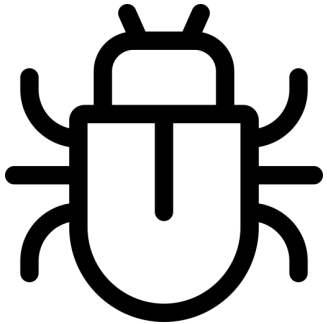
Team 3 CS453 Project Proposal

Muhammad Irfan Akbar 20180854
Irfan Ariq 20204831
Fathony Achmad 20180825
Muhammad Waleed Anwar 20170851

Introduction

Problem introduction

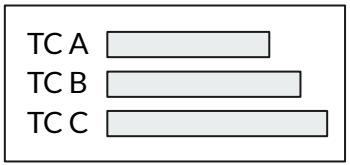
A robust and diverse test cases is needed in many occasions



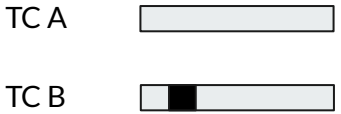
Failure to generate good test cases lead to possible bugs and crash in the future

We aim to use a **modified test case generation** that generates a better test cases with the help of delta debugging tool

Test cases



In most cases, Test Cases are made to find faults



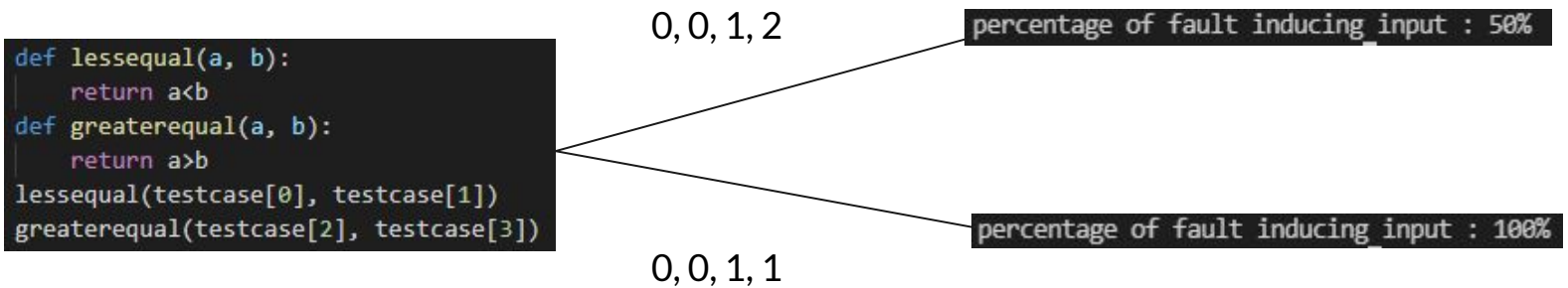
Test cases that have more fault inducing inputs are better

The use of better quality test cases can help reduce time needed to identify faults

What is “Percentage of fault inducing input”?

“Percentage of fault inducing input” is a measurement of input that causes fault to total lengths of input.

Basically, we count how many input causes fault with the help of delta debugging, and divide them to the total lengths.



Proposed Test Case Generation

We want to generate test cases that can find faults in a better way. Specifically, we want to generate test cases that have many fault inducing inputs, in order to trigger more bugs which are not found in previous test cases. In doing so, we calculate the percentage of fault inducing input of each generated test cases, and only keep the one that at least have the same value of percentage of fault inducing input from the previous maximum.

Methods

Baseline test case generation (statement coverage)

```
# 4. start fuzzing
i = 0
print("START TO RUN THE FUZZER")
start_time = time.time()
timer = 60
while True:
    if timer < (time.time() - start_time):
        break
    new_input = input_fuzzer.get_fuzzed_input()
    str_new_input = str(new_input)
    # pass_fail, raise_cov, total_cov = pyscript_runner.run_check([new_input])
    pass_fail, raise_cov, total_cov = pyscript_runner.run_check([str_new_input])
    if pass_fail == ScriptRunner.FAIL:
        input_fuzzer.add_crash(new_input)
    elif raise_cov:
        input_fuzzer.add_input(new_input)
    print("#{} - input:'{}' - total_cov:{}".format((i+1), new_input, total_cov))
    i += 1
```

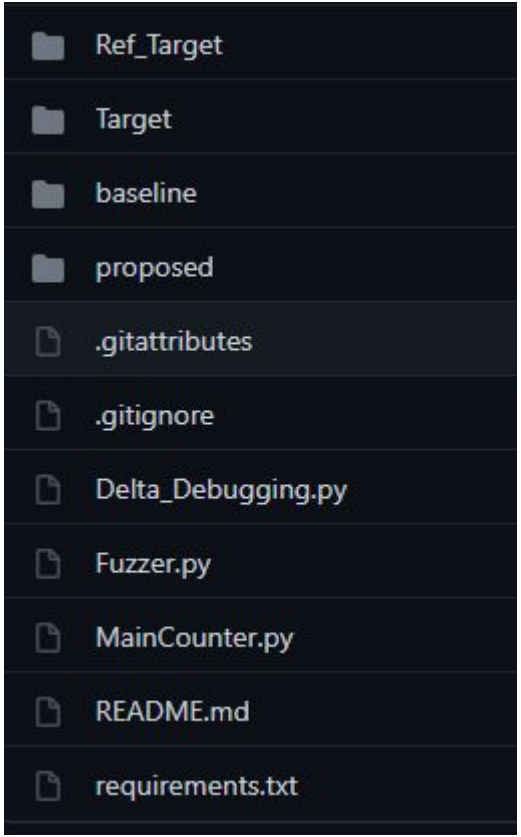
Proposed test case generation (percentage of fault inducing input)

```
current_time = time.time()
#set fuzzer timer
timer = 60

#fuzzing process
while True:
    elapsed = time.time() - current_time

    if timer < elapsed:
        break

    inp = random_fuzzer.fuzz()
    result, outcome = mystery.run(inp)
    if outcome == mystery.FAIL:
        results.append(result)
        dd_reducer = DeltaDebuggingReducer(mystery, log_test=False)
        ddres = dd_reducer.reduce(result)
        pfires = fault_percentage(ddres, result)
        if pfires >= maxpfi:
            push_queue(pfires, result)
```

Comparison to normal fuzzer

We will compare the average percentage of fault inducing input of each test cases generated by both the normal (baseline) fuzzer and our proposed test case generation.

The average percentage of fault inducing input will measure how fast our test case generation to generate such output in comparison to the normal fuzzer.



Baseline vs. Proposed test case generation

```
# 3. reduce the testcase
for test in base_seed:
    result, outcome = mystery.run(test)
    if(outcome == mystery.FAIL):
        dd_reducer = DeltaDebuggingReducer(mystery, log_test=False)
        ddres = dd_reducer.reduce(result)
        pfires = fault_percentage(ddres, result)
        base_pfi.append(pfires)
for test in proposed_seed:
    result, outcome = mystery.run(test)
    if(outcome == mystery.FAIL):
        dd_reducer = DeltaDebuggingReducer(mystery, log_test=False)
        ddres = dd_reducer.reduce(result)
        pfires = fault_percentage(ddres, result)
        proposed_pfi.append(pfires)


# 4. calculate and present the calculated pfi
base_avg_pfi = 100*sum(base_pfi)/len(base_pfi)
prop_avg_pfi = 100*sum(proposed_pfi)/len(proposed_pfi)
print("Average percentage of fault inducing input (baseline fuzzer) : %.2f%%" %(base_avg_pfi))
print("Average percentage of fault inducing input (proposed fuzzer) : %.2f%%" %(prop_avg_pfi))
```

Results

Results

Target program	Baseline	Proposed
Average pfi (dummy)	25%	32.92%
Average pfi (dummy1)	40%	44%
Average pfi (dummy2)	10%	10.43%

Fuzzing time : 60s

Further works

Found suitable target program from external sources to minimize threat to validity

Increase the diversity and number of programs used in the dataset

Run the fuzzer over a greater period of time

Scale the project to allow it to work with more complex and larger codebases

Q&A