Ho Chi Minh City University of Technology, Vietnam
Faculty of Computer Science and Engineering
Lab for Systems Analysis and VErification (SAVE)

Assisting Students in Finding Their Own Bugs in Programming Exercises using Verification and Group Testing Techniques

Long H. Pham, Nam P. Mai, Mai H. Dinh, Tho T. Quan, Hung Q. Ngo

November 6, 2013

http://prove.somee.com/slide.pdf

Outline

1. **Student Program Correctness Problem**

2. **Our approach**

3. **Framework**

4. **How it works**

5. **Case study**

6. **Experiment**

7. **Issues & Future works**

1. Student Program Correctness Problem

- Traditional method: read all programs to verify

  ⇒ 999 999 999 LOC ?

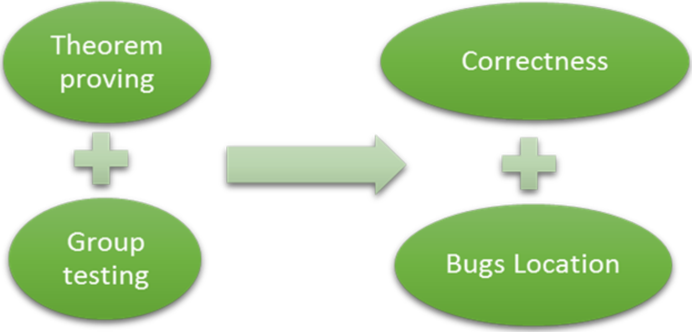
- Automated assessment system: execute, pass all test cases

  ⇒ test suite cover all possible errors ?

  ⇒ real execution is potentially dangerous for the system ?

2. Our approach

- Static methods & Automation


Theorem proving

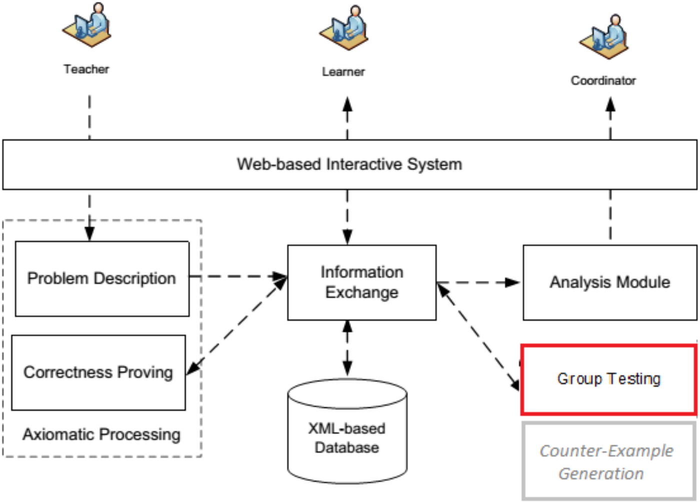
Group testing

Correctness

Bugs Location

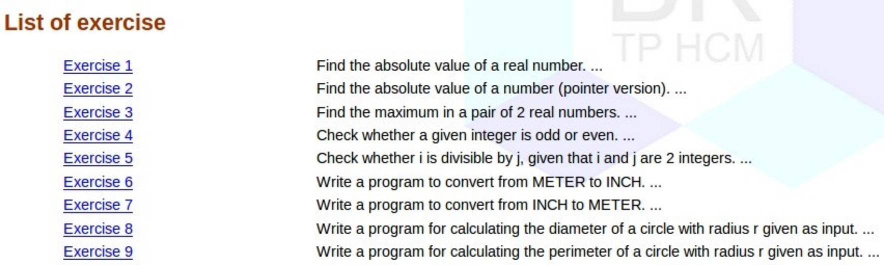
3. Framework

Verification and Group Testing Techniques



Verification and Group Testing Techniques

- Exercise list



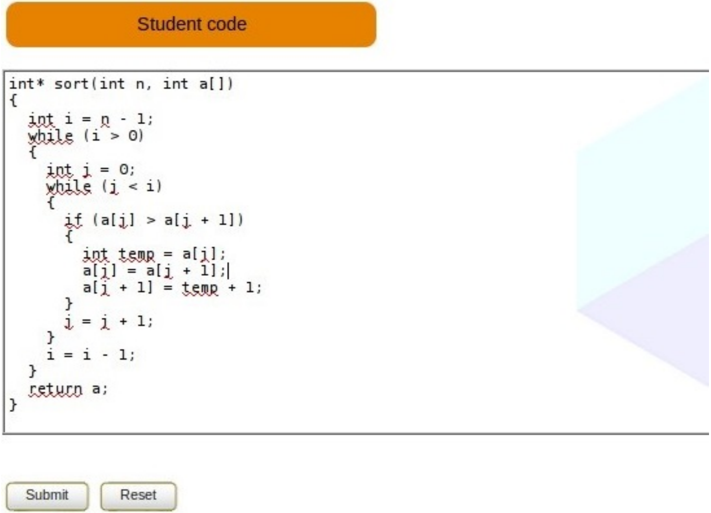
List of exercise

Exercise 1	Find the absolute value of a real number. ...
Exercise 2	Find the absolute value of a number (pointer version). ...
Exercise 3	Find the maximum in a pair of 2 real numbers. ...
Exercise 4	Check whether a given integer is odd or even. ...
Exercise 5	Check whether i is divisible by j, given that i and j are 2 integers. ...
Exercise 6	Write a program to convert from METER to INCH. ...
Exercise 7	Write a program to convert from INCH to METER. ...
Exercise 8	Write a program for calculating the diameter of a circle with radius r given as input. ...
Exercise 9	Write a program for calculating the perimeter of a circle with radius r given as input. ...

Verification and Group Testing Techniques

- Bubble sort problem

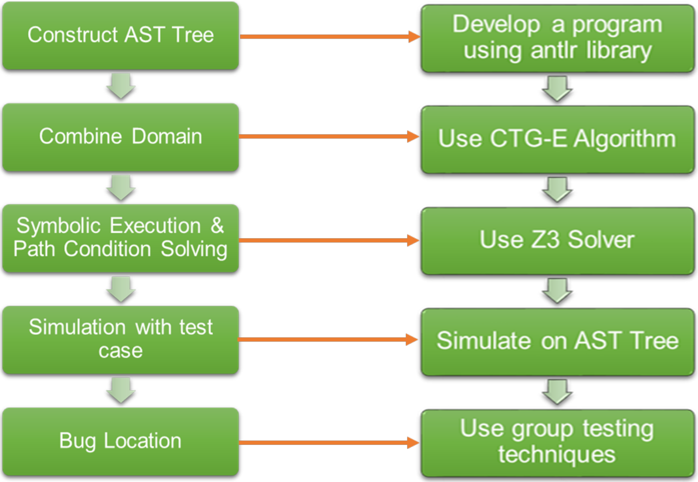
Verification and Group Testing Techniques



```
int* sort(int n, int a[])
{
  int i = n - 1;
  while (i > 0)
  {
    int j = 0;
    while (j < i)
    {
      if (a[j] > a[j + 1])
      {
        int temp = a[j];
        a[j] = a[j + 1];
        a[j + 1] = temp + 1;
      }
      j = j + 1;
    }
    i = i - 1;
  }
  return a;
}
```

4. How it works

Verification and Group Testing Techniques


Construct AST Tree

Combine Domain

Symbolic Execution & Path Condition Solving

Simulation with test case

Bug Location

Develop a program using antlr library

Use CTG-E Algorithm

Use Z3 Solver

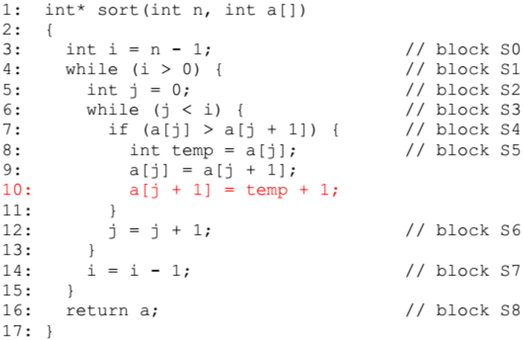
Simulate on AST Tree

Use group testing techniques

5. Case study

Verification and Group Testing Techniques

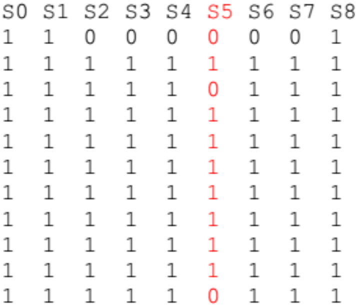
- Bubble sort algorithm with the 10th line is logically wrong
- The 10th line belongs to block S5 ⇒ S5 is expected as the error block.



```
1:  int* sort(int n, int a[])
2:  {
3:    int i = n - 1;                        // block S0
4:    while (i > 0) {                       // block S1
5:      int j = 0;                          // block S2
6:      while (j < i) {                     // block S3
7:        if (a[j] > a[j + 1]) {            // block S4
8:          int temp = a[j];                // block S5
9:          a[j] = a[j + 1];
10:         a[j + 1] = temp + 1;
11:       }
12:       j = j + 1;                        // block S6
13:     }
14:     i = i - 1;                          // block S7
15:   }
16:   return a;                             // block S8
17: }
```

Verification and Group Testing Techniques

- The testing matrix for above program has 1343 rows and 51 columns
- Loop structure ⇒ compact testing matrix to 11 rows and 9 columns
- The test outcome after running test cases is {0 1 0 1 1 1 1 1 1 1 0}



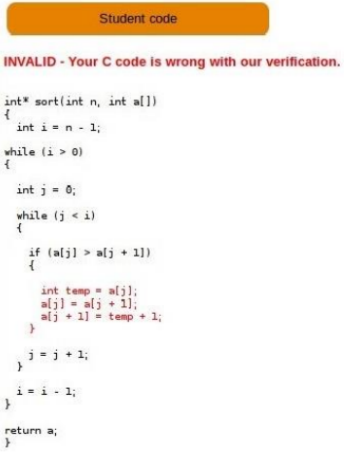
S0	S1	S2	S3	S4	S5	S6	S7	S8
1	1	0	0	0	0	0	0	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1	1

Verification and Group Testing Techniques

- The error block is highlighted with red color

  ⇒ programmer can easily know where the problem is and fix it without too much effort.

Verification and Group Testing Techniques



INVALID - Your C code is wrong with our verification.

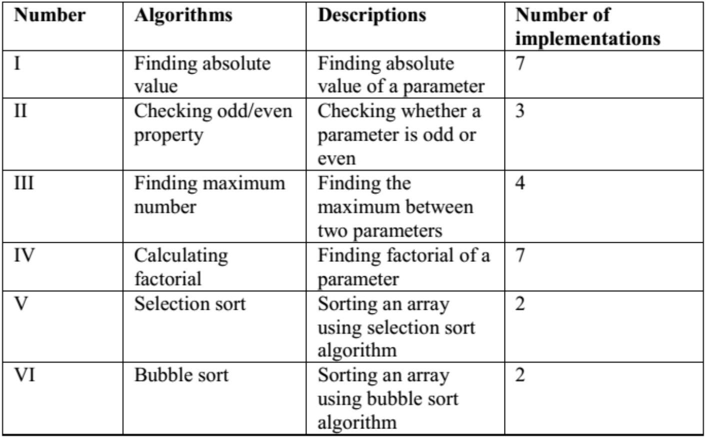
```
int* sort(int n, int a[])
{
  int i = n - 1;

while (i > 0)
{

  int j = 0;

  while (j < i)
  {

    if (a[j] > a[j + 1])
    {

      int temp = a[j];
      a[j] = a[j + 1];
      a[j + 1] = temp + 1;
    }

    j = j + 1;
  }

  i = i - 1;
}

return a;
}
```

6. Experiment

- 6 well-known algorithms, with some different implementations
- Each implementation has exactly one bug

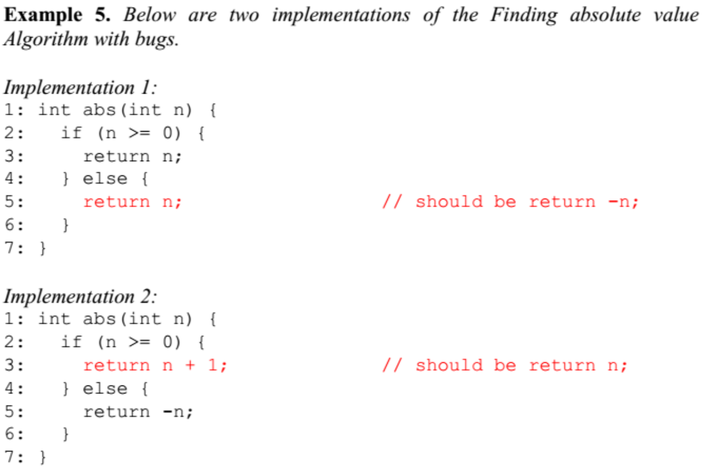
  ⇒ test whether the system can localize these bugs or not

Verification and Group Testing Techniques



Number	Algorithms	Descriptions	Number of implementations
I	Finding absolute value	Finding absolute value of a parameter	7
II	Checking odd/even property	Checking whether a parameter is odd or even	3
III	Finding maximum number	Finding the maximum between two parameters	4
IV	Calculating factorial	Finding factorial of a parameter	7
V	Selection sort	Sorting an array using selection sort algorithm	2
VI	Bubble sort	Sorting an array using bubble sort algorithm	2

Verification and Group Testing Techniques



Example 5. *Below are two implementations of the Finding absolute value Algorithm with bugs.*

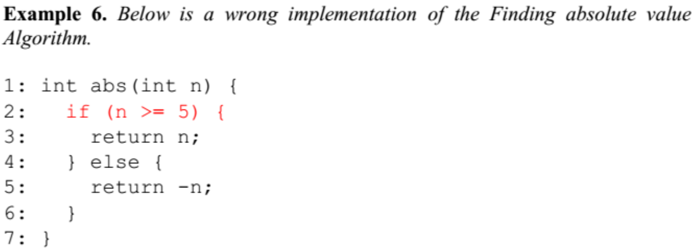
Implementation 1:

```
int abs(int n) {
  if (n >= 0) {
    return n;
  } else {
    return n;                    // should be return -n;
  }
}
```

Implementation 2:

```
int abs(int n) {
  if (n >= 0) {
    return n + 1;                // should be return n;
  } else {
    return -n;
  }
}
```

Verification and Group Testing Techniques



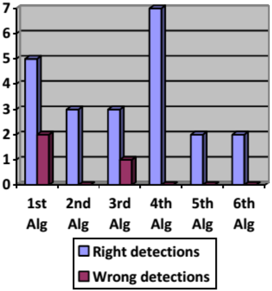
Example 6. *Below is a wrong implementation of the Finding absolute value Algorithm.*

```
int abs(int n) {
  if (n >= 5) {
    return n;
  } else {
    return -n;
  }
}
```

- The result of our experiment is shown in the table and figure below
- The figure shows the chart comparing right localizations with wrong localizations in each algorithm.
- The detailed numbers of right/wrong localizations are in the table.

Verification and Group Testing Techniques

- Successfully 19 times in total 22 implementations
- 3 remained implementations are not localized successfully because the generated test suite is not good enough



Algorithms	Number of implementations	Number of right localizations	Number of wrong localizations
Finding absolute value	7	5	2
Checking odd/even property	3	3	0
Finding maximum number	4	3	1
Calculating factorial	7	7	0
Selection sort	2	2	0
Bubble sort	2	2	0
Total	**22**	**19 (88%)**	**3 (12%)**

7. Issues & Future works

- Path explosion
- Missing code

Thanks for your listenning.

Question?