Trabajo Práctico #1: RandPy

Entrega: Miércoles 28 de Septiembre

Se desea construir un generador aleatorio para el lenguaje Python. Su implementación debe ser capaz de generar invocaciones a las funciones definidas en un módulo python. Este módulo es indicado usando una ruta a un archivo con extensión .py.

Para simplificar vamos a suponer que los programas bajo análisis no contendrán declaraciones de clases. También vamos a ignorar todas las funciones cuyo nombre comience con un "_".
El pseudo-código de RandPy es el siguiente:

```
INPUT: filePy, maxIters, maxTime, outputDir, random_seed

iterCount := 0
line_branch_coverage := 0
tests := {}
while (iterCount < maxIters or maxIters==-1) and time()<maxTime:
  t = generate_new_test(filePy, random_seed)
  r = execute_test(t)
  if r is TypeError:
     continue
  endif
  new_coverage := compute_line_branch_coverage(tests + t)
  if new_coverage > line_branch_coverage:
    line_branch_coverage := new_coverage
    tests.add(test)
    write_test_to_file(outputDir, test)
  endif
  iterCount:=iterCount+1
endwhile
```

donde:

- `filePy` es la ruta al archivo python con el módulo bajo test. Es una ruta absoluta (ejemplo: /home/johndoe/autotest/tp1/examples/arrays.py)
- `maxIters`: es la cantidad máxima de iteraciones permitidas para el algoritmo
- `maxTime`: es el tiempo máximo (en segundos) permitidos para ejecutar RandPy
- `outputDir`: es la ruta (absoluta) a la carpeta donde se almacenarán los tests generados
- `random_seed`: es la semilla para configurar el generador aleatorio de Python

RandPy debe ser capaz de generar argumentos de tipos básicos (bool, integer, float, long, string, etc.) así como de los tipos compuestos list, set, tuple, dict.

Para facilitar su trabajo, se cuenta con clases desarrolladas que permiten las siguientes funcionalidades:

- **TestCall**: representa un Test. Ya que sólo trabajaremos con módulos con funciones python, los test call son únicamente una invocación a una función python con sus argumentos. Un TestCall está compuesto por el nombre de la función a invocar y la lista de argumentos (TestArgument) a usar.
- **TestArgument**: son los argumentos (ie valores de parámetros) con que se necesitan para definir un TestCall.

- **TestExecutor**: ejecuta un TestCall.
- **TestWriter**: permite volcar el contenido de un TestCall en un archivo .py para su ejecución posterior como un unit test.

Evaluación

Una vez terminado el desarrollo de RandPy debe evaluar su desempeño completando la siguiente tabla:

Example	Avg. Line Cov.	Avg. Branch Cov.
arrays.py		
cgi_decode.py		
convexhull.py		
coord.py		
encryption.py		
levenshtein.py		
persons.py		
sets.py		
sorting.py		
triangle.py		
tuples.py		
years.py		

Para ello, deberá ejecutar cada uno de los módulos de la Tabla usando RandPy con la siguiente configuración:

- maxTime=60 (60 segundos)
- maxIters=-1 (sin importar la cantidad de iteraciones)

Dado que la efectividad de RandPy puede variar aleatoriamente, debe repetir la ejecución 10 veces usando distintas semillas de aleatoriedad (random_seed)