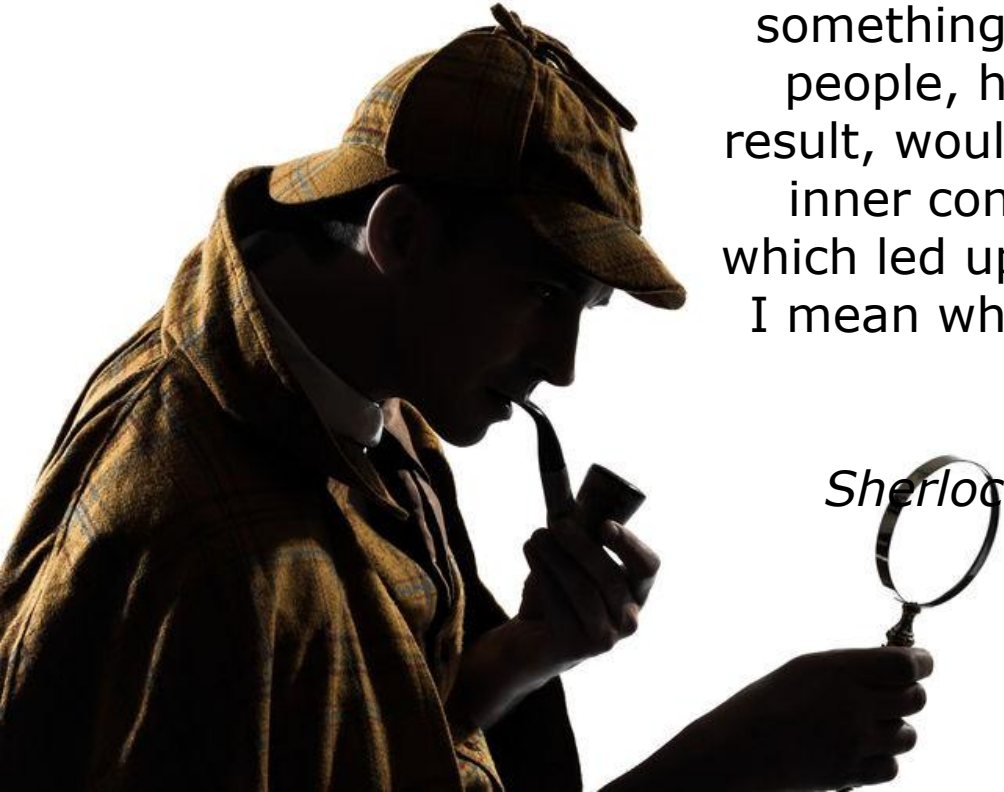
Debugging

Testing, debugging & verification

Atze van der Ploeg



"Most people, if you describe a train of events to them, will tell you what the result would be. They can put those events together in their minds, and argue from them that something will come to pass. There are few people, however, who, if you told them a result, would be able to evolve from their own inner consciousness what the steps were which led up to that result. This power is what I mean when I talk of reasoning backwards, or analytically."

Sherlock Holmes in A Study in Scarlet
by A. C. Doyle

Today

- Debugging steps and example
- Automatic problem minimization (shrinking)
- Backwards dependencies

A bug has been found, whodunnit?

1. Verify the Bug and Determine Correct Behavior
2. Isolate and Minimize (shrink)
3. Eyeball the code, where could it be?
4. Devise and run an experiment to test your hypothesis
5. Repeat 3,4 until you understand what's wrong
6. Fix the Bug and Verify the Fix
7. Create a Regression Test

Demo!

Test your hypothesis - observe outcome!

- Simple logging : print statements
- Advanced logging: configureable what is printed based on level (OFF < FINE . . . < INFO < WARNING < SEVERE)
- Debugging tools such as the eclipse debugger

“Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.”

- Brian Kernighan

Automatic problem minimization (shrinking)

paper

This input made mozilla crash in 2002, what was the problem?

```
...
<td align=left valign=top>
<SELECT NAME="op sys" MULTIPLE SIZE=7>
<OPTION VALUE="All">All<OPTION VALUE="Windows 3.1">Windows 3.1<OPTION VALUE="Windows 95">Windows 95<OPTION VALUE="Windows
98">Windows 98<OPTION VALUE="Windows ME">Windows ME<OPTION VALUE="Windows 2000">Windows 2000<OPTION VALUE="Windows
NT">Windows NT<OPTION VALUE="Mac System 7">Mac System 7<OPTION VALUE="Mac System 7.5">Mac System 7.5<OPTION VALUE="Mac
System 7.6.1">Mac System 7.6.1<OPTION VALUE="Mac System 8.0">Mac System 8.0<OPTION VALUE="Mac System 8.5">Mac System
8.5<OPTION VALUE="Mac System 8.6">Mac System 8.6<OPTION VALUE="Mac System 9.x">Mac System 9.x<OPTION VALUE="MacOS X">MacOS
X<OPTION VALUE="Linux">Linux<OPTION VALUE="BSDI">BSDI<OPTION VALUE="FreeBSD">FreeBSD<OPTION VALUE="NetBSD">NetBSD<OPTION
VALUE="OpenBSD">OpenBSD<OPTION VALUE="AIX">AIX<OPTION VALUE="BeOS">BeOS<OPTION VALUE="HP-UX">HP-UX<OPTION
VALUE="IRIX">IRIX<OPTION VALUE="Neutrino">Neutrino<OPTION VALUE="OpenVMS">OpenVMS<OPTION VALUE="OS/2">OS/2<OPTION
VALUE="OSF/1">OSF/1<OPTION VALUE="Solaris">Solaris<OPTION VALUE="SunOS">SunOS<OPTION VALUE="other">other</SELECT>
</td>
<td align=left valign=top>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<OPTION VALUE="--">--<OPTION VALUE="P1">P1<OPTION VALUE="P2">P2<OPTION VALUE="P3">P3<OPTION VALUE="P4">P4<OPTION
VALUE="P5">P5</SELECT>
</td>
<td align=left valign=top>
<SELECT NAME="bug severity" MULTIPLE SIZE=7>
<OPTION VALUE="blocker">blocker<OPTION VALUE="critical">critical<OPTION VALUE="major">major<OPTION
VALUE="normal">normal<OPTION VALUE="minor">minor<OPTION VALUE="trivial">trivial<OPTION VALUE="enhancement">enhancement</SELECT>
</tr>
</table>
...
```

<SELECT>

Shrinking revisited

Automatically go from failing test to minimal failing test

- *Shrink* the failing input somehow
- See if the test still fails
- If so shrink more, otherwise backtrack

Many possible ways of shrinking!
For example: try all sub-lists

Last week: greedy binary search. Results often good, but not garanteed to be *minimal*

What is minimal?

Global minimum: Smallest subsequence of input such that the test still fails

DDMin

1-minimal: Removing any single character from the input makes the test succeed (every character matters)

Subsequence of a string: a string that can be obtained by deleting elements of the input string

Example: “abd” is a subsequence of “xabcdef” (but not a substring)

Why 1-minimal instead of global minimum?

Global minimum is typically more expensive to compute (requires backtracking)

Example: Global minimum vs 1-minimal

Mozilla crashes on this input:

```
<html> <table> jada <select> ... </table> <chs> ... <br> <p> ...
</html>
```

Reason: crashes on <select> or <chs> (but we do not know that yet)!

1-Minimal

Global minimum

```
public static int checkSum(int[] a)
```

- is supposed to compute the checksum of an integer array
- gives wrong result, whenever a contains two identical consecutive numbers *(but we don't know that yet!)*
- we have a failed test case, e.g., from protocol transmission: {1,3,5,3,9,17,44,3,6,1,1,0,44,1,44,0}
- we have a method

```
boolean isCorrectCheckSum(int [] a, int chksum)
```

Want to get: {1,1},{3,3} or {44,44}

DDMin

```
interface Test{ boolean test(String s); }

String ddMin(Test t, String input)
```

Examples:
ddMin(hasInCorrectCheckSum, “1,3,5,3,9,17,44,3,6,1,1,0,44,1,44,0”) == “44,44”
ddMin(has2Xs, “BBBXCCDDDXD” == “XX”
ddMin(makeMozillaCrash, “<html> ... </html>”) == “<select>”

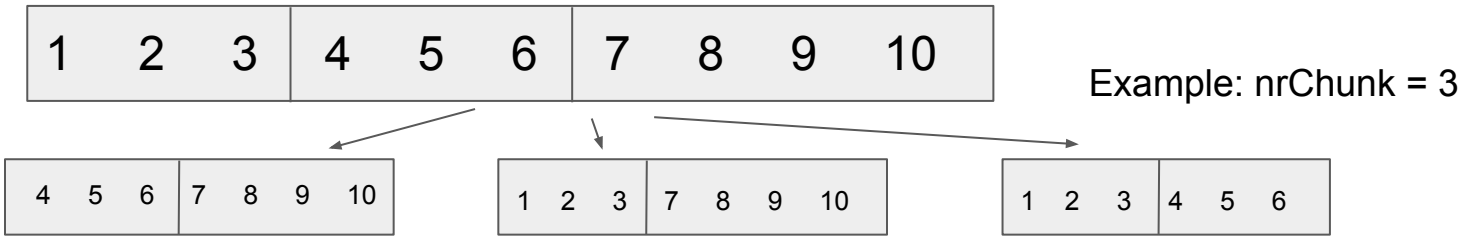
Requires: t.test(input) == false and input is non-null
Ensures: A *1-minimal subsequence* res of input with respect to t.test(res)

DDMin - overview

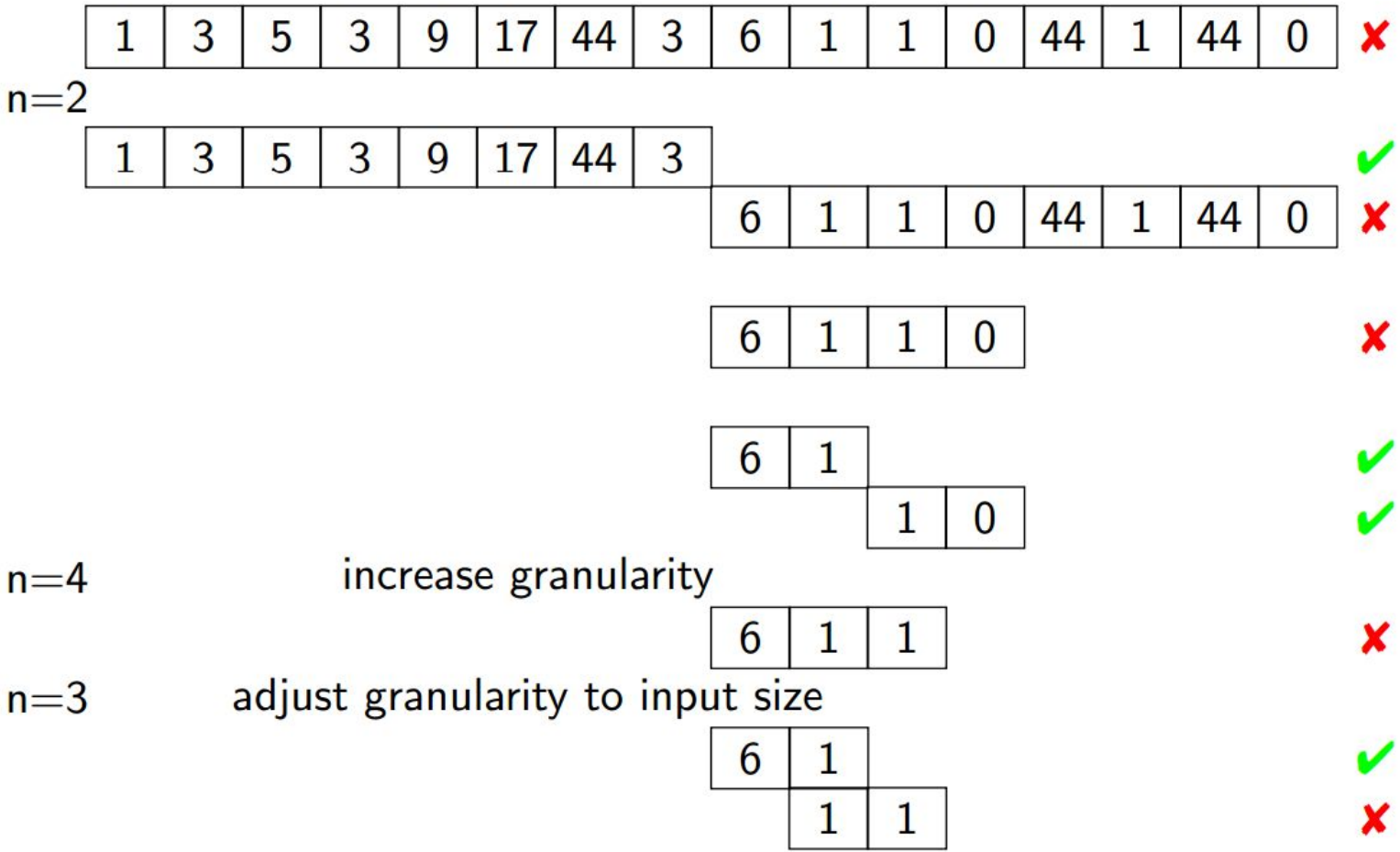
```
String ddMin(Test p, String inp) { return ddMin(p,inp,min(inp.lenght,2)); }

String ddMin(Test p, String inp, int nrChunks)
```

- Devide input into chunks
- Cut away a part, does the test still fail? If so, continue without that part
- Increase granularity (* 2) (number of chunks) when no failure occurs when we cut away any part
- Done when cutting away doesn't help anymore and nrChunks = length of input



Example: nrChunk = 3


1 3 5 3 9 17 44 3 6 1 1 0 44 1 44 0
n=2
1 3 5 3 9 17 44 3
6 1 1 0 44 1 44 0
6 1 1 0
6 1
1 0
n=4
increase granularity
6 1 1
n=3
adjust granularity to input size
6 1
1 1

DDMin - code

```
// Requires: t.test(input) == false and
//           input is non-null and
//           nrChunks >= 0 and nrChunks <= input.length
// Ensures: A subsequence res of input, such
//          that t.test(res) == false and res is 1-minimal
private static String ddMin(Test t, String input, int nrChucks){
    for(int i = 0 ; i < nrChucks ; i++){
        String withoutChunk = removeChuck(input, nrChucks, i);
        if(!t.test(withoutChunk)){
            return ddMin(t,withoutChunk,max(nrChucks-1,2));
        }
    }
    if(nrChucks == input.length()) return input;
    return ddMin(t,input, min(2*nrChucks, input.length()));
}

public static String ddMin(Test t, String input){
    return ddMin(t,input,min(input.length(),2));
}
```