/pachet1/PartitionareCategorii/3.14.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.14.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.15.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.15.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.16.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.16.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.17.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.17.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.18.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.18.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.19.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.19.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.20.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.20.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.21.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.21.txt")),  FileUtils.readLines(new File("licenta.out")));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.22.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.23.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.24.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.25.in"));  testClass.licenta("src/pachet1/PartitionareCategorii/3.26.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.26.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.27.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.27.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.28.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.28.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.29.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.29.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.30.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.30.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.31.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.31.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.32.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.32.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.33.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.33.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.34.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.34.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.35.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.35.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.36.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.36.txt")),  FileUtils.readLines(new File("licenta.out")));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.37.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.38.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.39.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.40.in"));  testClass.licenta("src/pachet1/PartitionareCategorii/3.41.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.41.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.42.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.42.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.43.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.43.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.44.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.44.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.45.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.45.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.46.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.46.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.47.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.47.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.48.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.48.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.49.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.49.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.50.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.50.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.51.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.51.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.52.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.52.txt")),  FileUtils.readLines(new File("licenta.out")));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.53.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.54.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.55.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.56.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.57.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.58.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.59.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/PartitionareCategorii/3.60.in"));  testClass.licenta("src/pachet1/PartitionareCategorii/3.61.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.61.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.62.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.62.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.63.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.63.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.64.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.64.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.65.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.65.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.66.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.66.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.67.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.67.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.68.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.68.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/PartitionareCategorii/3.69.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/PartitionareCategorii/expected3.69.txt")),  FileUtils.readLines(new File("licenta.out")));  }  … |

Aproximativ 110 teste.

1. **Testarea structurală – Transformarea programului într-un graf orientat**

|  |
| --- |
| public static void licenta(String inputFile) throws IOException {  List<Interval> intervals1 = new ArrayList<Interval>();  List<Interval> intervals2 = new ArrayList<Interval>();  Interval finalI = new Interval(0,0);  int k, m, n, i, j, x, y, m1, m2, p1, p2;  boolean gasit = false;  Scanner scanner = new Scanner(new File(inputFile));  **1** k = scanner.nextInt();  **2** if (k > 60) {  **3** throw new IllegalArgumentException("K trebuie sa fie mai mic decat 61!");  **4** }  **5** else if (k < 1) {  **6** throw new IllegalArgumentException("K trebuie sa fie mai mare decat 0!");  **7** }  **8** else {  **9** n = scanner.nextInt();  **10** if (n > 6) {  **11** throw new IllegalArgumentException("N trebuie sa fie mai mic decat 7!");  **12** }  **13** else if (n < 1) {  **14** throw new IllegalArgumentException("N trebuie sa fie mai mare decat 0!");  **15** }  **16** else {  **17** for(i = 0; i < n; i++) {  **18**  x = scanner.nextInt();  **19** y = scanner.nextInt();  **20** intervals1.add(new Interval(x, y));  **21** }  **22** m = scanner.nextInt();  **23** if (m > 6) {  **24** throw new IllegalArgumentException("M trebuie sa fie mai mic decat 7!");  **25** }  **26** else if (m < 1) {  **27** throw new IllegalArgumentException("M trebuie sa fie mai mare decat 0!");  **28** }  **29** else {  **30** for(i = 0; i < m; i++) {  **31** x = scanner.nextInt();  **32** y = scanner.nextInt();  **33** intervals2.add(new Interval(x, y));  **34** }  **35** scanner.close();  **36** Collections.sort(intervals1, new Compare());  **37** Collections.sort(intervals2, new Compare());  **38** for(i = 0; i < n && gasit == false; i++) {  **39** m1 = intervals1.get(i).a;  **40** m2 = intervals1.get(i).b;  **41** for(j = 0; j < m && gasit == false; j++) {  **42** p1 = intervals2.get(j).a;  **43** p2 = intervals2.get(j).b;  **44** if((m1 >= p1 && m2 <= p2) && (m2 - m1 >= k)) {  **45** finalI.a = m1;  **46** gasit = true;  **47** }  **48** else if((m1 < p1 && m2 > p1 && m2 < p2) && (m2 - p1 >= k)) {  **49** finalI.a = p1;  **50** gasit = true;  **51** }  **52** else if((m1 > p1 && m1 < p2 && m2 > p2) && (p2 - m1 >= k)) {  **53** finalI.a = m1;  **54** gasit = true;  **55** }  **56** else if((m1 <= p1 && m2 >= p2) && (p2 - p1 >= k)) {  **57** finalI.a = p1;  **58** gasit = true;  **59** }  **60** }  **61** }  **62** finalI.b = finalI.a + k;  **63** FileWriter writer = new FileWriter("licenta.out");  **64** if(gasit == false) {  **65** writer.write("-1");  **66** }  **67** else {  **68** writer.write(finalI.a + " " + finalI.b);  **69** }  **70** writer.close();  **71** }  **72** }  **73** }  } |



Pe baza grafului se pot defini diverse acoperiri:

* Acoperire la nivel de instrucțiune: fiecare instrucțiune (nod al grafului) este parcursă măcar o data
* Acoperire la nivel de ramură: fiecare ramură a grafului este parcursă măcar o data
* Acoperire la nivel de condiție: fiecare condiție individuală dintr-o decizie să ia atât valoarea adevărat cât și valoarea fals
* Acoperire la nivel de cale: fiecare cale din graf este parcursă măcar o data

1. **Acoperirea la nivel de instrucțiune**

Pentru a obține o acoperire la nivel de instrucțiune, trebuie să ne concentram asupra acelor instrucțiuni care sunt controlate de condiții (acestea corespund ramificațiilor din graf).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Intrări** | | | | | **Rezultatul afișat (expected)** | **Instrucțiuni parcurse** |
| **k** | **n** | **S** | **m** | **T** |
| 4 | 1 | {(1 10)} | 1 | {(5 12)} | (5, 9) | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,  49…51,60,41,61,38,62,63,64,  67…69,70,71,72,73. |

|  |
| --- |
| @Test  public void AcoperireNivelInstructiune() throws IOException {  testClass.licenta("src/pachet1/AcoperireInstructiune/4.1.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireInstructiune/expected4.1.txt")),  FileUtils.readLines(new File("licenta.out")));  } |

1. **Acoperirea la nivel de decizie**

Aceasta acoperire este o extindere naturală a metodei precendete, deoarece generează date de test care testează cazurile în care fiecare decizie este adevarată sau falsă.

|  |  |
| --- | --- |
|  | **Decizii** |
| (1) | if (k > 60) |
| (2) | else if (k < 1) |
| (3) | if (n > 6) |
| (4) | else if (n < 1) |
| (5) | for(i = 0; i < n; i++) |
| (6) | if (m > 6) |
| (7) | else if (m < 1) |
| (8) | for(i = 0; i < m; i++) |
| (9) | for(i = 0; i < n && gasit == false; i++) |
| (10) | for(j = 0; j < m && gasit == false; j++) |
| (11) | if((m1 >= p1 && m2 <= p2) && (m2 - m1 >= k)) |
| (12) | else if((m1 < p1 && m2 > p1 && m2 < p2) && (m2 - p1 >= k)) |
| (13) | else if((m1 > p1 && m1 < p2 && m2 > p2) && (p2 - m1 >= k)) |
| (14) | else if((m1 <= p1 && m2 >= p2) && (p2 - p1 >= k)) |
| (15) | if(gasit == false) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Intrări** | | | | | **Rezultatul afișat (expected)** | **Instructiuni parcurse** |
| **k** | **n** | **S** | **m** | **T** |
| 61 |  |  |  |  | Cere introducerea valorii k mai mică decât 61 | 1,2,3,4 |
| 1 | -5 |  |  |  | Cere introducerea valorii n mai mare decât 0 | 1,2,5,8,9,10,13,14,15,73 |
| 4 | 1 | {(1 6)} | 0 |  | Cere introducerea valorii m mai mare decât 0 | 1,2,5,8,9,10,13,16,17,18,19,20,  21,22,23,26,27,28,72,73 |
| 4 | 1 | {(1 10)} | 1 | {(5 12)} | (5, 9) | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,  49…51,60,41,61,38,62,63,64,  67…69,70,71,72,73. |
| 4 | 1 | {(1 10)} | 1 | {(16 23)} | -1 | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,52,56,  60,41,61,38,62,63,64,65,66,  70,71,72,73 |

|  |
| --- |
| @Test  public void AcoperireNivelDecizie() throws IOException {  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/AcoperireDecizie/5.1.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/AcoperireDecizie/5.2.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/AcoperireDecizie/5.3.in"));  testClass.licenta("src/pachet1/AcoperireDecizie/5.4.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireDecizie/expected5.4.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/AcoperireDecizie/5.5.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireDecizie/expected5.5.txt")),  FileUtils.readLines(new File("licenta.out")));  } |

1. **Acoperirea la nivel de condiție**

Generează date de test astfel încât fiecare condiție individuală dintr-o decizie să ia atât valoarea de adevar cât și valoarea de fals (dacă este posibil).

|  |  |  |
| --- | --- | --- |
|  | **Decizii** | **Conditii individuale** |
| (1) | if (k > 60) | k > 60 |
| (2) | else if (k < 1) | k < 1 |
| (3) | if (n > 6) | n > 6 |
| (4) | else if (n < 1) | n < 1 |
| (5) | for(i = 0; i < n; i++) | i < n |
| (6) | if (m > 6) | m > 6 |
| (7) | else if (m < 1) | m < 1 |
| (8) | for(i = 0; i < m; i++) | i < m |
| (9) | for(i = 0; i < n && gasit == false; i++) | i < n, gasit == false |
| (10) | for(j = 0; j < m && gasit == false; j++) | j < m, gasit == false |
| (11) | if((m1 >= p1 && m2 <= p2) && (m2 - m1 >= k)) | m1 >= p1, m2 <= p2, m2 - m1 >= k |
| (12) | else if((m1 < p1 && m2 > p1 && m2 < p2) && (m2 - p1 >= k)) | m1 < p1, m2 > p1, m2 < p2, m2 - p1 >= k |
| (13) | else if((m1 > p1 && m1 < p2 && m2 > p2) && (p2 - m1 >= k)) | m1 > p1, m1 < p2, m2 > p2, p2 - m1 >= k |
| (14) | else if((m1 <= p1 && m2 >= p2) && (p2 - p1 >= k)) | m1 <= p1, m2 >= p2, p2 - p1 >= k |
| (15) | if(gasit == false) | gasit == false |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Intrări** | | | | | **Rezultatul afișat (expected)** | **Instrucțiuni parcurse** |
| **k** | **n** | **S** | **m** | **T** |
| 61 |  |  |  |  | Cere introducerea valorii k mai mică decât 61 | 1,2,3,4 |
| -5 |  |  |  |  | Cere introducerea valorii k mai mare decât 0 | 1,2,5,6,7 |
| 1 | 7 |  |  |  | Cere introducerea valorii n mai mică decât 7 | 1,2,5,8,9,10,11,12,73 |
| 1 | -5 |  |  |  | Cere introducerea valorii n mai mare decât 0 | 1,2,5,8,9,10,13,14,15,73 |
| 4 | 1 | {(1 10)} | -5 |  | Cere introducerea valorii m mai mare decât 0 | 1,2,5,8,9,10,13,16,17,18,19,20,  21,22,23,26,27,28,72,73 |
| 4 | 1 | {(1 10)} | 7 |  | Cere introducerea valorii m mai mică decât 7 | 1,2,5,8,9,10,13,16,17,18,19,20,  21,22,23,24,25,72,73 |
| 4 | 1 | {(5 10)} | 1 | {(1 12)} | (5, 9) | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,45,46,47,  60,41,61,38,62,63,64,67…69,  70,71,72,73. |
| 4 | 1 | {(5 7)} | 1 | {(1 12)} | -1 | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,52,56,  60,41,61,38,62,63,64,65,66,  70,71,72,73. |
| 4 | 1 | {(1 10)} | 1 | {(5 12)} | (5, 9) | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,  49…51,60,41,61,38,62,63,64,  67…69,70,71,72,73. |
| 4 | 1 | {(1 10)} | 1 | {(8 12)} | -1 | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,52,56,  60,41,61,38,62,63,64,65,66,  70,71,72,73. |
| 4 | 1 | {(5 12)} | 1 | {(1 10)} | (5, 9) | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,52,  53…55,60,41,61,38,62,63,64,  67…69,70,71,72,73. |
| 4 | 1 | {(8 12)} | 1 | {(1 10)} | -1 | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,52,56,  60,41,61,38,62,63,64,65,66,  70,71,72,73. |
| 4 | 1 | {(1 12)} | 1 | {(5 10)} | (5, 9) | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,52,  56,57…59,60,41,61,38,62,63,64,  67…69,70,71,72,73. |
| 4 | 1 | {(1 12)} | 1 | {(5 7)} | -1 | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,52,56,  60,41,61,38,62,63,64,65,66,  70,71,72,73. |
| 4 | 2 | {(11 13),  (1 10)} | 2 | {(5 12),  (20 30)} | (5, 9) | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,  49…51,60,41,61,38,62,63,64,  67…69,70,71,72,73. |
| 4 | 1 | {(1 10)} | 2 | {(5 12),  (20 30)} | (5, 9) | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,  49…51,60,41,61,38,62,63,64,  67…69,70,71,72,73. |
| 4 | 2 | {(11 13),  (1 10)} | 1 | {(5 12)} | (5, 9) | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,  49…51,60,41,61,38,62,63,64,  67…69,70,71,72,73. |
| 4 | 1 | {(1 10)} | 1 | {(20 30)} | -1 | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,  49…51,60,41,61,38,62,63,64,  65, 66,70,71,72,73. |
| 4 | 1 | {(1 10)} | 1 | {(8 30)} | -1 | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,  49…51,60,41,61,38,62,63,64,  65, 66,70,71,72,73. |

|  |
| --- |
| @Test  public void AcoperireNivelConditie() throws IOException {  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/AcoperireConditie/6.1.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/AcoperireConditie/6.2.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/AcoperireConditie/6.3.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/AcoperireConditie/6.4.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/AcoperireConditie/6.5.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/AcoperireConditie/6.6.in"));  testClass.licenta("src/pachet1/AcoperireConditie/6.7.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireConditie/expected6.7.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/AcoperireConditie/6.8.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireConditie/expected6.8.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/AcoperireConditie/6.9.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireConditie/expected6.9.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/AcoperireConditie/6.10.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireConditie/expected6.10.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/AcoperireConditie/6.11.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireConditie/expected6.11.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/AcoperireConditie/6.12.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireConditie/expected6.12.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/AcoperireConditie/6.13.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireConditie/expected6.13.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/AcoperireConditie/6.14.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireConditie/expected6.14.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/AcoperireConditie/6.15.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireConditie/expected6.15.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/AcoperireConditie/6.16.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireConditie/expected6.16.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/AcoperireConditie/6.17.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireConditie/expected6.17.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/AcoperireConditie/6.18.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireConditie/expected6.18.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/AcoperireConditie/6.19.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/AcoperireConditie/expected6.19.txt")),  FileUtils.readLines(new File("licenta.out")));  } |

1. **Testarea circuitelor independente**

**a. Formula lui McCabe**

Aceasta este o modalitate de a identifica limita superioară pentru numărul de căi necesare pentru obținerea unei acoperiri la nivel de ramură. Ea se bazează pe formula lui McCabe pentru Complexitate Ciclomatica: V(G) = e - n + 1.

Unde adăugând un arc de la 73 la 1, am:

* e = 56 - numărul de muchii
* n = 43 - numărul de noduri
* p = 1 - numărul de componente conectate

Deci **V(G) = 56 – 43 + 1 = 14.**

**b. Circuite independente:**

1. 1,2,5,8..9,10,11..12,73,1
2. 1,2,5,8..9,10,13,14..15,73,1
3. 1,2,5,8..9,10,13,16,17,22,23,24..25,72,73,1
4. 1,2,5,8..9,10,13,16,17,22,23,26,27..28,72,73,1
5. 1,2,5,8..9,10,13,16,17,22,23,26,29,30,35..37,38,62..63,65..66,70..71,72,73,1
6. 1,2,5,8..9,10,13,16,17,22,23,26,29,30,35..37,38,62..63,67..69,70..71,72,73,1
7. 17,18..21,17
8. 30,31..34,30
9. 38,39..40,41,61,38
10. 41,42..42,44,48,52,56,60,41
11. 41,42..42,44,45..47,60,41
12. 41,42..42,44,48,49..51,60,41
13. 41,42..42,44,48,52,53..55,60,41
14. 41,42..42,44,48,52,56,57..59,60,41

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Intrări** | | | | | **Rezultatul afișat (expected)** | **Instructiuni parcurse** |
| **k** | **n** | **S** | **m** | **T** |
| 1 | 7 |  |  |  | Cere introducerea valorii n mai mică decât 7 | 1,2,5,8,9,10,11,12,73 |
| 1 | -5 |  |  |  | Cere introducerea valorii n mai mare decât 0 | 1,2,5,8,9,10,13,14,15,73 |
| 4 | 1 | {(1 10)} | -5 |  | Cere introducerea valorii m mai mare decât 0 | 1,2,5,8,9,10,13,16,17,18,19,20,  21,22,23,26,27,28,72,73 |
| 4 | 1 | {(1 10)} | 7 |  | Cere introducerea valorii m mai mică decât 7 | 1,2,5,8,9,10,13,16,17,18,19,20,  21,22,23,24,25,72,73 |
| 4 | 1 | {(5 10)} | 1 | {(1 12)} | (5, 9) | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,45,46,47,  60,41,61,38,62,63,64,67…69,  70,71,72,73. |
| 4 | 1 | {(5 7)} | 1 | {(1 12)} | -1 | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,52,56,  60,41,61,38,62,63,64,65,66,  70,71,72,73. |
| 4 | 2 | {(11 13),  (1 10)} | 1 | {(5 12)} | (5, 9) | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,  49…51,60,41,61,38,62,63,64,  67…69,70,71,72,73. |
| 4 | 1 | {(1 10)} | 1 | {(20 30)} | -1 | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,  49…51,60,41,61,38,62,63,64,  65, 66,70,71,72,73. |
| 4 | 1 | {(1 10)} | 1 | {(8 30)} | -1 | 1,2,5,8,9,10,13,16,17,18…21,  22,23,26,29,30,31…34,35…37,  38,39,40,41,42,43,44,48,  49…51,60,41,61,38,62,63,64,  65, 66,70,71,72,73. |

|  |
| --- |
| @Test  public void CircuiteIndependente() throws IOException {  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/CircuiteIndependente/7.1.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/CircuiteIndependente/7.2.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/CircuiteIndependente/7.3.in"));  assertThrows(IllegalArgumentException.class, () -> MyClass.licenta("src/pachet1/CircuiteIndependente/7.4.in"));  testClass.licenta("src/pachet1/CircuiteIndependente/7.5.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/CircuiteIndependente/expected7.5.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/CircuiteIndependente/7.6.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/CircuiteIndependente/expected7.6.txt")),  FileUtils.readLines(new File("licenta.out")));  } |

1. **Testare la nivel de cale**

**a. Expresia regulată**

Dacă un program este structurat, atunci, folosindu-se o tehnică descrisă de Paige și Holthouse, acesta poate fi caracterizat de o expresie regulată formată din nodurile grafului.

Pentru graful prezentat mai sus, obțin expresia regulată:

1.2.(3:4+5.(6:7+8:9.10.(11:12.73+13.(14:15.73+16.17.(18:21.17)\*.22.23.(24:25.72.73+26.(27:28.72.73+29.30.(31:34.30)\*.35:37.38.(39:40.41.(42:43.44.(45:47.60+48.(49:51.60+52.(53:55.60+ 56.(57:58.60+60)))).41)\*.61.38)\*.62:63.64.(65:66+67:69)70:71.72.73)))))).

Iar pentru n = 0 și n = 1 am:

1.2.(3:4+5.(6:7+8:9.10.(11:12.73+13.(14:15.73+16.17.(18:21.17+null).22.23.(24:25.72.73+26.(27:28.72.73+29.30.(31:34.30+null).35:37.38.(39:40.41.(42:43.44.(45:47.60+48.(49:51.60+52.(53:55.60+ 56.(57:58.60+60)))).41+null).61.38+null).62:63.64.(65:66+67:69)70:71.72.73)))))).

**b. Numărul de căi**

De aici, numărul de căi se obține înlocuind în expresia regulată fiecare nod cu 1, inclusiv pentru null, iar operația de concatenare devine inmulțire:

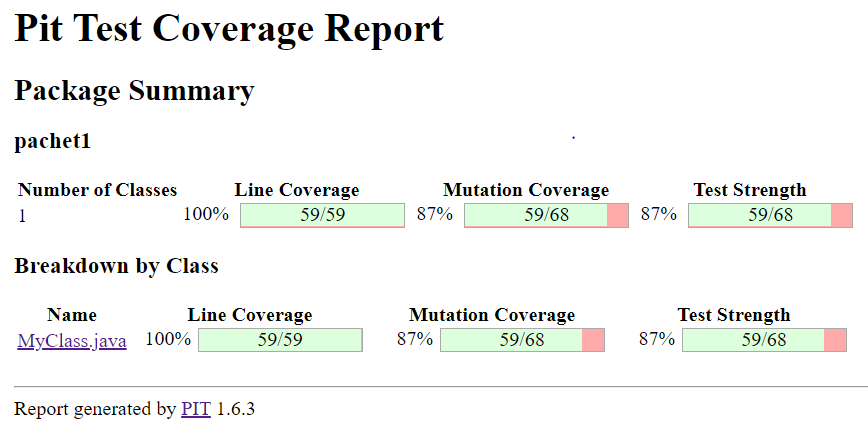
1.1.(1+1.(1+1.1.(1.1+1.(1.1+1.1.(1.1+1).1.1.(1.1.1+1.(1.1.1+1.1.(1.1+1).1.1.(1.1.(1.1.(1.1+1.(1. 1+1.(1.1+1.(1.1+1)))).1+1).1.1+1).1.1.(1+1)1.1.1))))))=(1+1\*(1+1\*(1+1\*(1+1\*(1+1)\*(1+1\*(1+1\*(1+1)\*(1\*(1\*(1+1\*(1+1\*(1+1(1+1))))\*1+1)\*1+1)\*(1+1)))))))=

(1+1\*(1+1\*(1+1\*(1+1\*2\*(1+(2\*2\*7\*2))))))=(1+1\*(1+1\*(1+1\*(1+1\*2\*57))))=(1+117) = **118 cai**.

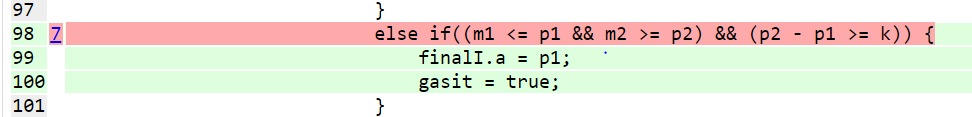
1. **Generarea mutanților**

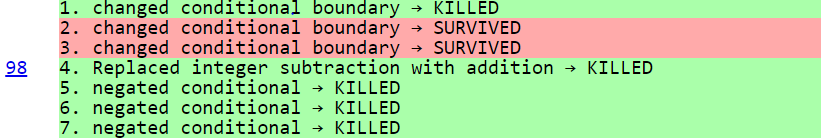
Am folosit generatorul de mutant Pitest pentru a crea mutanți pentru metoda implementată.

Apoi am rulat seturile de test de la punctele anterioare contra mutanților generați și am obținut următorul rezultat:



Printre muntanții neechivalenți rămași în viață se enumerau și următorii doi:





Pe care i-am omorât folosind datele de test:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Intrări** | | | | | **Rezultatul afișat (expected)** |
| **k** | **n** | **S** | **m** | **T** |
| 4 | 2 | {(1 10), (16 20)} | 2 | {(18 19), (5 9)} | (5, 9) |
| 4 | 2 | {(1 9), (16 20)} | 2 | {(18 19), (5 9)} | (5, 9) |

|  |
| --- |
| @Test  public void Mutanti() throws IOException {  testClass.licenta("src/pachet1/Mutanti/8.1.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/Mutanti/expected8.1.txt")),  FileUtils.readLines(new File("licenta.out")));  testClass.licenta("src/pachet1/Mutanti/8.2.in");  Assert.assertEquals(FileUtils.readLines(new File("tests/pachet2/Mutanti/expected8.2.txt")),  FileUtils.readLines(new File("licenta.out")));  } |

Astfel rezultând raportul:

