better scientific software

INTRODUCTION TO SOFTWARE TESTING FOR THE SCIENTIFIC COMMUNITY

Myra Cohen, BSSw Fellow

https://www.cs.iastate.edu/~mcohen

mcohen@iastate.edu

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Laboratory for Variability-Aware Assurance and
Testing of Organic Programs

LaVA-OPs

Module I: Overview

Motivation

What to test

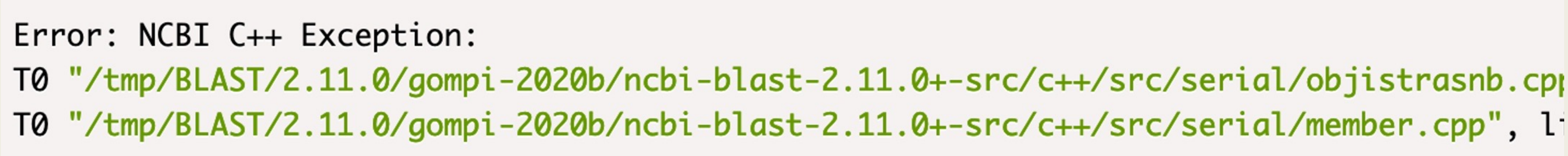
Types of Testing

Models

Coverage

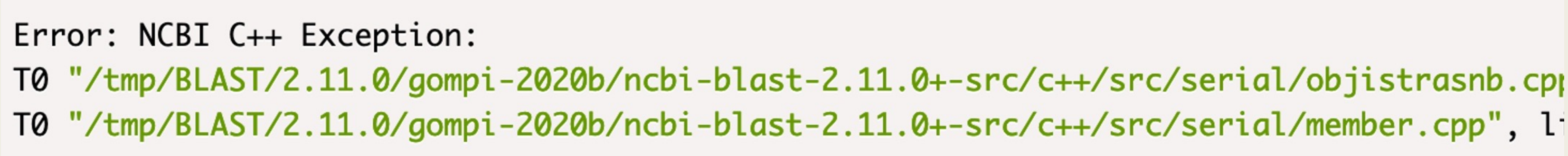
Oracles

Why Test Software?

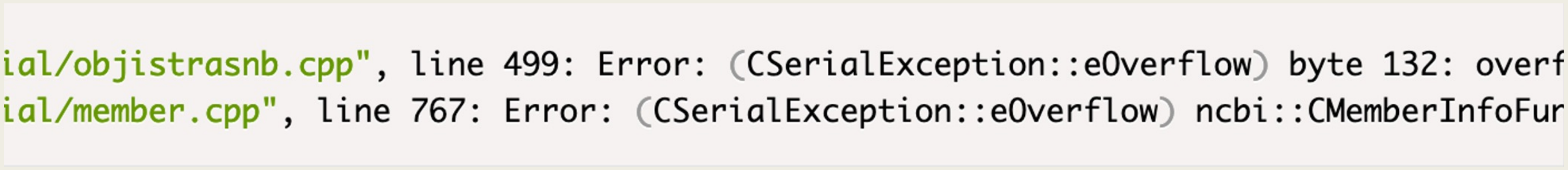


```
Error: NCBI C++ Exception:
T0 "/tmp/BLAST/2.11.0/gompi-2020b/ncbi-blast-2.11.0+-src/c++/src/serial/objistrasnb.cpp
T0 "/tmp/BLAST/2.11.0/gompi-2020b/ncbi-blast-2.11.0+-src/c++/src/serial/member.cpp", l
```

Why Test Software?

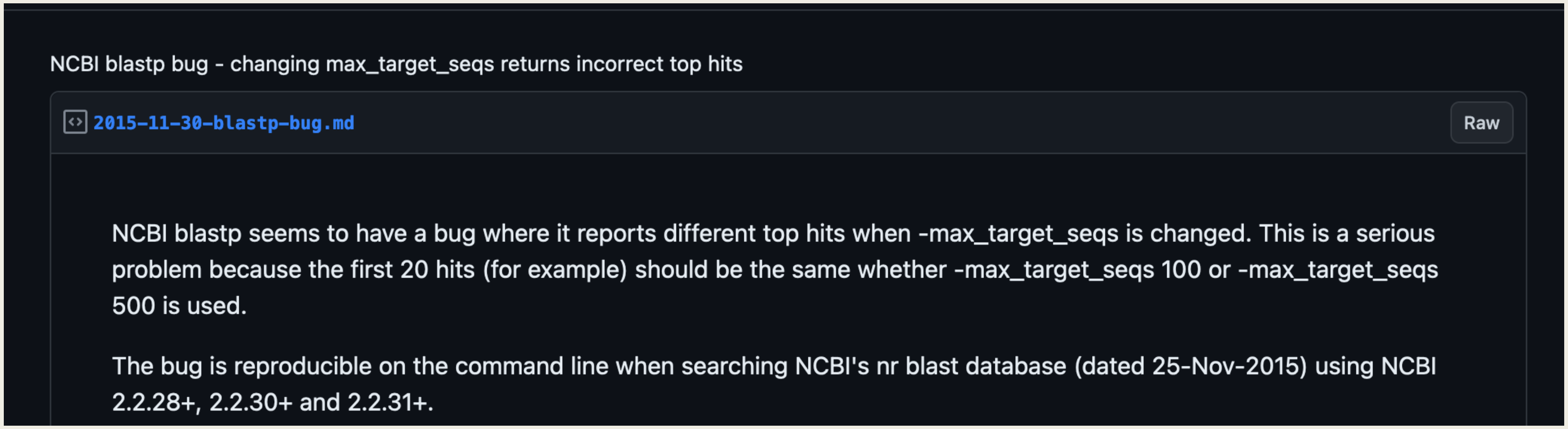


```
Error: NCBI C++ Exception:
T0 "/tmp/BLAST/2.11.0/gompi-2020b/ncbi-blast-2.11.0+-src/c++/src/serial/objistrasnb.cp
T0 "/tmp/BLAST/2.11.0/gompi-2020b/ncbi-blast-2.11.0+-src/c++/src/serial/member.cpp", l
```



```
ial/objistrasnb.cpp", line 499: Error: (CSerialException::eOverflow) byte 132: overf
ial/member.cpp", line 767: Error: (CSerialException::eOverflow) ncbi::CMemberInfoFu
```

Why Test Software?



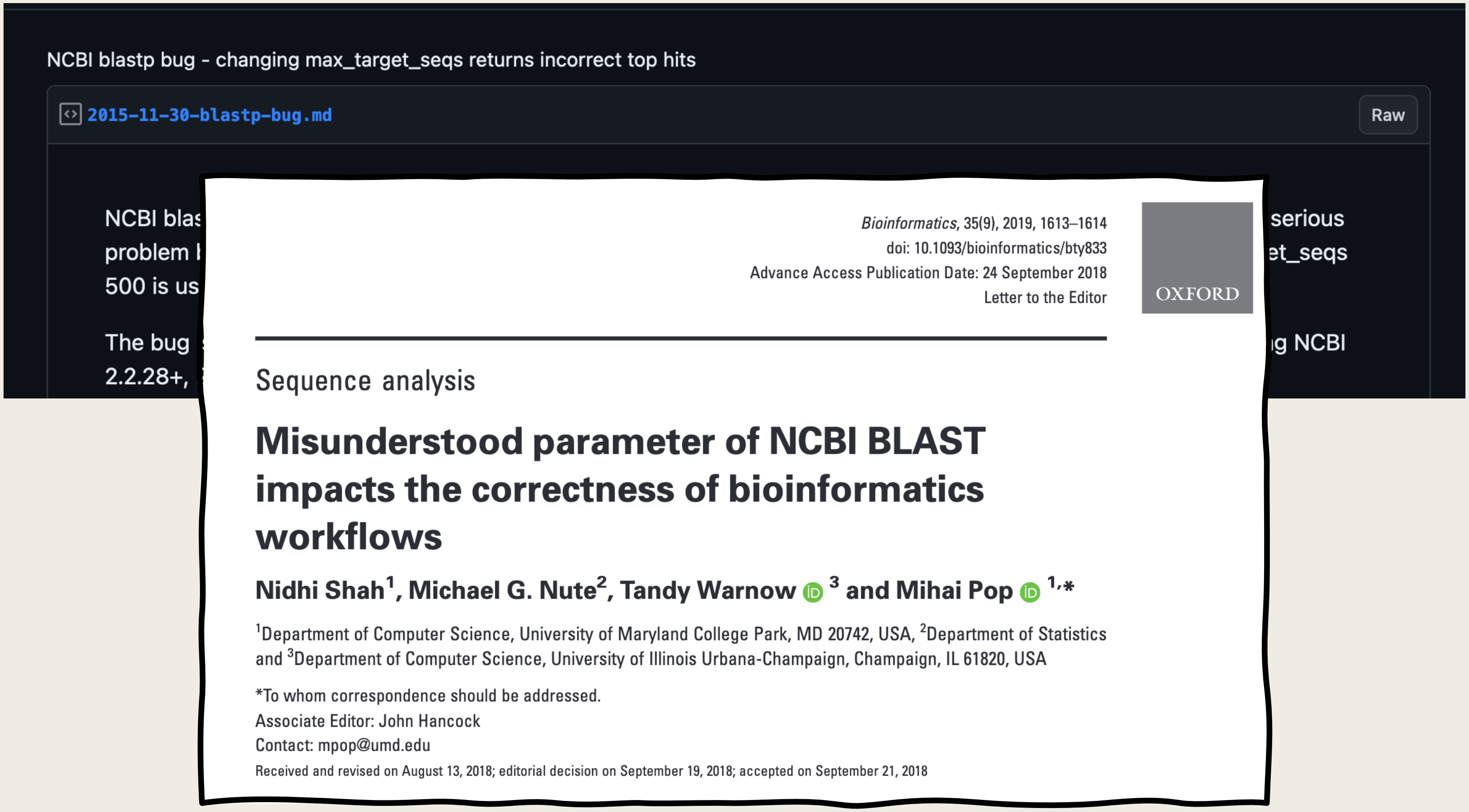
NCBI blastp bug - changing max_target_seqs returns incorrect top hits

2015-11-30-blastp-bug.md

NCBI blastp seems to have a bug where it reports different top hits when -max_target_seqs is changed. This is a serious problem because the first 20 hits (for example) should be the same whether -max_target_seqs 100 or -max_target_seqs 500 is used.

The bug is reproducible on the command line when searching NCBI's nr blast database (dated 25-Nov-2015) using NCBI 2.2.28+, 2.2.30+ and 2.2.31+.

Why Test Software?



NCBI blastp bug - changing max_target_seqs returns incorrect top hits

2015-11-30-blastp-bug.md

NCBI blas serious
problem i et_seqs
500 is us

The bug g NCBI
2.2.28+,

Bioinformatics, 35(9), 2019, 1613–1614
doi: 10.1093/bioinformatics/bty833
Advance Access Publication Date: 24 September 2018
Letter to the Editor

Sequence analysis

Misunderstood parameter of NCBI BLAST impacts the correctness of bioinformatics workflows

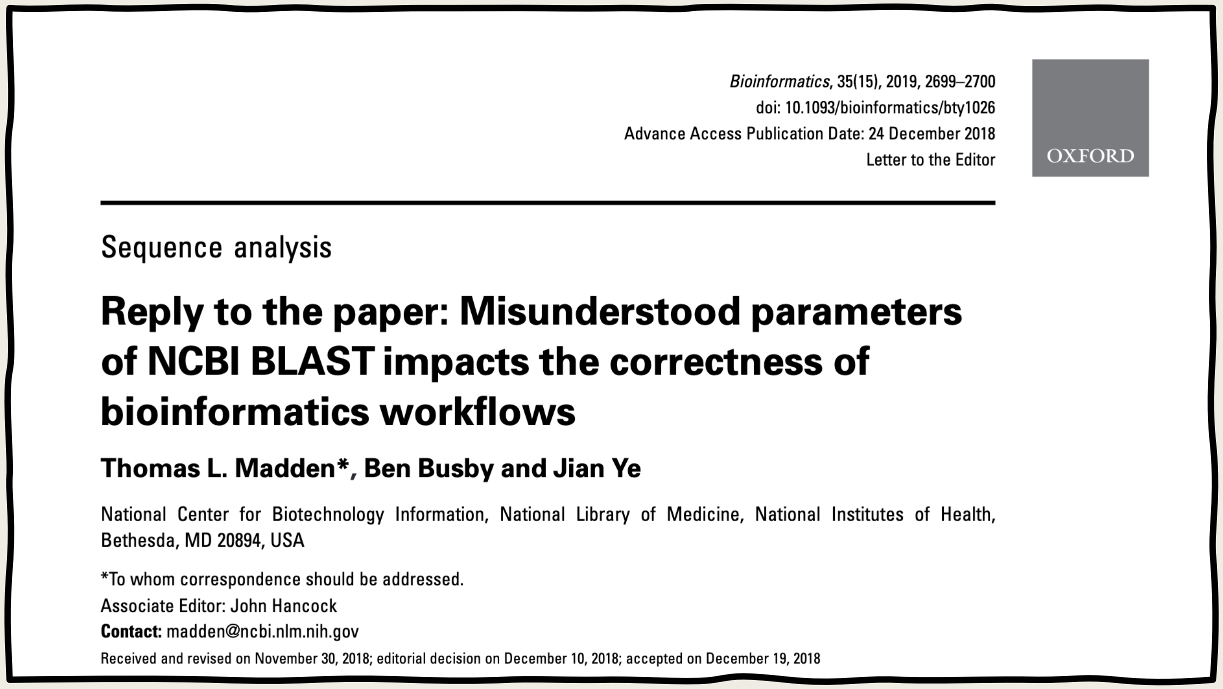
Nidhi Shah[1], Michael G. Nute[2], Tandy Warnow [3] and Mihai Pop [1,]*

[1]Department of Computer Science, University of Maryland College Park, MD 20742, USA, [2]Department of Statistics and [3]Department of Computer Science, University of Illinois Urbana-Champaign, Champaign, IL 61820, USA

*To whom correspondence should be addressed.
Associate Editor: John Hancock
Contact: mpop@umd.edu

Received and revised on August 13, 2018; editorial decision on September 19, 2018; accepted on September 21, 2018

Why Test Software?



Bioinformatics, 35(15), 2019, 2699–2700
doi: 10.1093/bioinformatics/bty1026
Advance Access Publication Date: 24 December 2018
Letter to the Editor

OXFORD

Sequence analysis

Reply to the paper: Misunderstood parameters of NCBI BLAST impacts the correctness of bioinformatics workflows

Thomas L. Madden*, Ben Busby and Jian Ye

National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD 20894, USA

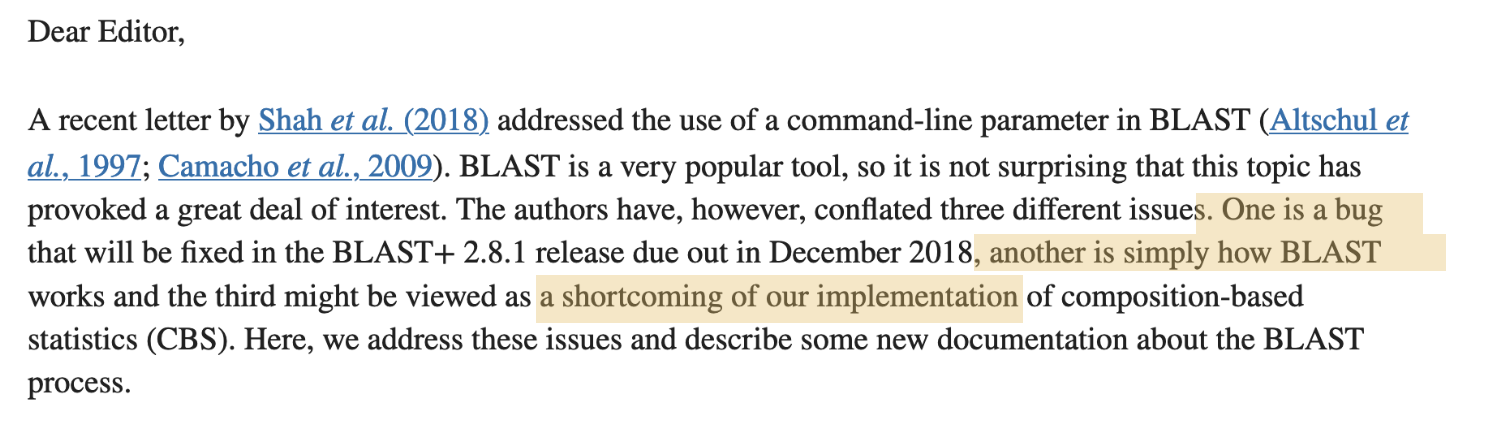
*To whom correspondence should be addressed.

Associate Editor: John Hancock

Contact: madden@ncbi.nlm.nih.gov

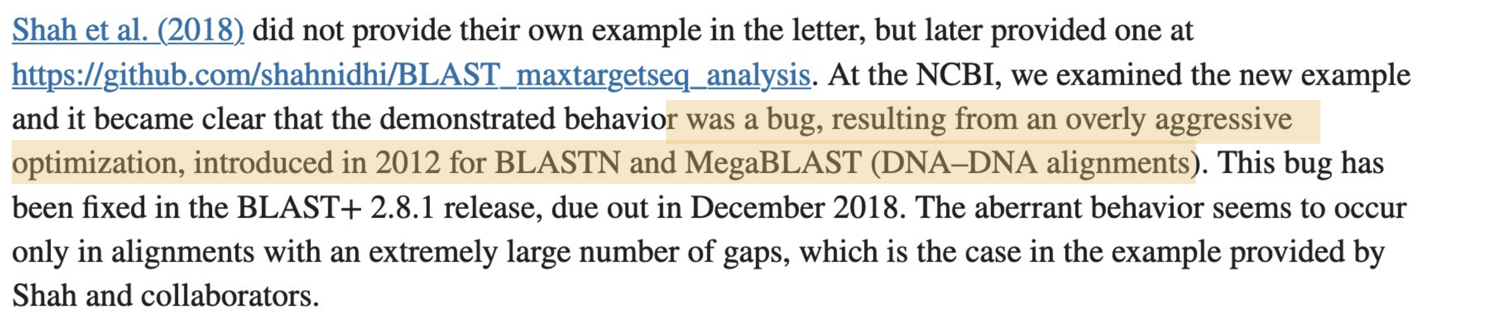
Received and revised on November 30, 2018; editorial decision on December 10, 2018; accepted on December 19, 2018

Why Test Software?



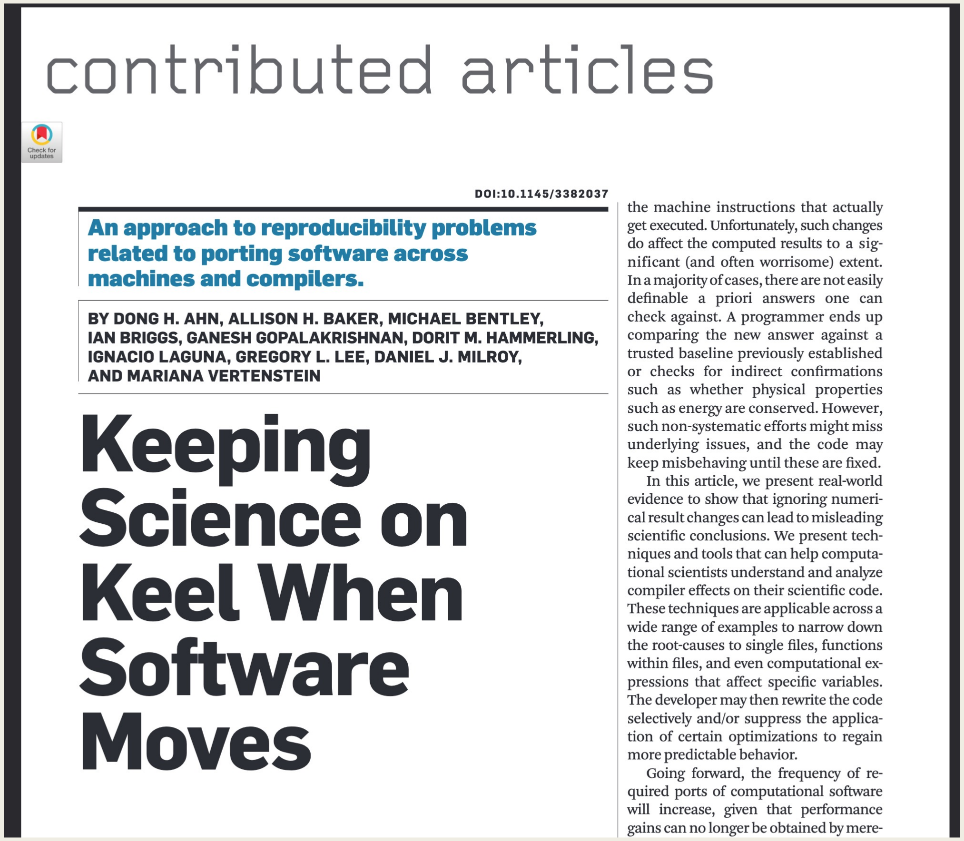
Dear Editor,

A recent letter by Shah *et al.* (2018) addressed the use of a command-line parameter in BLAST (Altschul *et al.*, 1997; Camacho *et al.*, 2009). BLAST is a very popular tool, so it is not surprising that this topic has provoked a great deal of interest. The authors have, however, conflated three different issues. One is a bug that will be fixed in the BLAST+ 2.8.1 release due out in December 2018, another is simply how BLAST works and the third might be viewed as a shortcoming of our implementation of composition-based statistics (CBS). Here, we address these issues and describe some new documentation about the BLAST process.



Shah et al. (2018) did not provide their own example in the letter, but later provided one at https://github.com/shahnidhi/BLAST_maxtargetseq_analysis. At the NCBI, we examined the new example and it became clear that the demonstrated behavior was a bug, resulting from an overly aggressive optimization, introduced in 2012 for BLASTN and MegaBLAST (DNA–DNA alignments). This bug has been fixed in the BLAST+ 2.8.1 release, due out in December 2018. The aberrant behavior seems to occur only in alignments with an extremely large number of gaps, which is the case in the example provided by Shah and collaborators.

Why Test Software?



contributed articles

Check for updates

DOI:10.1145/3382037

An approach to reproducibility problems related to porting software across machines and compilers.

BY DONG H. AHN, ALLISON H. BAKER, MICHAEL BENTLEY, IAN BRIGGS, GANESH GOPALAKRISHNAN, DORIT M. HAMMERLING, IGNACIO LAGUNA, GREGORY L. LEE, DANIEL J. MILROY, AND MARIANA VERTENSTEIN

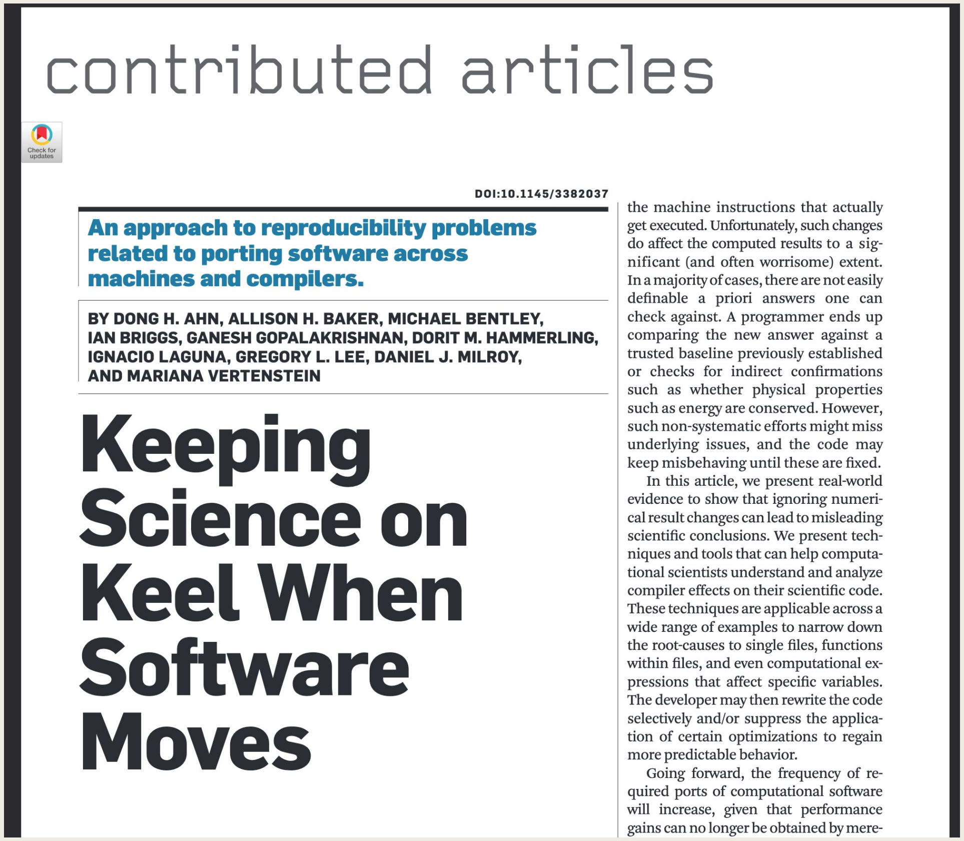
Keeping Science on Keel When Software Moves

the machine instructions that actually get executed. Unfortunately, such changes do affect the computed results to a significant (and often worrisome) extent. In a majority of cases, there are not easily definable a priori answers one can check against. A programmer ends up comparing the new answer against a trusted baseline previously established or checks for indirect confirmations such as whether physical properties such as energy are conserved. However, such non-systematic efforts might miss underlying issues, and the code may keep misbehaving until these are fixed.

In this article, we present real-world evidence to show that ignoring numerical result changes can lead to misleading scientific conclusions. We present techniques and tools that can help computational scientists understand and analyze compiler effects on their scientific code. These techniques are applicable across a wide range of examples to narrow down the root-causes to single files, functions within files, and even computational expressions that affect specific variables. The developer may then rewrite the code selectively and/or suppress the application of certain optimizations to regain more predictable behavior.

Going forward, the frequency of required ports of computational software will increase, given that performance gains can no longer be obtained by mere-

Why Test Software?



contributed articles

DOI:10.1145/3382037

An approach to reproducibility problems related to porting software across machines and compilers.

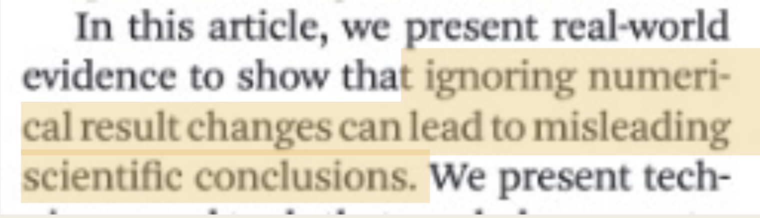
BY DONG H. AHN, ALLISON H. BAKER, MICHAEL BENTLEY, IAN BRIGGS, GANESH GOPALAKRISHNAN, DORIT M. HAMMERLING, IGNACIO LAGUNA, GREGORY L. LEE, DANIEL J. MILROY, AND MARIANA VERTENSTEIN

Keeping Science on Keel When Software Moves

the machine instructions that actually get executed. Unfortunately, such changes do affect the computed results to a significant (and often worrisome) extent. In a majority of cases, there are not easily definable a priori answers one can check against. A programmer ends up comparing the new answer against a trusted baseline previously established or checks for indirect confirmations such as whether physical properties such as energy are conserved. However, such non-systematic efforts might miss underlying issues, and the code may keep misbehaving until these are fixed.

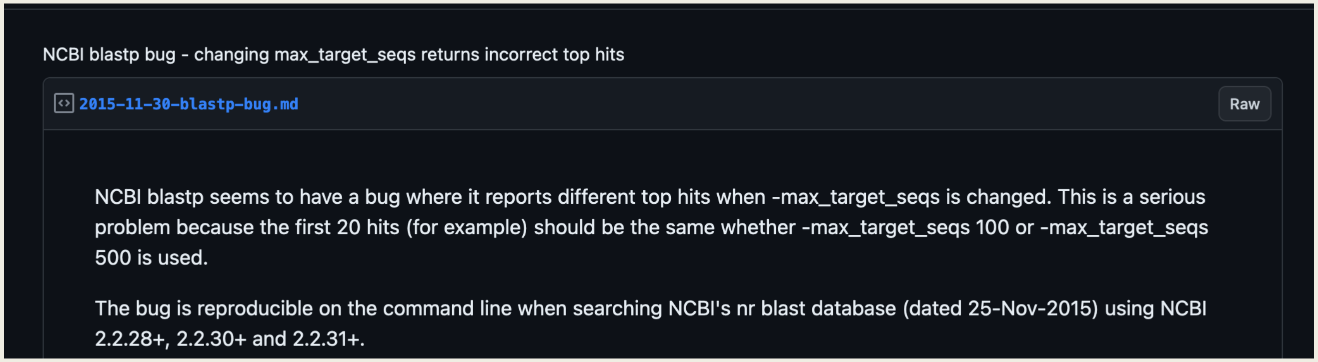
In this article, we present real-world evidence to show that ignoring numerical result changes can lead to misleading scientific conclusions. We present techniques and tools that can help computational scientists understand and analyze compiler effects on their scientific code. These techniques are applicable across a wide range of examples to narrow down the root-causes to single files, functions within files, and even computational expressions that affect specific variables. The developer may then rewrite the code selectively and/or suppress the application of certain optimizations to regain more predictable behavior.

Going forward, the frequency of required ports of computational software will increase, given that performance gains can no longer be obtained by mere-



In this article, we present real-world evidence to show that ignoring numerical result changes can lead to misleading scientific conclusions. We present tech-

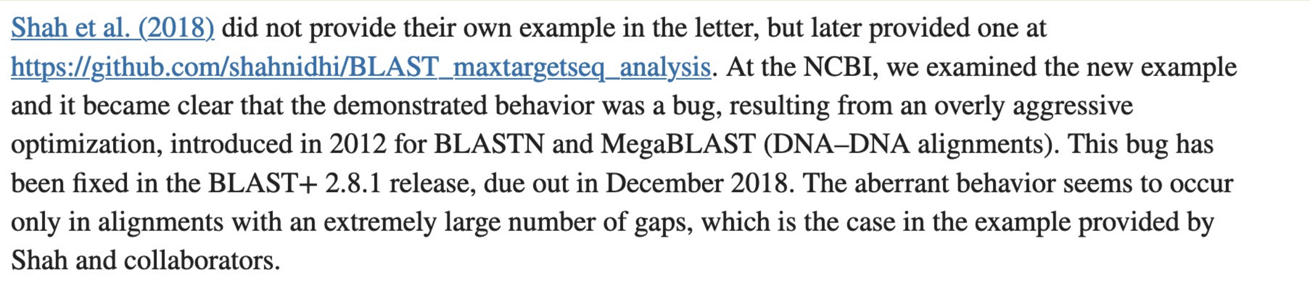
Why Test Software?



NCBI blastp bug - changing max_target_seqs returns incorrect top hits

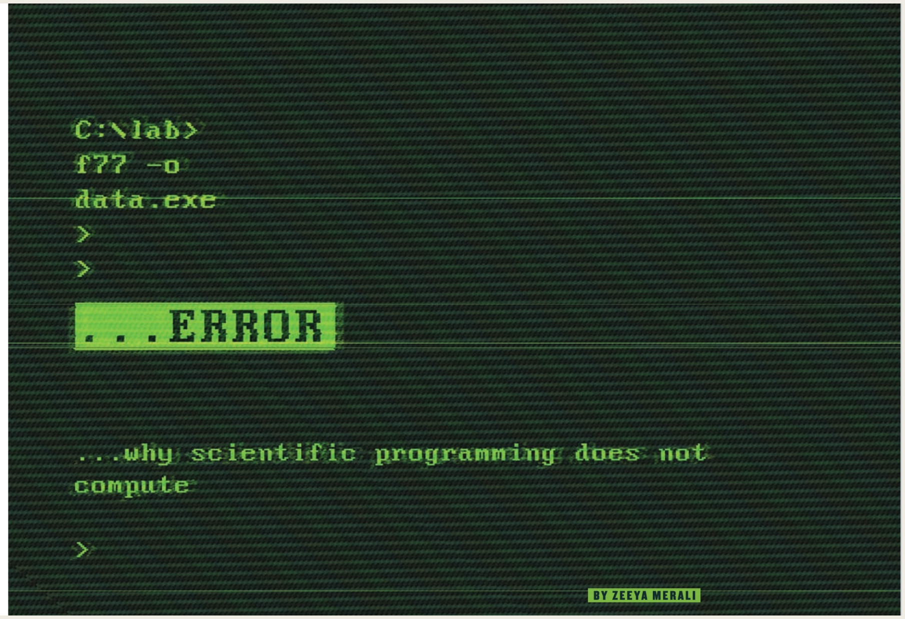
NCBI blastp seems to have a bug where it reports different top hits when -max_target_seqs is changed. This is a serious problem because the first 20 hits (for example) should be the same whether -max_target_seqs 100 or -max_target_seqs 500 is used.

The bug is reproducible on the command line when searching NCBI's nr blast database (dated 25-Nov-2015) using NCBI 2.2.28+, 2.2.30+ and 2.2.31+.



Shah et al. (2018) did not provide their own example in the letter, but later provided one at https://github.com/shahnidhi/BLAST_maxtargetseq_analysis. At the NCBI, we examined the new example and it became clear that the demonstrated behavior was a bug, resulting from an overly aggressive optimization, introduced in 2012 for BLASTN and MegaBLAST (DNA–DNA alignments). This bug has been fixed in the BLAST+ 2.8.1 release, due out in December 2018. The aberrant behavior seems to occur only in alignments with an extremely large number of gaps, which is the case in the example provided by Shah and collaborators.

Why Test Software?



```
C:\lab>
f77 -o
data.exe
>
>
...ERROR


...why scientific programming does not
compute

>
```

BY ZEEYA MERALI

Merali, Zeeya. “Computational Science: ...Error.” *Nature* 467, no. 7317 (Oct, 2010): 775–77

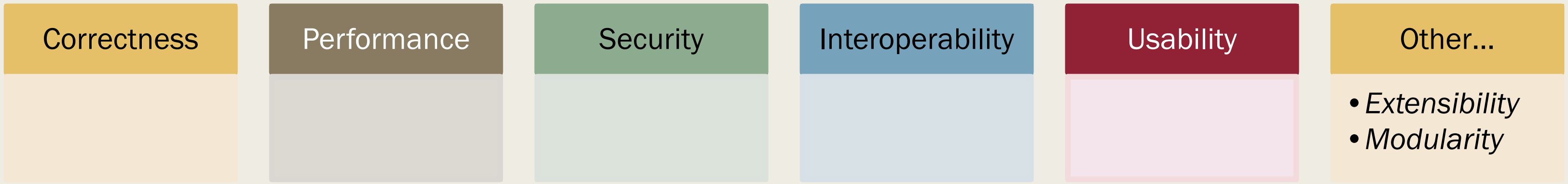
What Should We Test?

- Correctness
- Performance
- Security
- Interoperability
- Usability
- Other...

What Should We Test?

Correctness	Performance	Security	Interoperability	Usability	Other…
					• *Extensibility* • *Modularity*

What Should We Test?



Correctness

Performance

Security

Interoperability

Usability

Other…

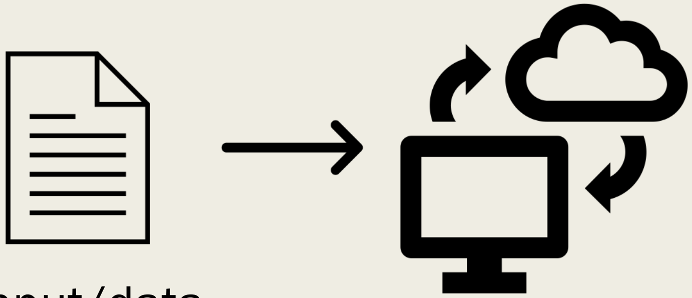
- *Extensibility*
- *Modularity*

Our focus will be on correctness and interoperability

What is Testing

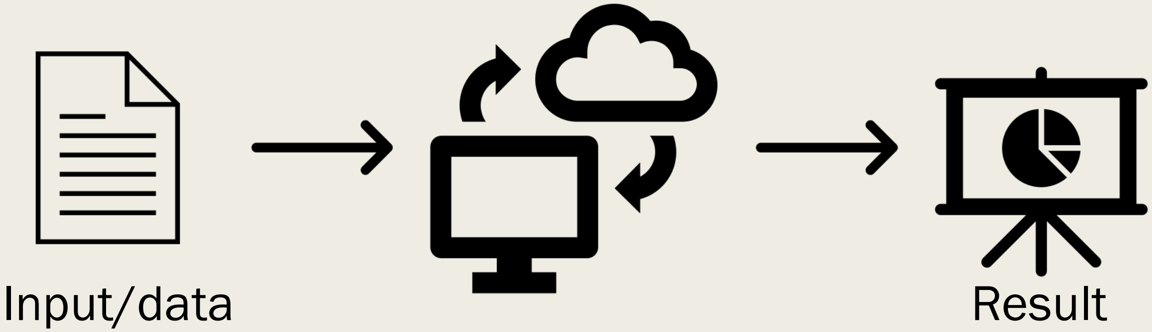
Input/data

What is Testing



Input/data

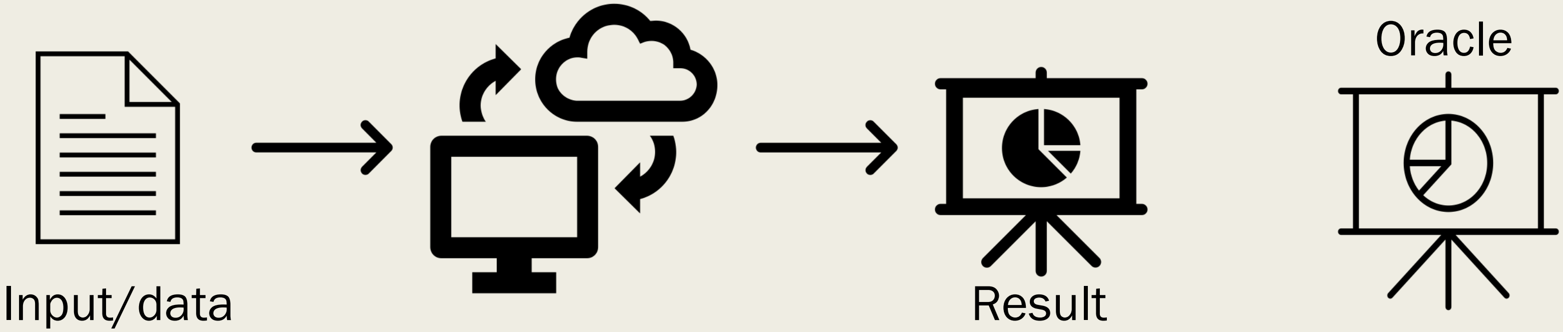
What is Testing



Input/data

Result

What is Testing

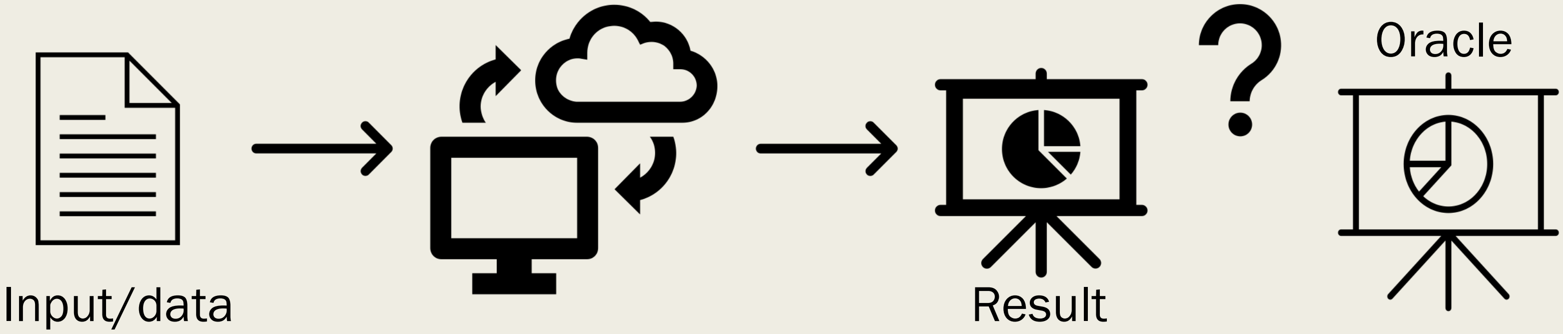


Input/data

Result

Oracle

What is Testing



Limitations

Testing can only show the presence of faults.
It cannot determine their absence.

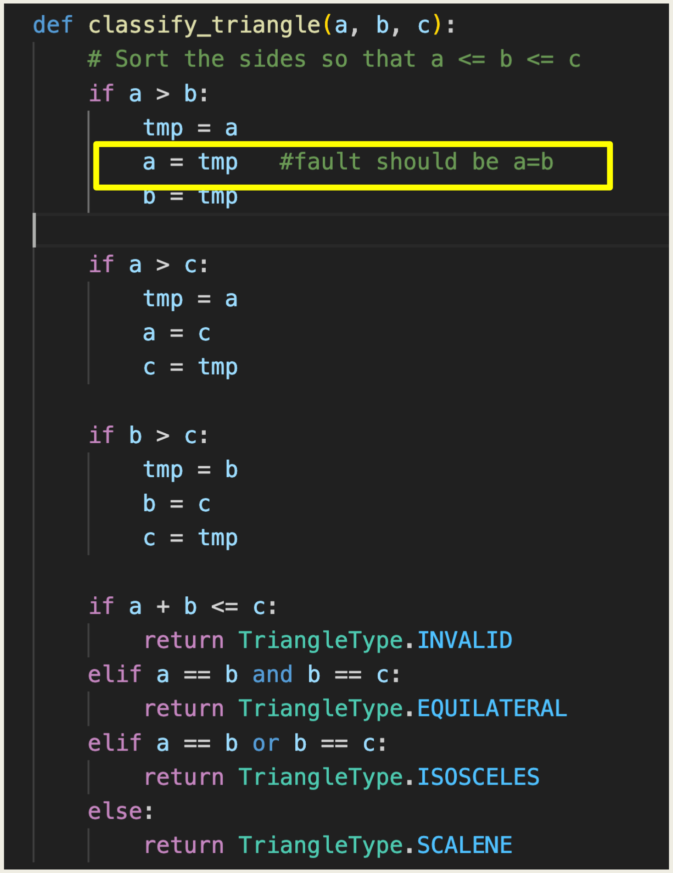
Edsger W. Dijkstra

Challenge

- To detect a program FAILURE we need to:
  - Reach a FAULT in the code
  - Infect the code (change to incorrect state) - ERROR
  - Propagate the error out of program
  - Reveal (detect) the error – (ORACLE)

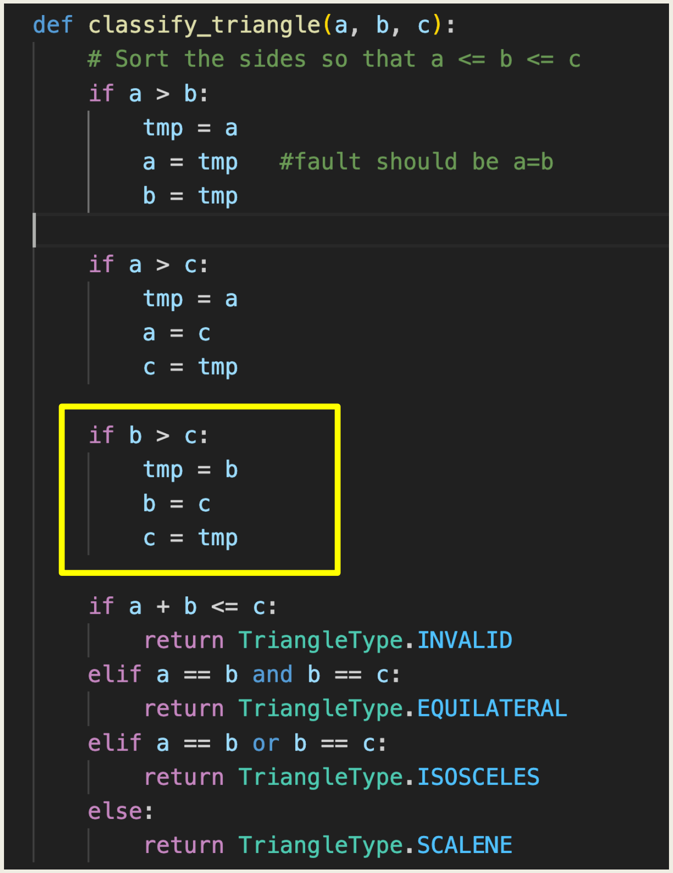
RIPR model *Ammann, Offutt (Introduction to Software Testing, 2016)*

Tests Can Miss Faults



```python
def classify_triangle(a, b, c):
    # Sort the sides so that a <= b <= c
    if a > b:
        tmp = a
        a = tmp    #fault should be a=b
        b = tmp

    if a > c:
        tmp = a
        a = c
        c = tmp

    if b > c:
        tmp = b
        b = c
        c = tmp

    if a + b <= c:
        return TriangleType.INVALID
    elif a == b and b == c:
        return TriangleType.EQUILATERAL
    elif a == b or b == c:
        return TriangleType.ISOSCELES
    else:
        return TriangleType.SCALENE
```

Tests Can Miss Faults

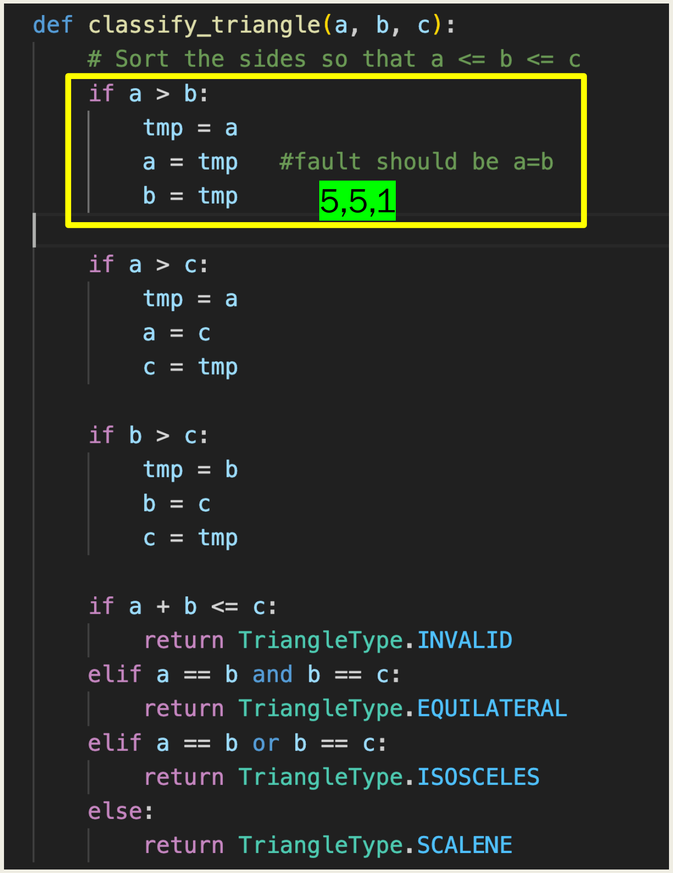


```python
def classify_triangle(a, b, c):
    # Sort the sides so that a <= b <= c
    if a > b:
        tmp = a
        a = tmp    #fault should be a=b
        b = tmp

    if a > c:
        tmp = a
        a = c
        c = tmp

    if b > c:
        tmp = b
        b = c
        c = tmp

    if a + b <= c:
        return TriangleType.INVALID
    elif a == b and b == c:
        return TriangleType.EQUILATERAL
    elif a == b or b == c:
        return TriangleType.ISOSCELES
    else:
        return TriangleType.SCALENE
```

1. Test Case 3, 4, 5 (scalene)
    - doesn't reach fault

✕

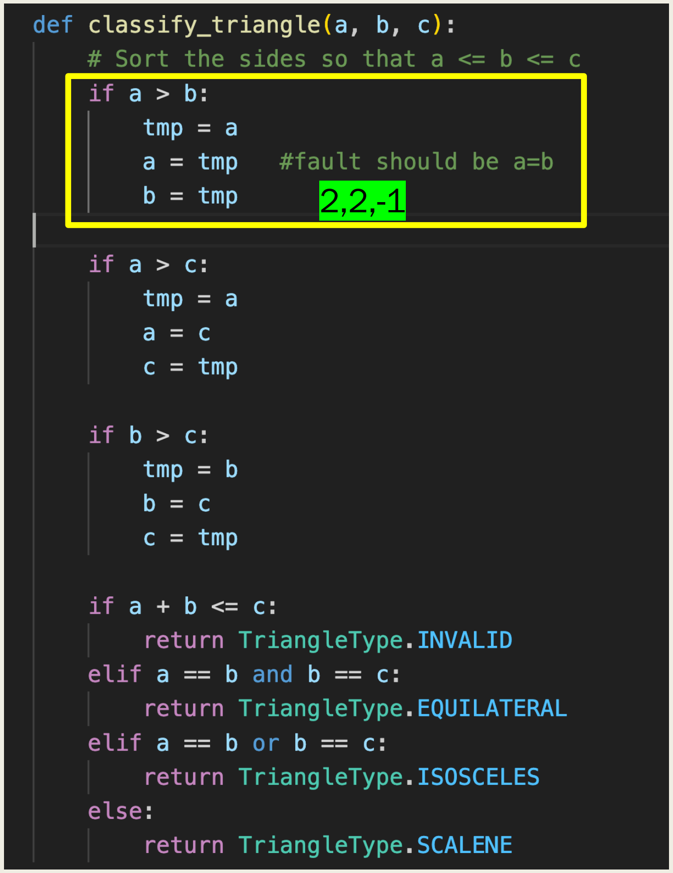
Tests Can Miss Faults



```
def classify_triangle(a, b, c):
    # Sort the sides so that a <= b <= c
    if a > b:
        tmp = a
        a = tmp    #fault should be a=b
        b = tmp        5,5,1

    if a > c:
        tmp = a
        a = c
        c = tmp

    if b > c:
        tmp = b
        b = c
        c = tmp

    if a + b <= c:
        return TriangleType.INVALID
    elif a == b and b == c:
        return TriangleType.EQUILATERAL
    elif a == b or b == c:
        return TriangleType.ISOSCELES
    else:
        return TriangleType.SCALENE
```

1. Test Case 3, 4, 5 (scalene) ✗
   - doesn’t reach fault
2. Test Case 5, 1, 1 (invalid) ✓
   - reaches fault and infects
   - reveals (returns isosceles)

Tests Can Miss Faults



```
def classify_triangle(a, b, c):
    # Sort the sides so that a <= b <= c
    if a > b:
        tmp = a
        a = tmp    #fault should be a=b
        b = tmp

    if a > c:
        tmp = a
        a = c
        c = tmp

    if b > c:
        tmp = b
        b = c
        c = tmp

    if a + b <= c:
        return TriangleType.INVALID
    elif a == b and b == c:
        return TriangleType.EQUILATERAL
    elif a == b or b == c:
        return TriangleType.ISOSCELES
    else:
        return TriangleType.SCALENE
```

1. Test Case 3, 4, 5 (scalene) ✗
   - doesn't reach fault
2. Test Case 5, 1, 1 (invalid) ✓
   - reaches fault and infects
   - reveals (returns isosceles)
3. Test Case 2, 1, -1 (invalid) ✗
   - reaches fault and infects
   - Doesn't propagate (2, 2,-1) is still INVALID

Software vs. Data

- Scientific software is often data (or model driven)
- Both software and data can lead to faults
- We start by focusing on software

But I Have Unit Tests…

- These are an essential part of testing
- Focus is on individual modules
- Can be re-used each time system changes (regression testing)
- Can be packaged with software when released
- Other testing focuses on the system specifications and overall program behavior

Module I: Overview

Motivation

What to test

Types of Testing

Models

Coverage

Oracles

This work was supported by the Better Scientific Software Fellowship Program, funded by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy (DOE) Office of Science and the National Nuclear Security Administration; and by the National Science Foundation (NSF) under Grant No. 2154495.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DOE or NSF.