

Domeniu de intrari:

- Un intreg "N", ia valorile [1], [2] si [orice alta valoare], deci se disting clasele de echivalenta:

$N_1 = 1$

$N_2 = 2$

$N_3 = \{n \mid n \neq 1; n \neq 2\}$

- N = 1 (solve):

- Un string "str" care trebuie sa contina cel putin 15 mutari, iar mutarile trebuie sa fie valide, deci apar clasele de echivalenta:

$Str_1 = \{str \mid len(str) \geq 15, str \text{ valid}\}$

$Str_2 = \{str \mid len(str) < 15, str \text{ valid}\}$

$Str_3 = \{str \mid len(str) > 15, str \text{ invalid (are alte caractere decat cele asignare mutarilor valide)}\}$

- N = 2 (scramble):

- Un numar intreg "seed", intre 0 si 1000 inclusiv, avem clasele de echivalenta:

$Seed_1 = \{seed \mid seed \geq 0; seed \leq 1000\}$

$Seed_2 = \{seed \mid seed < 0; seed > 1000\}$

- Un caracter "opt" care poate fi {y} sau orice alta valoare

$Opt_1 = \{y\}$

$Opt_2 = \{opt \mid opt \neq "y"\}$

Domeniul de iesiri:

- Solutia pentru N=1 si amestecarea pentru N=2, nu separa programul in clase de echivalenta diferite mai mult decat o face domeniul de iesiri

Clase de echivalenta globale:

$C_{11} = \{(n, str) \mid n \in N_1, str \in Str_1\}$

$C_{12} = \{(n, str) \mid n \in N_1, str \in Str_2\}$

$C_{13} = \{(n, str) \mid n \in N_1, str \in Str_3\}$

$C_{211} = \{(n, seed, opt) \mid n \in N_2, seed \in Seed_1, opt \in Opt_1\}$

$C_{221} = \{(n, seed, opt) \mid n \in N_2, seed \in Seed_2, opt \in Opt_1\}$

$C_{212} = \{(n, seed, opt) \mid n \in N_2, seed \in Seed_1, opt \in Opt_2\}$

$C_{222} = \{(n, seed, opt) \mid n \in N_2, seed \in Seed_2, opt \in Opt_2\}$

$C_3 = \{n \mid n \in N_3\}$

Valori de frontiera:

Pentru a acorda testa valorile de frontiera ale programului nostru, vom alege valori la frontiera tuturor intervalelor existente in program, in testele alese dupa clasele de echivalenta,.

Intervalele existente sunt:

- $\text{len}(\text{str}) < \text{si} \geq 15$ .
- $\text{seed} \in \text{sau} \notin \{1, 2 \dots 1000\}$

Avand aceste considerente in vedere, vom alege urmatorul set de date de test:

$c_{11} = (1, \text{"B R2 B D R F D' R L B U2 F2 R F' R"})$ ;  $\text{len}(\text{str}) = 15$   
 $c_{12} = (1, \text{"F2 L' R U D2 L' U F2 U2 L B' R' L' B"})$ ;  $\text{len}(\text{str}) = 14$   
 $c_{13} = (1, \text{"F B2 U R' L F L' EROARE F D2 F' D' L B' D2"})$ ;  $\text{len}(\text{str}) = 15$   
 $c_{211} = (2, 1000, y)$   
 $c_{221} = (2, 1001, y)$   
 $c_{212} = (2, 1000, n)$   
 $c_{222} = (2, 1001, n)$   
 $c_3 = (3)$

Acestea se gasesc in fisierul test\_main\_scrambler.py, in clasa TestFunctional.

Testare structurala:

Au fost testate structural clasele main.py si scrambler.py. Numerele cu ' reprezinta liniile din clasa scrambler.py.

(a) Statement coverage (acoperire la nivel de instructiune)

Nume test	Intrari N Str Seed Opt	Rezultat afisat	Instructiuni Parcuse
s_1	1 "B R2 B D R F D' R L B U2 F2 R F' R" - -	"solution: R F' D2 B2 L B U2 B2 R D' F D2 L2 D2 L2 D' R2 D' L2 D R2"	9, 10, 11, 13, 14, 15, 17, 19, 20

s_2	1 "B R2 B D R F D' R L B U2 F2 R F' A" - -	"Exception: scramble contains non-legal moves"	9, 10, 11, 13, 14, 15, 17, 18
s_3	1 "B R2" - -	"Exception: scramble should have at least 15 moves"	9, 10, 11, 13, 14, 15, 16
s_4	3 - - -	-program ends-	9, 10, 11, 12
s_5	2 - 5 y	"scramble generated: F' B' L2 U R' F2 U2 D' B' L F2 R L' D F2 U F R2 F' D' Do you want to solve the scramble ? (y/n)y solution: U R U2 B' U2 L F2 D B U' D2 B U' R2 B2 U' R2 U2 F2 L2 D2 F2"	9, 10, 11, 13, 21, 22, 23, 10', 11', 13', 14', 15', 16', 17', 18', 19', 20', 24, 25, 26, 27
s_6	2 - 1005 y	"Exception: seed should be 0-1000"	9, 10, 11, 13, 21, 22, 23, 10', 11', 12'

Testele se gasesc in fisierul test\_main\_scrambler.py, in clasa TestStructural

(b) Decision coverage (acoperire la nivel de decizie) sau branch coverage

Nume test	Intrari N Str Seed Opt	Rezultat afisat	Instructiuni Parcuse
s_1	1 "B R2 B D R F D' R L B U2 F2 R F' R" - -	"solution: R F' D2 B2 L B U2 B2 R D' F D2 L2 D2 L2 D' R2 D' L2 D R2"	9, 10, 11, 13, 14, 15, 17, 19, 20
s_2	1 "B R2 B D R F D' R L B U2 F2 R F' A" - -	"Exception: scramble contains non-legal moves"	9, 10, 11, 13, 14, 15, 17, 18
s_3	1 "B R2" - -	"Exception: scramble should have at least 15 moves"	9, 10, 11, 13, 14, 15, 16
s_4	3 - - -	-program ends-	9, 10, 11, 12
s_5	2 - 5 y	"scramble generated: F' B' L2 U R' F2 U2 D' B' L F2 R L' D F2 U F R2 F' D' Do you want to solve the scramble ? (y/n)y solution: U R U2 B' U2 L F2 D B U' D2 B U' R2 B2 U' R2 U2 F2 L2 D2 F2"	9, 10, 11, 13, 21, 22, 23, 10', 11', 13', 14', 15', 16', 17', 18', 19', 20', 24, 25, 26, 27
s_6	2 - 1005	"Exception: seed should be 0-1000"	9, 10, 11, 13, 21, 22, 23, 10', 11', 12'

	y		
s_7	2 - 1000 n	"scramble generated: R2 D R B' U' R' D' L' R2 D2 F2 B' L F B' F2 R' D' U2 L Do you want to solve the scramble ? (y/n)n"	

Explicatie:

Acum este evaluata, pe ultima linie a tabelului, si evaluarea negativa a deciziei de rezolvare a scramble-ului.

Testele se gasesc in fisierul test\_main\_scrambler.py, in clasa TestStructural

## (c) Condition coverage (acoperire la nivel de condiție)

Nume test	Intrari N Str Seed Opt	Rezultat afisat	Instructiuni Parcuse
s_1	1 "B R2 B D R F D' R L B U2 F2 R F' R" - -	"solution: R F' D2 B2 L B U2 B2 R D' F D2 L2 D2 L2 D' R2 D' L2 D R2"	9, 10, 11, 13, 14, 15, 17, 19, 20
s_2	1 "B R2 B D R F D' R L B U2 F2 R F' A" - -	"Exception: scramble contains non-legal moves"	9, 10, 11, 13, 14, 15, 17, 18
s_3	1 "B R2" - -	"Exception: scramble should have at least 15 moves"	9, 10, 11, 13, 14, 15, 16
s_4	3 - - -	-program ends-	9, 10, 11, 12
s_5	2 - 5 y	"scramble generated: F' B' L2 U R' F2 U2 D' B' L F2 R L' D F2 U F R2 F' D' Do you want to solve the scramble ? (y/n)y solution: U R U2 B' U2 L F2 D B U' D2 B U' R2 B2 U' R2 U2 F2 L2 D2 F2"	9, 10, 11, 13, 21, 22, 23, 10', 11', 13', 14', 15', 16', 17', 18', 19', 20', 24, 25, 26, 27
s_6	2 - 1005 y	"Exception: seed should be 0-1000"	9, 10, 11, 13, 21, 22, 23, 10', 11', 12'

s_7	2 - 1000 n	“scramble generated: R2 D R B' U' R' D' L' R2 D2 F2 B' L F B' F2 R' D' U2 L Do you want to solve the scramble ? (y/n)n”	
s_8	2 - -3 n	“Exception: seed should be 0-1000”	

Explicatie:

Acum, pe ultima linie, este evaluata si conditia individuala a deciziei de la linia 11', care verifica daca seed-ul este incadrat corespunzator.

Testele se gasesc in fisierul test\_main\_scrambler.py, in clasa TestStructural

## Testare Pe Mutanti

Pentru Mutanti, am folosit libraria mutpy si am obtinut urmatorul raport pentru clasa main.py (in care se afla interfata text-based pentru interactiunea cu utilizatorul):

```
- [# 7] COI main: [0.00345 s] killed by test_c_11 (test_main_scrambler.TestFunctional)
- [# 8] COI main: [0.00408 s] killed by test_c_11 (test_main_scrambler.TestFunctional)
- [# 9] COI main: [0.00968 s] killed by test_c_211 (test_main_scrambler.TestFunctional)
- [# 10] COI main: 1 for solver, 2 for scrambler, anything else for exit: 1
input valid scramble for the solver: F F
[0.00000 s] incompetent
- [# 11] ROR main: [0.00373 s] killed by test_c_11 (test_main_scrambler.TestFunctional)
- [# 12] ROR main: solution: U2 R' L2 D2 R2 F R' L F D2 R' F2 U' L2 U' R2 L2
[0.02399 s] killed by test_c_12 (test_main_scrambler.TestFunctional)
- [# 13] ROR main: [0.00371 s] killed by test_c_11 (test_main_scrambler.TestFunctional)
- [# 14] ROR main: [0.00962 s] killed by test_c_211 (test_main_scrambler.TestFunctional)
- [# 15] ROR main: 1 for solver, 2 for scrambler, anything else for exit: 2
Introduce a seed (0-1000) for the scrambler: 2
scramble generated: U2 B' D' B F' U2 D' R2 L2 B' L' R' L2 F' U2 B' R' B2 R L2
Do you want to solve the scramble ? (y/n)n
1 for solver, 2 for scrambler, anything else for exit: 2
Introduce a seed (0-1000) for the scrambler: n
[0.00000 s] incompetent
[*] Mutation score [21.20429 s]: 100.0%
- all: 15
- killed: 12 (80.0%)
- survived: 0 (0.0%)
- incompetent: 2 (13.3%)
- timeout: 1 (6.7%)
o stefan@stefan-ASUS:~/TOATE/Facultate/An 3 sem 2/Testare/TSS-PythonUnitTesting$
```

O rata de ucidere de 80% la prima iteratie de teste demonstreaza ca am aplicat strategiile prezentate la curs in mod eficient.

Cei 3 mutanti care nu au fost ucisi au scapat deoarece faptul ca generatorul de mutanti este derutat de faptul ca noi cerem input de la utilizator. In textul de mai sus, de deasupra raportului final, tag-ul "Incompetent" apare la mutantii care au cerut input in timpul generarii.

In afara cazului in care gasim un mod de a mock-ui input-ul de la utilizator in timpul generarii de mutanti, credem ca rata de ucidere nu se poate imbunatati.



Comparatie cu teste propuse de GPT:

Dupa ce am introdus tot codul nostru din clasele main.py si scrambler.py in Chat-GPT 3.5 (versiunea gratuita), am primit urmatorul raspuns:

---

"Here are some tests you could perform for the provided Python code:

1. **Test for Solver Functionality:**
  - Input: Choose option 1 for the solver.
    - Input a valid scramble sequence.
  - Expected Output: A solution for the provided scramble sequence.
2. **Test for Scrambler Functionality:**
  - Input: Choose option 2 for the scrambler.
    - Input a seed value within the range (0-1000).
    - Optionally, choose to solve the generated scramble.
  - Expected Output: A scramble generated based on the given seed.
3. **Test for Minimum Scramble Length:**
  - Input: Choose option 1 for the solver.
    - Input a scramble sequence with fewer than 15 moves.
  - Expected Output: An Exception should be raised indicating that the scramble should have at least 15 moves.
4. **Test for Valid Scramble Moves:**
  - Input: Choose option 1 for the solver.
    - Input a scramble sequence containing non-legal moves.
  - Expected Output: An Exception should be raised indicating that the scramble contains non-legal moves.
5. **Test for Seed Range:**
  - Input: Choose option 2 for the scrambler.
    - Input a seed value less than 0 or greater than 1000.
  - Expected Output: An Exception should be raised indicating that the seed should be within the range of 0-1000.
6. **Test for Consistency of Scramble Generation:**
  - Input: Generate multiple scrambles with the same seed.
  - Expected Output: The same scramble should be generated for the same seed.
7. **Test for Solver Consistency:**
  - Input: Generate a scramble using the scrambler and solve it using the solver.
  - Expected Output: The solution should be consistent for the same scramble.
8. **Test for Input Validation:**
  - Input: Enter an invalid option (anything other than '1' or '2').
  - Expected Output: The program should exit the loop.
9. **Test for Optional Solver Prompt:**
  - Input: Choose option 2 for the scrambler and choose not to solve the generated scramble.

- Expected Output: The program should return to the main menu without attempting to solve the scramble.

These tests should cover most scenarios and ensure the functionality and robustness of the code.”

---

Observam ca testele sunt grupate dupa niciun fel de strategie de testare, ci sunt oferite in mod arbitrar, si cele mai multe sunt destul de evidente, ca 1, 2, sau 3.

Totusi, suntem surprinsi sa observam idei interesante, la care noi nu ne-am gandit, de tipul sugestiei 6 (consistenta seed-ului) si 7 (care foloseste o parte a aplicatiei in testarea celeilalte parti).

Cand, mai departe, am intrebat GPT-ul sa ofere teste bazate pe partitii de echivalenta, am primit urmatorul raspuns:

---

“Certainly! Equivalent partitioning, also known as equivalence class partitioning, is a technique used to divide the input space into partitions such that each partition behaves in the same way. Here are the tests grouped by equivalent partitions:

### **Option Selection:**

1. **Valid Option (Solver):**
  - Input: '1'
  - Expected: Proceed to solver functionality.
2. **Valid Option (Scrambler):**
  - Input: '2'
  - Expected: Proceed to scrambler functionality.
3. **Invalid Option (Exit):**
  - Input: Any value not equal to '1' or '2'.
  - Expected: Program should exit the loop.

### **Scramble Length:**

1. **Valid Scramble Length:**

- Input: Scramble length  $\geq 15$  moves.
- Expected: Proceed with solving the scramble.
- 2. **Invalid Scramble Length:**
  - Input: Scramble length  $< 15$  moves.
  - Expected: Raise Exception indicating scramble should have at least 15 moves.

## Scramble Validity:

1. **Valid Scramble Moves:**
  - Input: Scramble with only legal moves.
  - Expected: Proceed with solving the scramble.
2. **Invalid Scramble Moves:**
  - Input: Scramble with non-legal moves.
  - Expected: Raise Exception indicating scramble contains non-legal moves.

## Seed Range:

1. **Valid Seed Range:**
  - Input: Seed value between 0 and 1000.
  - Expected: Proceed with generating the scramble.
2. **Invalid Seed Range (Lower Bound):**
  - Input: Seed value  $< 0$ .
  - Expected: Raise Exception indicating seed should be 0-1000.
3. **Invalid Seed Range (Upper Bound):**
  - Input: Seed value  $> 1000$ .
  - Expected: Raise Exception indicating seed should be 0-1000.

## Optional Solver Prompt:

1. **Solve Scramble (Yes):**
  - Input: 'y' after scramble generation.
  - Expected: Print the solution for the generated scramble.
2. **Do Not Solve Scramble (No):**
  - Input: 'n' after scramble generation.
  - Expected: Return to the main menu without solving the scramble.

## Other Tests:

1. **Consistency of Scramble Generation:**
  - Input: Generate multiple scrambles with the same seed.
  - Expected: The same scramble should be generated for the same seed.
2. **Consistency of Solver:**
  - Input: Generate a scramble and solve it multiple times.
  - Expected: The solution should be consistent for the same scramble.

These tests cover various scenarios and ensure that the program behaves correctly under different conditions.”

---

Observam o asemanare deosebita intre clasele de echivalenta conturate de GPT si cele identificate de noi, ceea ce demonstreaza faptul ca strategia este precisa si nu lasa loc de interpretare.

O singura diferenta ce poate fi identificata este faptul ca GPT-ul a desemnat 2 categorii diferite ce creeaza partitii din string-ul de mutari (lungimea si validitatea), pe cand noi le-am unit intr-o singura categorie, mai concisa, creatoare de mai multe partitii.

De asemenea, GPT-ul nu a reusit sa reuneasca partiile locale generate pentru a construi partiile globale.