327: Object-oriented programming

Lecture 12

10/11/2021

Professor Barron

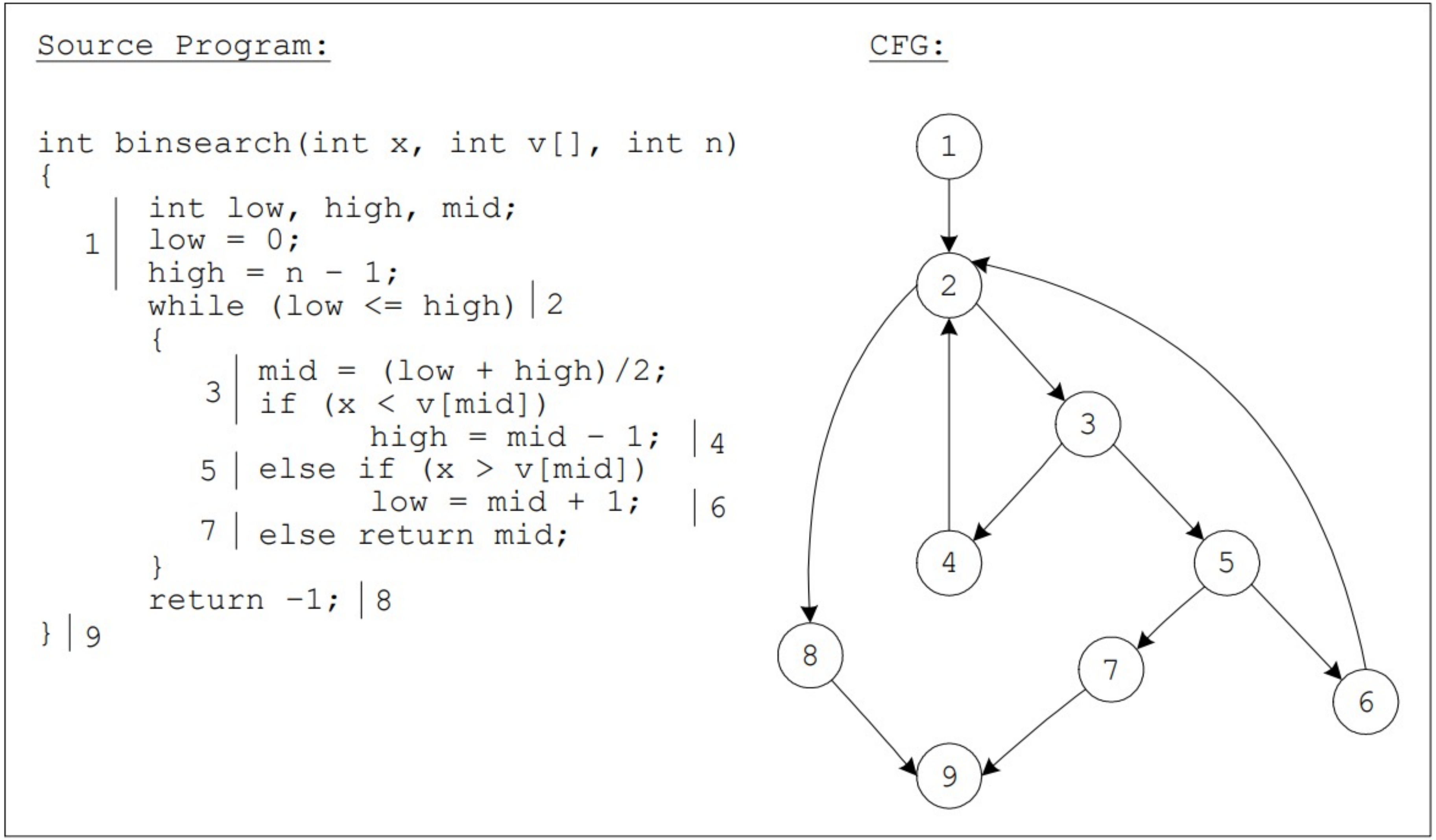
How much testing do we need?

- Have we tested enough cases?
    - checking boundary conditions
    - hard to quantify
- How much of the code was tested?
    - code coverage
    - easier to quantify
- Could we automate the generation of tests?
    - symbolic execution
    - fuzzing

Boundary values

- Values at the boundary of conditions
- For example, a function that converts a numeric grade to A,B,C,D,F
  - 89 and 90 are boundary values
  - 81-88 are less important to check
- We should focus testing on these boundaries

Control flow graph



Source Program:

```
int binsearch(int x, int v[], int n)
{
      | int low, high, mid;
   1  | low = 0;
      | high = n - 1;
        while (low <= high) | 2
        {
            3 | mid = (low + high)/2;
              | if (x < v[mid])
                      high = mid - 1;  | 4
            5 | else if (x > v[mid])
                      low = mid + 1;   | 6
            7 | else return mid;
        }
        return -1; | 8
} | 9
```

CFG:

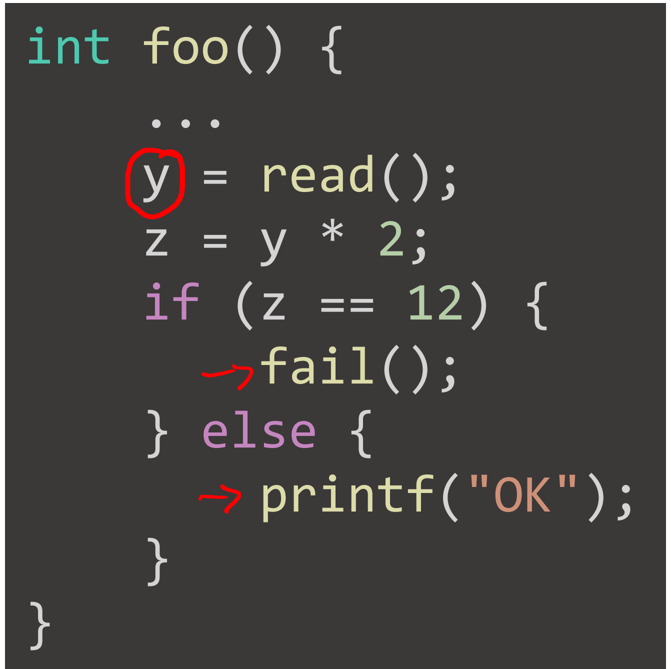
figure from Raihan Al-Ekram and Kostas Kontogiannis

Code coverage

- **Function coverage**
    - has each function in the program been called?
- **Statement coverage**
    - has each statement in the program been executed?
- **Branch coverage**
    - has each branch of each control structure been executed?
- **Edge coverage**
    - has every edge in the Control flow graph been executed?
- **Condition coverage**
    - has each Boolean sub-expression evaluated both to true and false?

Symbolic execution

- Analyze the code to determine what inputs lead to different paths in the CFG
- Find constraints on inputs that could possibly lead to bugs
- Difficulties with memory aliasing and path explosion



```
int foo() {
    ...
    y = read();
    z = y * 2;
    if (z == 12) {
       fail();
    } else {
       printf("OK");
    }
}
```

y == 6

y != 6

Fuzzing

- Very popular security research area in recent years
- Essentially large-scale randomized input testing
- Instead of worrying about finding the right boundary values, you could just try everything!
- Ideally inputs have some structure so that we don’t waste time with rejected inputs
  - Could be randomly mutated from a set of normal inputs
  - Could have a grammar or protocol
- Chrome is continually being fuzzed
  - 14 trillion test inputs in 30 days found 112 bugs.
  - https://github.com/google/oss-fuzz