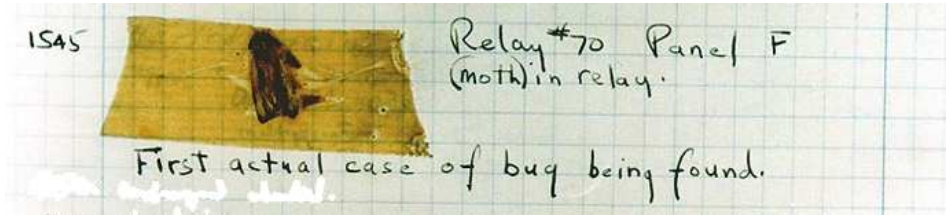
Why Computers Go Wrong (And How To Prevent This)

Joe Hurd
joe.hurd@magd.ox.ac.uk

Magdalen College
Oxford University

The First Actual Bug



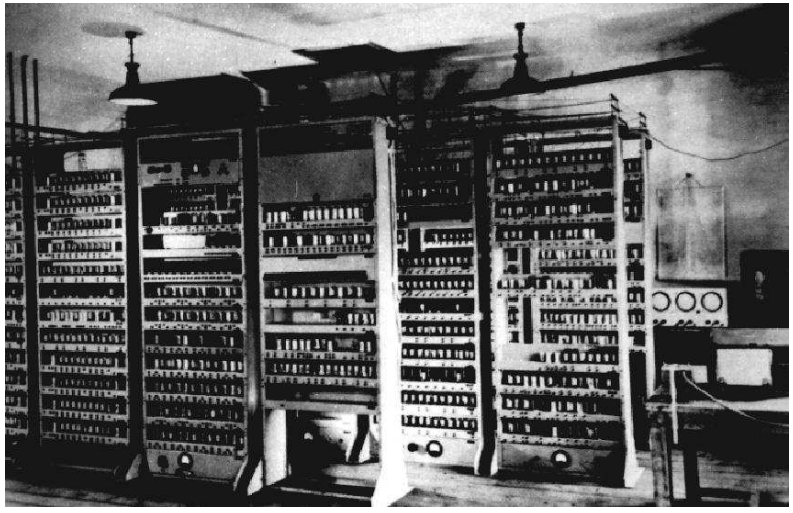
This is the Harvard Mark II Machine, an early computer boasting magnetic drum storage.

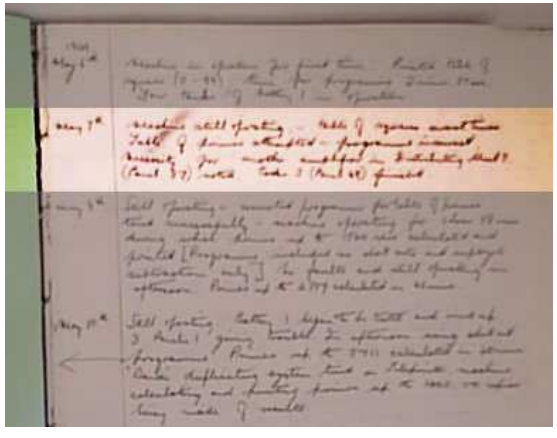
On 9 September 1945 it broke down because a moth got caught between the points of Relay #70 in Panel F.

At 3:45pm Grace Murray Hopper extracted it and taped it into the log book with the comment “First actual case of bug being found”.

In fact the term *bug* to mean a snag or defect was used by Edison as early as 1878.

The First Software Bug





This is the EDSAC I, the first computer with a stored program.

It became operational on 6 May 1949, printing a table of square numbers. On 7 May 1949 the log entry reads

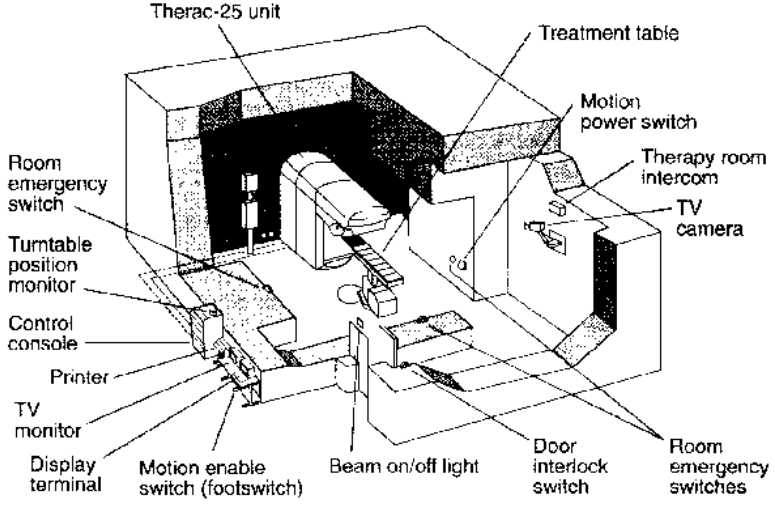
> *Machine still operating, - table of squares several times. Table of primes attempted - programme incorrect.*

Maurice Wilkes recalls the experience of debugging a program in June 1949:

> *[T]he realization came over me with full force that a good part of the remainder of my life was going to be spent in finding errors in my own programs.*

Big Software Bugs

In 1996, the $2B Ariane 5 rocket exploded on its maiden flight because there wasn't enough room to store a 64 bit number in a 16 bit buffer. This caused the Inertial Reference System to crash and output a test pattern, which the rocket controller interpreted as real flight data. The rocket changing direction caused it to disintegrate, which in turn triggered the self destruct mechanism.



Between 1985 and 1987 several installations of the Therac 25 linear accelerator, a radiation machine to treat cancer, caused the deaths of at least 2 people and injured several more. A particular combination of operator key presses caused the patient to be blasted with X-rays at 125 times the recommended dose for their condition.

What are Bugs?

- Bugs are deviations in behaviour from a specification.
- Hard problem: What are programs supposed to do?
  - The Therac 25 software was supposed to blast people with radiation (just not as much as it did).
  - What is a word processor supposed to do?
  - What about ambiguities in English?
- Easier: What are programs *not* supposed to do?
  - Crash the computer.
  - Interfere with other programs.
  - Corrupt data.

Debugging Successes

- Memory protection
  - Prevents programs interfering with each other.
- High level languages
  - Prevents data corruption and crashes.
- Types
  - Detect nonsense operations such as “a” + 1.
  - Prevents data corruption.
- What else can be done to help prevent programmers from inserting bugs into programs?
- Warning: In his book *The Mythical Man Month*, Brooks claims that there will be “no silver bullet”.

Testing

- Q: Why don't programmers simply test their programs?
  - A: They do!
  - Testing is an effective technique for finding bugs that appear *frequently*.
- If you have a bug in your software that crashes the computer every 1,000,000 hours on average, then:
  - you need 1,000,000 hours of testing to spot the bug;
  - but every day it will crash one of your 50,000 users.
- Another problem: how to know when to stop testing?
  - "Program testing can be used to show the presence of bugs, but never to show their absence!" [Dijkstra]

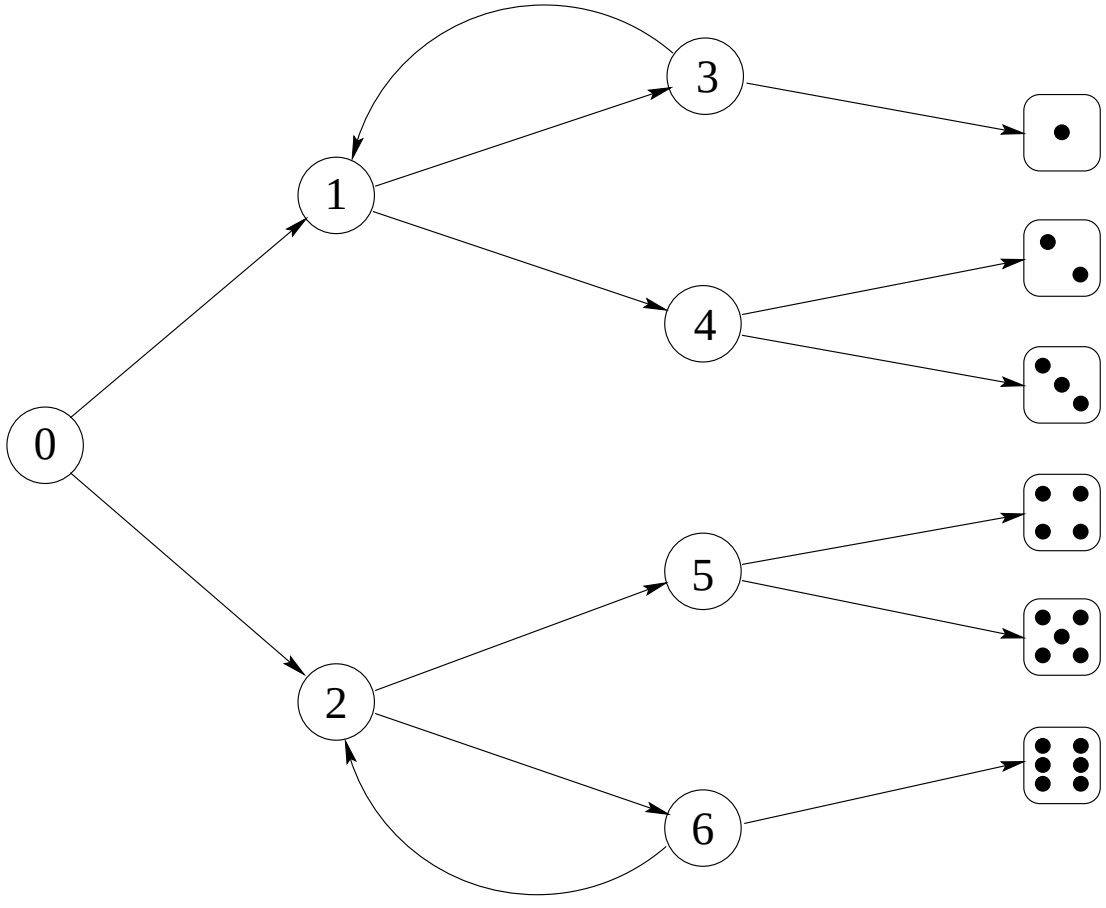
Static Analysis

- Consider the following buggy procedure for squaring a number:
  1. Let i be the input number.
  2. If i is 1000 then set r to be 17, else set r to be $i * i$.
  3. Return r as the answer.
- Testing is going to have a hard time finding this bug.
  - But it's obvious from looking at the program.
- Static analysis finds bugs by examining the program before it is executed.
  - It's doesn't replace testing, it's complementary.
  - It tends to find different kinds of bugs.

Proving Programs Correct

- Idea! Prove programs are correct [Turing, 1949]
- Programs and specifications are both formal objects, so no philosophical objections.
  - Warning: it's a different matter if you run the program on a real computer :-)
- Bonus: There are no ambiguities in a logic specification.
- Bonus: Completing the proof guarantees correctness.
- Expensive, but starting to be commercially viable.
  - Intel now proves floating point operations correct to avoid another $1B Pentium bug.
  - Microsoft uses SLAM to check all device drivers.

Probabilistic Programs



A dice throw generator [Knuth 1976].

Probabilistic Programs

- How to test a dice throw generator?
  - 5,2,3,1,2,1,1,2,5,3,5,4,2,3,1,6,5,4,2,3,1 $\rightarrow$ OK?
  - 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 $\rightarrow$ OK?
  - Would it perturb you if a coin came up heads 85 times in a row? [Rosencrantz & Guildenstern]
- In my research I prove the correctness of probabilistic programs:

$$\vdash \quad \forall n.\, 1 \leq n \leq 6 \;\Rightarrow\; \mathbb{P}\{s \mid \mathsf{dice}\ s = n\} = \tfrac{1}{6}$$

Conclusions

- People will continue to demand highly complex software that inevitably contain bugs.
  - Perhaps even buggy software to automate a task is better than doing the task manually (or not at all)?
- New verification techniques to keep a lid on this complexity is a *grand challenge* of computer science.
  - And will allow programmers to write even more complex software with new kinds of bugs :-)
- Logic is too useful to be left to theoreticians.