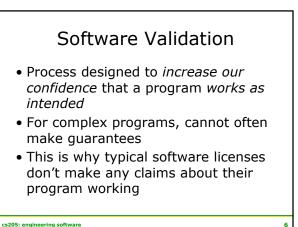


Java's License

2. RESTRICTIONS. Unless enforcement is prohibited by applicable law, you may not modify, decompile, or reverse engineer Software. You acknowledge that Software is not designed, licensed or intended for use in the design, construction, operation or maintenance of any nuclear facility. Sun disclaims any express or implied warranty of fitness for such uses.





- Verification
 - Argue formally or informally that the program always works as intended
- Analysis
 - Poor programmer's verification: examine the source code to increase confidence that it works as intended

cs205: engineering software

Testing and Fishing

Using some successful tests to conclude that a program has no bugs, is like concluding there are no fish in the lake because you didn't catch one!

s205: engineering software



10

Exhaustive Testing

- Test all possible inputs
- PS1: 50x50 grid, all cells can be either dead or alive before starting 2²⁵⁰⁰ =

R5880002444 601 1208833624 18972386204487738651 7582586770565238397862231 681 4880377085367367257257502584 433370245774852865778038227891 551 61 7758551 190731058970825844695446351 4258521 687 445 145 113 18252 5533955302477457459401 610768684295875718225446459446351 42585241647244 547 580 113 18252 5533955302477459401 6107686942955751 182 61 7758551 19073107897802544459445931 4258524162744003739382503 21 888025836109005699 7097531 432 43921 191051023075449463543051025854267469254951 2658528772240077 7702239914 1446025185755261 2053954728091 1185032037574496354837699515695514185041 7772458444655 279671 19230528574555368601 307024798821 183474395521 677469529862551 75858826750271589400788772725007 70850526952877214028422771402842274865587878797475332290261 488504377

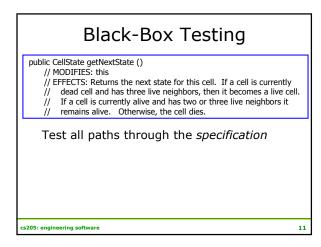
But that's not all: all possible start stop step clicks, different platforms, how long to you need to run it, etc.

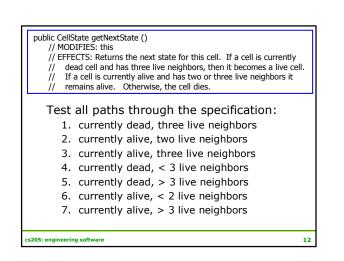
s205: engineering software

Selective Testing

- We can't test everything, pick test cases with high probability of finding flaws
- Black-Box Testing: design tests looking only at specification
- Glass-Box Testing: design tests looking at code
 - Path-complete: at least one test to exercise each path through code

s205: engineering software





Black-Box Testing

public CellState getNextState ()
 // MODIFIES: this

- // EFFECTS: Returns the next state for this cell. If a cell is currently
- // dead cell and has three live neighbors, then it becomes a live cell.

13

17

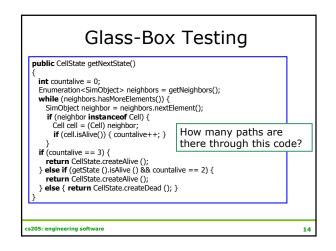
// If a cell is currently alive and has two or three live neighbors it // remains alive. Otherwise, the cell dies.

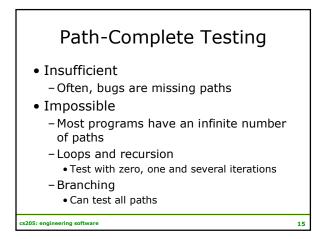
Test all (7) paths through the specification

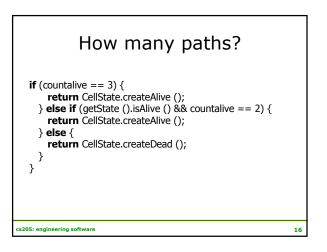
Test boundary conditions

- 1. all neighbors are dead
- 2. all neighbors are alive
- 3. cell is at a corner of the grid
- 4. cell is at an edge of the grid

cs205: engineering software



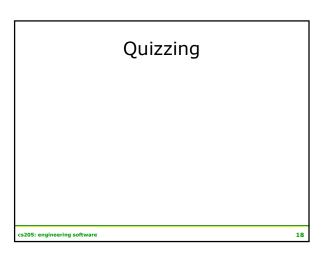




Testing Recap

- Testing can find problems, not to prove your program works
 - Since exhaustive testing is impossible, select test cases with maximum probability of finding bugs
 - A successful test case is one that reveals a bug in your program!
- Typically at least 40% of cost of software project is testing, often ~80% of cost for safety-critical software

cs205: engineering software



Testing Recap

- Testing can find problems, but can't prove your program works
 - Since exhaustive testing is impossible, select test cases with maximum probability of finding bugs
 - A successful test case is one that reveals a bug in your program!
- If we can't test all possible paths through a program, how can we increase our confidence that it works?

19

21

cs205: engineering software

s205: engineering software

Analysis Make claims about *all* possible paths by examining the program code directly, not executing it Use formal semantics of programming language to know what things mean

20

22

 Use formal specifications of procedures to know that they do

Hopelessness of Analysis It is impossible to correctly determine if any interesting property is true for an arbitrary program! The Halting Problem: it is impossible to write a program that determines if an arbitrary program halts.

Compromises Use imperfect automated tools: Accept unsoundness and incompleteness False positives: sometimes an analysis tool will report warnings for a program, when the program is actually okay (unsoundness) False negatives: sometimes an analysis tool will report no warnings for a program, even when the program violates properties it checks (incompleteness) Use informal reasoning Design programs to modularize reasoning

s205: engineering software

s205: engineering software

