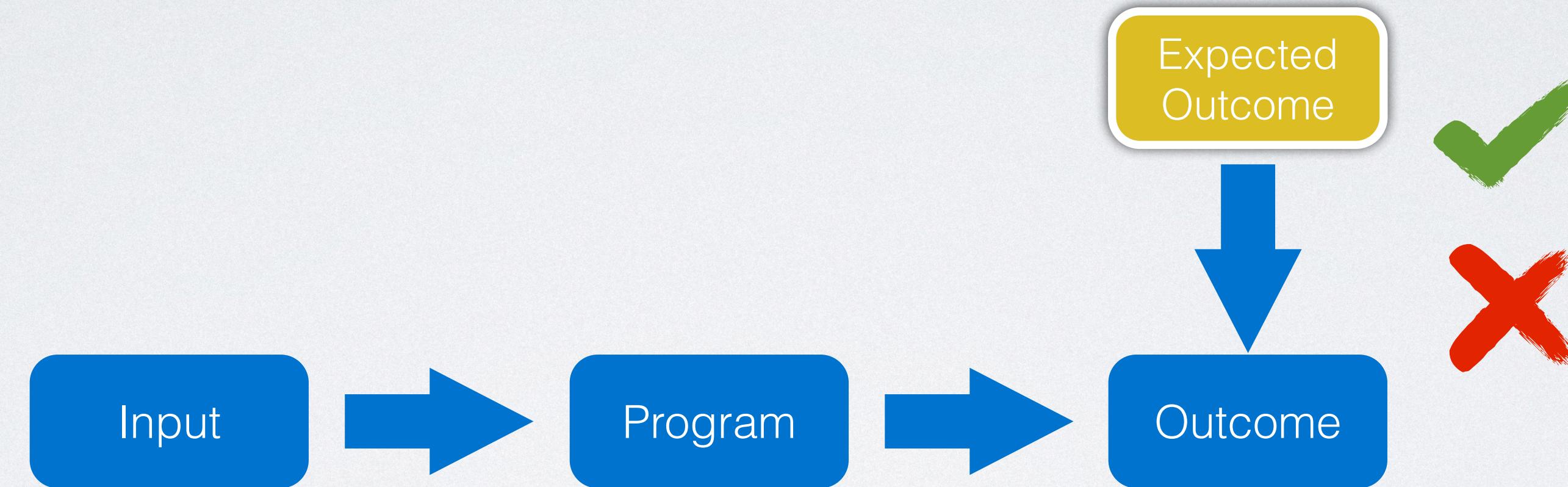


SEARCH-BASED AUTOMATED TEST GENERATION FOR JAVA PROGRAMS

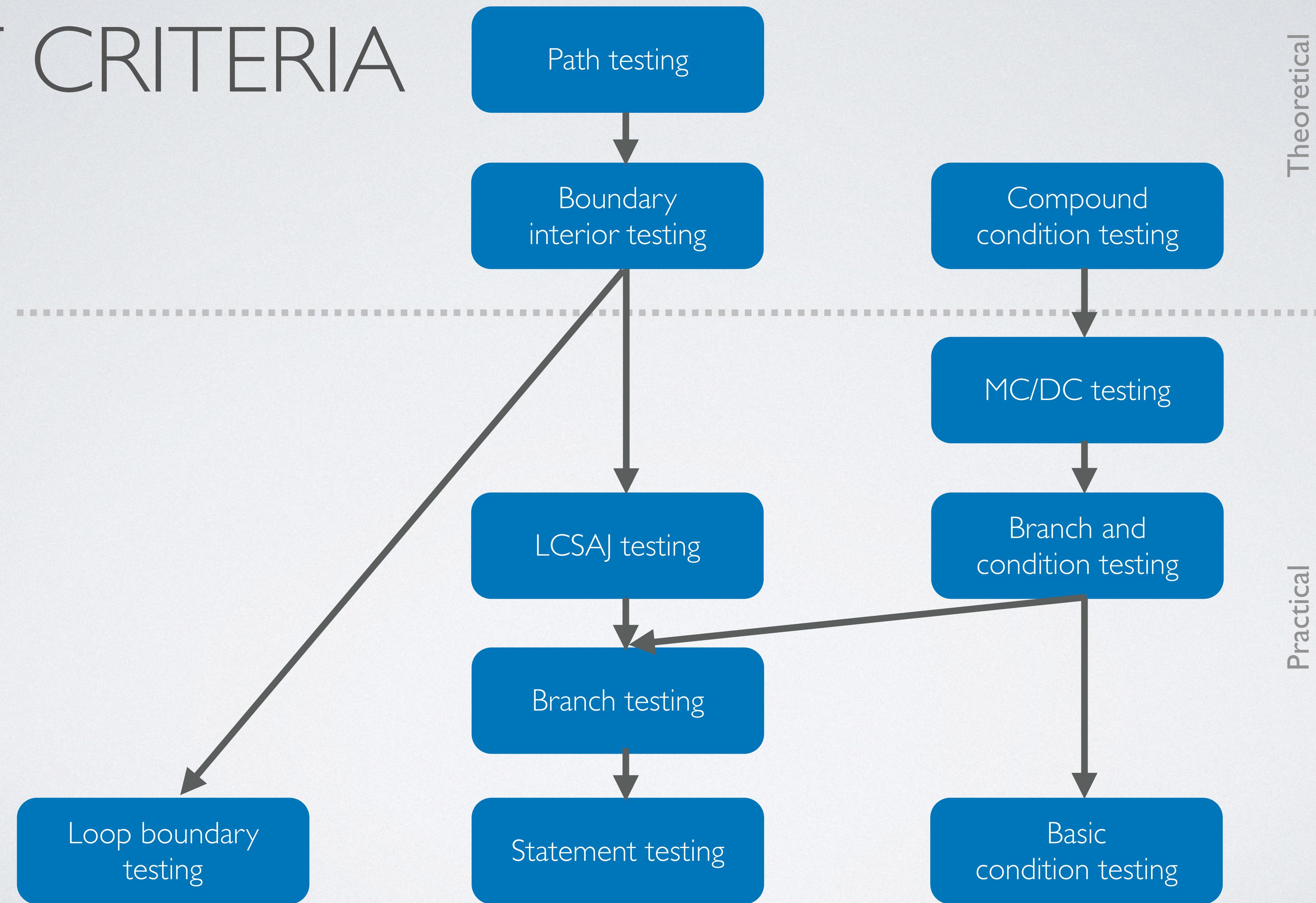
José Miguel Rojas
Department of Computer Science
The University of Sheffield

UNIT TESTING

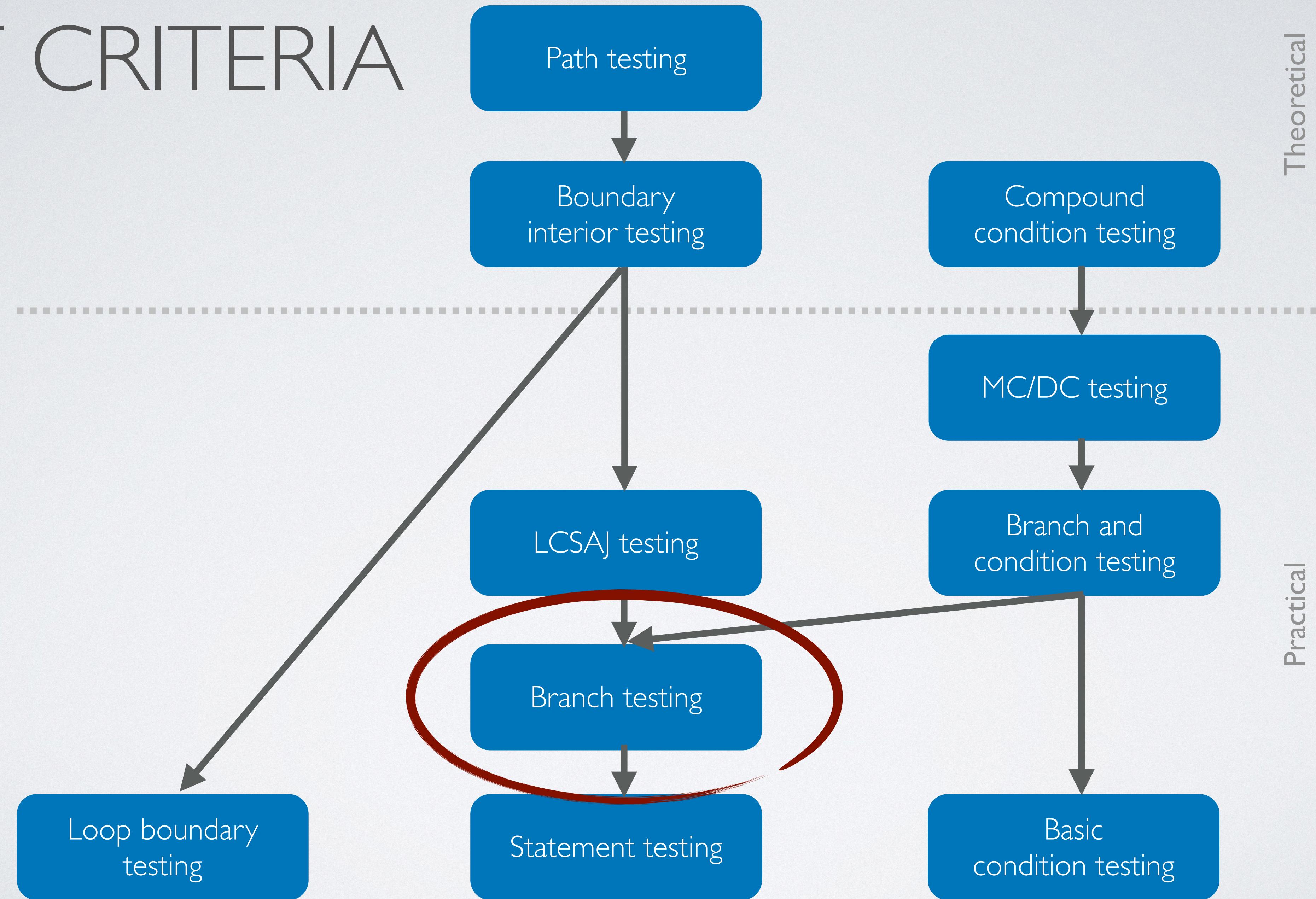


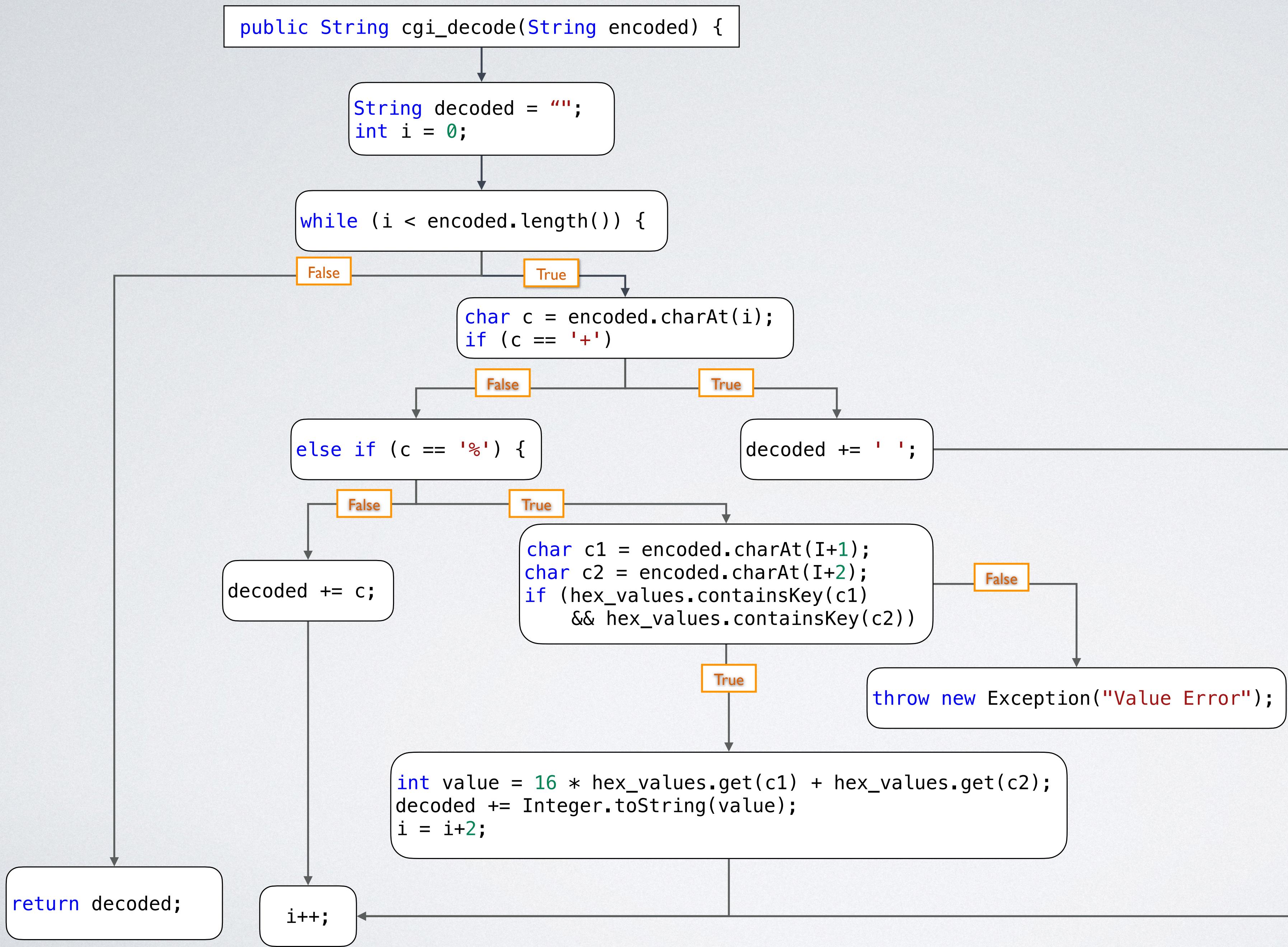
1. Execute program using test data as *input*
2. Verify program *output* using test oracle

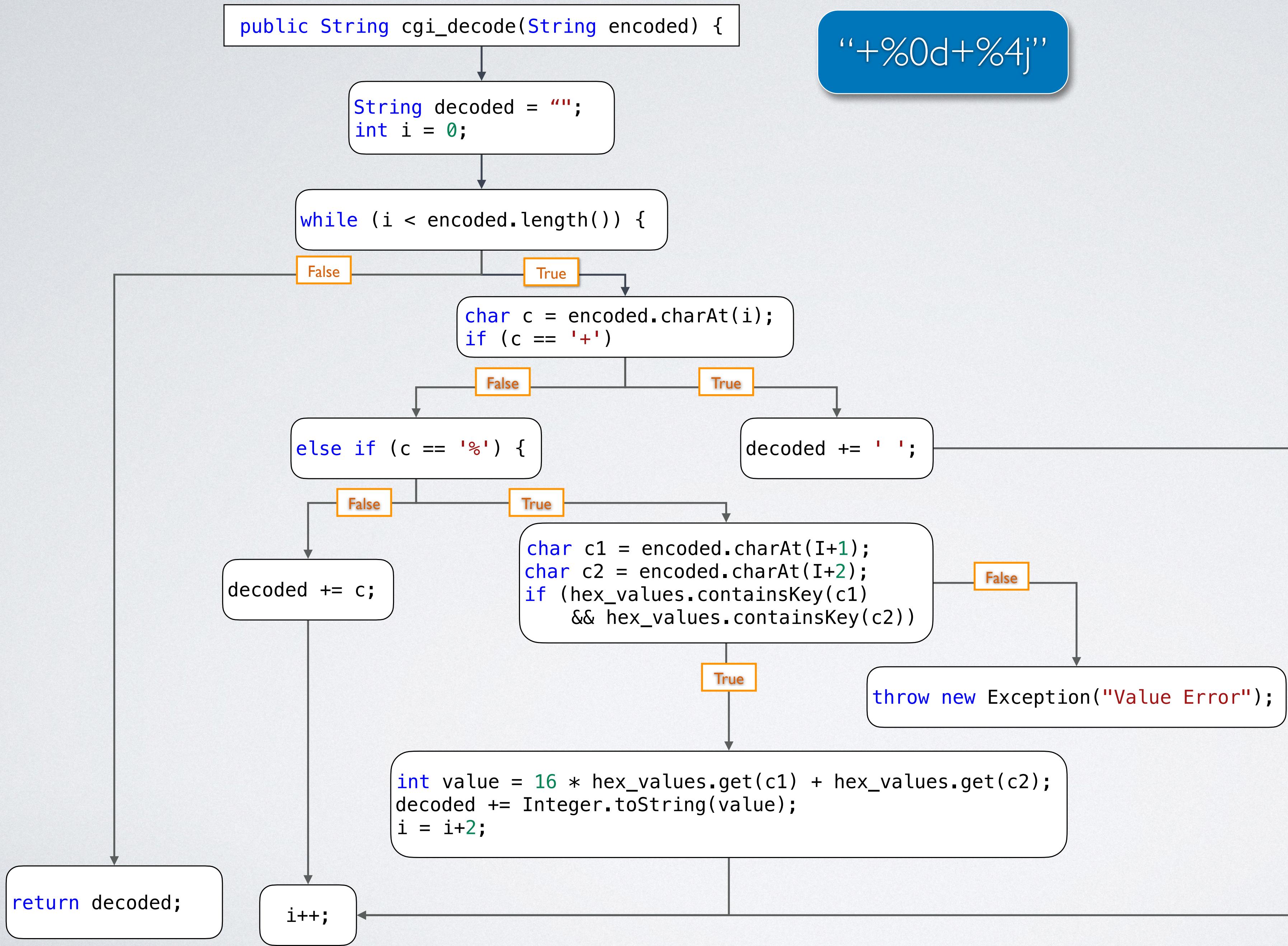
TEST CRITERIA



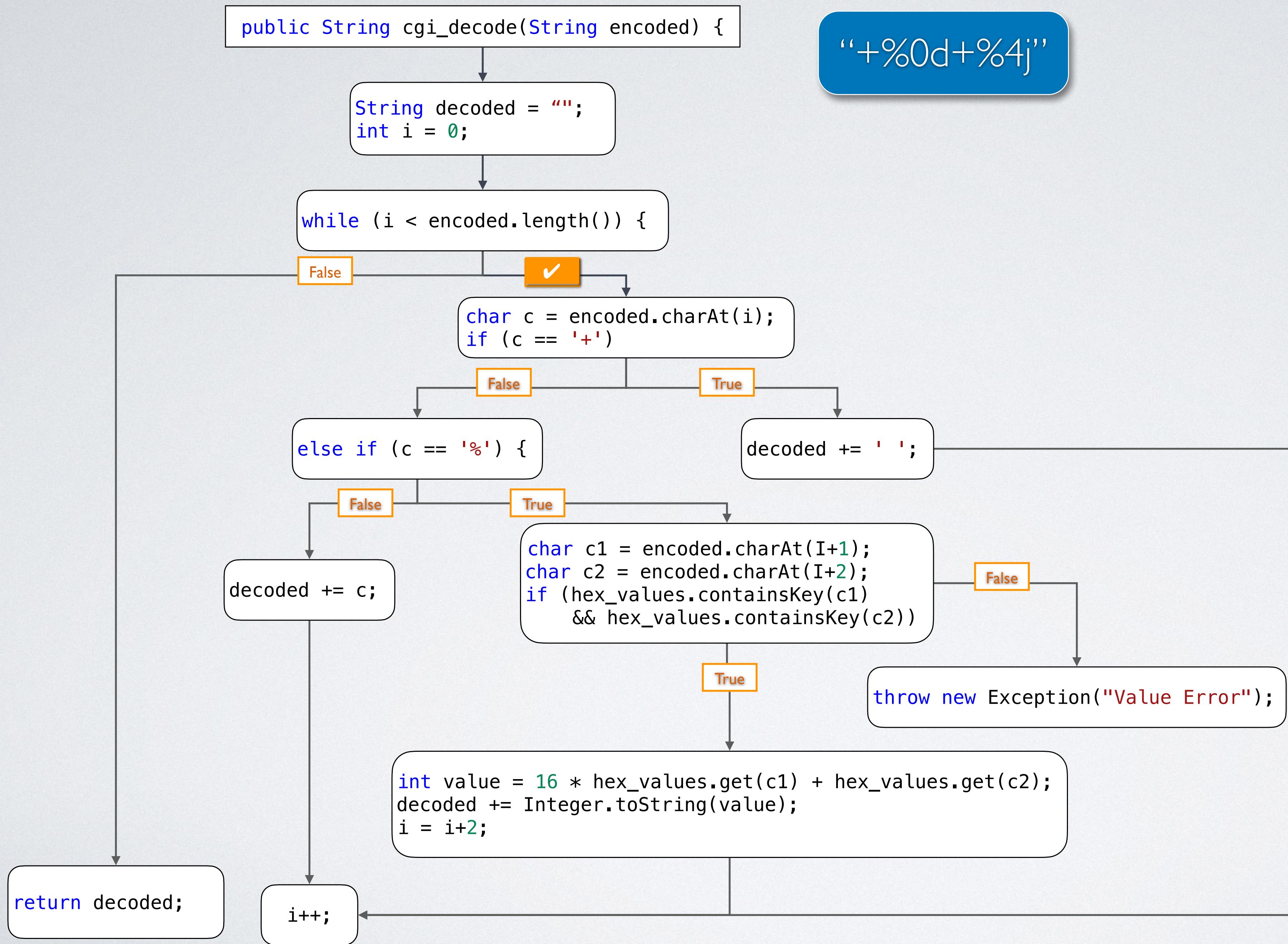
TEST CRITERIA

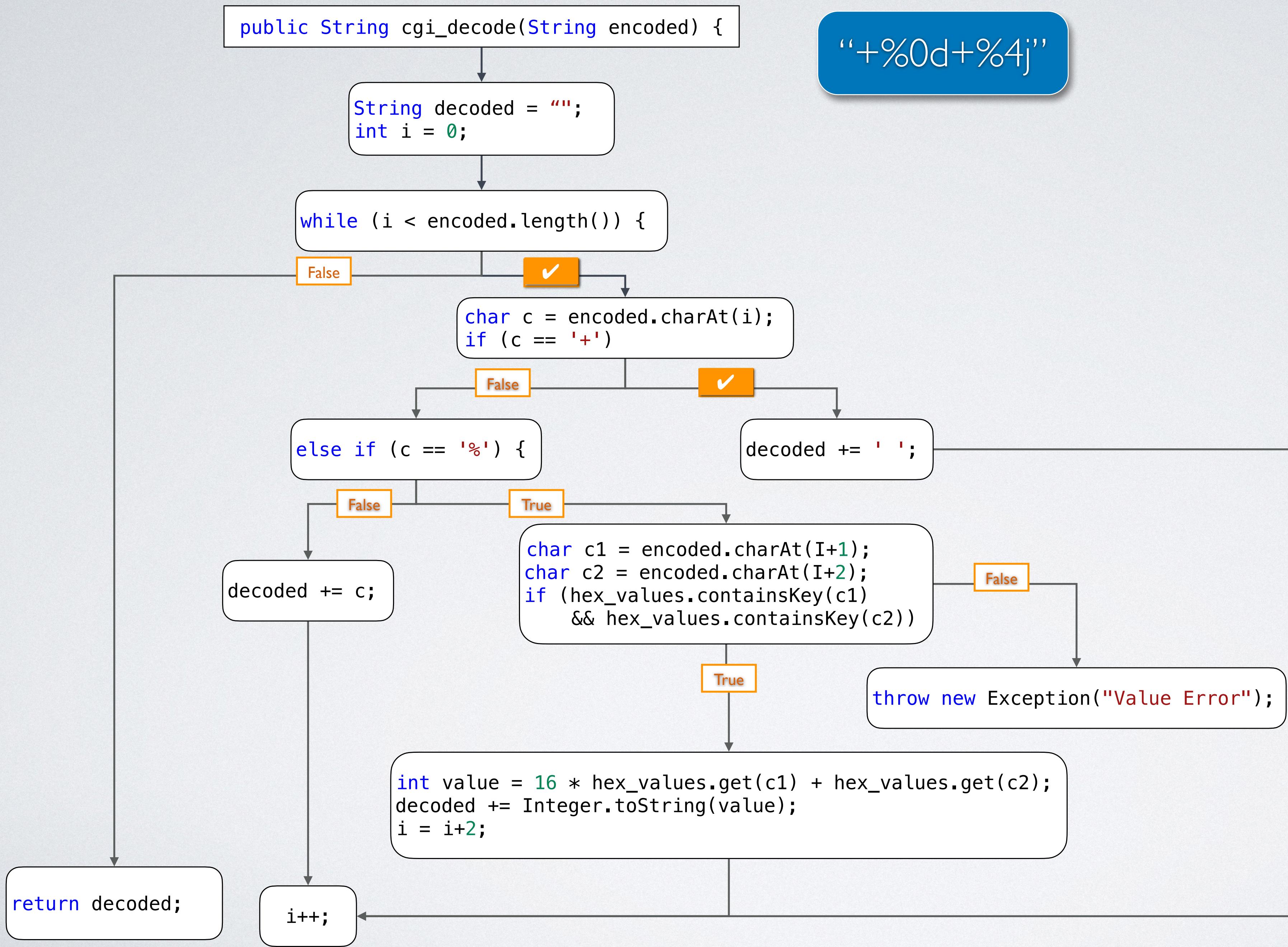




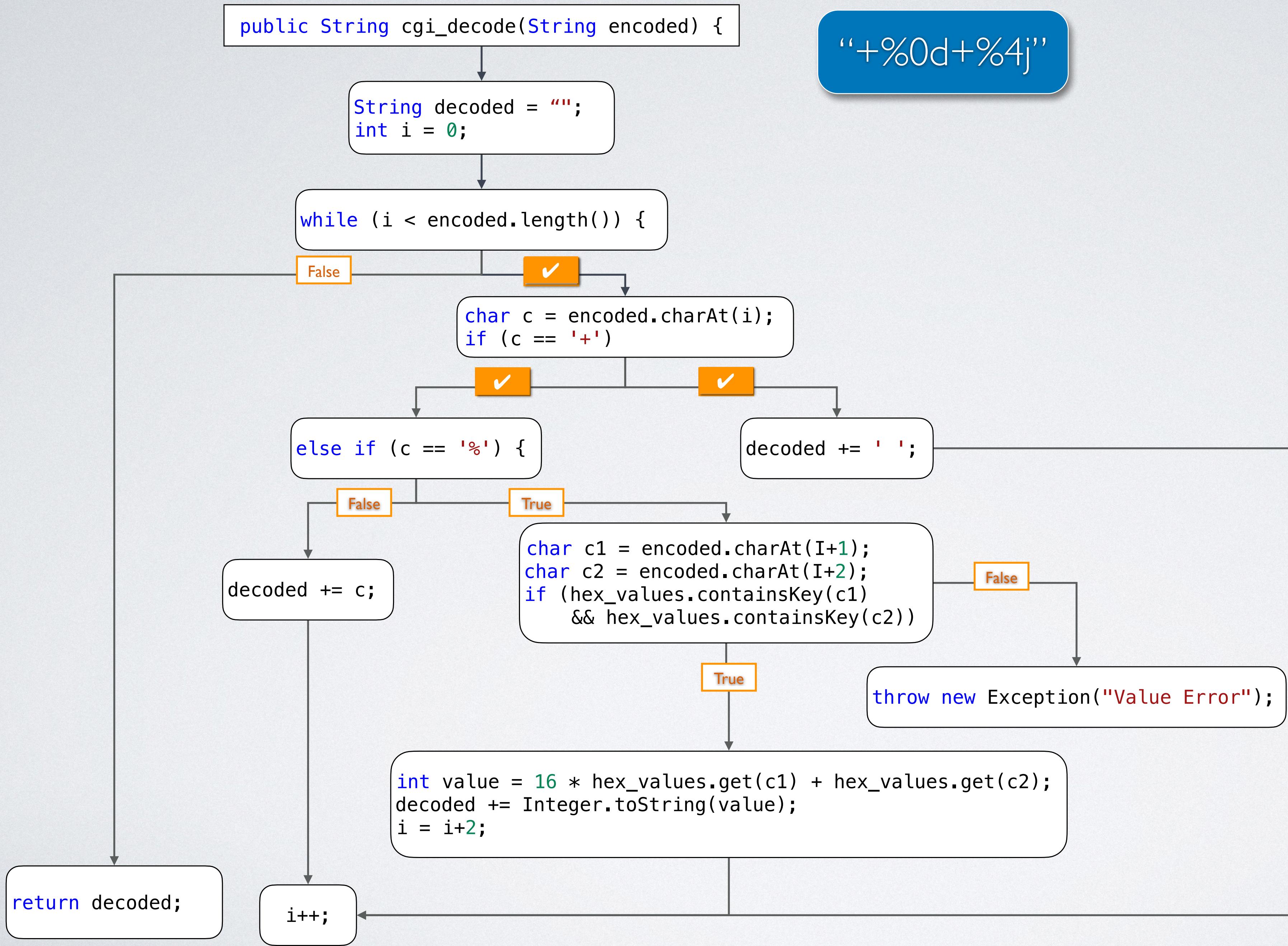


“+%0d+%4j”

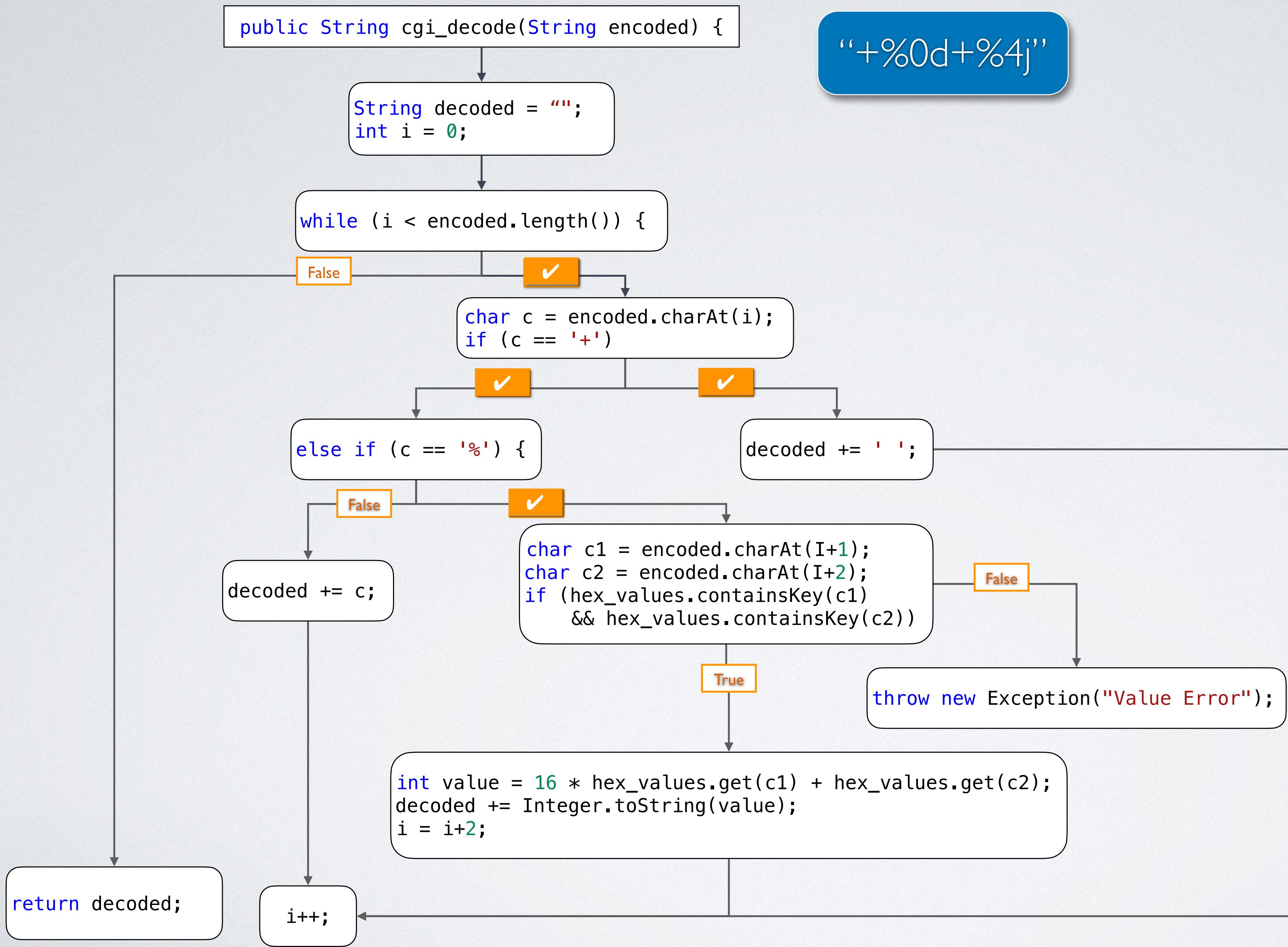




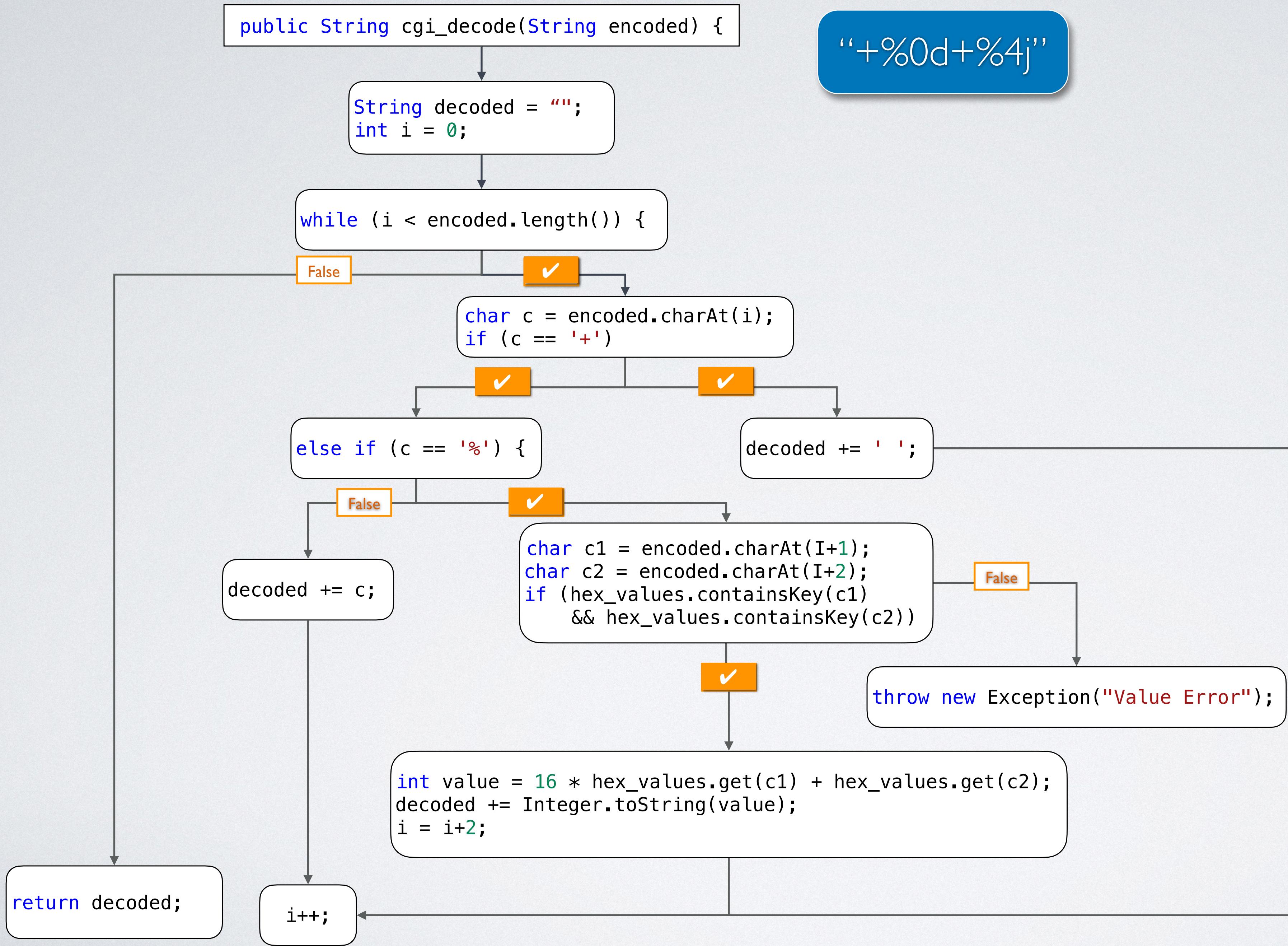
“+%0d+%4j”

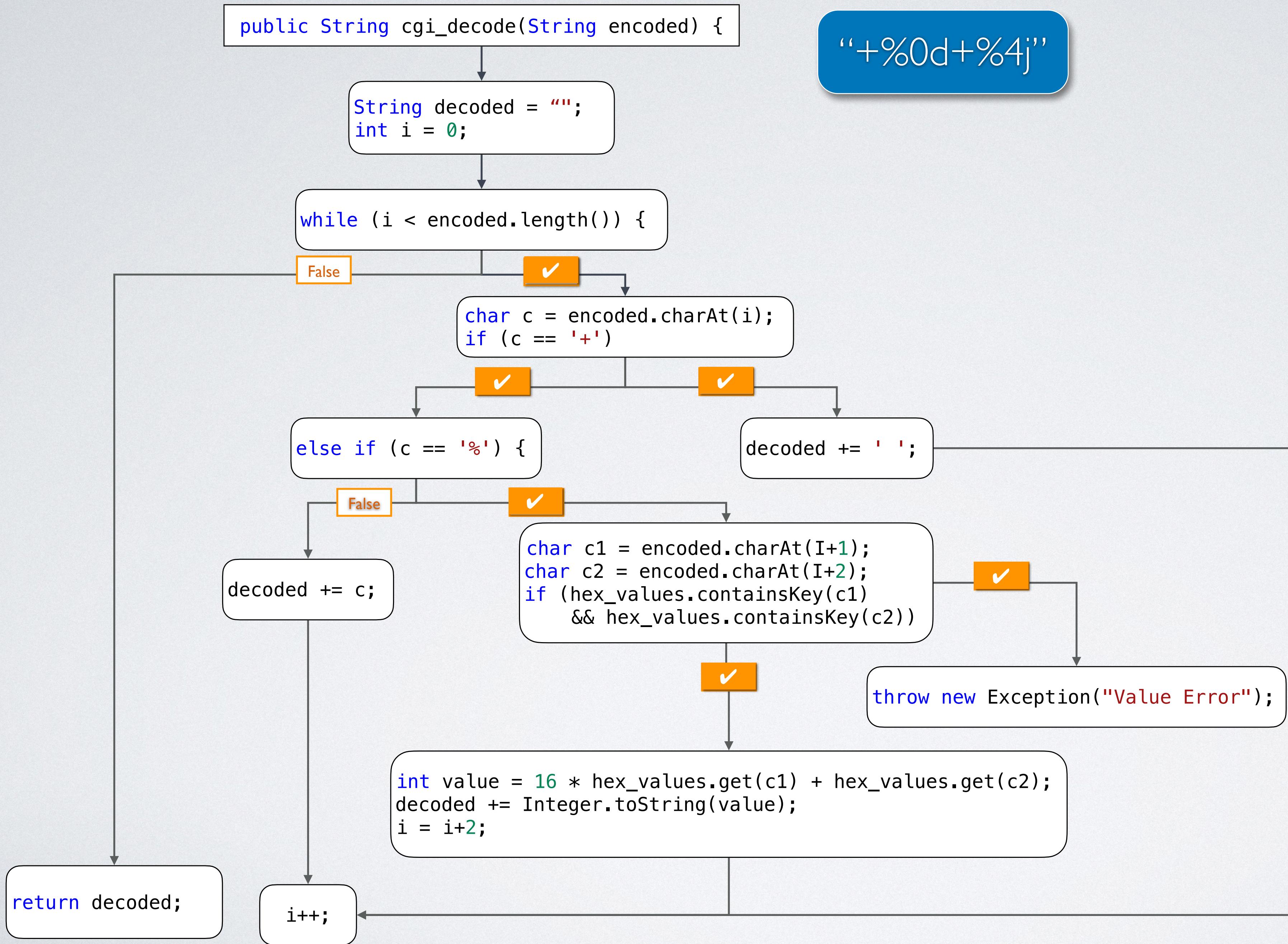


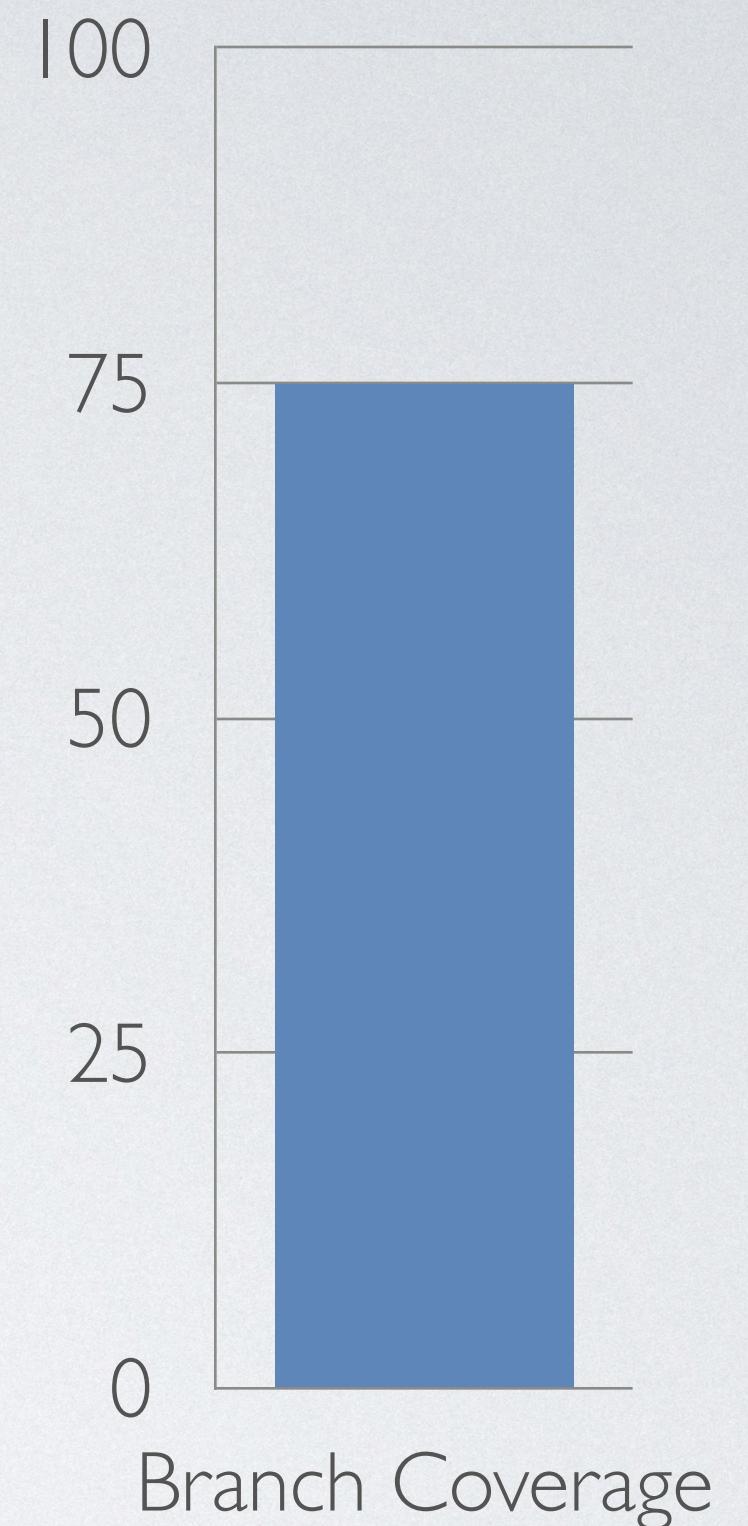
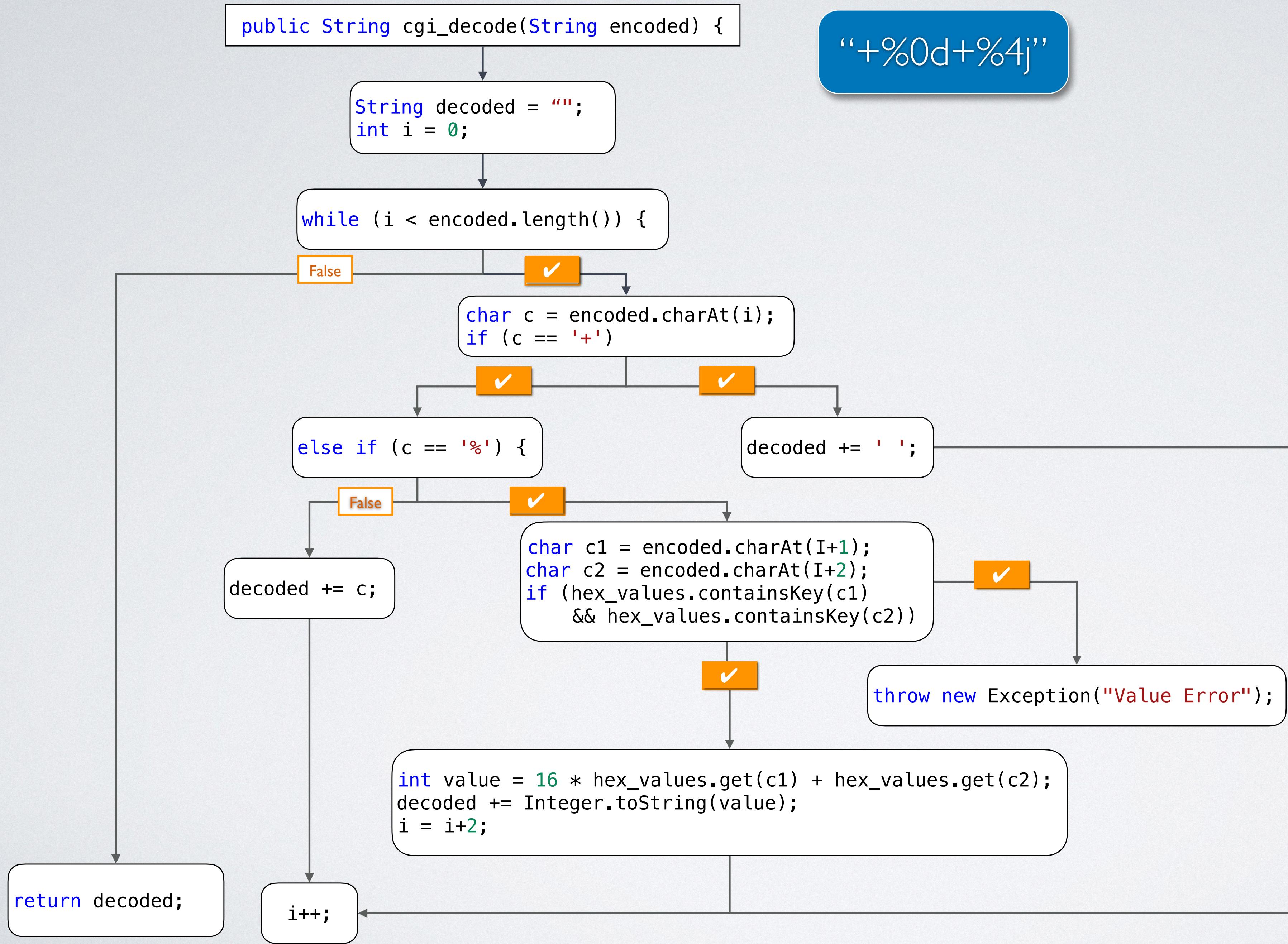
“+%0d+%4j”

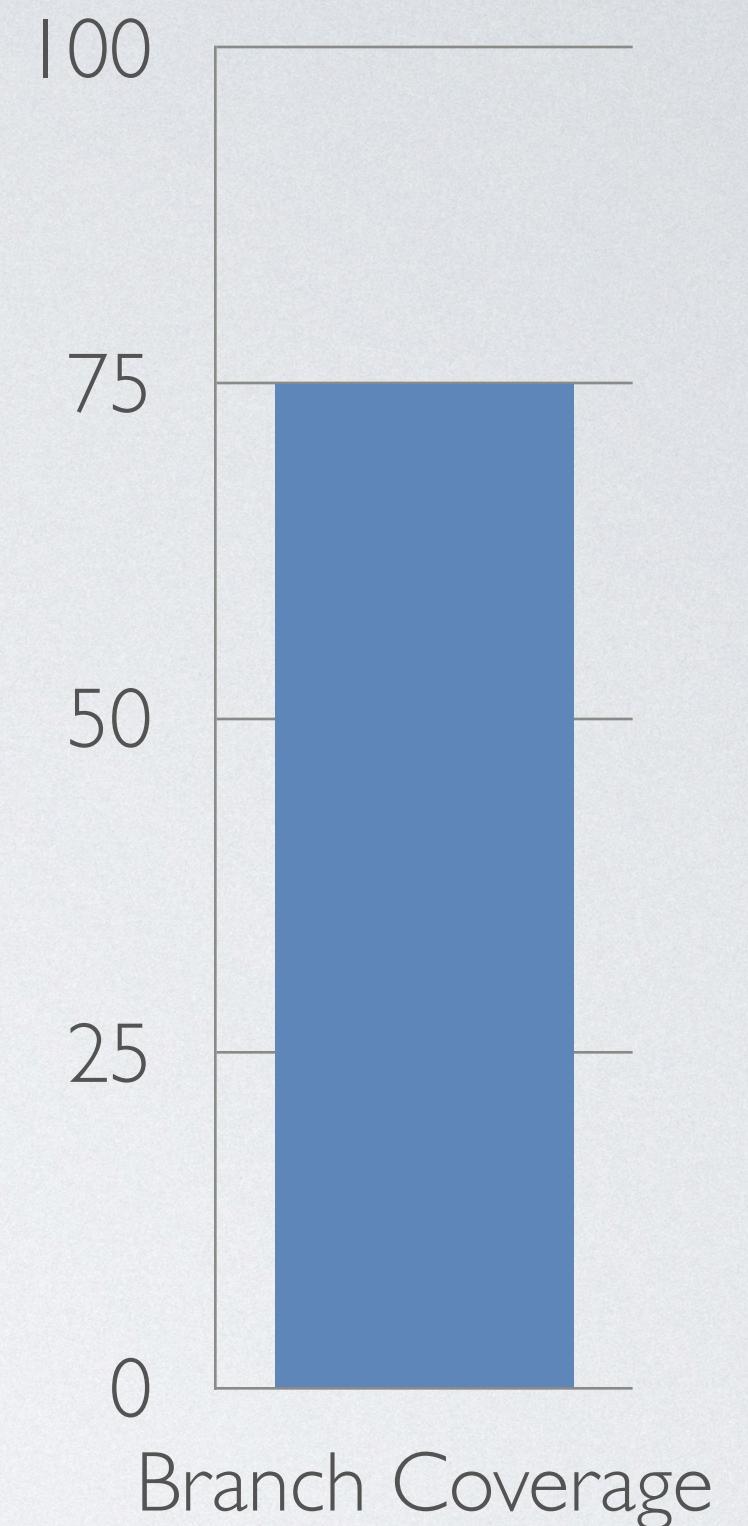
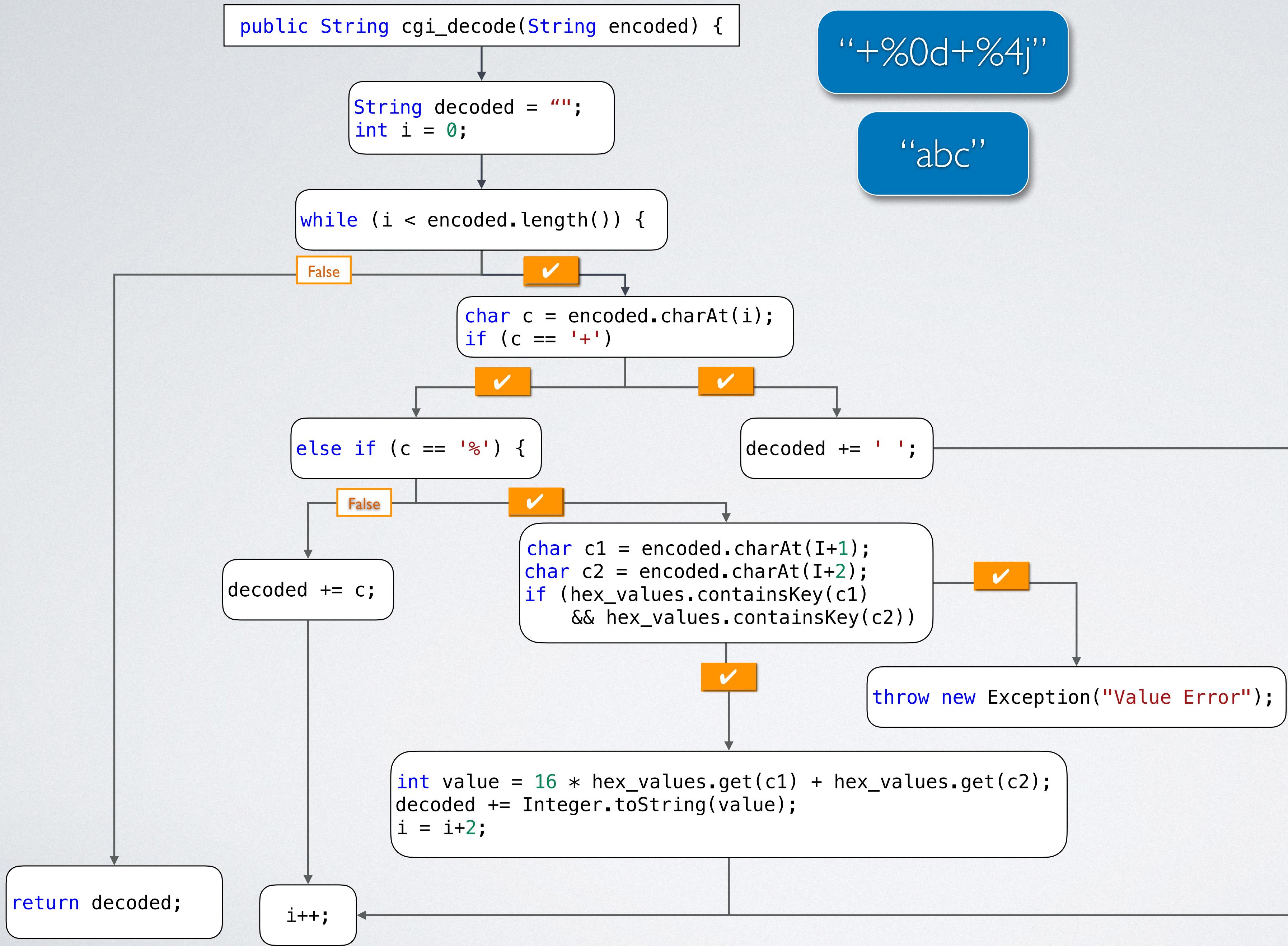


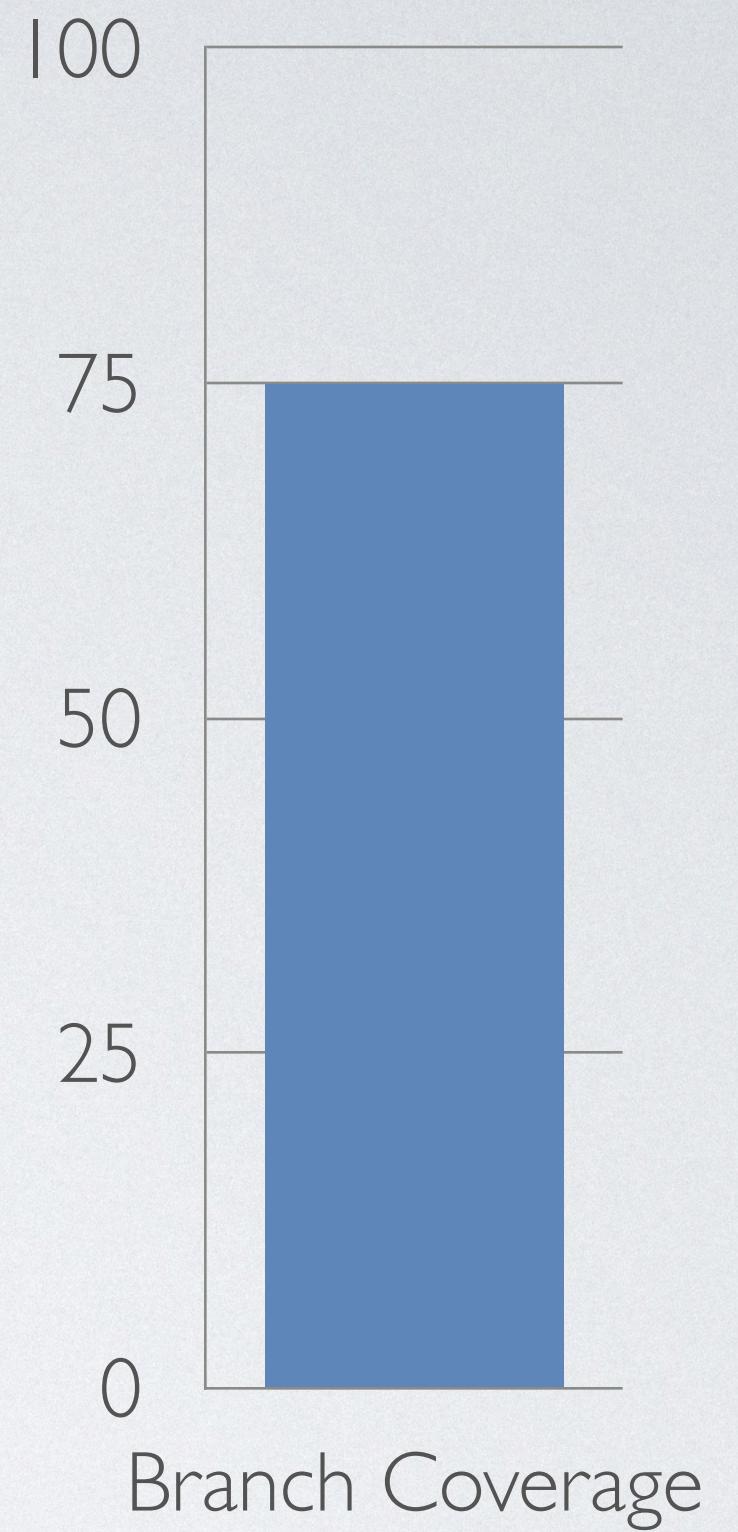
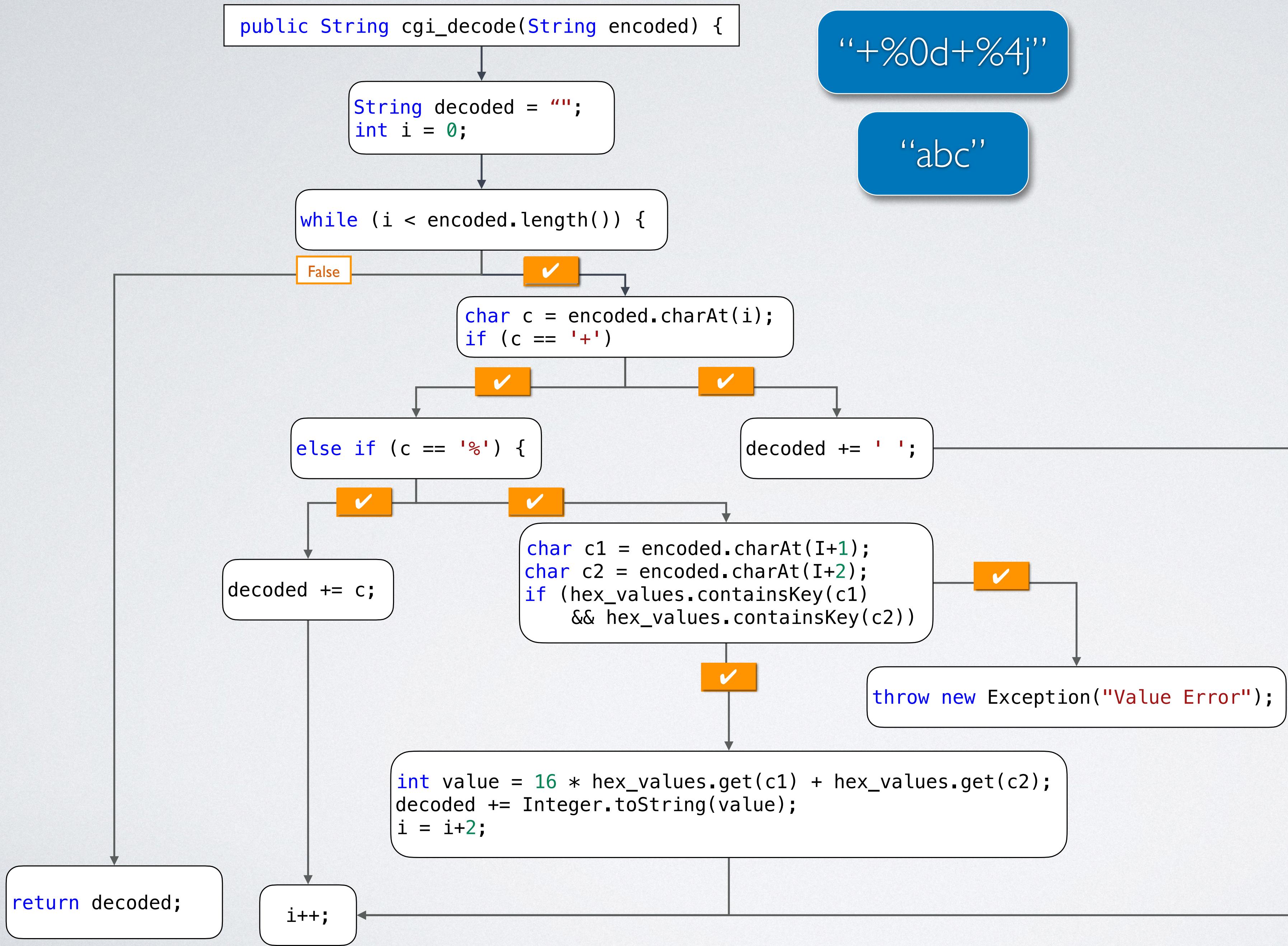
“+%0d+%4j”

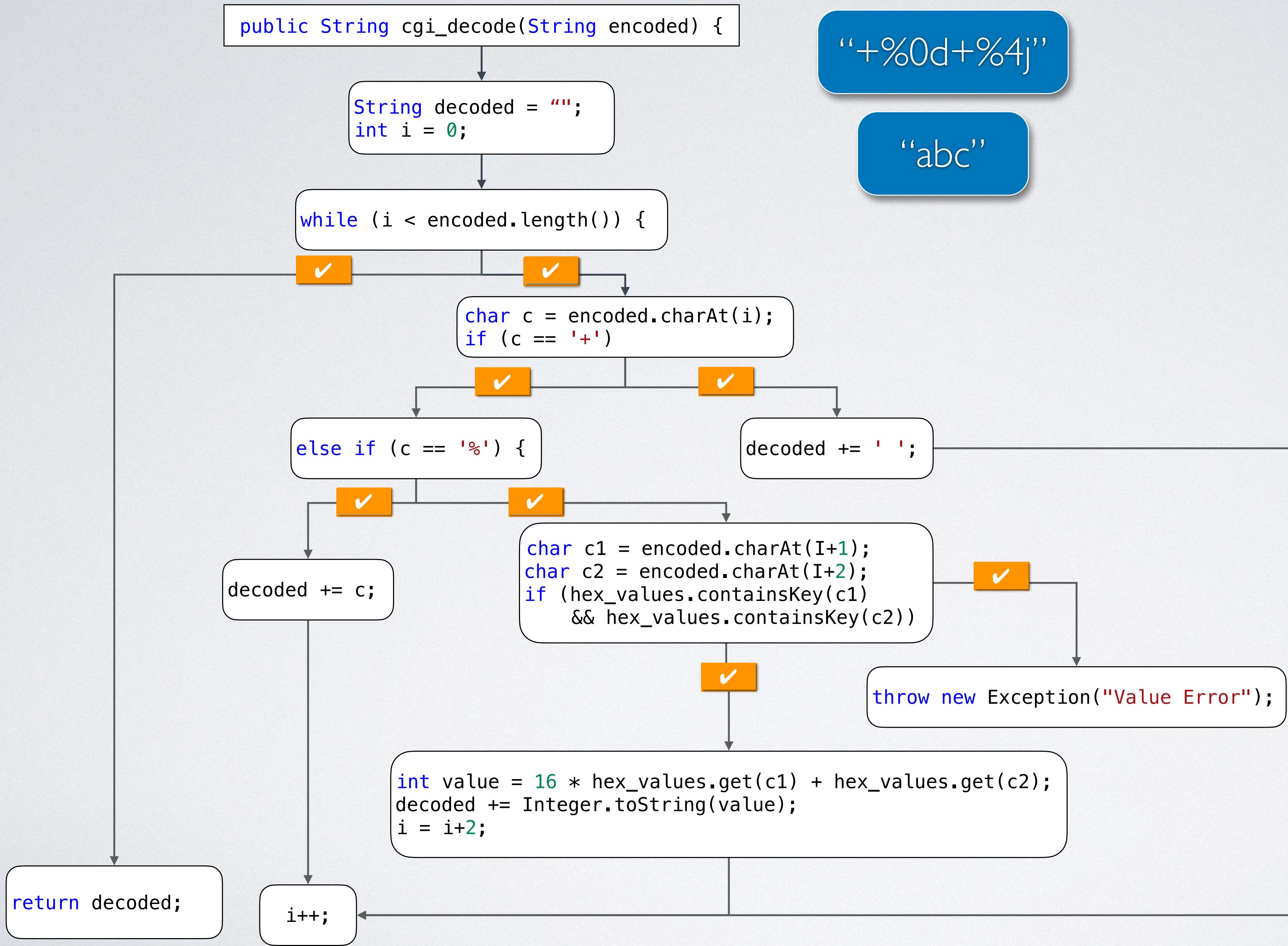






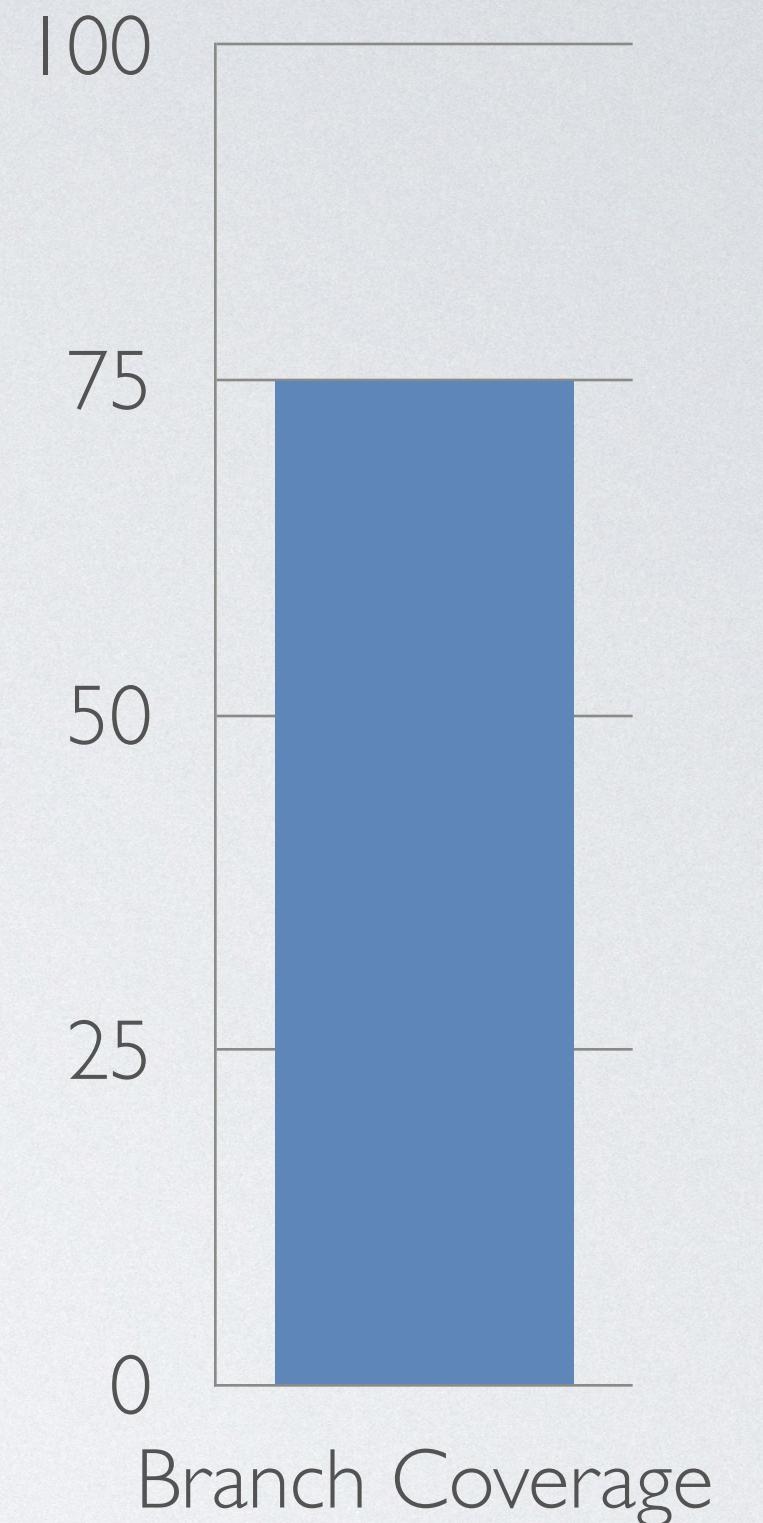


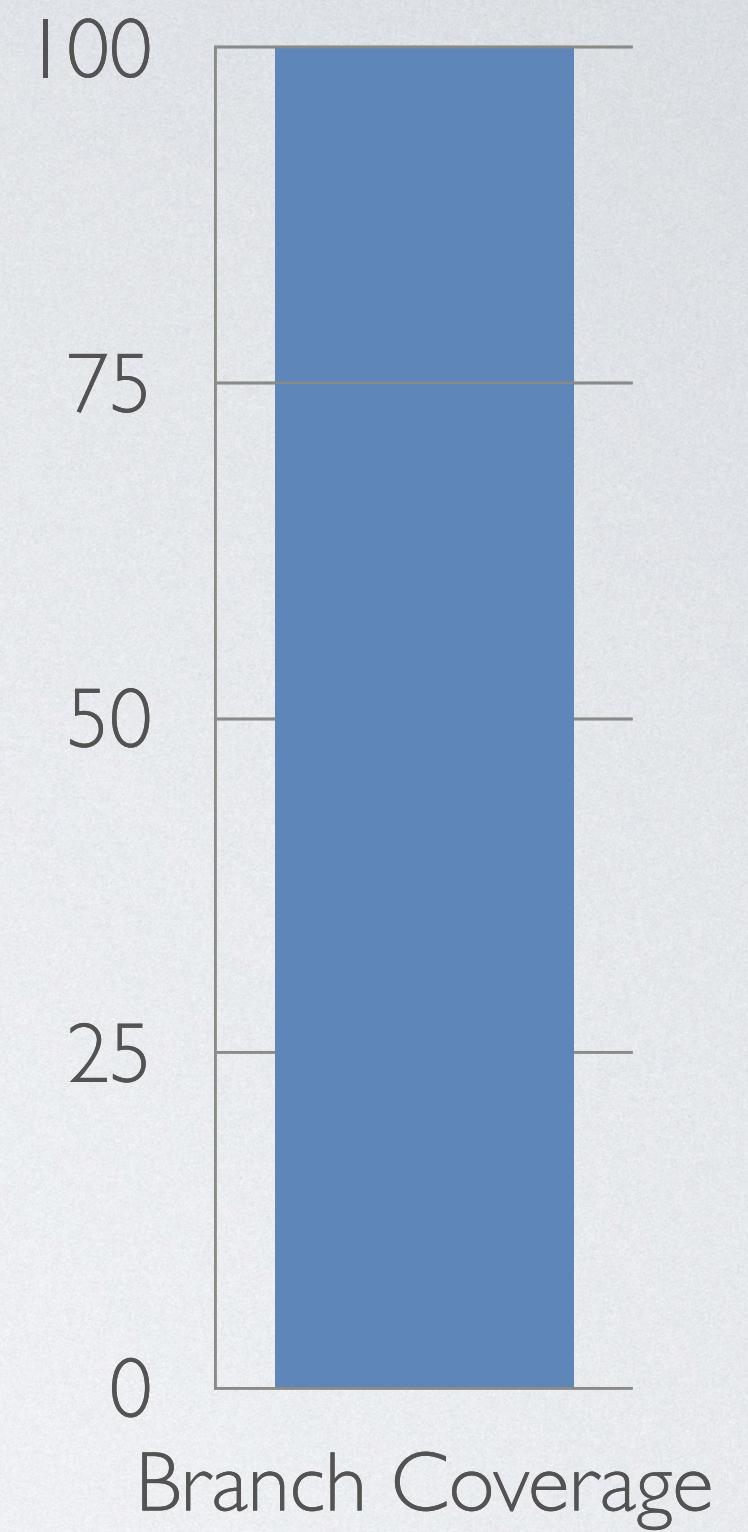
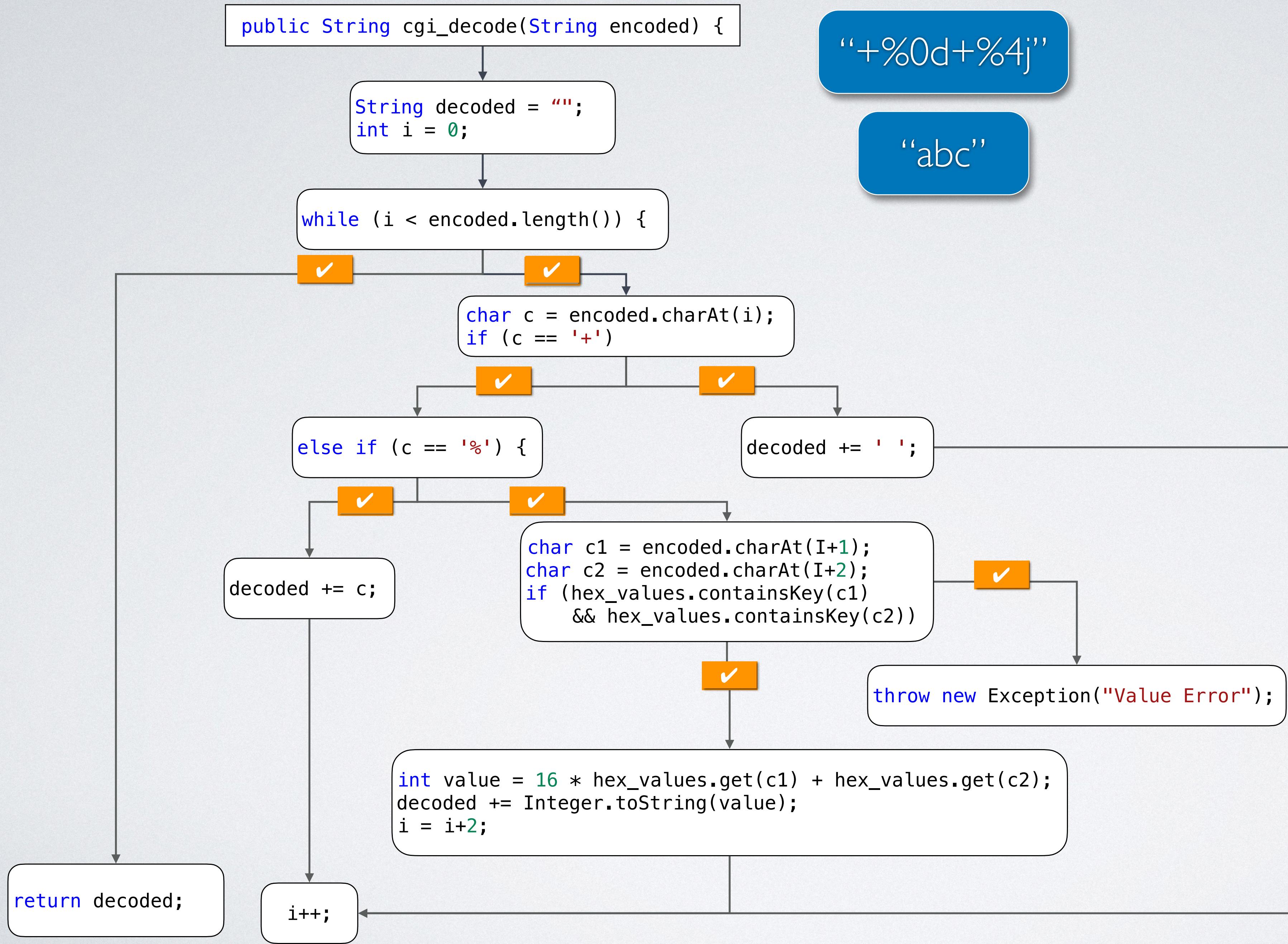




“+%0d+%4j”

“abc”





BRANCH COVERAGE

- Each branch in the program must be executed at least once
- Branch Coverage:
$$\frac{\# \text{ Executed Branches}}{\# \text{ Branches}}$$
- Subsumes statement coverage
- Most used structural criterion in practice
 - Complementary *fault-based* criteria: mutation score

SEARCH-BASED SOFTWARE TESTING

Search-Based Software Testing (SBST) is the application of optimisation search techniques, e.g., Genetic Algorithms, to solve problems in software testing, e.g., test generation.

Stress Testing

L. C. Briand, Y. Labiche, and M. Shousha, "Stress testing real-time systems with genetic algorithms". GECCO 2005.

Combinatorial Interaction Testing

M.B. Cohen, M.B. Dwyer and J. Shi, "Interaction testing of highly-configurable systems in the presence of constraints", ISSTA 2007.

GUI Testing

X.Yuan, M.B. Cohen and A.M. Memon, GUI Interaction Testing: Incorporating Event Context, IEEE Trans on Software Engineering, 2010.

Mobile App Testing

K. Mao, Y. Jia, M. Harman, "Sapienz: Multi-objective Automated Testing for Android Applications". ISSTA 2016.

SBST TECHNIQUES

Local Search

Hill climbing simple programs
Alternating Variable Method

Evolutionary Algorithms

Genetic Algorithms
Multi-Objective Optimisation

Hybrid Approaches

Interactive GA
Memetic Algorithms
Search + DSE

The Test Oracle Problem

SEARCH-BASED TEST GENERATION

Random Search

Input

SEARCH-BASED TEST GENERATION

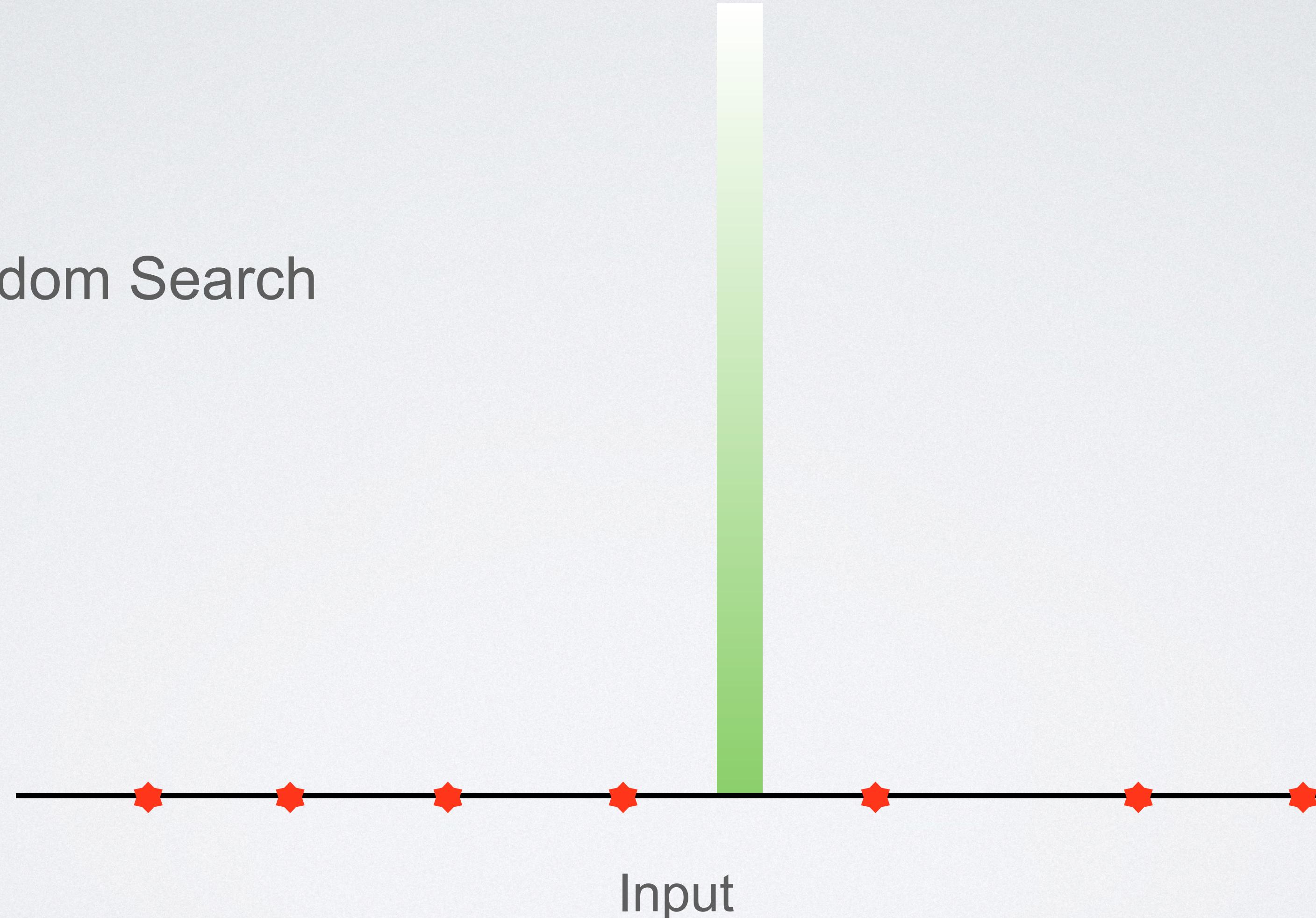
Random Search



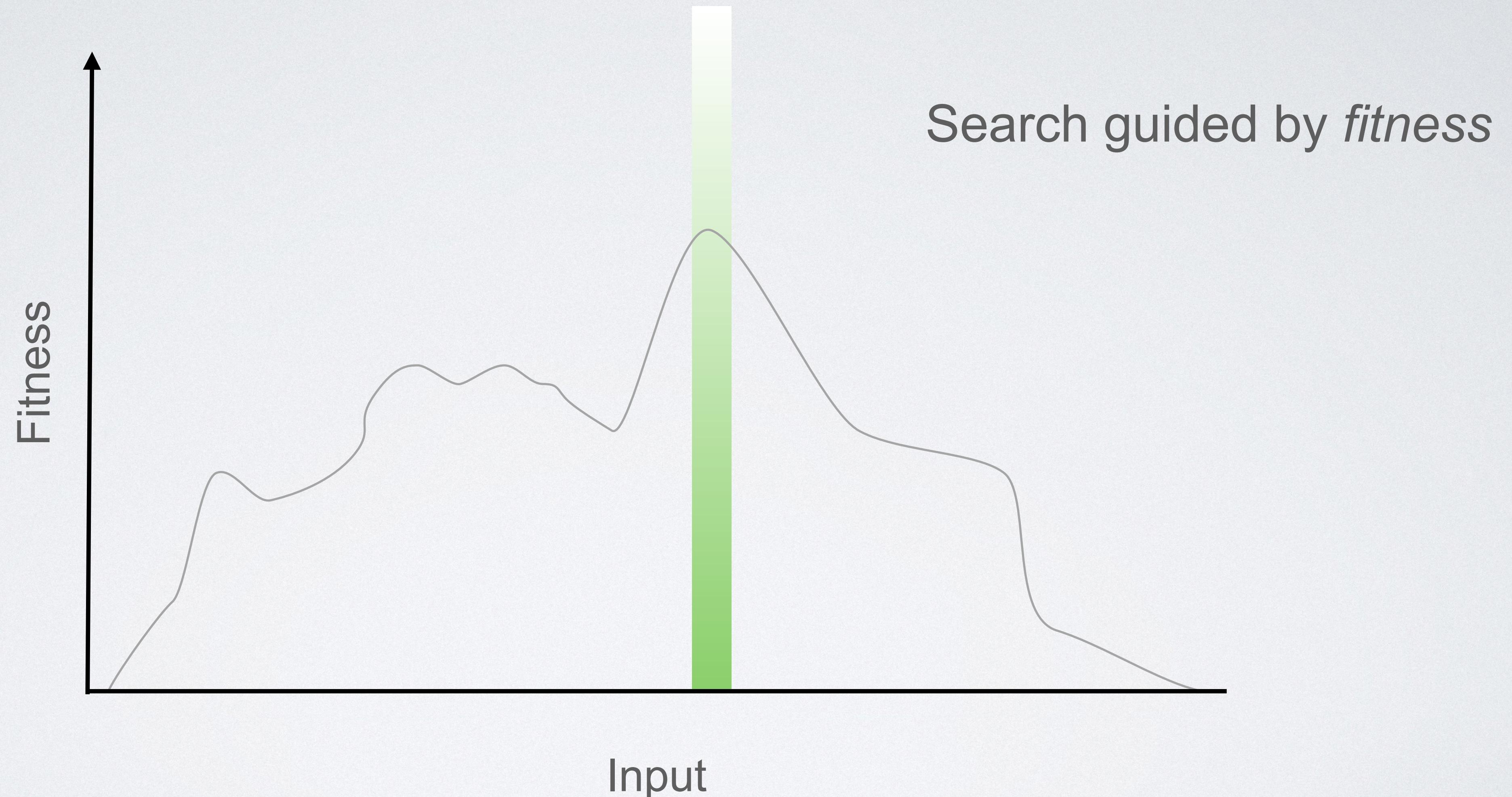
Input

SEARCH-BASED TEST GENERATION

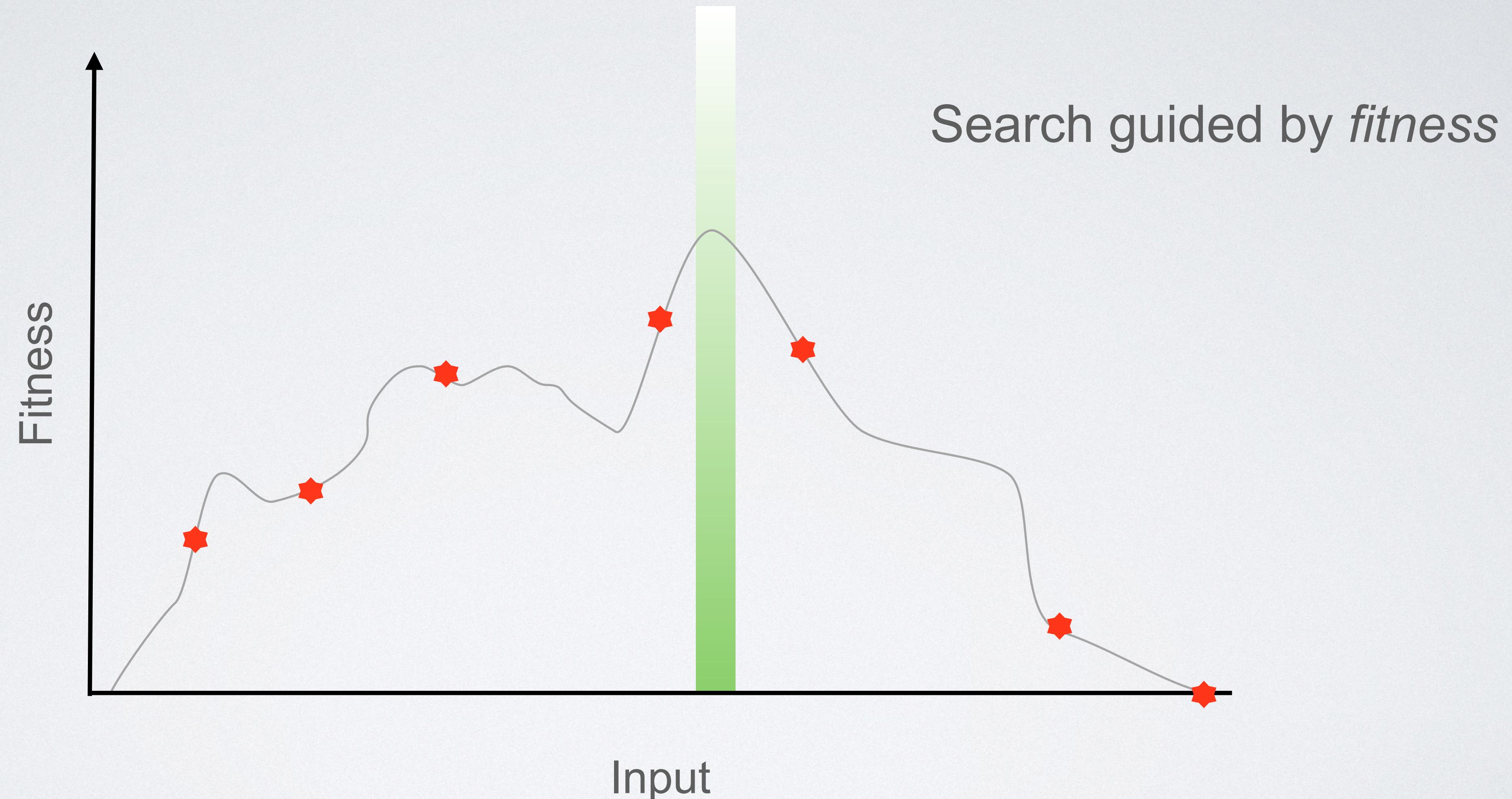
Random Search



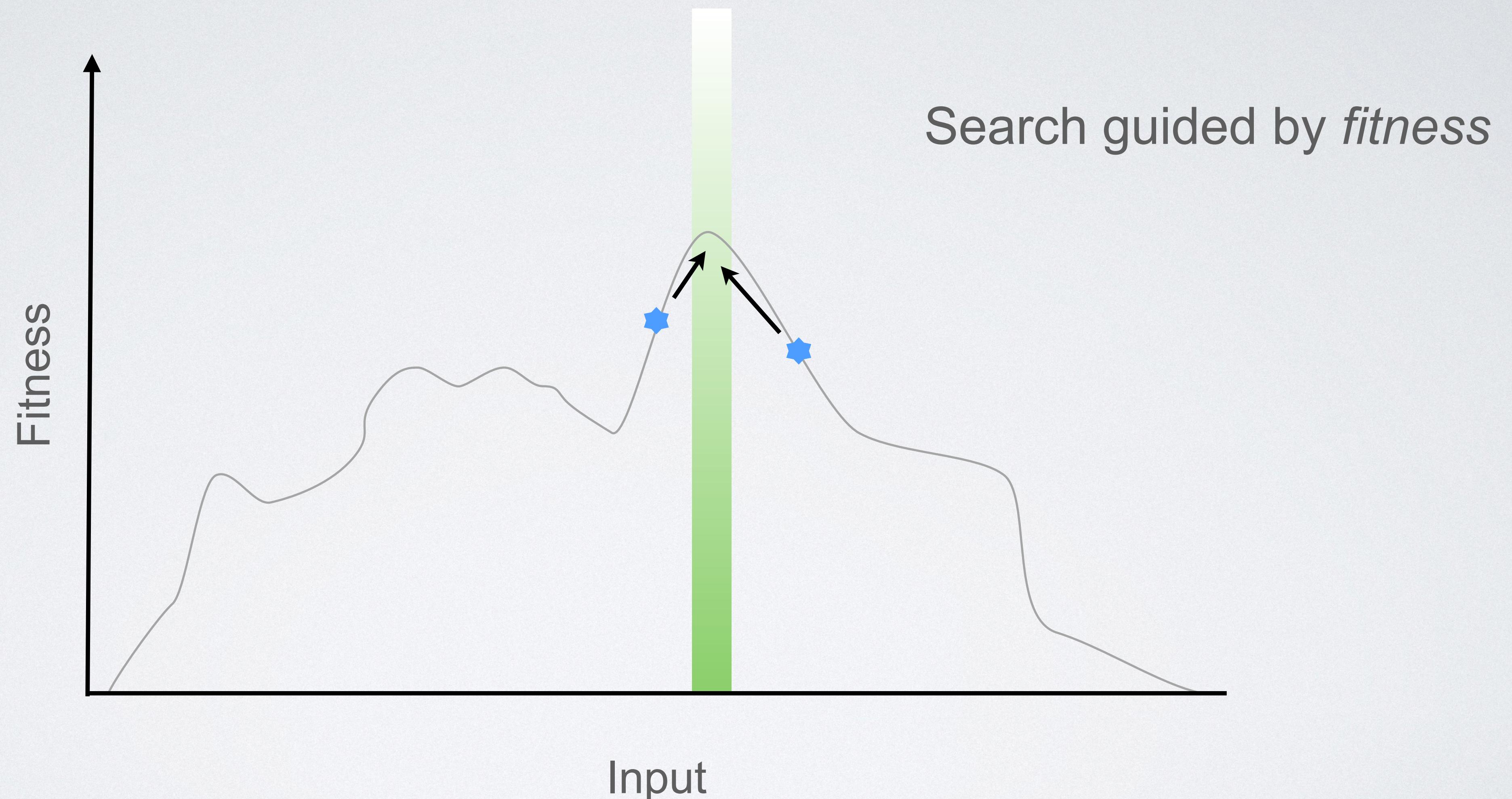
SEARCH-BASED TEST GENERATION



SEARCH-BASED TEST GENERATION

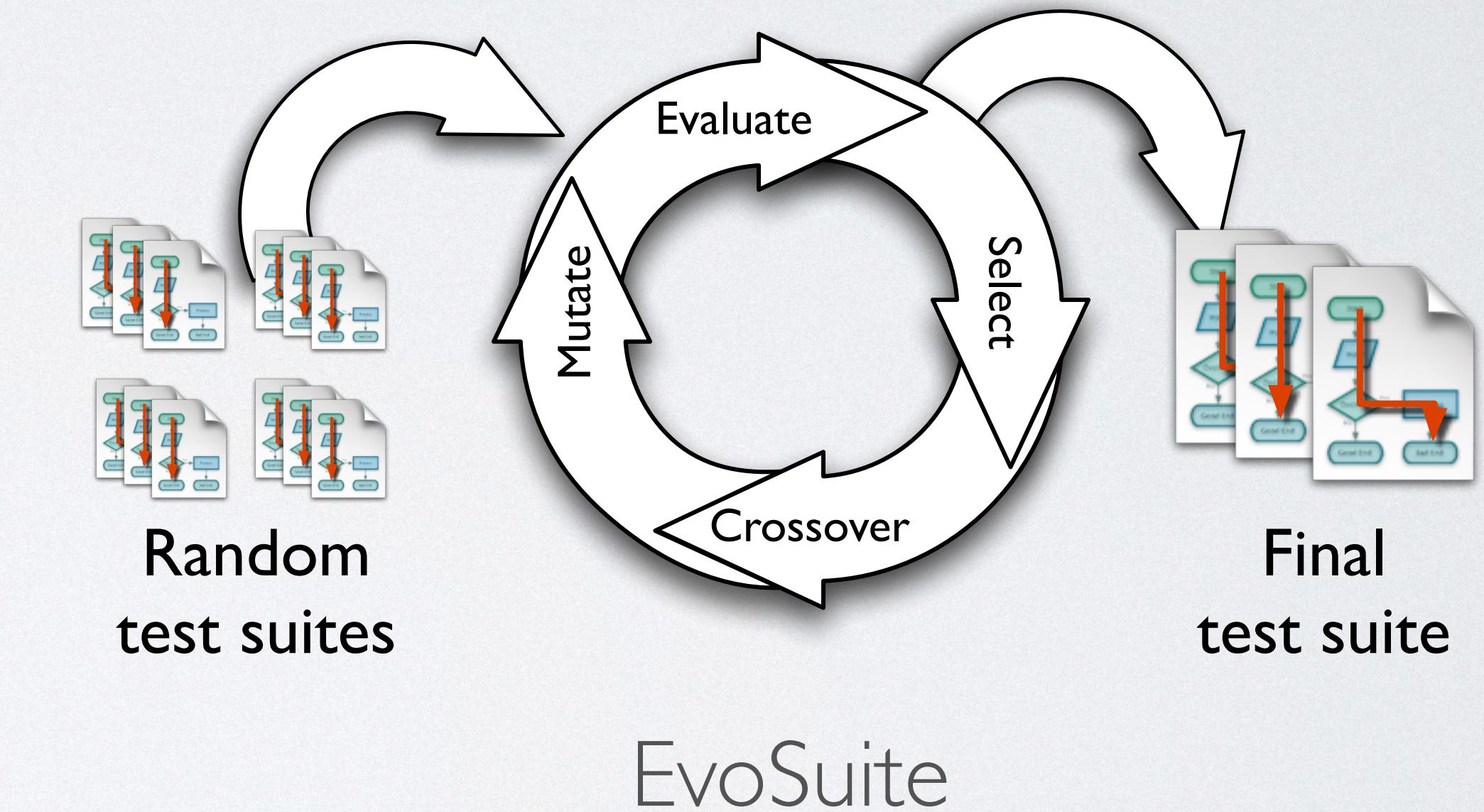
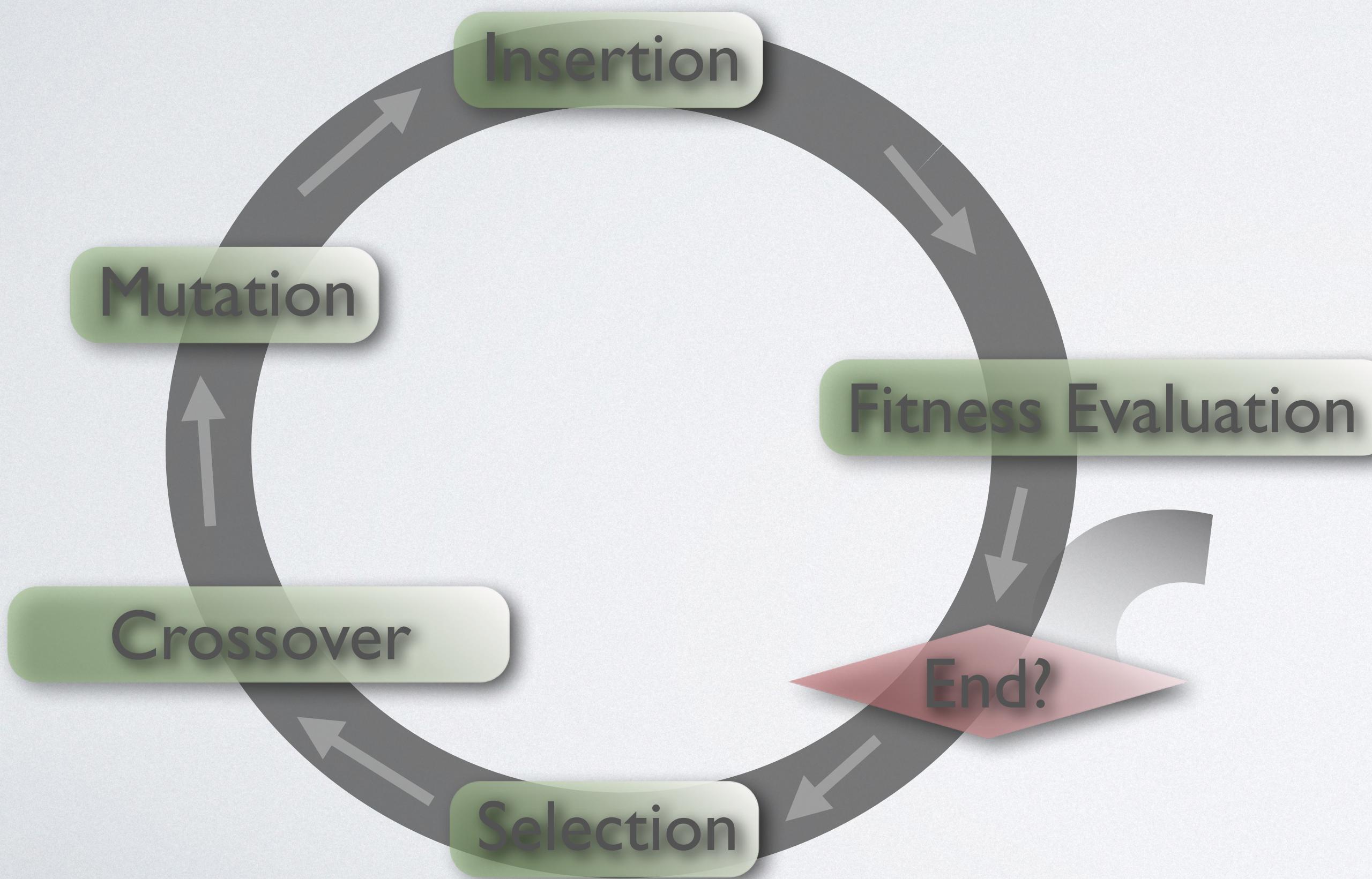


SEARCH-BASED TEST GENERATION



GENETIC ALGORITHM

Search algorithms inspired by the theory of evolution



TESTS AS INDIVIDUALS

An individual in the search is a test suite, i.e., a collection of unit tests:

```
@Test  
public void test()  
{  
    // set up  
  
    // execute code  
  
    // assertions  
}
```

TESTS AS INDIVIDUALS

At first, each test contains a randomly generated sequence of statements

@Test

public void test()

{

int var0 = 10

YearMonthDay var1 = new YearMonthDay(var0);

TimeOfDay var2 = new TimeOfDay();

DateTime var3 = var1.toDateTime(var2);

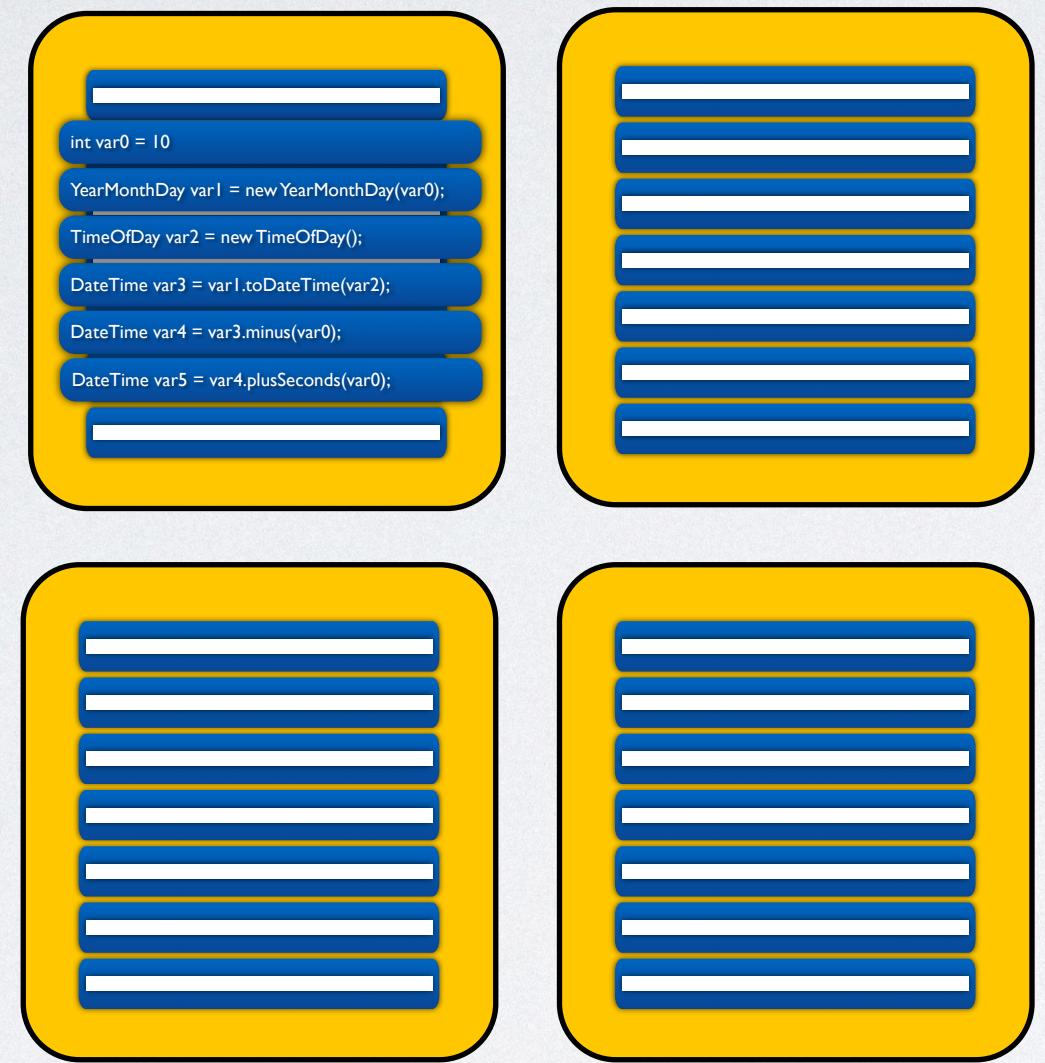
DateTime var4 = var3.minus(var0);

DateTime var5 = var4.plusSeconds(var0);

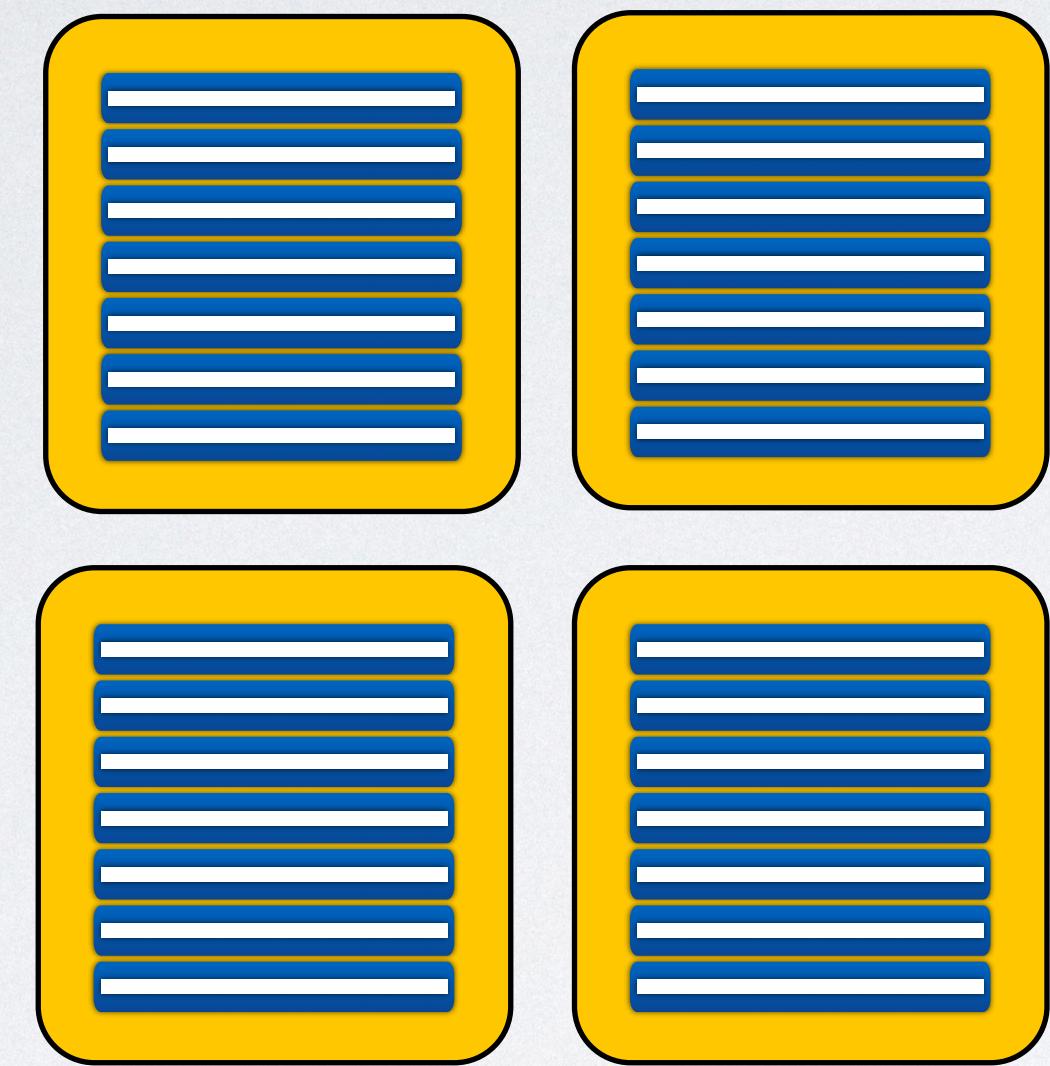
}

Assertions are not generated at this stage, they are added as a post-processing step.

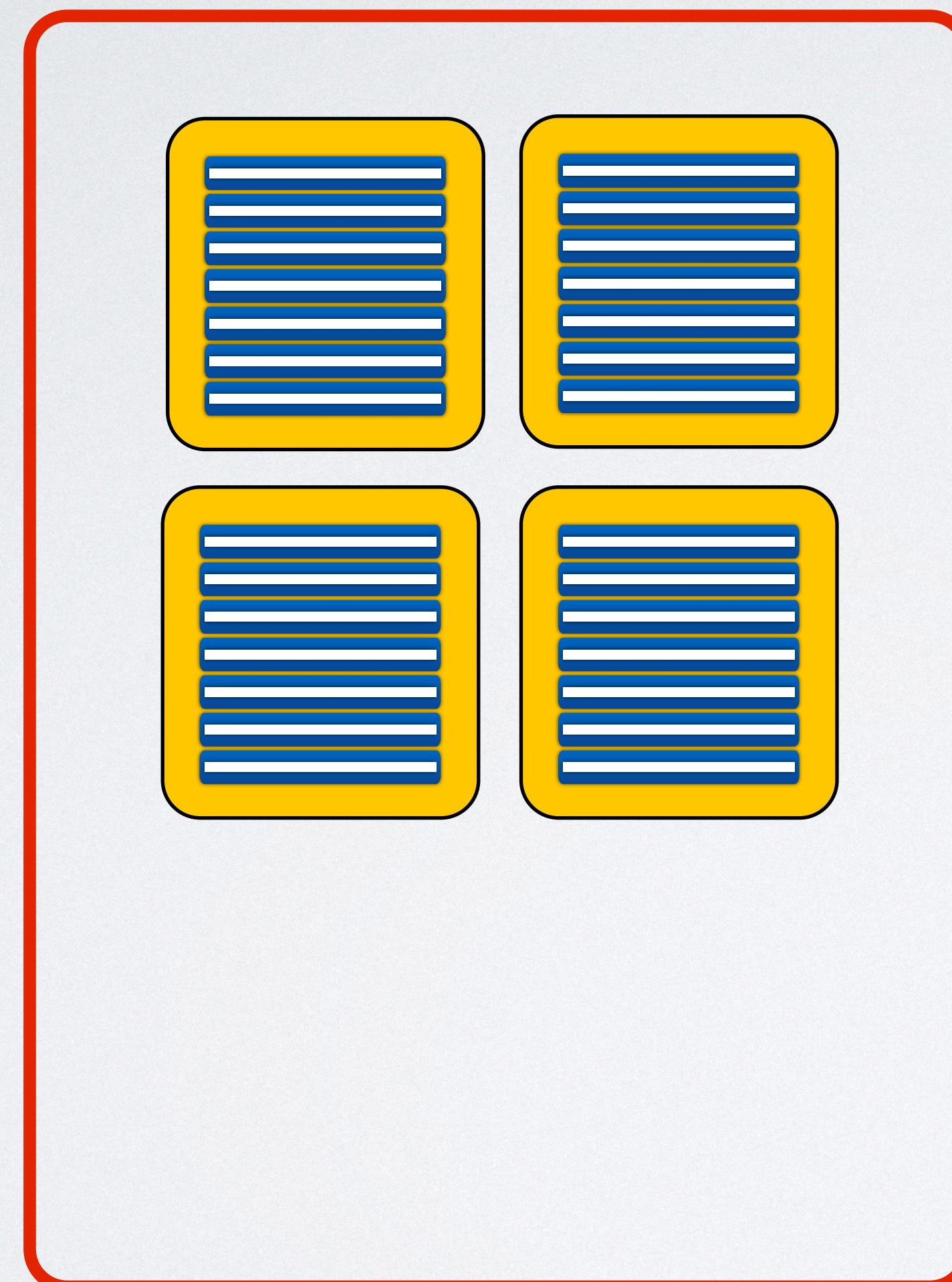
TEST SUITE GENERATION



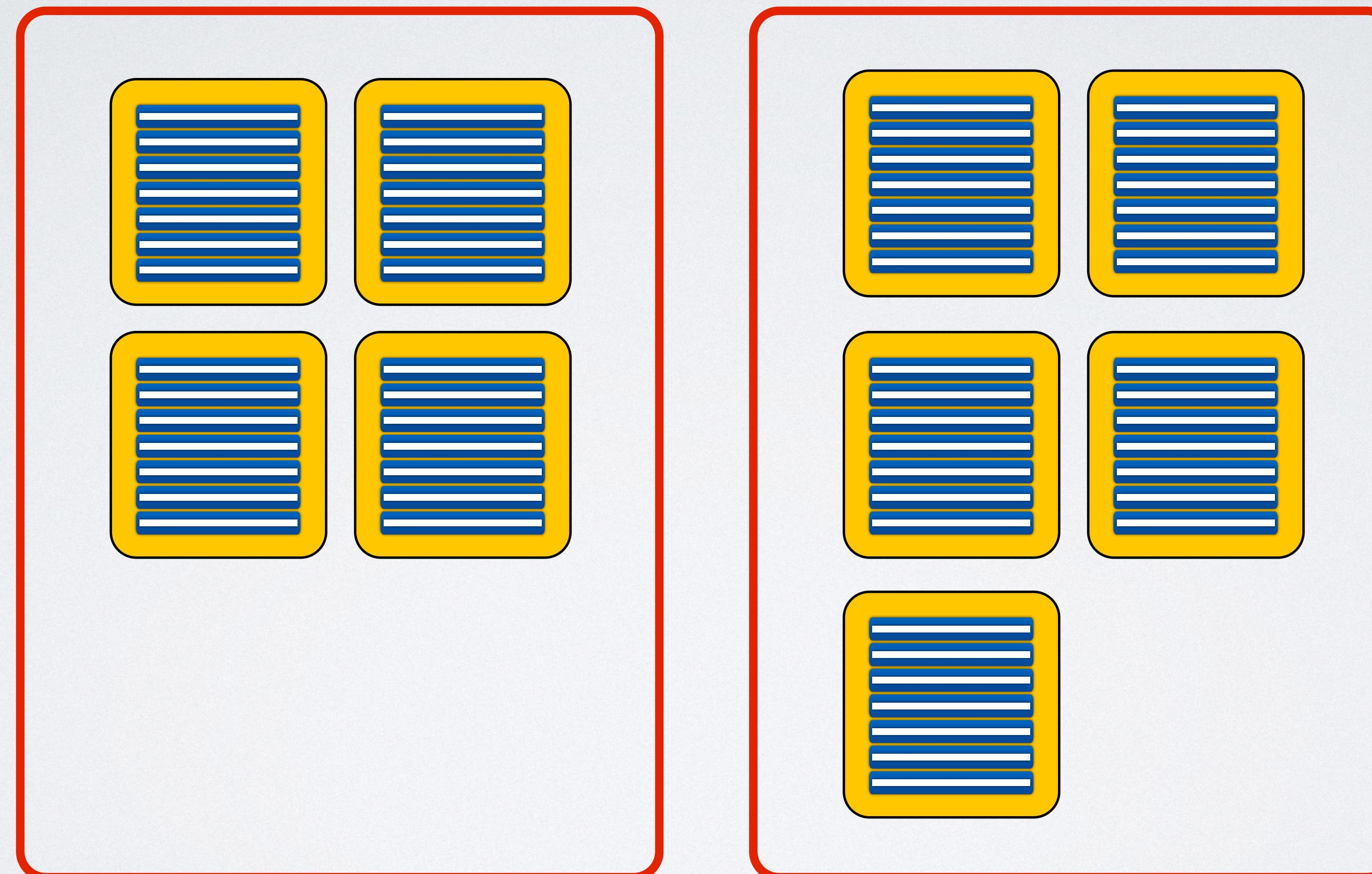
TEST SUITE GENERATION



TEST SUITE GENERATION



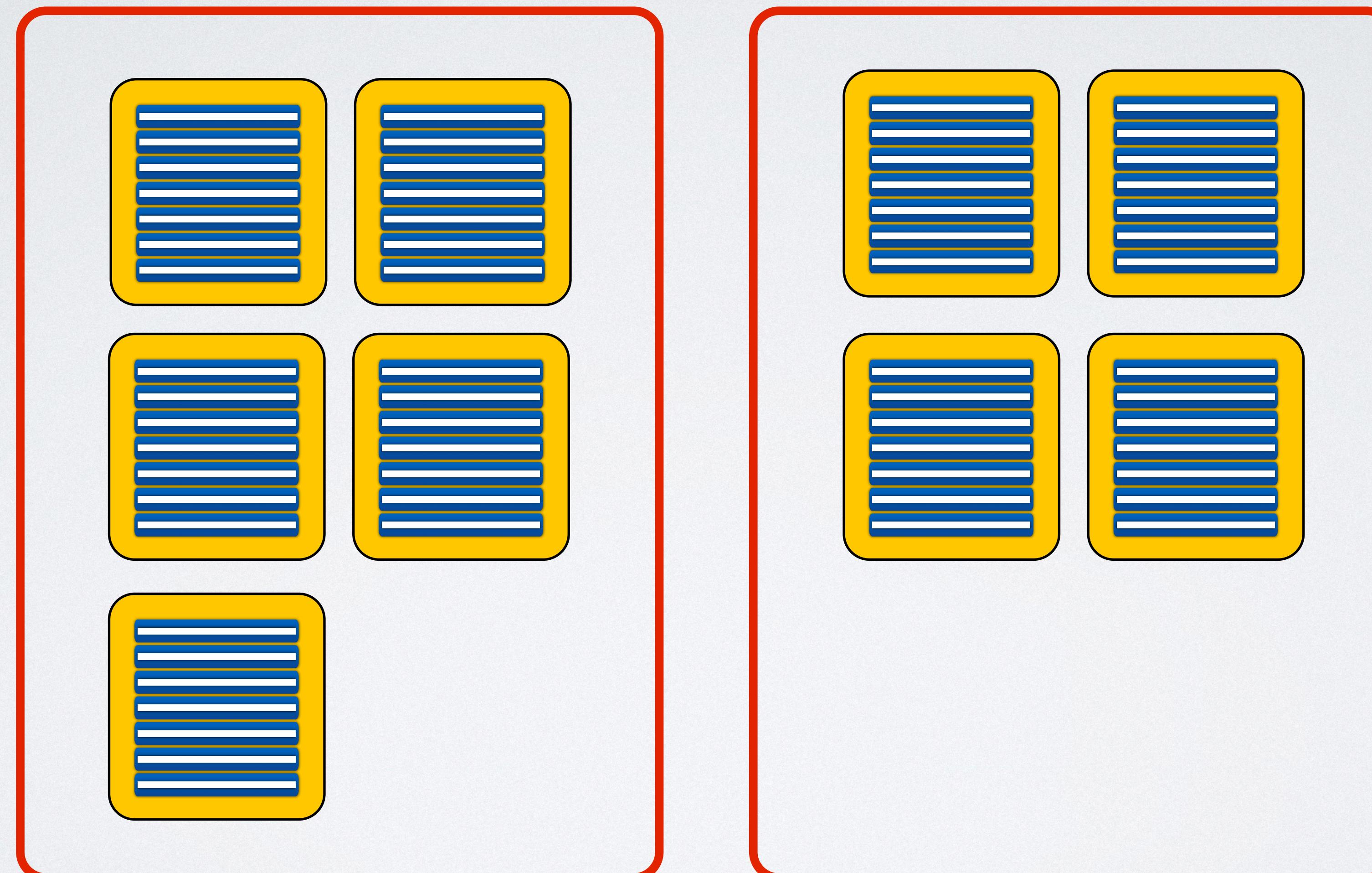
TEST SUITE GENERATION



The size of the initial population is configurable

GO-BX-KT

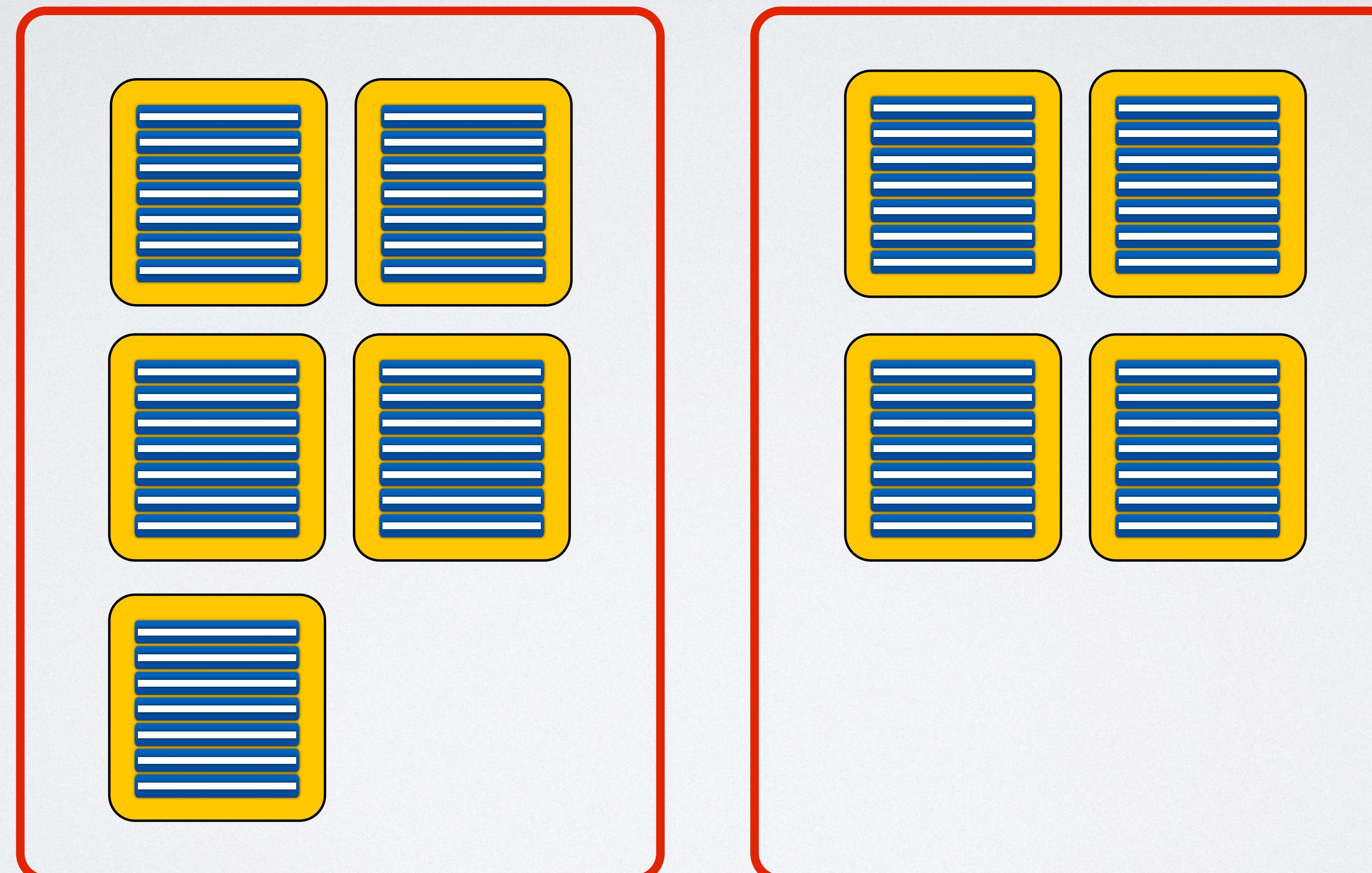
Crossover



Crossover happens with a certain probability

GO-BX-KT

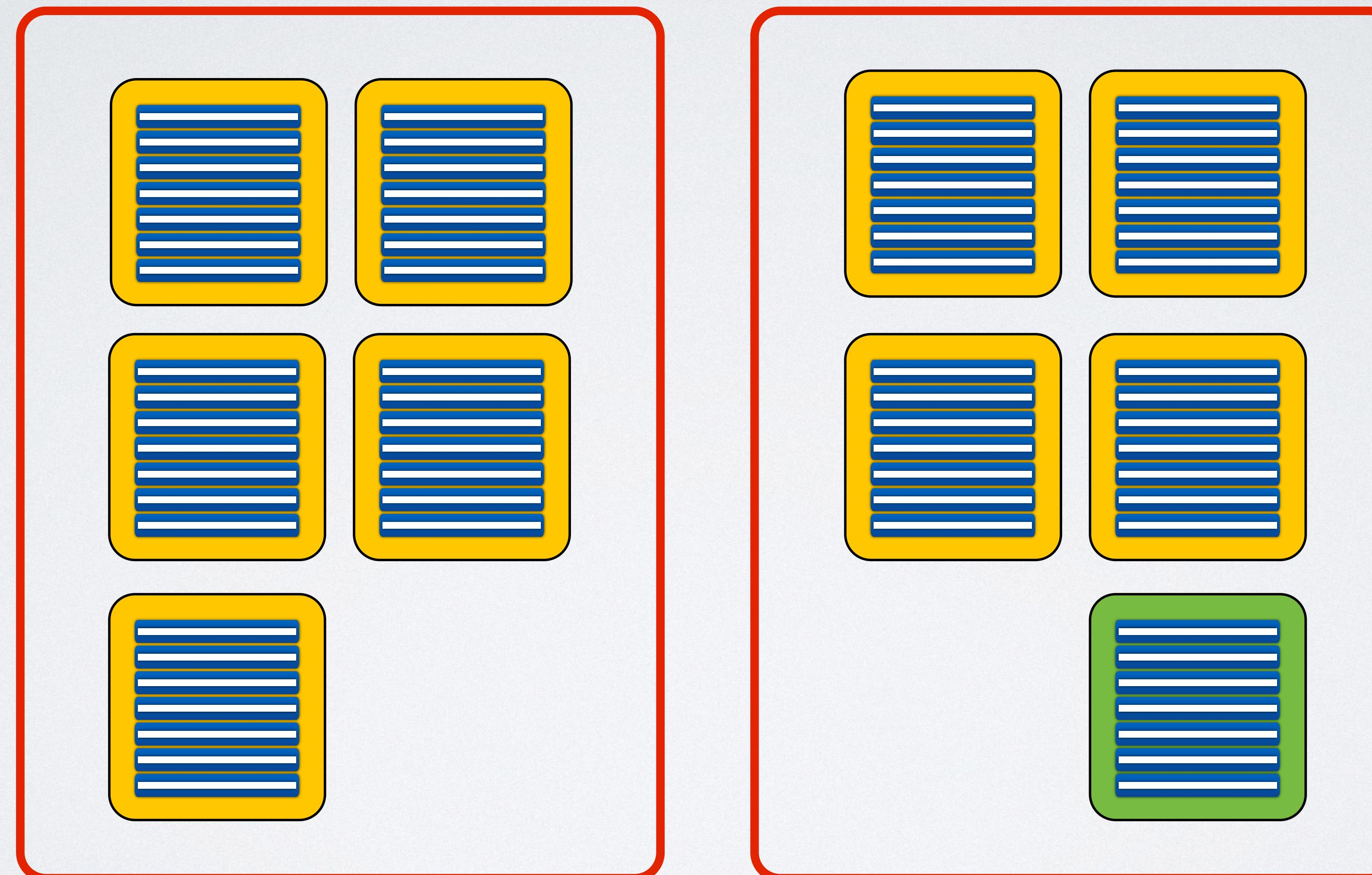
MUTATION



Mutation happens with a certain probability

GO-BX-KT

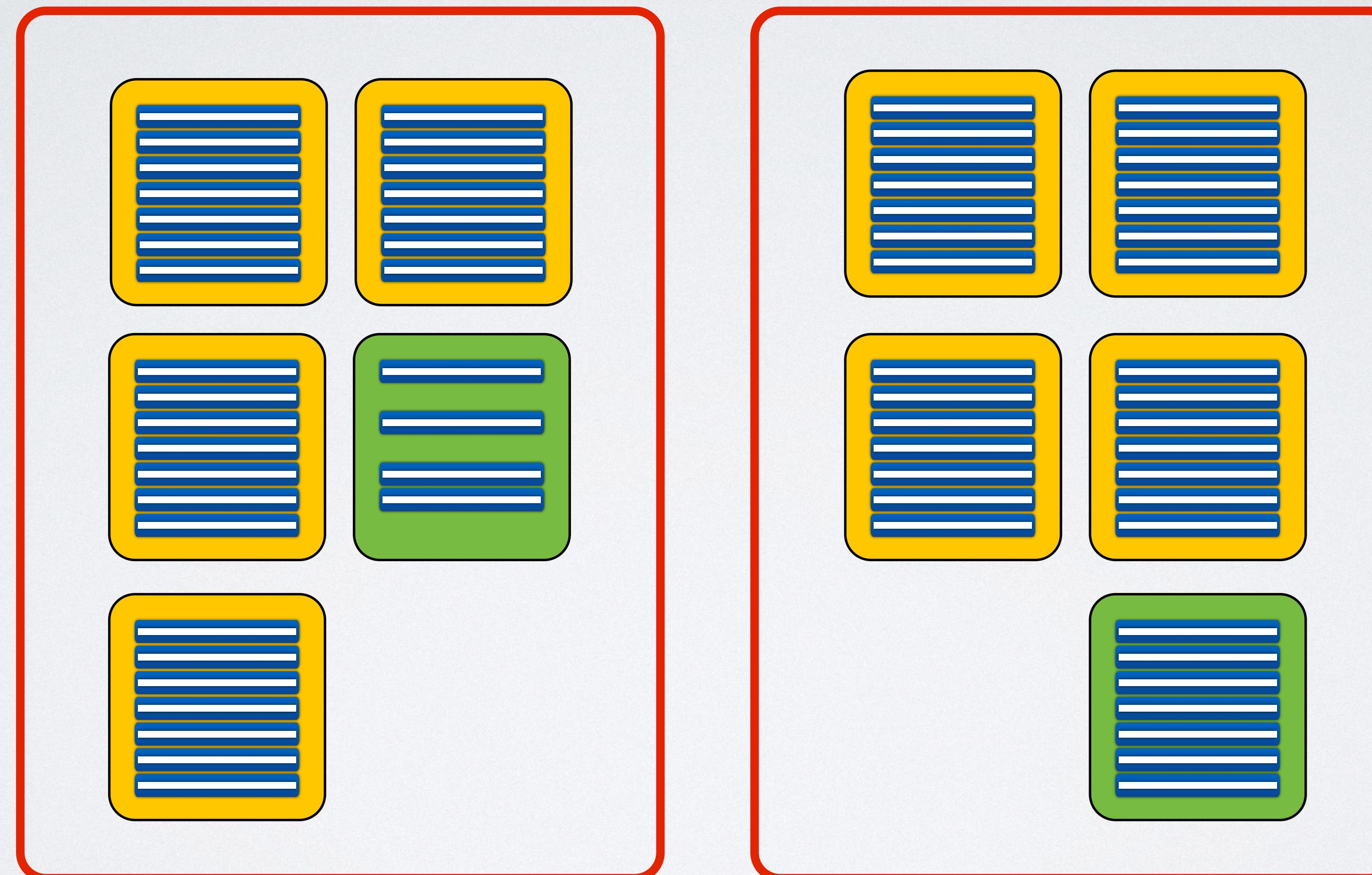
MUTATION



Mutation happens with a certain probability

GO-BX-KT

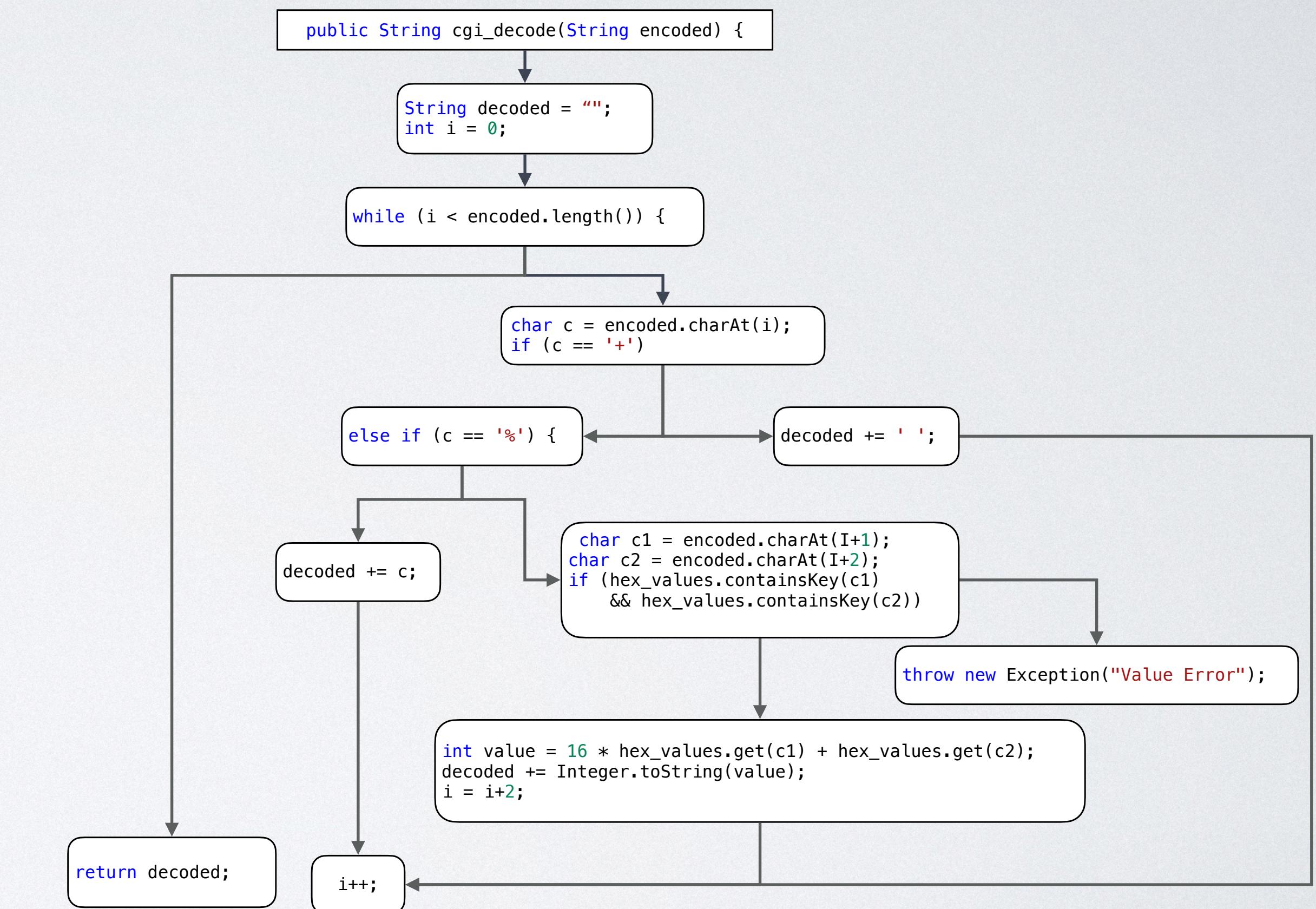
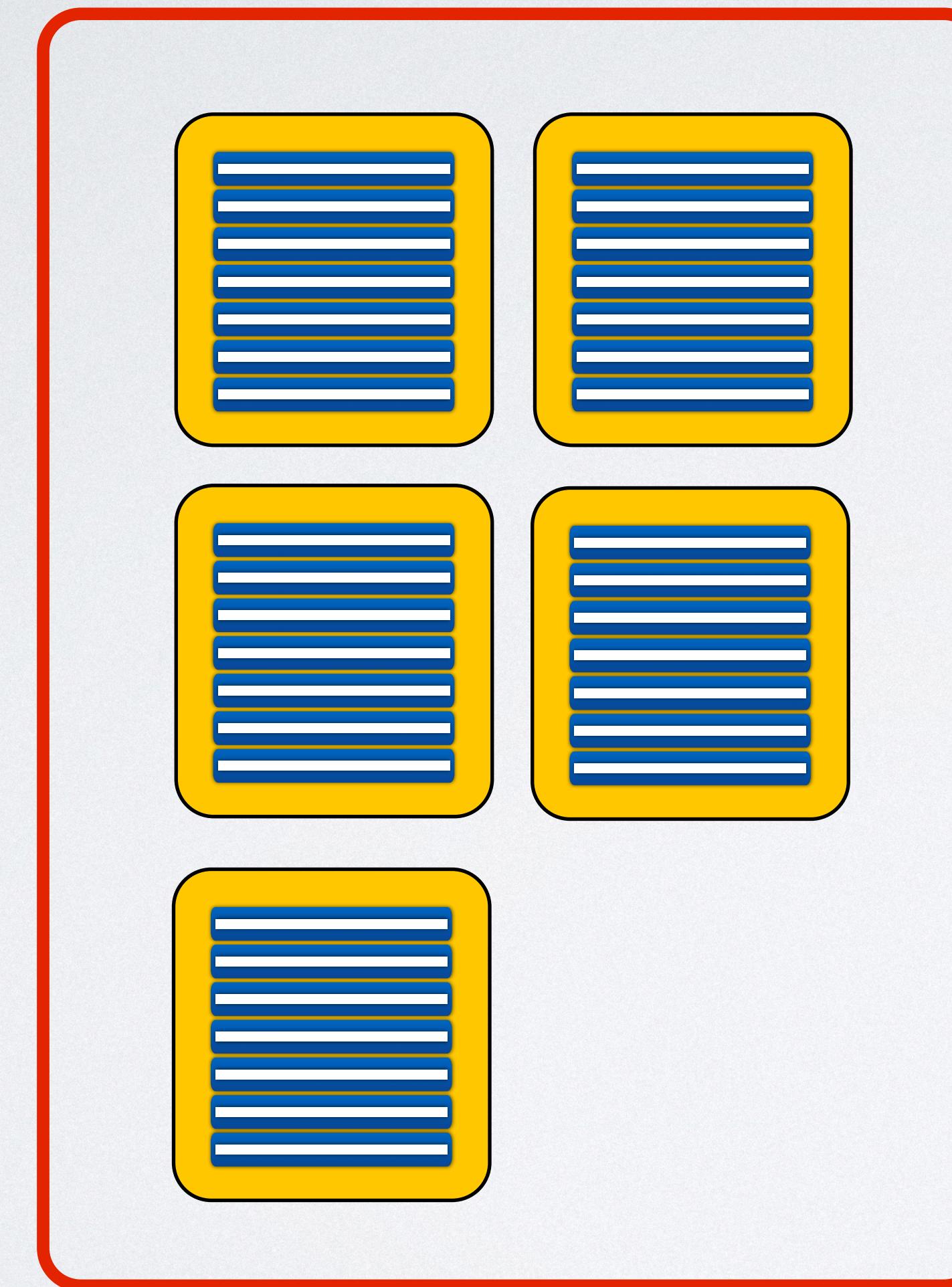
MUTATION



Mutation happens with a certain probability

GO-BX-KT

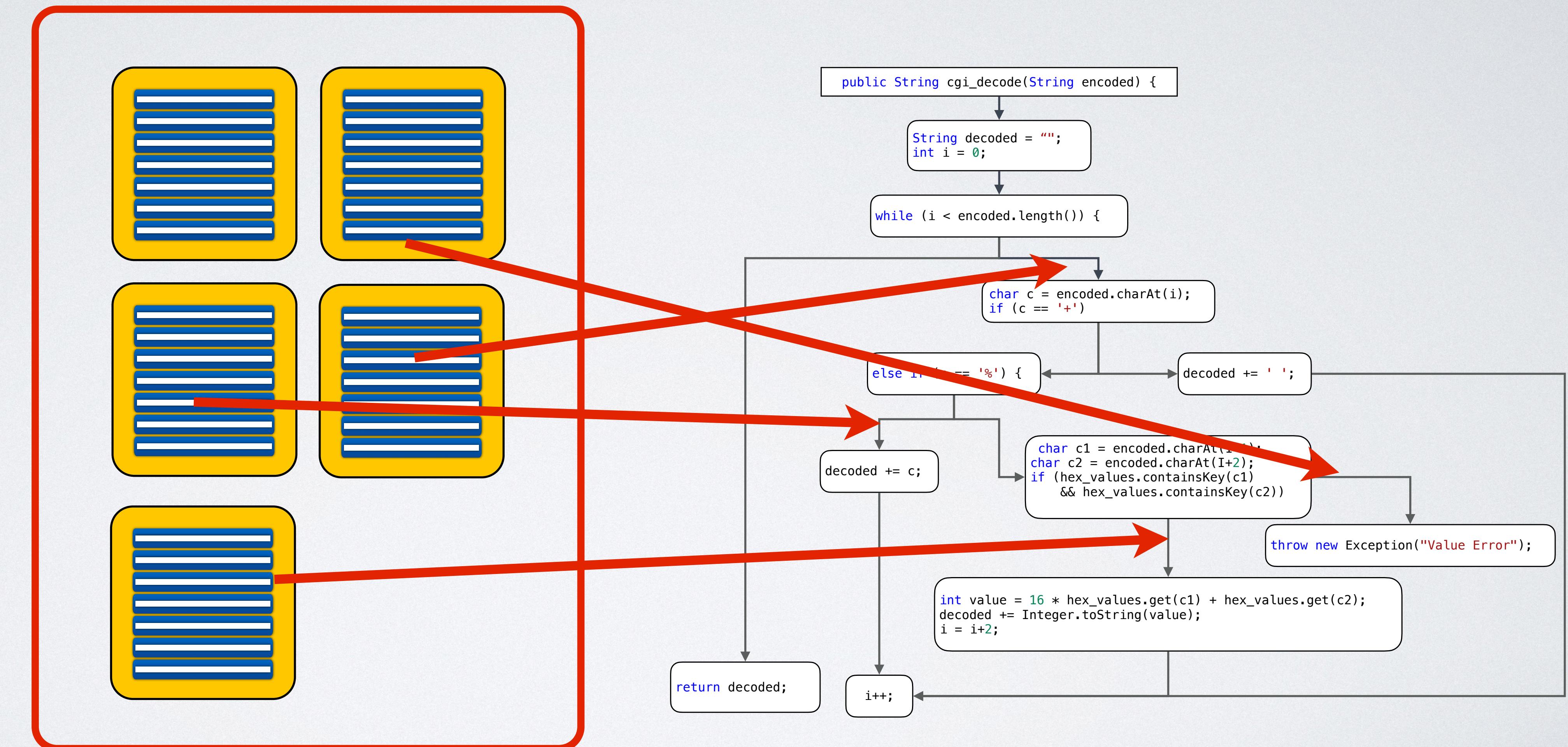
FITNESS EVALUATION



Fitness is computed using *branch distance* and *approach level*

GO-BX-KT

FITNESS EVALUATION



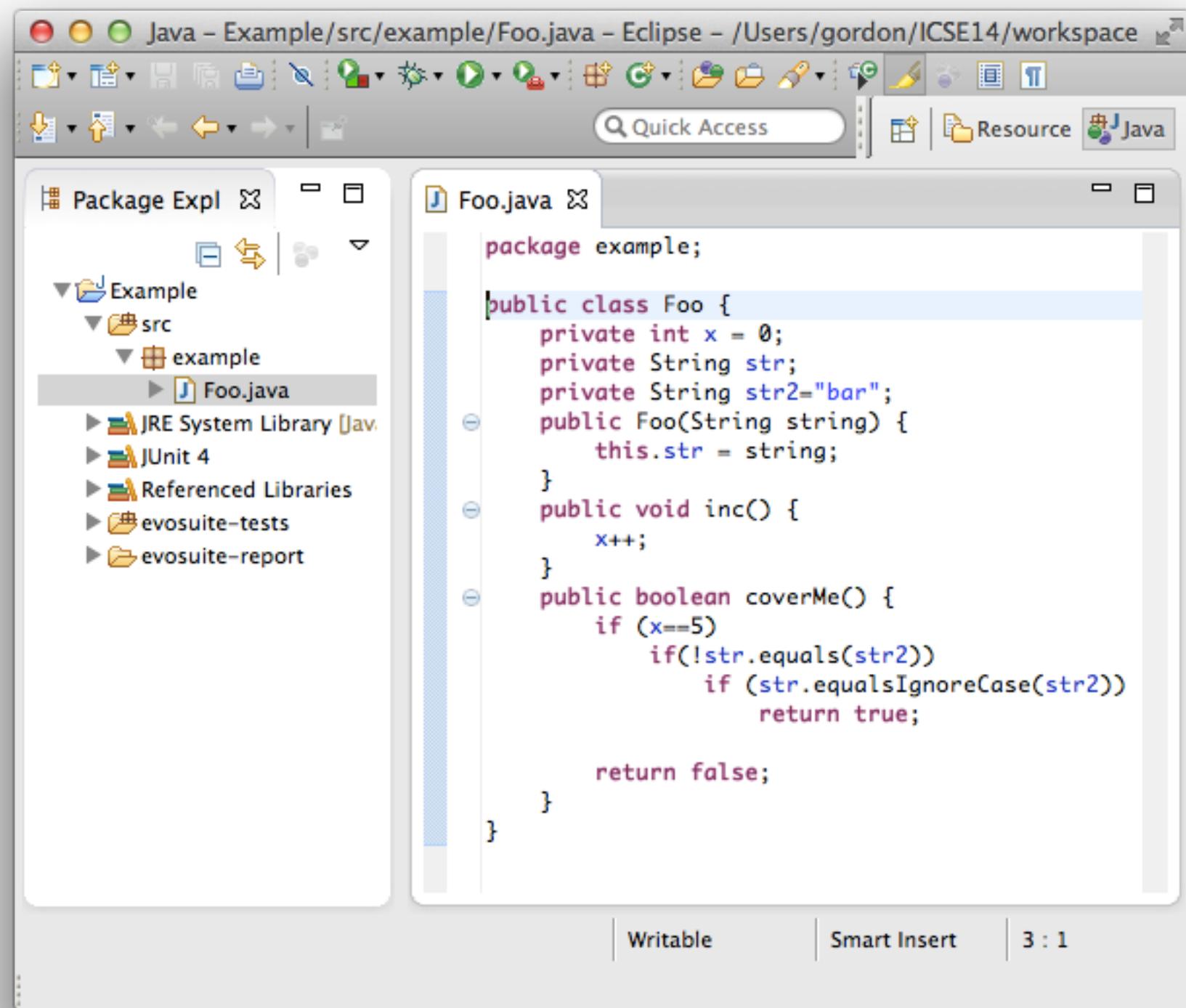
Fitness is computed using *branch distance* and *approach level*

GO-BX-KT

TEST ORACLES

- Assertions are introduced in resulting tests
- These assertions reflect **implemented** behaviour
- The algorithm knows nothing about **intended** behaviour
- **Regression test suite:** All generated tests will pass

SOFTWARE DEVELOPMENT USE CASE



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Java - Example/src/example/Foo.java - Eclipse - /Users/gordon/ICSE14/workspace
- Toolbar:** Standard Eclipse toolbar with various icons for file operations, search, and preferences.
- Quick Access Bar:** Contains a search field and links to Resource and Java perspectives.
- Package Explorer View:** Shows a project structure with a single package named "Example". Inside "Example", there is a "src" folder containing an "example" folder which contains a "Foo.java" file. Other items in "src" include "JRE System Library [Java]", "JUnit 4", "Referenced Libraries", "evoSuite-tests", and "evoSuite-report".
- Editor View:** Displays the source code of "Foo.java". The code defines a class "Foo" with a private int "x", a private String "str", and a private String "str2" initialized to "bar". It has a constructor "public Foo(String string)" that sets "str" to "string". It also has a method "public void inc()" that increments "x". A third method, "public boolean coverMe()", checks if "x" is 5 and if "str" is not equal to "str2" (ignoring case). If both conditions are met, it returns true; otherwise, it returns false.
- Status Bar:** Shows "Writable", "Smart Insert", and a character count of "3 : 1".

Source code

SOFTWARE DEVELOPMENT USE CASE

The diagram illustrates the software development use case for generating test cases. It shows two Eclipse IDE windows side-by-side.

Left Window (Source code):

- Shows the Java file `Foo.java` in the `src/example` directory.
- Contains the following code:

```
package example;

public class Foo {
    private int x = 0;
    private String str;
    private String str2="bar";
    public Foo(String string) {
        this.str = string;
    }
    public void inc() {
        x++;
    }
    public boolean coverMe() {
        if (x==5)
            if(!str.equals(str2))
                if (str.equalsIgnoreCase(str2))
                    return true;
        return false;
    }
}
```

Right Window (Test cases optimised for branch coverage):

- Shows the Java file `FooEvoSuiteTest.java` in the `evoSuite-tests/example` directory.
- Contains the following code:

```
* This file was automatically generated by EvoS

package example;

import org.junit.Test;

public class FooEvoSuiteTest {

    @Test
    public void test0() throws Throwable {
        Foo foo0 = new Foo("bar");
        foo0.inc();
        foo0.inc();
        foo0.inc();
        foo0.inc();
        boolean boolean0 = foo0.coverMe();
        assertEquals(false, boolean0);
    }

    @Test
    public void test1() throws Throwable {
    }
}
```

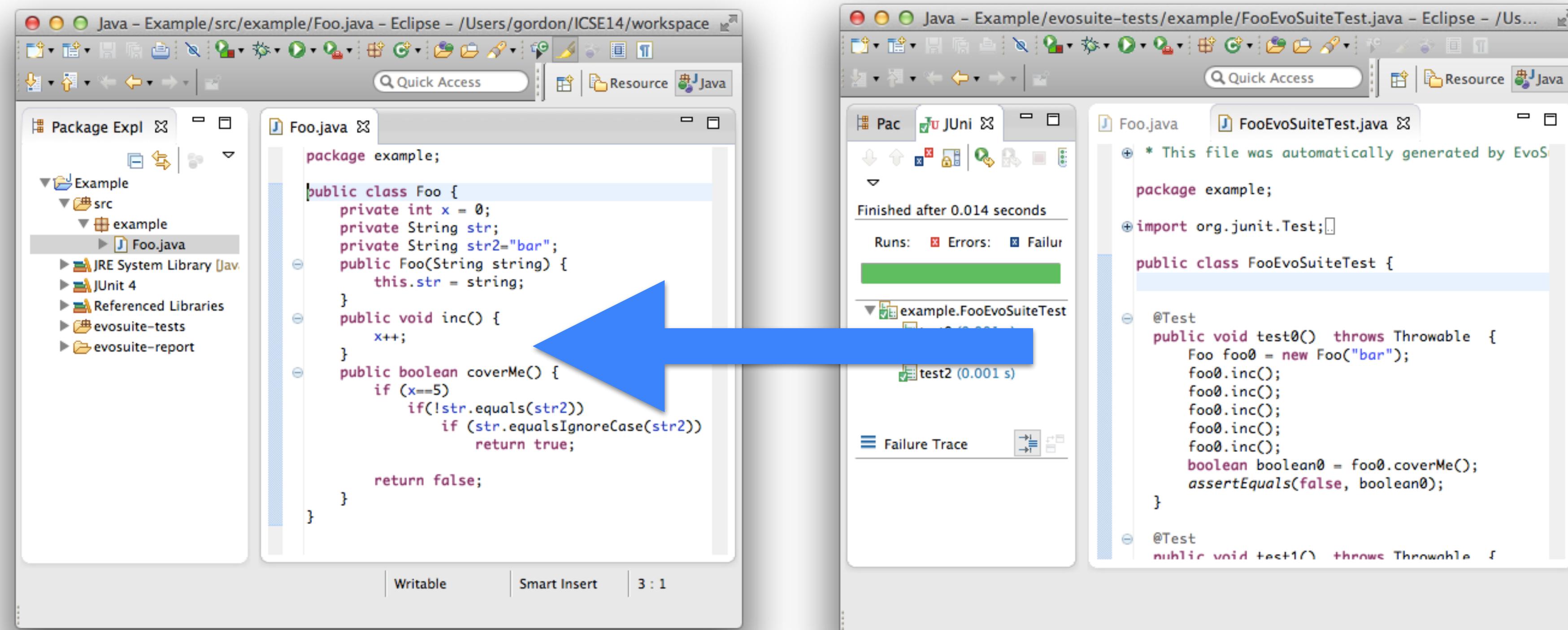
Source code

Test cases optimised for branch coverage

Automated generation

GO-BX-KT

SOFTWARE DEVELOPMENT USE CASE



Source code

Test cases optimised for branch coverage

Automated generation

GO-BX-KT

ADVANTAGES

A word cloud centered around the word "coverage". The words are arranged in a roughly circular pattern around the central word. The colors of the words vary, including shades of blue, green, purple, and red. The words include: consistency, corner, less, tests, better design, present, cases, stop, regression, effective, cost, bug, test, time, manual, efficiency, design, future, conferences, risk, fast, speed, feesback, better, failproof, framework, inspiration, toatestnewcases, you, get, to, present.

DISADVANTAGES

expected output honesty
dependencies-and-datatypes control
can't truth false-pasd cost
oracles negatives same
time falerun money
maintenance
test repair test usefulness variation
tests boring understanding new
test oracle false-fails tell sense common
became-uncompilavle-with-complex
is-this-a-meaningful-test false negatives
want
hoping
hardware
pre-determined
expected changes
missed-bugs
lack of oracle

Do Automatically Generated Unit Tests Find Real Faults? An Empirical Study of Effectiveness and Challenges

Sina Shamshiri*, René Just†, José Miguel Rojas*, Gordon Fraser*, Phil McMinn* and Andrea Arcuri‡

*Department of Computer Science, University of Sheffield, UK

†Department of Computer Science & Engineering, University of Washington, Seattle, WA, USA

‡Scienta, Norway, and University of Luxembourg

*{sina.shamshiri, j.rojas, gordon.fraser, p.mcminn}@sheffield.ac.uk, †rjust@cs.washington.edu, ‡aa@scienta.no

Abstract—Rather than tediously writing unit tests manually, tools can be used to generate them automatically — sometimes

faults from open source projects [25]. We applied three state-of-the-art unit test generation tools for Java, RANDOOP [30],

Fault Detection (ASE 2015)

Empir Software Eng (2017) 22:852–893
DOI 10.1007/s10664-015-9424-2



**A detailed investigation of the effectiveness
of *whole* test suite generation**

José Miguel Rojas¹ · Mattia Vivanti² ·
Andrea Arcuri^{3,4} · Gordon Fraser¹

Code Coverage (EMSE 2017)

OPTIMISATIONS

- Seeding (Rojas et al., STVR 2015)
 - Static values, e.g., constants appearing in code or strings representing email addresses
 - Dynamic values
 - Existing tests
 - Parameter tuning (Arcuri and Fraser, ESE 2013)
 - Multiple coverage criteria at once (Rojas et al., SSBSE 2015)
 - Many objectives at once (Panichela et al., TSE 2017)
- Email addresses, such as `jsmith@example.org`, have two parts. The part before the `@` sign is the *local-part* of the address, often the `username` of the recipient (`jsmith`), and the part after the `@` sign is a *domain name* to which the email message will be sent (`example.org`).
- P. McMinn, M. Shahbaz and M. Stevenson. "Search-Based Test Input Generation for String Data Types Using the Results of Web Queries". ICST 2012.



- Search-based Whole Test Suite Generation of JUnit tests for Java programs
- Publicly available at <http://www.evosuite.org/>
 - Open-source on GitHub: <https://github.com/evosuite/evosuite>
- Different interfaces
 - Jar file for use from command line
 - Plug-ins for IntelliJ and Eclipse IDEs for use during software development
 - Plug-in for Maven for build integration and Plug-in for Jenkins for continuous integration

```
@Test(timeout = 4000)
public void test01() throws Throwable {
    LinkedList<String> linkedList0 = new LinkedList<String>();
    LinkedList<Object> linkedList1 =
        new LinkedList<Object>((Collection<?>) linkedList0);
    FixedOrderComparator fixedOrderComparator0 =
        new FixedOrderComparator((List) linkedList1);
    linkedList1.add((Object) linkedList1);
    // Undeclared exception!
    try {
        fixedOrderComparator0.compare(linkedList1, linkedList0);
        fail("Expecting exception: StackOverflowError");
    } catch(StackOverflowError e) {
        //
        // no message in exception (getMessage() returned null)
        //
    }
}
```

```
@Test
public void test002() throws Throwable {

    if (debug) { System.out.format("%n%s%n", "RegressionTest2.test002"); }

    java.lang.Object[] obj_array1 = new java.lang.Object[] { (short)10 };
    collections.comparators.FixedOrderComparator fixedOrderComparator2 =
        new collections.comparators.FixedOrderComparator(obj_array1);
    java.util.Comparator comparator3 = fixedOrderComparator2.reversed();
    java.util.Comparator comparator4 = comparator3.reversed();
    java.lang.Object[] obj_array6 = new java.lang.Object[] { (short)10 };
    collections.comparators.FixedOrderComparator fixedOrderComparator7 =
        new collections.comparators.FixedOrderComparator(obj_array6);
    java.util.Comparator comparator8 =
        comparator4.thenComparing((java.util.Comparator)fixedOrderComparator7);
    boolean b10 = fixedOrderComparator7.add((java.lang.Object)10L);
    int i11 = fixedOrderComparator7.getUnknownObjectBehavior();
    boolean b12 = fixedOrderComparator7.isLocked();
    fixedOrderComparator7.setUnknownObjectBehavior( 0 );
    ...
    ...
}
```



Randoop

GO-BX-KT

```

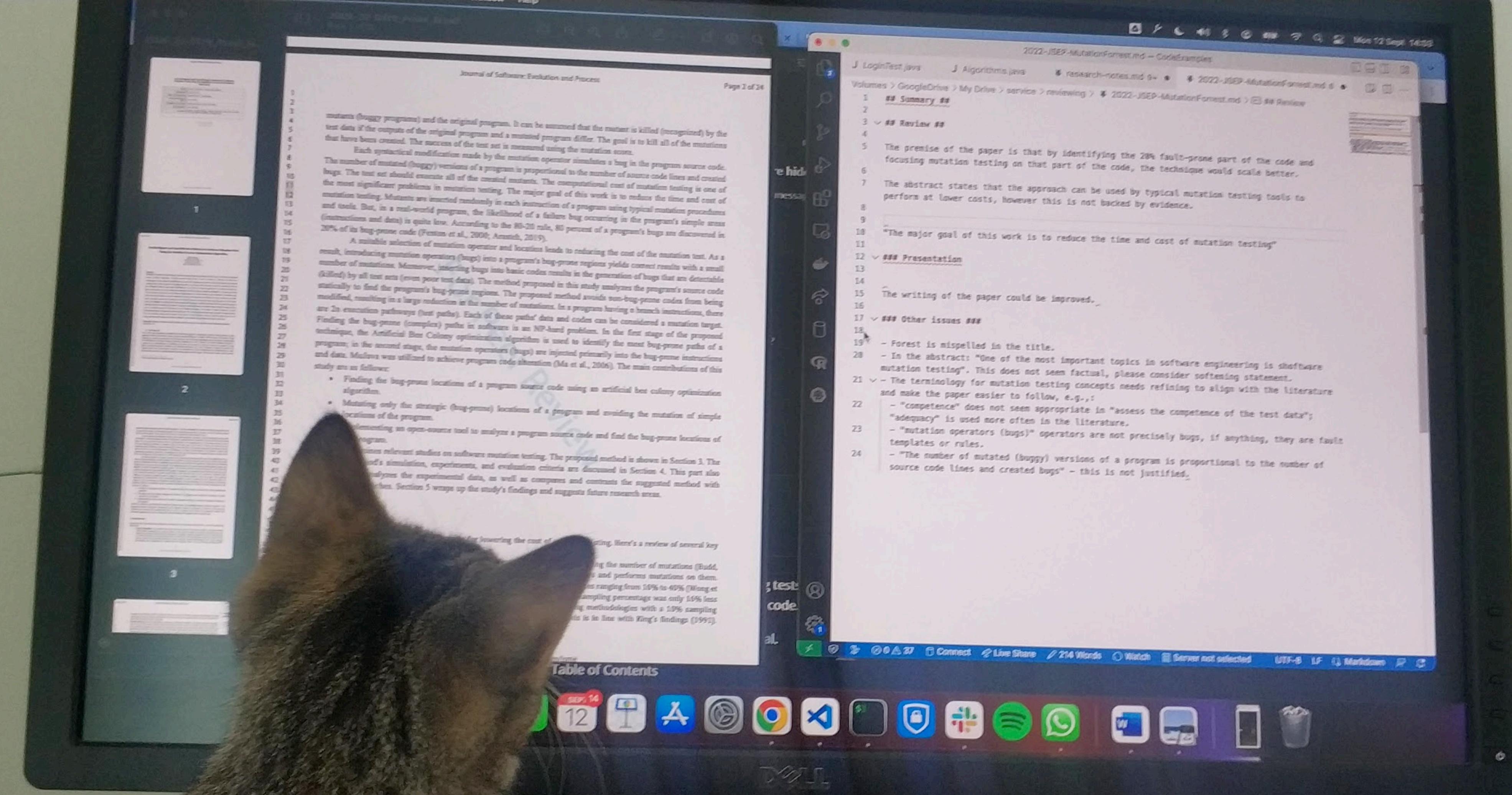
/**
 * Chromosome :
1)---->org.apache.commons.lang3.math.NumberUtils[]
2)---->min[[30823,57]],
3)---->toString[]
Covered Branches:[1, 113, 114, 115, 109, 111]
*/
@Test public void TestCase1() throws Throwable {
    NumberUtils clsUTNumberUtils=null;
    clsUTNumberUtils=new NumberUtils();
    short[] clsUTNumberUtilsP2P1=new short[]{30823,57};
    short clsUTNumberUtilsP2R=0;
    clsUTNumberUtilsP2R=NumberUtils.min(clsUTNumberUtilsP2P1);
    assertTrue(Arrays.equals(new short[]{30823,57},clsUTNumberUtilsP2P1));
    assertEquals(57,clsUTNumberUtilsP2R);
    String clsUTNumberUtilsP3R=null;
    clsUTNumberUtilsP3R=clsUTNumberUtils.toString();
    String clsUTNumberUtilsP3RP0P1=new
String("YfJ0ufsFnsYRrFHBpCyLgDWEmhiRPOGISMxFfCYjMOSisquxCuSHI0rrEBbqqwfBtBmhuiUWUSsKUAfloL1YPBYmyihySJwWEVETnqzBRd1RCMk");
    Double clsUTNumberUtilsP3RP0P2P100=-11.825526454314883D;
    Object clsUTNumberUtilsP3RP0P2P1=clsUTNumberUtilsP3RP0P2P100;
    String clsUTNumberUtilsP3RP0P2P200=new String(">?])$#qxd_ (;jiah+un");
    Object clsUTNumberUtilsP3RP0P2P2=clsUTNumberUtilsP3RP0P2P200;
    Short clsUTNumberUtilsP3RP0P2P300=1689;
    Object clsUTNumberUtilsP3RP0P2P3=clsUTNumberUtilsP3RP0P2P300;
    Object[] clsUTNumberUtilsP3RP0P2=new Object[]{clsUTNumberUtilsP3RP0P2P1,clsUTNumberUtilsP3RP0P2P2,clsUTNumberUtilsP3RP0P2P3};
    String clsUTNumberUtilsP3RP0R=null;
    clsUTNumberUtilsP3RP0R=String.format(clsUTNumberUtilsP3RP0P1,(Object[])clsUTNumberUtilsP3RP0P2);

    assertEquals("YfJ0ufsFnsYRrFHBpCyLgDWEmhiRPOGISMxFfCYjMOSisquxCuSHI0rrEBbqqwfBtBmhuiUWUSsKUAfloL1YPBYmyihySJwWEVETnqzBRd1RCMk",clsUTNumberUtilsP3RP0P1.toString());
    assertEquals("YfJ0ufsFnsYRrFHBpCyLgDWEmhiRPOGISMxFfCYjMOSisquxCuSHI0rrEBbqqwfBtBmhuiUWUSsKUAfloL1YPBYmyihySJwWEVETnqzBRd1RCMk",clsUTNumberUtilsP3RP0R);
}

```



BX-KT



TEST READABILITY

“[Developers] read tests [...] 77% of the total time they spend in them”

M. Beller, G. Gousios, A. Panichella, and A. Zaidman, “When, How, and Why Developers (Do Not) Test in Their IDEs”. FSE 2015.

TEST READABILITY

“[Developers] read tests [...] 77% of the total time they spend in them”

M. Beller, G. Gousios, A. Panichella, and A. Zaidman, “When, How, and Why Developers (Do Not) Test in Their IDEs”. FSE 2015.

But what does readability mean?

TEST READABILITY

“[Developers] read tests [...] 77% of the total time they spend in them”

M. Beller, G. Gousios, A. Panichella, and A. Zaidman, “When, How, and Why Developers (Do Not) Test in Their IDEs”. FSE 2015.

But what does readability mean?

| Feature name | Correlation | | | One feature a time | | |
|---------------------------|--------------|--------------|--------------|--------------------|-------|-------|
| | total | max | avg | total | max | avg |
| identifier length | -0.50 | -0.42 | -0.46 | 0.50 | 0.41 | 0.45 |
| commas | -0.15 | -0.20 | -0.15 | 0.13 | 0.14 | 0.13 |
| line length | -0.45 | -0.50 | -0.43 | 0.4 | 0.49 | 0.41 |
| Halstead difficulty | -0.15 | - | - | - | - | - |
| constructor calls | -0.45 | - | -0.24 | 0.44 | - | 0.20 |
| has exceptions | 0.15 | - | - | 0.11 | - | - |
| byte entropy | -0.39 | - | - | 0.31 | - | - |
| identifier ratio | 0.15 | - | - | 0.11 | - | - |
| unique identifiers | -0.37 | -0.25 | -0.14 | 0.36 | 0.22 | 0.11 |
| method invocations | -0.14 | -0.06 | -0.03 | 0.09 | -0.00 | -0.07 |
| identifiers | -0.36 | -0.23 | -0.29 | 0.36 | 0.20 | 0.27 |
| string length | -0.14 | -0.24 | -0.20 | 0.09 | 0.23 | 0.19 |
| assignments | -0.33 | - | -0.16 | 0.32 | - | 0.13 |
| arrays | -0.14 | -0.05 | -0.06 | 0.11 | -0.05 | -0.01 |
| casts | -0.33 | -0.33 | -0.28 | 0.32 | 0.32 | 0.26 |
| indentation | -0.13 | 0.08 | -0.01 | 0.10 | 0.02 | -0.25 |
| parentheses | -0.31 | - | -0.28 | 0.30 | - | 0.26 |
| field accesses | -0.13 | -0.14 | -0.17 | 0.11 | 0.11 | 0.15 |
| keywords | -0.30 | -0.28 | -0.17 | 0.28 | 0.27 | 0.14 |
| Halstead effort | -0.13 | - | - | 0.13 | - | - |
| Halstead volume | -0.27 | - | - | 0.18 | - | - |
| assertions | -0.12 | - | -0.04 | 0.09 | - | -0.10 |
| distinct methods | -0.27 | -0.05 | 0.00 | 0.26 | -0.02 | -0.18 |
| additional assertions | -0.12 | - | - | 0.09 | - | - |
| single characters | -0.26 | -0.32 | -0.24 | 0.17 | 0.22 | 0.16 |
| nulls | -0.12 | -0.20 | -0.15 | 0.06 | 0.19 | 0.12 |
| periods | -0.25 | -0.22 | -0.17 | 0.24 | 0.20 | 0.13 |
| class ratio | -0.10 | - | - | 0.04 | - | - |
| comparison operations | -0.25 | - | -0.23 | 0.24 | - | 0.23 |
| blank lines | 0.08 | - | 0.21 | 0.03 | - | 0.19 |

E. Daka, J. Campos, G. Fraser, J. Dorn, and W. Weimer,
“Modeling readability to improve unit tests”. FSE 2015.

GO-BX-KT

```
@Test(timeout = 4000)
public void test1() throws Throwable {
    LinkedList<String> linkedList0 = new LinkedList<String>();
    LinkedList<Object> linkedList1 =
        new LinkedList<Object>((Collection<?>) linkedList0);
    FixedOrderComparator fixedOrderComparator0 =
        new FixedOrderComparator((List) linkedList1);
    linkedList1.add(0, linkedList1);
    // Undeclared exception!
    try {
        fixedOrderComparator0.compare(linkedList1, linkedList0);
        fail("Expecting exception: StackOverflowError");
    } catch(StackOverflowError e) {
        //
        // no message in exception (getMessage() returned null)
        //
    }
}
```

```
@Test(timeout = 1000)
public void test1() throws Throwable {
    LinkedList<String> linkedList0 = new LinkedList<String>();
    LinkedList<Object> linkedList1 =
        new LinkedList<Object>((Collection<?>) linkedList0);
    FixedOrderComparator fixedOrderComparator0 =
        new FixedOrderComparator((List) linkedList1);
    linkedList1.add(0, linkedList1);
    // Undeclared exception!
    try {
        fixedOrderComparator0.compare(linkedList1, linkedList0);
        fail("Expecting exception: StackOverflowError");
    } catch(StackOverflowError e) {
        //
        // no message in exception (getMessage() returned null)
        //
    }
}
```

COVERAGE-BASED NAMING

COVERAGE-BASED NAMING

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

COVERAGE-BASED NAMING

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

Method Coverage

GO-BX-KT

COVERAGE-BASED NAMING

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

Method Coverage

GO-BX-KT

COVERAGE-BASED NAMING

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

testCreateShoppingCart
testAddPrice
testGetTotal

Method Coverage

GO-BX-KT

COVERAGE-BASED NAMING

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

Exception Coverage

GO-BX-KT

COVERAGE-BASED NAMING

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

Exception Coverage

GO-BX-KT

COVERAGE-BASED NAMING

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

testAddPriceThrowsIllegalArgExc

Exception Coverage

GO-BX-KT

COVERAGE-BASED NAMING

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

Output Coverage

GO-BX-KT

COVERAGE-BASED NAMING

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

Output Coverage

GO-BX-KT

COVERAGE-BASED NAMING

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

testGetTotalReturnsZero
testGetTotalReturnsPositive
testGetTotalReturnsNegative

Output Coverage

GO-BX-KT

COVERAGE-BASED NAMING

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

Input Coverage

GO-BX-KT

COVERAGE-BASED NAMING

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

Input Coverage

GO-BX-KT

COVERAGE-BASED NAMING

```
public class ShoppingCart {  
    private int total = 0;  
    private final static int MAX = 1000;  
  
    public boolean addPrice(int cost)  
        throws IllegalArgumentException {  
        if (cost <= 0)  
            throw new IllegalArgumentException("Negative cost");  
        if (cost < MAX) {  
            total += cost;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    public int getTotal() {  
        return total;  
    }  
}
```

testAddPriceWithZero
testAddPriceWithPositive
testAddPriceWithNegative

SYNTHESIS OF DESCRIPTIVE TEST NAMES



SYNTHESIS OF DESCRIPTIVE TEST NAMES



SYNTHESIS OF DESCRIPTIVE TEST NAMES



- I. Identify goals covered uniquely by each test case

SYNTHESIS OF DESCRIPTIVE TEST NAMES



- I. Identify goals covered uniquely by each test case

SYNTHESIS OF DESCRIPTIVE TEST NAMES



- I. Identify goals covered uniquely by each test case

SYNTHESIS OF DESCRIPTIVE TEST NAMES



- I. Identify goals covered uniquely by each test case

SYNTHESIS OF DESCRIPTIVE TEST NAMES



- I. Identify goals covered uniquely by each test case

SYNTHESIS OF DESCRIPTIVE TEST NAMES



1. Identify goals covered uniquely by each test case
2. Sort covered goals
Exception > Method > Output > Input

SYNTHESIS OF DESCRIPTIVE TEST NAMES



1. Identify goals covered uniquely by each test case
2. Sort covered goals
Exception > Method > Output > Input

SYNTHESIS OF DESCRIPTIVE TEST NAMES



1. Identify goals covered uniquely by each test case
2. Sort covered goals
Exception > Method > Output > Input

SYNTHESIS OF DESCRIPTIVE TEST NAMES



1. Identify goals covered uniquely by each test case
2. Sort covered goals
Exception > Method > Output > Input

SYNTHESIS OF DESCRIPTIVE TEST NAMES



1. Identify goals covered uniquely by each test case
2. Sort covered goals
Exception > Method > Output > Input

SYNTHESIS OF DESCRIPTIVE TEST NAMES



1. Identify goals covered uniquely by each test case
2. Sort covered goals
Exception > Method > Output > Input
3. Synthesise readable names

SYNTHESIS OF DESCRIPTIVE TEST NAMES



1. Identify goals covered uniquely by each test case
2. Sort covered goals
Exception > Method > Output > Input
3. Synthesise readable names
4. Resolve conflicts when possible based on pairs of goals

SYNTHESIS OF DESCRIPTIVE TEST NAMES



1. Identify goals covered uniquely by each test case
2. Sort covered goals
Exception > Method > Output > Input
3. Synthesise readable names
4. Resolve conflicts when possible based on pairs of goals
5. Add numerical suffixes

EMPIRICAL EVALUATION



Online study



47 participants



10 Java classes

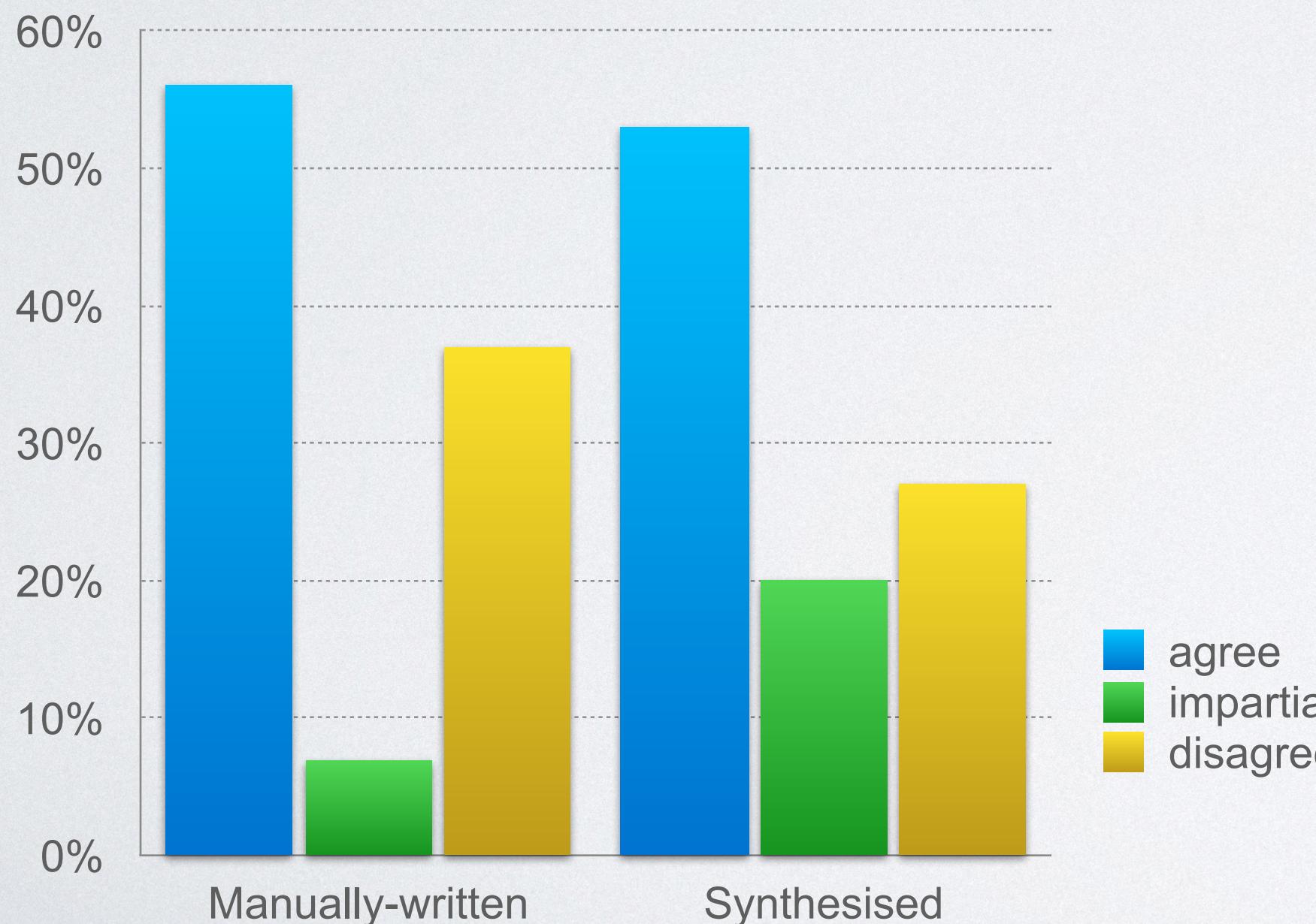


2 experts as control



No time limit

"Do you agree with the name of the following test?"



RQ1: Do developers agree with synthesised test names?

Developers agreed similarly—and disagreed less—with synthesised test names than with manually given names.

RQ2: Can developers match tests with synthesised test names?

Developers were slightly more accurate and faster at matching tests with synthesised names.

RQ3: Can synthesised test names help developers to match tests to code?

Developers were more accurate at identifying relevant tests for given pieces of code using synthesised names.

RELATED WORK

- Natural Language Generation *from test code*
- Test data generation for RESTful APIs – www.evomaster.org
- Unit test generation for Python programs – <https://www.pynguin.eu/>
- Fuzzing – <https://www.fuzzingbook.org/>

CAN MODERN AI HELP?

- Software Testing  AI
- Building large datasets of annotated unit tests for training can be very costly
- Inferring test oracles from test code: assertions that capture the developer's *intention*
- GitHub CoPilot
 - Trained on billions of lines of code in a number of programming languages
 - Turns natural language prompts into code snippets
 - Similar work is being done for tests by DiffBlue (<https://www.diffblue.com/>)

WRAP-UP

- Genetic Algorithms used to generate tests automatically
 - Other optimisation algorithms can also be suitable
- Test readability and understandability remain major challenges
- Opportunities for modern AI approaches