CITS5501 Software Testing and Quality Assurance CITS5501 Semester 1, 2018 Workshop 4 – Mutation testing

1. Mutation testing packages

We have noted two mutation testing packages, PIT for Java, and mutpy for Python. If you are familiar with another language, see if you can find mutation testing libraries for it.

2. Sample code

Ensure mutpy is installed in your Python IDE.

Save the following code in a file named `search.py`:

```
def searchList(x, xs):
  i = 0
  while i < len(xs):
    if x == xs[i]:
      return i
  return -1
```

If we applied input space partitioning, what would be some appropriate test cases for this code?

If we check for prime path coverage, do those tests cases cover all the prime paths?

3. mutpy

Save the following code in a file named `test_search.py`:

```python
from unittest import TestCase
import unittest
from search import searchList
from mutpy import commandline

class SearchTest(TestCase):

    def test_alwaysFail(self):
        self.assertTrue(False)


def mutTest():
  import sys
  args = "--target search --unit-test test_search -m".split()
  sys.argv[1:] = args
  commandline.main(args)

if __name__ == '__main__':
  unittest.main()
  # mutTest()
```

Try running this script – it should always fail. Now replace `test_alwaysFail` with *one* of the test cases you derived from input space partitioning. Debug any resulting errors in the `searchList` code.

Now, comment out the `unittest.main()` line, and uncomment the `mutTest()` line, and try running your script again. You should see a very large number of outputs, representing cases where the `searchList` function could be randomly *mutated*, yet the mutant still passed all (in this cases, we only have one) of our tests.

Look at one of the mutants, and see if you can identify what test would be needed in order for that mutant to be "killed".

Now add in the rest of your tests from input space partitioning. Are any mutants still generated and not killed? Are there any test cases which are not exercised by any mutant?

Another use of mutation testing is that it can identify cases where two tests always seem to give the same result (for a particular mutant). In that case, one of the tests might be redundant. Does mutpy offer a way of identifying such test cases? Do any other mutation testing packages you can find?