

Lecture 5:

Correctness and Efficiency

Martin Henz

Overview of this Lecture

- Correctness of algorithms
 - Kinds of errors
 - Sources of errors
 - Correctness
- Efficiency of algorithms
 - Example
 - Big-O
 - Recurrences

Summary of Previous Lecture

- Searches and traversals
- Divide and conquer
- Greedy algorithms
- Dynamic programming

Computer Errors Do Happen

- Mars Polar Lander project started February 1994, lost on December 3, 1999 after 11 months in space, traveled at least 35 million miles, cost of approx. \$165 million, and only 40 meters from landing.
- Premature engine shutdown most likely cause when Touchdown Monitor (TDM) software falsely indicated landing.
- Verification activities were comprehensive and performed by dedicated, experienced engineers.
- Fault not malicious, just difficult to find.
- According to Bob Knickerbocker (Director of Software at Lockheed Martin), Lockheed Martin had serendipitously found the bug a couple months after the crash.

Language Errors

- Errors in programs can result from the wrong usage of language constructs.
- Example:
for I in 0..10 then {Browse I} end
instead of
for I in 0..10 do {Browse I} end
- Language processors (compilers, interpreters) can catch these errors

Other Errors

- Today's language processors can catch other errors such as
 - {Browse 1+nil}
 - declare X=1 X=2
 - declare proc {P X} {Browse X} end
{P 1 2}
- Techniques:
 - Type checking,
 - Type inference,
 - Abstract interpretation,
 - Code verification

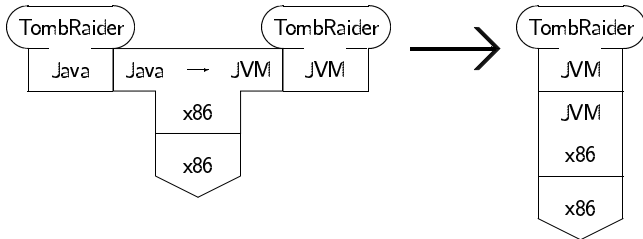
Logical Errors: Example

- Remember the program that counts the number of sentences in which “money” appears.
- This program recognizes the end of a sentence by looking for “. ”.
- What if we look for “.” instead?
- “The amount was \$322.56, a truly remarkable sum.”

The Source of Errors in Cooking

- Recipe
- Cook
- Hardware (oven, dishes, etc.)

The Source of Errors in Computing: Example



Errors can occur at any phase of the process.

The Usual Suspects

- Hardware?
- Compilers, interpreters?
- Software!

Example

- A few years ago, a Danish lady received, around her 107th birthday, a computerized letter from the local school authorities with instructions as to the registration procedure for first grade in elementary school.
- The software used only two digits for the “age” field in the database.

Fighting Errors in Software

- Prevent errors from being made (rigorous, disciplined software design)
- Find errors and remove them (debugging)
- Systematically search for erroneous behavior and correct it (testing)
- Software verification, correctness proofs

Infinite Loops

- Some programs never finish!
- Example:

```
declare
proc {Loop X}
  {Browse X}
  {Loop X+1}
end
{Loop 0}
```

Program Specification

1. Specification of set of legal inputs
2. The relationship between the inputs and the desired outputs

Partial and Total Correctness

- Partial correctness: Whenever the program terminates on a legal input, the desired relationship olds on the input and the output.
- Termination: The program terminates.
- Total correctness: Partial correctness + Termination

Floyd's Method

- Annotate program with assertions and convergents
- Prove these assertions and convergents
- Conclude correctness from assertions and convergents

Example: Reversing a List

```
fun {Rev X Y}
  if X==nil
  then Y
  else {Rev X.2 X.1|Y}
  end
end
fun {Reverse S}
  {Rev S nil}
end
```

Partial Correctness: Prove Statement S

Assume that the peg names A, B, and C are associated, in some order, with the variables X, Y, and Z. Then a terminating execution of the call “move N from X to Y using Z” lists a sequence of ring-moving instructions, which, if started in any legal configuration of rings in which at least the N smallest rings are on peg X, correctly moves those N rings from X to Y. The sequence adheres to the rules and leaves all other rings untouched.

Example: Towers of Hanoi

Subroutine: move N from X to Y using Z

1. if N is 1 then output “move X to Y”
2. otherwise
 - (a) call move $N - 1$ from X to Z using Y
 - (b) output “move X to Y”
 - (c) call move $N - 1$ from Z to Y using X

Proof by Induction

- Statement is true when N is 1
- Assume that the statement is true for some given $N - 1$, and show that the statement is then also true for N .

An Iterative Solution to Towers of Hanoi

1. Do the following repeatedly until all rings are correctly piled up on some other peg:
 - (a) move the smallest ring from its current peg to the next peg in clockwise order;
 - (b) make the only move possible that does not involve the smallest ring.

Automated Proofs

- Computers are used to proof correctness of programs and of other mathematical theorems.
- Example: Four-color problem was proven correct in 1976 by
 - Proving that the theorem is equivalent to about 1700 other statements.
 - Proving all 1700 statements using a computer.

Can we trust computerized proofs?

- Is the translation to 1700 statements correct?
- Is the software correct?
- Is the hardware correct?
- Are the language processors correct?

Overview of this Lecture

- Correctness of algorithms
 - Kinds of errors
 - Sources of errors
 - Correctness
- Efficiency of algorithms
 - Example
 - Big-O
 - Recurrences

Efficiency of Algorithms

- Example: normalize the score of a class of students
- 1. compute MAX
- 2. for I from 1 to N do:
 - (a) $L(I) \leftarrow L(I) \times 100/MAX$

Efficiency of Algorithms

- Second version: Only one arithmetic operation in loop
- 1. compute MAX
- 2. $FACTOR \leftarrow 100/MAX$
- 3. for I from 1 to N do:
 - (a) $L(I) \leftarrow L(I) \times FACTOR$

Big-O Notation

- Let us look at a numeric aspect of the input, say N
- An algorithm runs in time $O(N)$ if there exists a constant K such that for any input of size N , the algorithm takes $K \times N$ seconds.

Classes of Big-O

- Logarithmic algorithms: $O(\log_2 N)$, example: binary search
- Linear algorithms: $O(N)$, example: Reverse
- Pseudo-linear algorithms: $O(N \times \log_2 N)$, example: merge sort
- Quadratic algorithms: $O(N^2)$, example: bubble sort

Time Analysis of Recurrence: Example

1. If L has one element, this element is the smallest and largest
2. If L has two elements, take the smaller and larger ones
3. Otherwise:
 - (a) Split L in two halves: L_{left} and L_{right}
 - (b) Find their extremal elements: MIN_{left} , MIN_{right} , MAX_{left} , MAX_{right} .
 - (c) The smaller one of MIN_{left} and MIN_{right} is the smallest element of L .
 - (d) The larger one of MAX_{left} and MAX_{right} is the largest element of L .

Recurrence in the case that N is a Power of 2

Count the number $C(N)$ of comparisons used for a list with N elements

1. $C(2) = 1$
2. $C(N) = 2 \times C(N/2) + 2$

Average Case Complexity

- What is the runtime for a **typical** input?
- Characterize **typical!**
- Much more effort required for good estimates.

Upper and Lower Bounds

- Often it is useful to get general bounds on the possible efficiency
- Lower bounds: What is the least number of operations required?
- Example: Search in an unsorted list requires $O(N)$ comparisons.
- Upper bounds: A given big-O result provides an upper bound for other possible algorithms

Next Week

Inefficient algorithms!