Tutorial 7

Swinburne University of Technology

Software Testing and Reliability (SWE30009)

Semester 2, 2023

Lecturer: Prof T. Y. Chen

Tutor: Dr Hung Q Luu

Project Report

Final Assignment

Important notices

- This is an individual assignment.
- Worth 60% of the total unit score.
- Due date: 11:00pm Mon 16 Oct 2023

Task 1

- **Subtask 1.1.** Present your understanding of the random testing methodology.
- **Subtask 1.2.** Apply random testing methodology to generate concreate test cases to test a program.
  - This program is designed to sort a non-empty list of integer numbers that may contain duplicated numbers.

Task 2

- **Subtask 2.1.** Present your understanding of the metamorphic testing methodology.
- **Subtask 2.2.** You are required to apply metamorphic testing to propose at least two MRs to test a program.
  - This program is designed to sort a non-empty list of integer numbers that may contain duplicated numbers.
- **Subtask 2.3.** You are also required to compare the advantage and disadvantage of random testing and metamorphic testing.

Task 3

- You are required to test a real-world program of your choice. Its requirements are
  - The original program under test is implemented correctly and must be obtained from GitHub;
  - The program must be written in either Python, Java, JavaScript, Ruby, C/C++, C#, Swift, Visual Basic, Fortran, R, Go, Perl, PHP or MATLAB; and
  - It is neither too large and complex nor too simple so that you can generate at least 20 non-equivalent mutants.
- You must use metamorphic testing technique and evaluate it using the mutation analysis.
  - You are required to propose and describe at least two metamorphic relations.

Requirement about report

- Report submission
  - Must submit the report as a single PDF file.
  - Must be self-contained and complete.
  - Coverage page is not required.
- Report format
  - Must use 12-point font size on A4 papers.
  - Must contain full name & student number on the first line of first page.
- Report volume
  - Must have no more than 10 pages.
  - Must be smaller than 10 MB in size.

Requirement about codes

- Code must be in a single ZIP file consisting of
  - Complete source codes of the program
  - Complete set of non-equivalent mutants used
  - Test script (if applicable) and test cases
- Code language must be one of the followings
  - Python, Java, JavaScript, Ruby, C/C++, C#, Swift, Visual Basic, Fortran, R, Go, Perl, PHP or MATLAB.
- ZIP volume
  - Must be smaller than 10 MB in size

Other requirements

- Report must have the filename specified in this format “FinalReport-YourStudentID-YourSurname.pdf”
  - Example: FinalReport-12345678-Nash.pdf
- ZIP must have the filename specified in this format “FinalReport-YourStudentID-YourSurname.zip”
  - Example: FinalReport-12345678-Nash.zip
- Must use the submission for Assignment in the unit’s Canvas.
- Submission must be before the due date.

Marking criteria

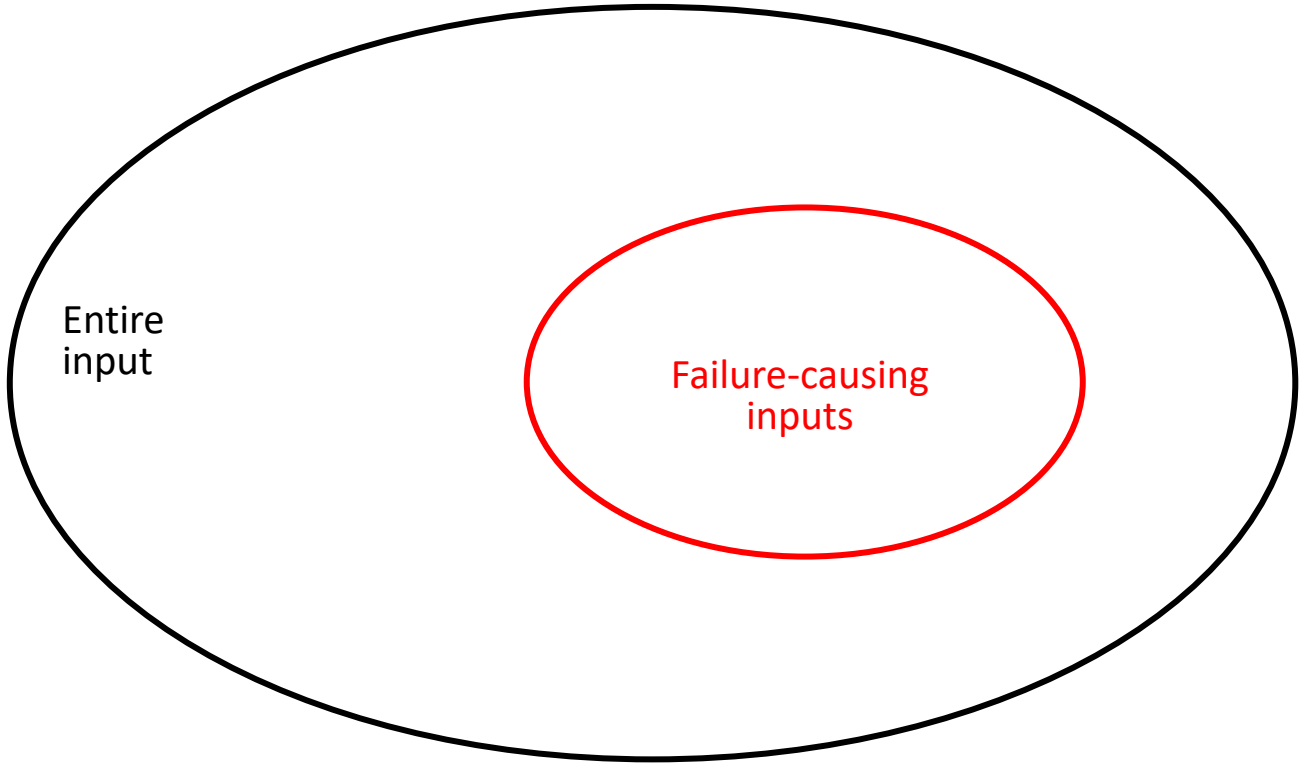
- A maximum of **15 marks for Task 1**.
- A maximum of **25 marks for Task 2**.
- A maximum of **50 marks for Task 3**.
- A maximum of **10 marks is for** presentation, completeness compliance, and cohesion.

Marking penalty

- Penalty will be applied for late submission and plagiarism.
- Refer to the Unit Outline for the policy on late submission and plagiarism.

Random testing

Input domain

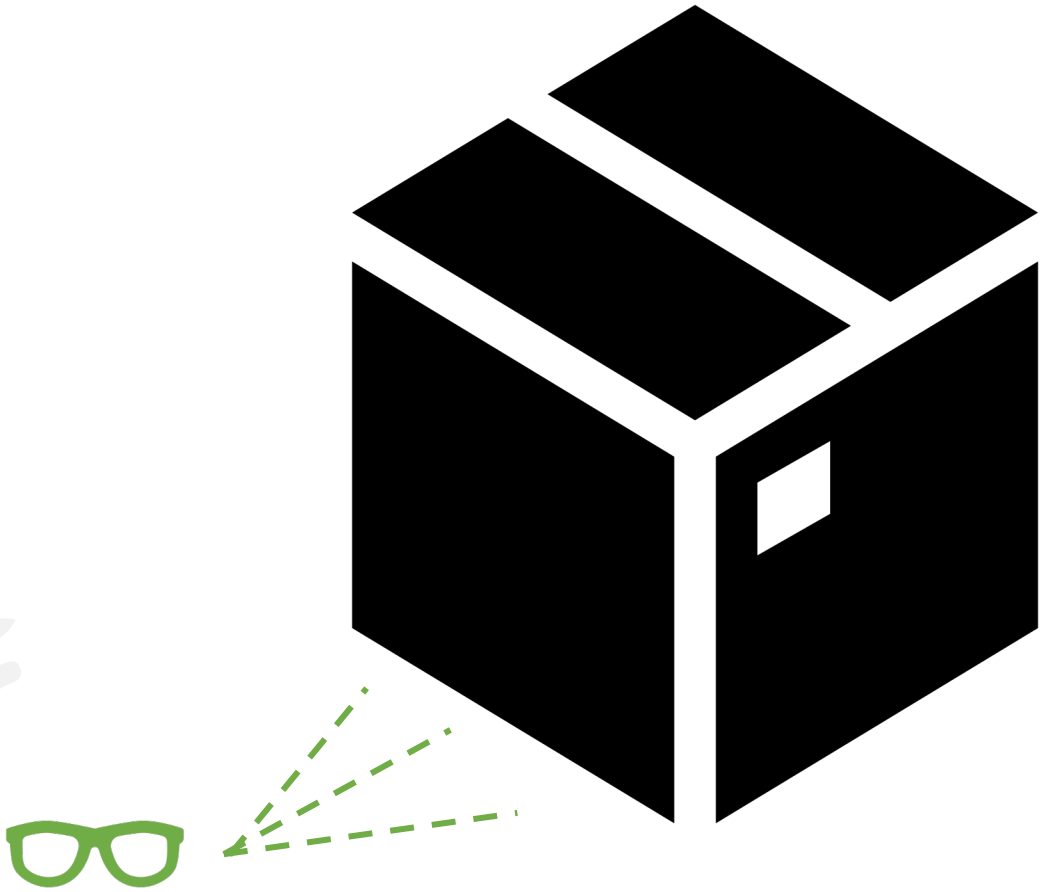


Discussion

Can we test all inputs?

Blackbox testing

Does not refer to the program code



Task: Assignment 2

Blackbox testing: advantage

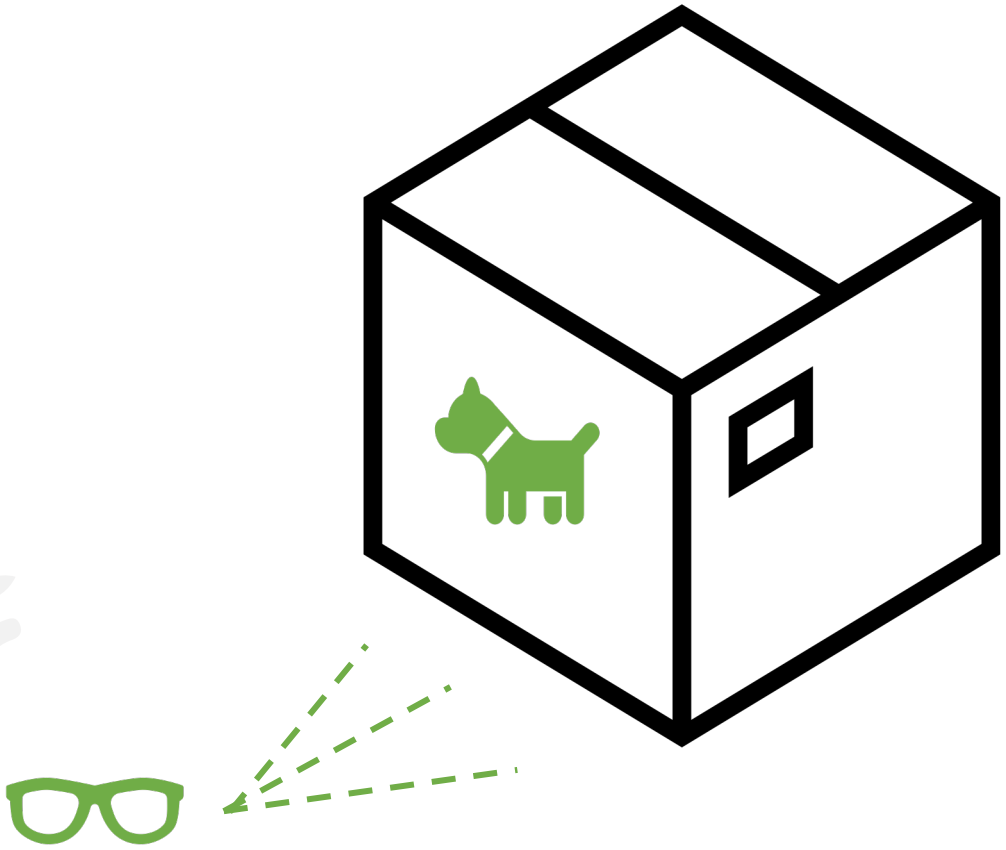
- Design of test cases and coding can be done in parallel
- Can reveal missing functions / features
  - Missing functions / features are those specified but not implemented
- Test cases are independent of the source code
- Test cases can be reused as long as the specifications are not changed

Blackbox testing: disadvantage

- Cannot test extra functions / features
  - Extra functions / features are those implemented but never specified

Whitebox testing

Makes use of the program code



Task: Assignment 1

Whitebox testing: advantage

- Can test those implemented functionality
- Uncover types of errors different from those detected by Black-Box Testing

Whitebox testing: disadvantage

- Cannot test those specified functions which are not implemented
- Need to wait until the source code is ready to start design / generate test case
- Need to know the language syntax to generate test case

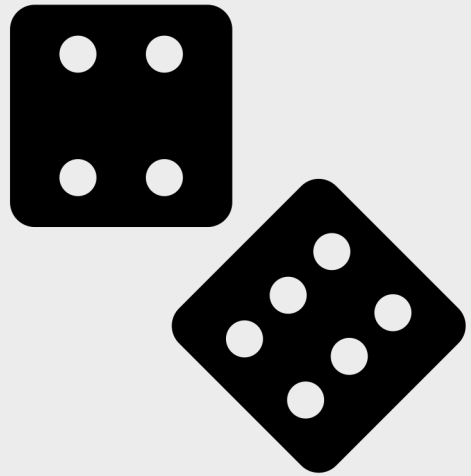
A core blackbox testing method

Random testing

Selects test cases from the

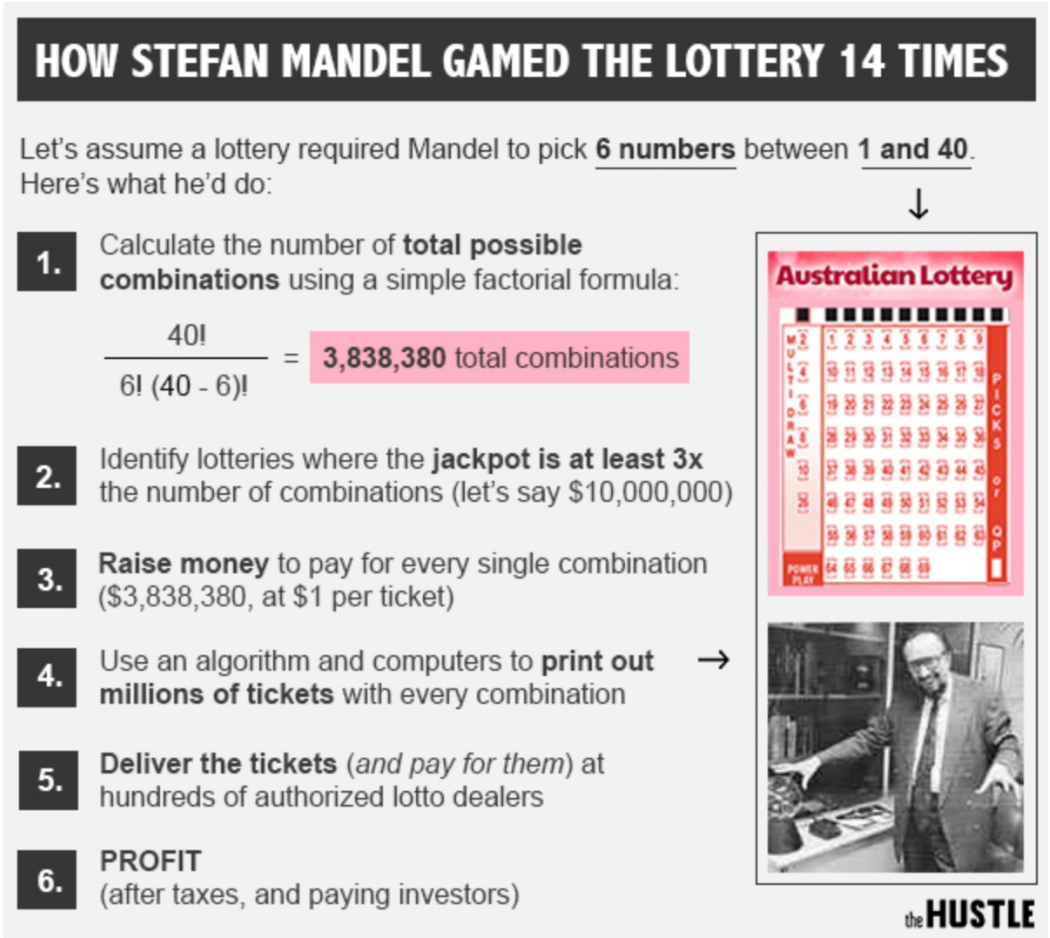
entire input domain **randomly**

and independently



Random story

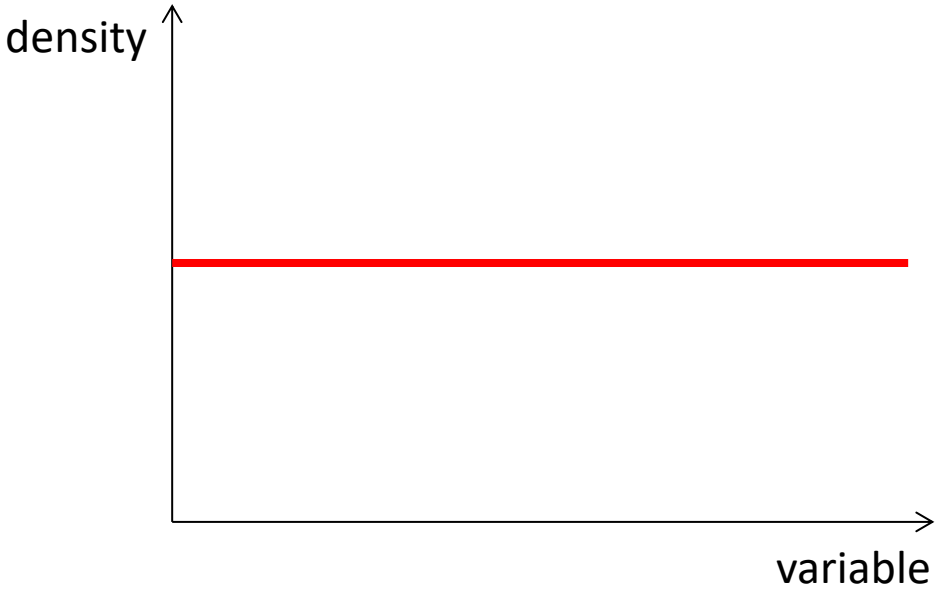
Top: Stefan Mandel spent more than a decade reading mathematical theories before winning his first lottery in the 1960s; Bottom: Mandel's lottery feats made headlines in his hometown Romanian newspaper (Via Busra; Illustrations: The Hustle)



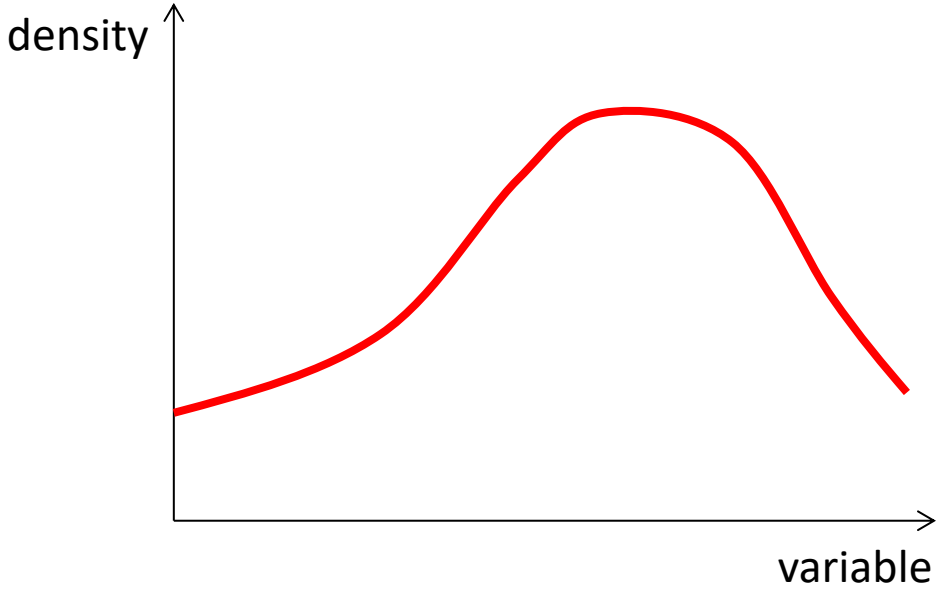
Mandel's system was simple — but incredibly complex from a logistical standpoint (The Hustle)

Source: https://thehustle.co/the-man-who-won-the-lottery-14-times/

Input probability distribution



Uniform distributions



Non-uniform distributions

Discussion on picking a random sample

Melbourne

Sydney

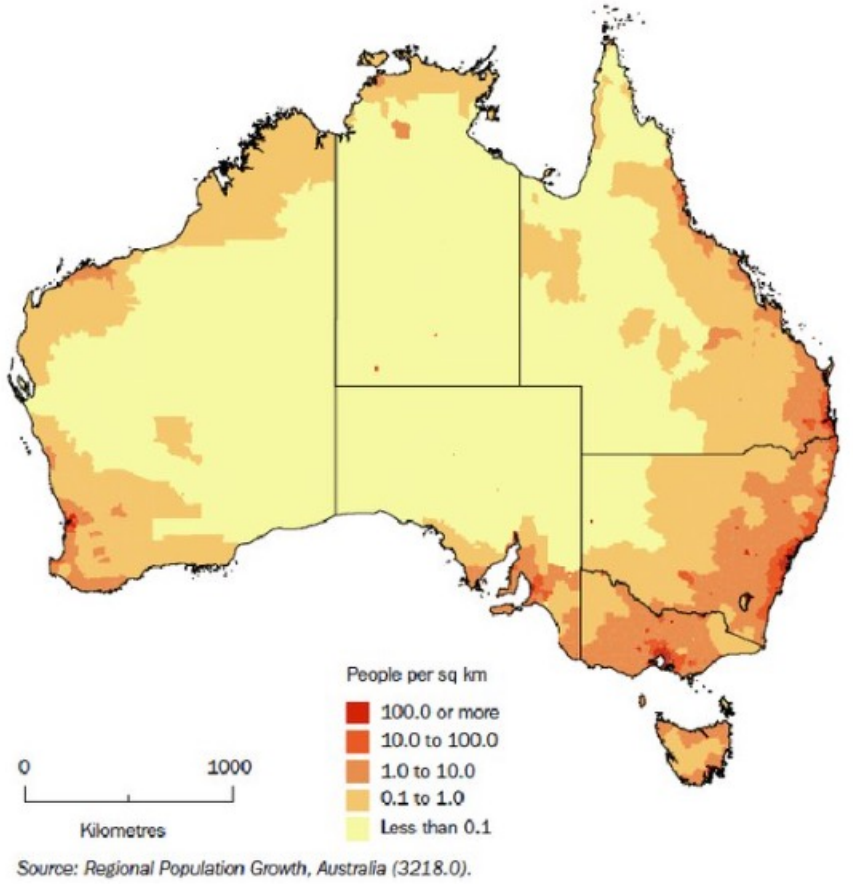
Brisbane

Adelaide

Perth

Darwin

Hobart



Random testing for numeric inputs

```
4  # set reproducable random algorithm
5  random.seed(10)
6  print(numpy.random.seed(0))
```

- Numpy Library: https://docs.python.org/3/library/random.html

```
25 print(numpy.random.rand(10),'random values in a given shape')
26 print(numpy.random.randn(2),'sample from "standard normal" distribution')
27 print(numpy.random.randint(100),'random integers 10[, high, size, dtype]')
28 print(numpy.random.exponential(scale=1.0),'draw samples from an exponential distribution')
29 print(numpy.random.pareto(5),'draw samples from a pareto distribution')
```

- Random Library: https://numpy.org/doc/1.16/reference/routines.random.html

```
32 # random algorithm in numpy
33 print(random.random(),'random value in the range of [0,1]')
34 print(random.uniform(5,15),'random value in the range of [a,b]')
35 print(random.expovariate(5),'draw samples from an exponential distribution with lamda')
36 print(random.choice(['a','b'],'uniform samples from a list')
37 print(random.choices(['a','b'],weights=[0.1,0.9]),'non-uniform samples from a list following an distribution')
```

Random testing for string inputs

- Uniform
- Non-uniform

```python
import random

# inputs and weights (optional)
names = ['John','Peter','Mary']
weights = [0.5, 0.2, 0.3]

# uniform distribution
print('Uniform distribution of names:')
for _ in range(10):
    print(random.choice(names))

# non-uniform distribution
print('Non-uniform distribution of names:')
for _ in range(10):
    print(random.choices(names, weights=weights)[0])

```