Testarea Sistemelor Software

Proiect

454 Costea Maria Alexandra

Facultatea de Matematica si Informatica

Universitatea din Bucuresti

1. Cuprins

2. Enuntul problemei

Plagiat

Giani, un Mai Mare al Oraşului, este acuzat de plagiat în lucrarea sa de licenţă intitulată "Despre cromatica semnelor de circulaţie". S-a format, bineînţeles, o comisie care să decidă asupra veridicităţii acestei acuzaţii. Aceasta este formată dintr-un semn de circulaţie, un cetăţean care se ocupă cu trafic de fier vechi, un delfin şi un cetăţean care citeşte în stele. În această problemă ne vom preocupa de metoda prin care se va decide cititorul în stele. Acesta se va uita la harta cerului, formată din N stele punctiforme. Apoi va lua în considerare toate triunghiurile care se pot forma cu vârfurile în stele, iar dacă găseşte două triunghiuri astfel încât primul poate fi obţinut din al doilea doar printr-o translaţie atunci consideră că acesta este un semn, iar Giani este vinovat.

Dându-se T hărţi ale cerului, să se decidă pentru fiecare dacă Giani este vinovat de plagiat sau nu.

Date de intrare

Fişierul de intrare plagiat.in conţine pe prima sa linie T, numărul de hărţi pe care trebuie să le analizaţi. Fiecare hartă va fi descrisă printr-un număr N, numărul de stele, şi N perechi de numere naturale care descriu coordonatele stelelor.

Date de ieşire

În fişierul de ieşire plagiat.out se vor afla T linii. Linia cu numărul i va conţine cuvântul "DA" dacă Giani este vinovat conform hărţii cu numărul i din input. Altfel va conţine cuvântul "NU".

Restricţii

$1 \leq T \leq 5$

$1 \leq N \leq 400$

Coordonatele stelelor se află în intervalul [0, 109]

Nu vor exista două stele cu aceleaşi coordonate.

Exemplu

plagiat.in	**plagiat.out**
2 5 1 1 2 2 1 2 0 0 100 105 5 1 1 2 2 1 2 0 0 0 1	NU DA

Explicaţie

În primul caz Giani este absolvit de orice vină, toate triunghiurile posibile fiind distincte relativ la translaţii.
În cel de al doilea caz, triunghiurile (1, 2, 3) respectiv (1, 4, 5) se pot obţine unul din altul prin translaţii. În acest caz Giani îşi pune speranţele în votul delfinului.

3. Să se genereze date de test folosind metode funcționale:

a) Partitionare in clase de echivalenta

Domeniul de intrari:

T – numar de harti, numar intreg intre 1 si 5

N – numar de stele pe o harta, numar intreg intre 1 si 400 (pentru proiect voi utiliza N < 10)

x, y – coordonatele stelelor, intregi intre 0 si 109; nu vor exista 2 stele cu aceleasi coordinate

T_1 = { T / T < 1 }

T_2 = { T/ T > 5 }

T_3 = { T / 1 <= T <= 5 }

N_1 = { N / N < 1 }

N_2 = { N / N > 10 }

N_3 = { N / 1 <= N <= 10 }

XY_1 = { (x, y) / x < 0 || y < 0 }

XY_2 = { (x, y) / x > 109 || y > 109 }

XY_3 = { (x, y) / 0 <= x <= 109 && 0 <= y <= 109 && (x, y), nu se repeta }

XY_4 = { (x, y) / 0 <= x <= 109 && 0 <= y <= 109 && (x, y), se repeta }

Domeniul de iesiri:

- este reprezentat de un string, R, „DA” sau „NU”

R_1= { „DA” }

R_2= { „NU” }

Prin combinarea claselor individuale obtinem urmatoarele clase de echivalență:

C_3331 = { (T, N, x, y, R) / T din T_3, N din N_3, (x, y) din XY_3, R din R_1}

C_3332 = { (T, N, x, y, R) / T din T_3, N din N_3, (x, y) din XY_3, R din R_2}

C_334 = {(T, N, x, y, R) / T din T_3, N din N_3, (x, y) din XY_4 }

C_331 = { (T, N, x, y, R) / T din T_3, N din N_3, (x, y) din XY_1 }

C_332 = { (T, N, x, y, R) / T din T_3, N din N_3, (x, y) din XY_2 }

C_31 = { (T, N, x, y, R) / T din T_3, N din N_1 }

C_32 = { (T, N, x, y, R) / T din T_3, N din N_2 }

C_1 = { (T, N, x, y, R) / T din T_1 }

C_2 = { (T, N, x, y, R) / T din T_2 }

Intrari				Rezultat afisat (expected)
T	N	x,y	R	
0				T < 1
6				T > 5
1	0			N < 1
1	11			N > 10
1	3	-1,y		X < 0
1	3	110,y		X > 109
1	3	X,-1		Y < 0
1	3	X,110		Y > 109
1	3	2 2 1 1 2 2		Doua stele cu aceleasi coordonate
1	5	1 1 2 2 1 2 0 0 0 1	DA	Rezultat “ DA”
1	5	1 1 2 2 1 2 0 0 100 105	NU	Rezultat “NU”

```
@Test
public void equivalencePartitioning() {

    tester.main(new String[]{"0"});
    tester.main(new String[]{"6"});
    tester.main(new String[]{"1", "0"});
    tester.main(new String[]{"1", "11"});
    tester.main(new String[]{"1", "3", "-1"});
    tester.main(new String[]{"1", "3", "110"});
    tester.main(new String[]{"1", "3", "0", "-1"});
    tester.main(new String[]{"1", "3", "0", "110"});
    tester.main(new String[]{"1", "3", "2", "2", "1", "1", "2", "2"});
    tester.main(new String[]{"1", "5", "1", "1", "2", "2", "1", "2", "0", "0", "100", "105", "NU"});
```

```
    tester.main(new String[]{"1", "5", "1", "1", "2", "2", "1", "2", "0", "0", "0", "1", "DA"});

}
```

b) Analiza valorilor de frontiera

Intrari				Rezultat afisat (expected)
T	N	X,Y	R	
0				T < 1
6				T > 5
1	0			N < 1
1	11			N > 10
1	1	-1,y		X < 0
1	1	110,y		X > 109
1	1	X,-1		Y < 0
1	1	X,110		Y > 109
1	1	5,5	NU	Rezultat “ NU”
1	1	100,100	NU	Rezultat “NU”
1	10	1 1 2 2 1 1 3 3 4 4 5 5 6 6 7 7 8 8 9 9		Doua stele cu aceleasi coordonate
1	10	0 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9	DA	Rezultat “DA”
1	10	0 0 1 1 2 2 10 10 13 13 42 42 53 53 65 65 67 67 18 18	NU	Rezultat “NU”
1	10	109 109 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9	DA	Rezultat “DA”
1	10	109 109 1 1 2 2 10 10 13 13 42 42 53 53 65 65 67 67 18 18	NU	Rezultat “NU”
5	0			N < 1

5	11			N > 10
5	1	-1,y		X < 0
5	1	110,y		X > 109
5	1	X,-1		Y < 0
5	1	X,110		Y > 109
5	10	1 1 2 2 1 1 3 3 4 4 5 5 6 6 7 7 8 8 9 9 3 3		Doua stele cu aceleasi coordonate
5	10	0 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9		Rezultat “DA”
5	10	0 0 1 1 2 2 10 10 13 13 42 42 53 53 65 65 67 67 18 18		Rezultat “NU”
5	10	109 109 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9		Rezultat “DA”
5	10	109 109 1 1 2 2 10 10 13 13 42 42 53 53 65 65 67 67 18 18		Rezultat “NU”

```
@Test
public void boundaryValueAnalysis() {

   tester.main(new String[]{"0"});
   tester.main(new String[]{"6"});
   tester.main(new String[]{"1", "0"});
   tester.main(new String[]{"1", "11"});
   tester.main(new String[]{"1", "1", "-1"});
   tester.main(new String[]{"1", "1", "110"});
   tester.main(new String[]{"1", "1", "0", "-1"});
   tester.main(new String[]{"1", "1", "0", "110"});
   tester.main(new String[]{"1", "1", "0", "0", "NU"});
   tester.main(new String[]{"1", "1", "109", "109", "NU"});
   tester.main(new String[]{"1", "1", "-1", "2"});
   tester.main(new String[]{"1", "1", "110", "2"});
   tester.main(new String[]{"1", "1", "0", "-1"});
   tester.main(new String[]{"1", "1", "0", "110"});
   tester.main(new String[]{"1", "10", "1", "1", "2", "2", "1", "1", "3", "3", "4", "4", "5", "5", "6", "6", "7", "7",
"8", "8", "9", "9", "NU"});
   tester.main(new String[]{"1", "10", "0", "0", "1", "1", "2", "2", "3", "3", "4", "4", "5", "5", "6", "6", "7", "7",
"8", "8", "9", "9", "DA"});
   tester.main(new String[]{"1", "10", "0", "0", "1", "1", "2", "2", "10", "10", "13", "13", "42", "42", "53", "53",
"65", "65", "67", "67", "18", "18", "NU"});
   tester.main(new String[]{"1", "10", "109", "109", "1", "1", "2", "2", "3", "3", "4", "4", "5", "5", "6", "6", "7",
"7", "8", "8", "9", "9", "DA"});
   tester.main(new String[]{"1", "10", "109", "109", "1", "1", "2", "2", "10", "10", "13", "13", "42", "42", "53",
"53", "65", "65", "67", "67", "18", "18", "NU"});
   tester.main(new String[]{"5", "0"});
   tester.main(new String[]{"5", "11"});
```

```
    tester.main(new String[]{"5", "1", "-1", "3"});
    tester.main(new String[]{"5", "1", "110", "3"});
    tester.main(new String[]{"5", "1", "3", "-1"});
    tester.main(new String[]{"5", "1", "3", "110"});
    tester.main(new String[]{"5", "10", "1", "1", "2", "2", "1", "1", "3", "3", "4", "4", "5", "5", "6", "6", "7", "7",
"8", "8", "9", "9", "NU"});
    tester.main(new String[]{"5", "10", "0", "0", "1", "1", "2", "2", "3", "3", "4", "4", "5", "5", "6", "6", "7", "7",
"8", "8", "9", "9", "DA"});
    tester.main(new String[]{"5", "10", "0", "0", "1", "1", "2", "2", "10", "10", "13", "13", "42", "42", "53", "53",
"65", "65", "67", "67", "18", "18", "NU"});
    tester.main(new String[]{"5", "10", "109", "109", "1", "1", "2", "2", "3", "3", "4", "4", "5", "5", "6", "6", "7",
"7", "8", "8", "9", "9", "DA"});
    tester.main(new String[]{"5", "10", "109", "109", "1", "1", "2", "2", "10", "10", "13", "13", "42", "42", "53",
"53", "65", "65", "67", "67", "18", "18", "NU"});

}
```

c) Partitionare in categorii

Avem o unitate de testare

Parametri si conditii de mediu: T, N, (x,y), R

Specificatiile de testare:

- T:
  1) { T / T < 1 }
  2) 0
  3) 1
  4) { T / T > 1 && T < 5}
  5) 5
  6) 6
  7) { T / T > 5 }

- N:
  1) { N / N < 1 }
  2) 0
  3) 1
  4) { N / N > 1 && N < 10}
  5) 10
  6) 11
  7) { N / N > 10 }

- (x, y):

1) { (x, y) / x < 0 || y < 0 }
2) -1
3) 0 && (x, y) se repeta
4) 0 &&(x, y) nu se repeta
5) { (x, y) / (x > 0 || y > 0) && (x < 109 || y < 109) && (x, y) se repeta }
6) { (x, y) / (x > 0 || y > 0) && (x < 109 || y < 109) && (x, y) nu se repeta }
7) 109 && (x, y) se repeta
8) 109 && (x, y) nu se repeta
9) 110
10) { (x, y) / x > 109 || y > 109 }

- R:

1) „NU”
2) „DA”

Cazuri de testare: 7 * 7 * 10 * 2 = 980

Eliminand unele combinatii obtinem 33 de cazuri de testare

Intrari				Rezultat afisat (expected)
T	N	X,Y	R	
-20				T << 1
0				T < 1
6				T > 5
20				T >> 5
1	-20			N << 1
1	0			N < 1
1	11			N > 10
1	20			N >> 10
1	1	-20,y		X << 0

1	1	-1,y		X < 0
1	1	110,y		X > 109
1	1	120,y		X >> 109
1	1	X, -20		Y << 0
1	1	X,-1		Y < 0
1	1	X,110		Y > 109
1	1	X,120		Y >> 109
1	1	0,0	NU	Rezultat “NU”
1	1	35,35	NU	Rezultat “NU”
1	1	109,109	NU	Rezultat “NU”
2	5	0 0 1 1 2 2 1 1 3 3		Doua stele cu aceleasi coordonate
2	5	0 0 5 5 6 6 7 7 8 8	NU	Rezultat “NU”
2	5	0 0 1 1 2 2 3 3 4 4	DA	Rezultat “DA”
2	5	35 35 36 36 47 47 48 48 79 79	NU	Rezultat “NU”
2	5	35 35 36 36 37 37 38 38 39 39	DA	Rezultat “DA”
2	5	95 95 96 96 105 105 106 106 109 109	NU	Rezultat “NU”
2	5	105 105 106 106 107 107 108 108 109 109	DA	Rezultat “DA”
5	10	0 0 1 1 2 2 1 1 3 3 4 4 5 5 6 6 7 7 8 8		Doua stele cu aceleasi coordonate
5	10	1 1 6 6 7 7 8 8 11 11 12 12 13 13 14 14 15 15 16 16	NU	Rezultat “NU”
5	10	0 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9	DA	Rezultat “DA”
5	10	35 35 36 36 47 47 48 48 79 79 53 53 65 65 66 66 67 67 68 68	NU	Rezultat “NU”
5	10	35 35 36 36 37 37 38 38 39 39 40 40 41 41 42 42 43 43 44 44	DA	Rezultat “DA”

5	10	90 90 91 91 92 92 95 95 96 96 101 101 102 102 105 105 106 106 109 109	NU	Rezultat “NU”
5	10	101 101 102 102 103 103 104 104 105 105 106 106 107 107 108 108 109 109	DA	Rezultat “DA”

```
@Test
public void categoryPartitioning() {

    tester.main(new String[]{"-5"});
    tester.main(new String[]{"0"});
    tester.main(new String[]{"6"});
    tester.main(new String[]{"10"});
    tester.main(new String[]{"1", "-20"});
    tester.main(new String[]{"1", "0"});
    tester.main(new String[]{"1", "11"});
    tester.main(new String[]{"1", "20"});
    tester.main(new String[]{"1", "1", "-20"});
    tester.main(new String[]{"1", "1", "-1"});
    tester.main(new String[]{"1", "1", "110"});
    tester.main(new String[]{"1", "1", "120"});
    tester.main(new String[]{"1", "1", "0", "-20"});
    tester.main(new String[]{"1", "1", "0", "-1"});
    tester.main(new String[]{"1", "1", "0", "110"});
    tester.main(new String[]{"1", "1", "0", "120"});
    tester.main(new String[]{"1", "1", "0", "0", "NU"});
    tester.main(new String[]{"1", "1", "35", "35", "NU"});
    tester.main(new String[]{"1", "1", "109", "109", "NU"});
    tester.main(new String[]{"2", "5", "0", "0", "1", "1", "2", "2", "1", "1", "3", "3", "NU"});
    tester.main(new String[]{"2", "5", "0", "0", "5", "5", "6", "6", "7", "7", "8", "8", "NU", "5", "0", "0", "4", "4",
"5", "5", "6", "6", "7", "7", "NU"});
    tester.main(new String[]{"2", "5", "0", "0", "1", "1", "2", "2", "3", "3", "4", "4", "DA", "5", "0", "0", "1", "1",
"2", "2", "3", "3", "4", "4", "DA"});
    tester.main(new String[]{"2", "5", "35", "35", "36", "36", "47", "47", "48", "48", "79", "79", "NU", "5", "47",
"47", "48", "48", "79", "79", "80", "80", "81", "81", "NU"});
    tester.main(new String[]{"2", "5", "35", "35", "36", "36", "37", "37", "38", "38", "39", "39", "DA", "5", "50",
"50", "51", "51", "52", "52", "53", "53", "54", "54", "DA"});
    tester.main(new String[]{"2", "5", "95", "95", "96", "96","105", "105", "106", "106", "109", "109", "NU", "5",
"100", "100", "101", "101", "105", "105", "106", "106", "109", "109", "NU"});
    tester.main(new String[]{"2", "5","105", "105", "106", "106", "107", "107", "108", "108", "109", "109", "DA",
"5", "109", "109", "108", "108", "107", "107", "106", "106", "105", "105", "DA"});
    tester.main(new String[]{"5", "10", "0", "0", "1", "1", "2", "2", "1", "1", "3", "3", "4", "4", "5", "5", "6", "6",
"7", "7", "8", "8", "NU"});
    tester.main(new String[]{"5", "10", "1", "1", "6", "6", "7", "7", "8", "8", "11", "11", "12", "12", "13", "13",
"14", "14", "15", "15", "16", "16", "NU", "10", "0", "0", "1", "1", "13", "13", "14", "14", "15", "15", "16", "16",
"28", "28", "26", "26", "35", "35", "NU", "10", "1", "1", "6", "6", "7", "7", "8", "8", "11", "11", "12", "12", "13",
"13", "14", "14", "15", "15", "16", "16", "NU", "10", "0", "0", "1", "1", "6", "6", "15", "15", "28", "28", "7", "7",
"8", "8", "9", "9", "36", "36", "NU", "10", "0", "0", "1", "1", "6", "6", "15", "15", "28", "28", "7", "7", "8", "8",
"9", "9", "36", "36", "NU"});
    tester.main(new String[]{"5", "10", "0", "0", "1", "1", "2", "2", "3", "3", "4", "4", "5", "5", "6", "6", "7", "7",
"8", "8", "9", "9", "DA", "10", "0", "0", "1", "1", "2", "2", "3", "3", "4", "4", "5", "5", "6", "6", "7", "7", "8", "8",
"9", "9", "DA", "10", "0", "0", "1", "1", "2", "2", "3", "3", "4", "4", "5", "5", "6", "6", "7", "7", "8", "8", "9", "9",
"DA", "10", "0", "0", "1", "1", "2", "2", "3", "3", "4", "4", "5", "5", "6", "6", "7", "7", "8", "8", "9", "9", "DA",
```

```
"10", "0", "0", "1", "1", "2", "2", "3", "3", "4", "4", "5", "5", "6", "6", "7", "7", "8", "8", "9", "9", "DA"});
   tester.main(new String[]{"5", "10", "35", "35", "36", "36", "47", "47", "48", "48", "79", "79", "53", "53", "65",
"65", "66", "66", "67", "67", "68", "68", "NU", "10", "50", "50", "51", "51", "47", "47", "48", "48", "79", "79",
"80", "80", "81", "81", "70", "70", "89", "89", "60", "60", "NU", "10", "50", "50", "51", "51", "57", "57", "79",
"79", "80", "80", "81", "81", "70", "70", "29", "29", "30", "30", "31", "31", "NU", "10", "50", "50", "51", "51",
"57", "57",  "79", "79", "80", "80", "81", "81", "70", "70", "29", "29", "89", "89", "32", "32", "NU", "10", "80",
"80", "81", "81", "70", "70", "29", "29", "89", "89", "32", "32", "50", "50", "51", "51", "57", "57","34", "34",
"NU"});
   tester.main(new String[]{"5", "10", "35", "35", "36", "36", "37", "37", "38", "38", "39", "39", "40", "40", "41",
"41", "42", "42", "43", "43", "44", "44", "DA", "10", "36", "36", "37", "37", "38", "38", "39", "39", "40", "40",
"41", "41", "42", "42", "43", "43", "44", "44", "45", "45", "DA", "10", "36", "36", "37", "37", "38", "38", "39",
"39", "40", "40", "41", "41", "42", "42", "43", "43", "44", "44", "45", "45", "46", "46", "DA", "10", "36", "36",
"37", "37", "38", "38", "39", "39", "40", "40", "41", "41", "42", "42", "43", "43", "44", "44", "45", "45", "46",
"46", "47", "47", "DA", "10", "50", "50", "51", "51", "52", "52", "53", "53", "54", "54", "55", "55", "56", "56",
"57", "57", "58", "58", "59", "59", "DA"});
   tester.main(new String[]{"5", "10", "109", "109", "108", "108", "105", "105", "102", "102", "14", "14", "93",
"93", "55", "55", "81", "81", "21", "21", "60", "60", "NU", "10", "109", "109", "108", "108", "105", "105", "102",
"102", "14", "14", "93", "93", "55", "55", "81", "81", "21", "21", "60", "60", "NU", "10", "109", "109", "108",
"108", "105", "105", "102", "102", "14", "14", "93", "93", "55", "55", "81", "81", "21", "21", "60", "60", "NU",
"10", "109", "109", "108", "108", "105", "105", "102", "102", "14", "14", "93", "93", "55", "55", "81", "81", "21",
"21", "60", "60", "NU", "10", "109", "109", "108", "108", "105", "105", "102", "102", "14", "14", "93", "93",
"55", "55", "81", "81", "21", "21", "60", "60", "NU"});
   tester.main(new String[]{"5", "10", "100", "100", "101", "101", "102", "102", "103", "103", "104", "104",
"105", "105", "106", "106", "107", "107", "108", "108", "109", "109", "DA", "10", "109", "109", "108", "108",
"107", "107", "106", "106", "105", "105", "104", "104", "103", "103", "102", "102", "101", "101", "100", "100",
"DA", "10", "100", "100", "101", "101", "102", "102", "103", "103", "104", "104", "105", "105", "106", "106",
"107", "107", "108", "108", "109", "109", "DA", "10", "109", "109", "108", "108", "107", "107", "106", "106",
"105", "105", "104", "104", "103", "103", "102", "102", "101", "101", "100", "100", "DA", "10", "109", "109",
"108", "108", "107", "107", "106", "106", "105", "105", "104", "104", "103", "103", "102", "102", "101", "101",
"100", "100", "DA"});

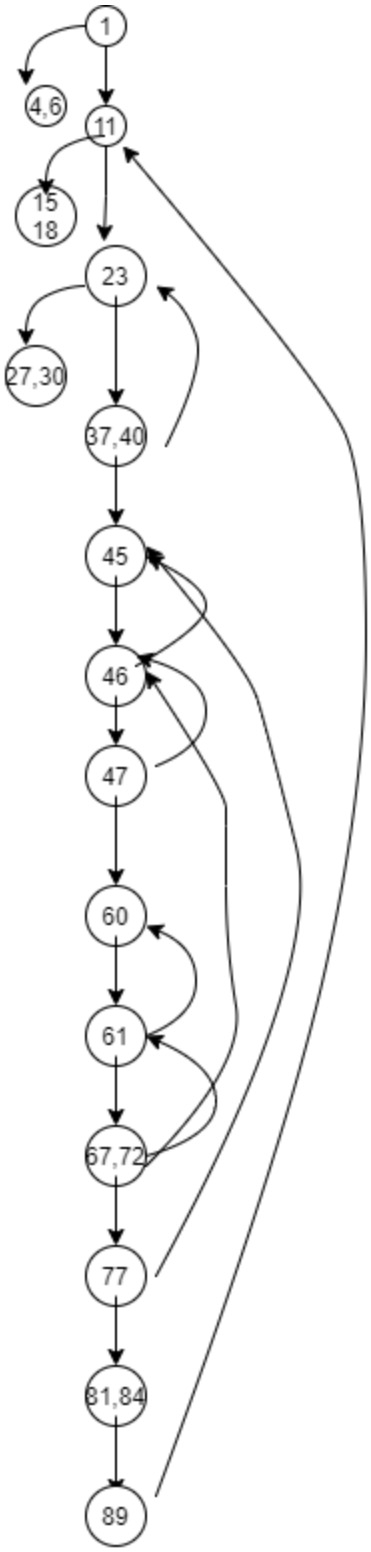
}
```

4. Testarea structurala

```
1    T = Integer.parseInt(args[a]);
2       a++;
3       if(T < 1) {
4          System.out.println("T < 1");
5          return;
6       } else if (T > 5) {
7          System.out.println("T > 5");
8          return;
9       }
10
11      for(t = 0; t < T; t++) {
```

```
12
13          N = Integer.parseInt(args[a]);
14          a++;
15          if(N < 1) {
16              System.out.println("N < 1");
17              return;
18          } else if (N > 10) {
19              System.out.println("N > 10");
20              return;
21          }
22
23          for(i = 0; i < N; i ++) {
24              x[i] = Integer.parseInt(args[a]);
25              a++;
26
27              if(x[i] < 0) {
28                  System.out.println("x < 0");
29                  return;
30              } else if (x[i] > 109) {
31                  System.out.println("x > 109");
32                  return;
33              }
34
35              y[i] = Integer.parseInt(args[a]);
36              a++;
37              if(y[i] < 0) {
38                  System.out.println("y < 0");
39                  return;
40              } else if (y[i] > 109) {
41                  System.out.println("y > 109");
42                  return;
43              }
44          }
45          for(i = 0; i < N-1; i ++) {
46              for(j = i+1; j < N; j++) {
47                  if ((x[i] == x[j]) && (y[i] == y[j])) {
48                      System.out.println("2 stele cu aceleasi coordonate");
49                      return;
50                  }
51              }
52          }
53
54          plagiat = args[a];
55          a++;
56           System.out.println("plagiat = " + plagiat);
57
58          map.clear();
59          for(i = 0; i < N-1; i ++) {
60              for(j = i+1; j < N; j++) {
61
62                  int X = Math.abs(x[j] - x[i]);
63                  int Y = Math.abs(y[j] - y[i]);
64                  Pair pair = new Pair(X, Y);
65                  Integer value = map.get(pair);
66                  if (value == null) {
67                      value = 0;
```

```
68                }
69                value++;
70                map.put(pair, value);
71                if(map.get(pair) >= 3) {
72                    ok = true;
73                    break;
74                }
75            }
76            if(ok == true) {
77                break;
78            }
79        }
80        if(ok == true){
81            plagiat = "DA";
82            System.out.println(plagiat);
83        } else {
84            plagiat = "NU";
85            System.out.println(plagiat);
86        }
87    }
88 }
89
```


1
4,6
11
15
18
23
27,30
37,40
45
46
47
60
61
67,72
77
81,84
89

a) Acoperire la nivel de instructiune

Intrari				Rezultat afisat (expected)
T	N	x,y	R	
1	5	1 1 2 2 1 2 0 0 0 1	DA	Rezultat “ DA”
1	5	1 1 2 2 1 2 0 0 100 105	NU	Rezultat “NU”

```
@Test
public void statementCoverage(){

   tester.main(new String[]{"1", "5", "1", "1", "2", "2", "1", "2", "0", "0", "100", "105", "NU"});
   tester.main(new String[]{"1", "5", "1", "1", "2", "2", "1", "2", "0", "0", "0", "1", "DA"});

}
```

b) Acoperire la nivel de decizie

- if(T < 1) {
  System.out.println("T < 1");
  return;
  } else if (T > 5) {
  System.out.println("T > 5");
  return;
  }

- if(N < 1) {

  System.out.println("N < 1");

  return;

  } else if (N > 10) {

  System.out.println("N > 10");

  return;

  }

- ```
  if(x[i] < 0) {
      System.out.println("x < 0");
      return;
  } else if (x[i] > 109) {
      System.out.println("x > 109");
      return;
  }
  ```
- ```
  if(y[i] < 0) {
      System.out.println("y < 0");
      return;
  } else if (y[i] > 109) {
      System.out.println("y > 109");
      return;
  }
  ```
- ```
  if ((x[i] == x[j]) && (y[i] == y[j])) {
      System.out.println("2 stele cu aceleasi coordonate");
      return;
  }
  ```
- ```
  if (value == null) {
      value = 0;
  }
  ```
- ```
  if(map.get(pair) >= 3) {
      ok = true;
      break;
  }
  ```
- ```
  if(ok == true) {
      break;
  }
  }
  ```
- ```
  if(ok == true){
      plagiat = "DA";
      System.out.println(plagiat);
  } else {
      plagiat = "NU";
  ```
  ```

System.*out*.println(plagiat);
}

```
@Test
public void branchCoverage() {

    tester.main(new String[]{"0"});
    tester.main(new String[]{"1", "0"});
    tester.main(new String[]{"1", "1", "-1"});
    tester.main(new String[]{"1", "2", "0", "0", "0", "0"});
    tester.main(new String[]{"1", "5", "1", "1", "2", "2", "1", "2",
"0", "0", "100", "105", "NU"});
    tester.main(new String[]{"1", "5", "1", "1", "2", "2", "1", "2",
"0", "0", "0", "1", "DA"});

}
```

c) Acoperire la nivel de conditie

- T < 1 , T > 5
- iN < 1, N > 10
- x[i] < 0, x[i] > 109, y[i] < 0, y[i] > 109
- (x[i] == x[j]) && (y[i] == y[j])
- ok == true

Intrari				Rezultat afisat (expected)	Conditii individuale
T	N	x,y	R		
0				T < 1	T < 1
6				T > 5	T > 5
1	0			N < 1	N < 1
1	11			N > 10	N > 10
1	3	-1,y		X < 0	x[i] < 0
1	3	110,y		X > 109	x[i] > 109
1	3	X,-1		Y < 0	y[i] < 0
1	3	X,110		Y > 109	y[i] > 109

1	3	2 2 1 1 2 2		Doua stele cu aceleasi coordonate	(x[i] == x[j]) && (y[i] == y[j])
1	5	1 1 2 2 1 2 0 0 0 1	DA	Rezultat “ DA”	Ok == true
1	5	1 1 2 2 1 2 0 0 100 105	NU	Rezultat “NU”	Ok == true

```
@Test
public void conditionCoverage() {

    tester.main(new String[]{"0"});
    tester.main(new String[]{"6"});
    tester.main(new String[]{"1", "0"});
    tester.main(new String[]{"1", "11"});
    tester.main(new String[]{"1", "1", "-1"});
    tester.main(new String[]{"1", "1", "110"});
    tester.main(new String[]{"1", "1", "0", "-1"});
    tester.main(new String[]{"1", "1", "0", "110"});
    tester.main(new String[]{"1", "2", "0", "0", "0", "0"});
    tester.main(new String[]{"1", "5", "1", "1", "2", "2", "1", "2",
"0", "0", "100", "105", "NU"});
    tester.main(new String[]{"1", "5", "1", "1", "2", "2", "1", "2",
"0", "0", "0", "1", "DA"});

}
```

5. Complexitatea metodei si numarul de circuite independente

Formula lui McCabe:

V(G) = e − n + 2.

e = numărul de muchii ale graficului.

n = numărul de noduri ale graficului.

e = 23

n = 16

V(G) = 23 – 16 + 2 = 9

Circuite independente:

- 11,23,37,40,45,46,47,60,61,67,72,77,81,84,89,11
- 23,37,40,23
- 45,46,45
- 46,47,46
- 46,47,60,61,67,72,46
- 45,46,47,60,61,67,72,77,45

```
@Test
public void circuitsCoverage() {

    tester.main(new String[]{"1", "5", "1", "1", "2", "2", "1", "2", "0",
"0", "100", "105", "NU"});
    tester.main(new String[]{"1", "5", "1", "1", "2", "2", "1", "2", "0",
"0", "0", "1", "DA"});

}
```

6. Testarea la nivel de cale

Expresia regulate:

1.(4+6).11*((15+16).23*((27+30).(37+40)).45*(4*47.60*61.(67+72).77).(81+84).89)

Numar de cai: 1.(1+1).1.(1+1).1.(1+1).(1+1).1.(1.1.1.1.(1+1).1.(1+1).1 = 64