

CSE 403

Software Engineering

Coverage-based Testing

This week: test adequacy

- Coverage-based testing
- In-class exercise: software testing gamification

Structural code coverage: Cobertura

Classes in this File	Line Coverage	Branch Coverage	Complexity
Avg	100% 10/10	100% 8/8	6

```
1 package avg;
2
3 4 public class Avg {
4
5     /*
6      * Compute the average of the absolute values of an array of doubles
7      */
8     public double avgAbs(double ... numbers) {
9         // We expect the array to be non-null and non-empty
10 4         if (numbers == null || numbers.length == 0) {
11 2             throw new IllegalArgumentException("Array numbers must not be null or empty!");
12         }
13
14 2         double sum = 0;
15 8         for (int i=0; i<numbers.length; ++i) {
16 6             double d = numbers[i];
17 6             if (d < 0) {
18 2                 sum -= d;
19             } else {
20 4                 sum += d;
21             }
22         }
23 2         return sum/numbers.length;
24     }
25 }
```

Structural code coverage: Jacoco

 testing-mock >  cse403.testing.mock.service

cse403.testing.mock.service

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 UserService.java		100%		100%	0	4	0	8	0	3	0	1
Total	0 of 29	100%	0 of 2	100%	0	4	0	8	0	3	0	1

 testing-mock >  [cse403.testing.mock.service](#) >  [UserService.java](#)

UserService.java

```
1. package cse403.testing.mock.service;
2.
3. import cse403.testing.mock.model.User;
4. import cse403.testing.mock.repository.UserRepository;
5.
6. public class UserService {
7.     private final UserRepository repository;
8.
9.     public UserService(UserRepository repository) {
10.         this.repository = repository;
11.     }
12.
13.     public void registerUser(int id, String name) {
14.         User user = new User(id, name);
15.         repository.save(user);
16.     }
17.
18.     public String getUserByNameById(int id) {
19.         User user = repository.findById(id);
20.         return user != null ? user.getName() : null;
21.     }
22. }
```

<https://github.com/rjust/testing-mock>

Code coverage metrics



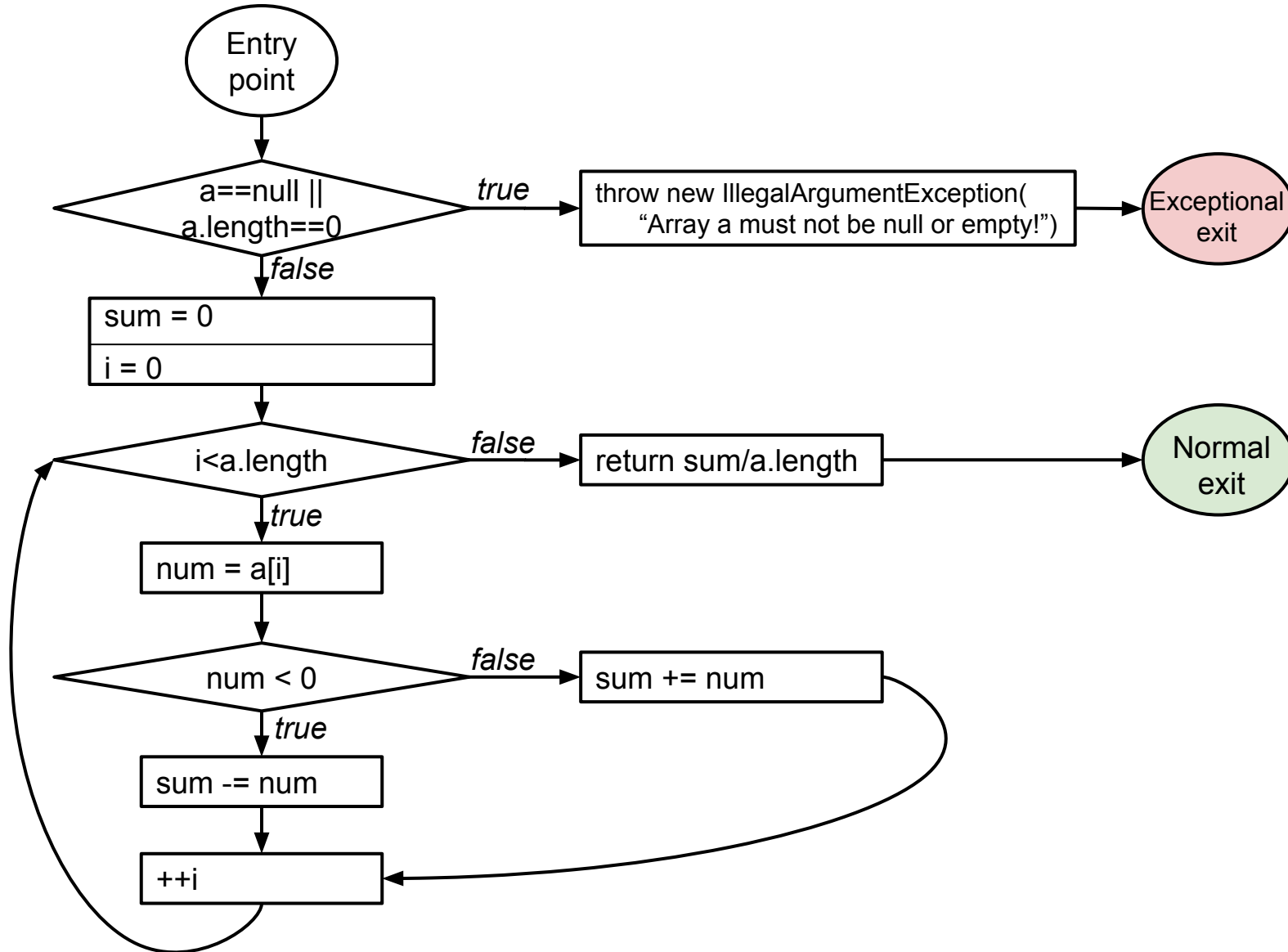
Structural code coverage: the basics

Average of the absolute values of an array of doubles

```
public double avgAbs(double ... numbers) {  
  
    // We expect the array to be non-null and non-empty  
    if (numbers == null || numbers.length == 0) {  
        throw new IllegalArgumentException("Array numbers must not be null or empty!");  
    }  
  
    double sum = 0;  
    for (int i=0; i<numbers.length; ++i) {  
        double d = numbers[i];  
        if (d < 0) {  
            sum -= d;  
        } else {  
            sum += d;  
        }  
    }  
  
    return sum/numbers.length;  
}
```

What's the control flow graph (CFG) for this method?

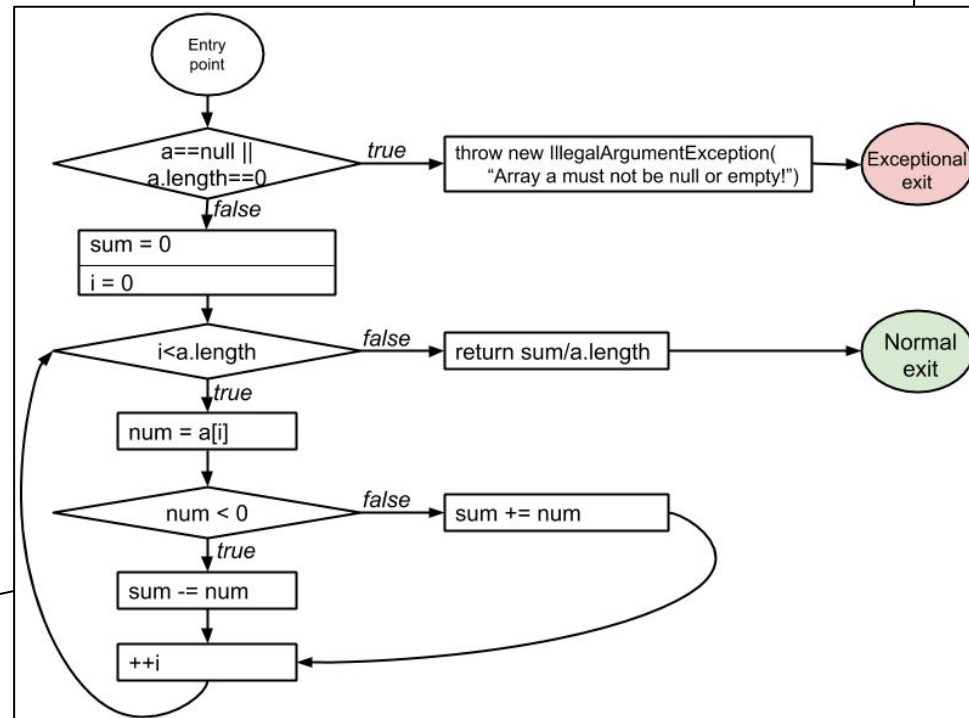
Structural code coverage: the basics



Structural code coverage: the basics

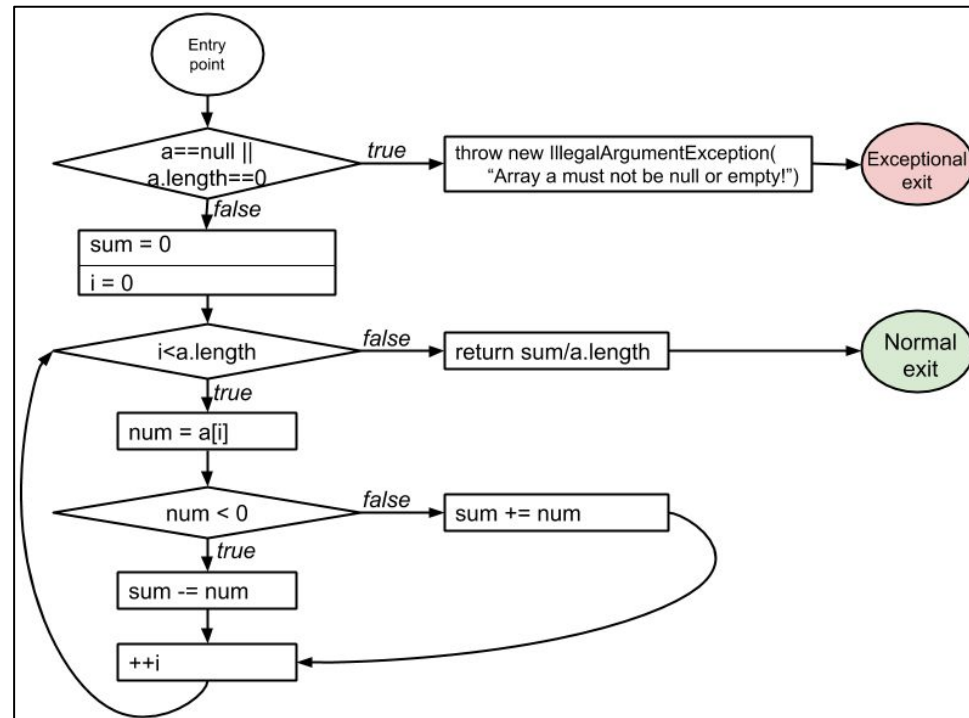
Average of the absolute values of an array of doubles

```
public double avgAbs(double ... numbers) {  
  
    // We expect the array to be non-null and non-empty  
    if (numbers == null || numbers.length == 0) {  
        throw new IllegalArgumentException("Array numbers must not be null or empty!");  
    }  
  
    double sum = 0;  
    for (int i=0; i<numbers.length; ++i) {  
        double d = numbers[i];  
        if (d < 0) {  
            sum -= d;  
        } else {  
            sum += d;  
        }  
    }  
  
    return sum/numbers.length;  
}
```

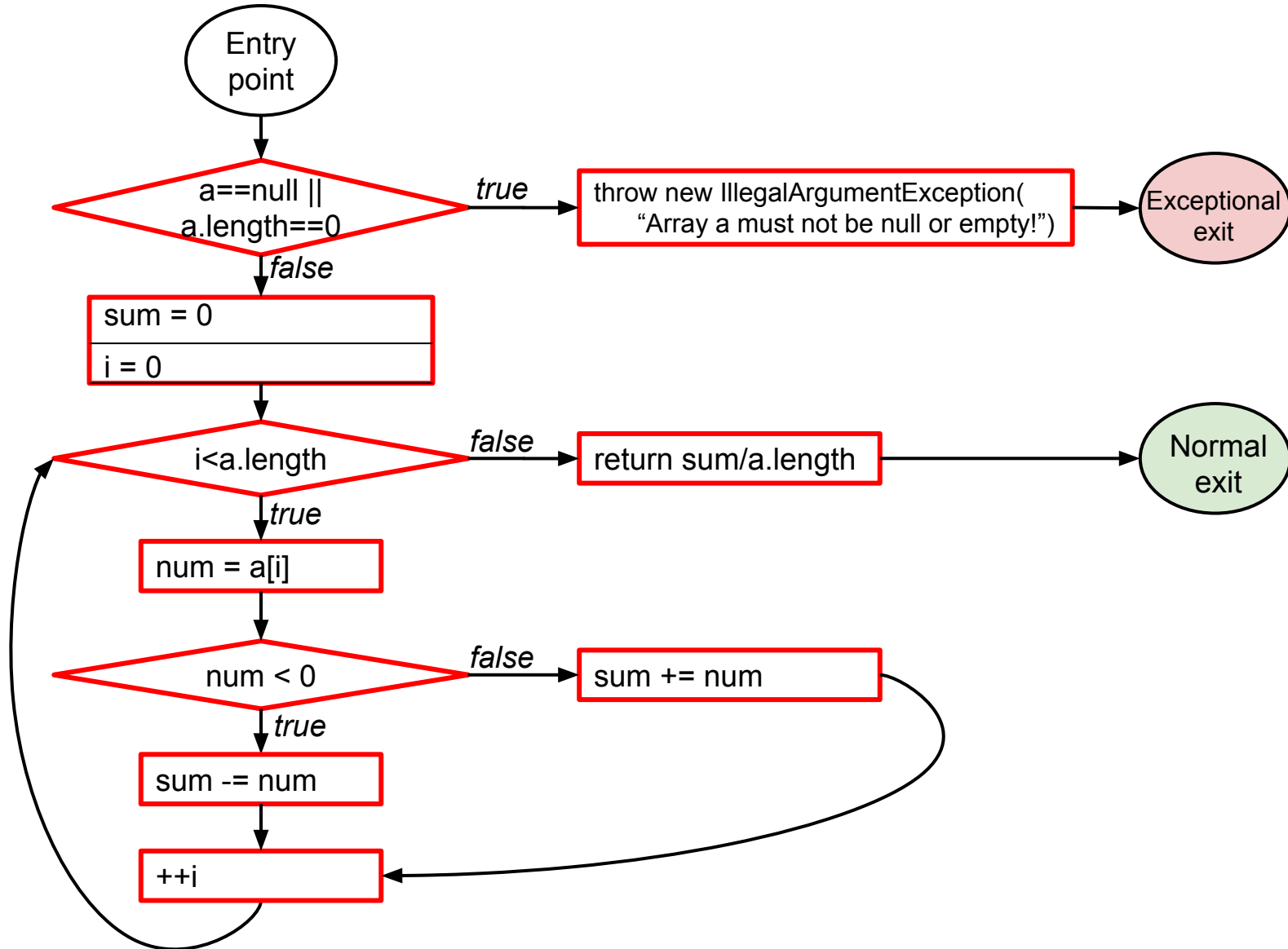


Statement coverage

- **Every statement in the program must be executed at least once.**

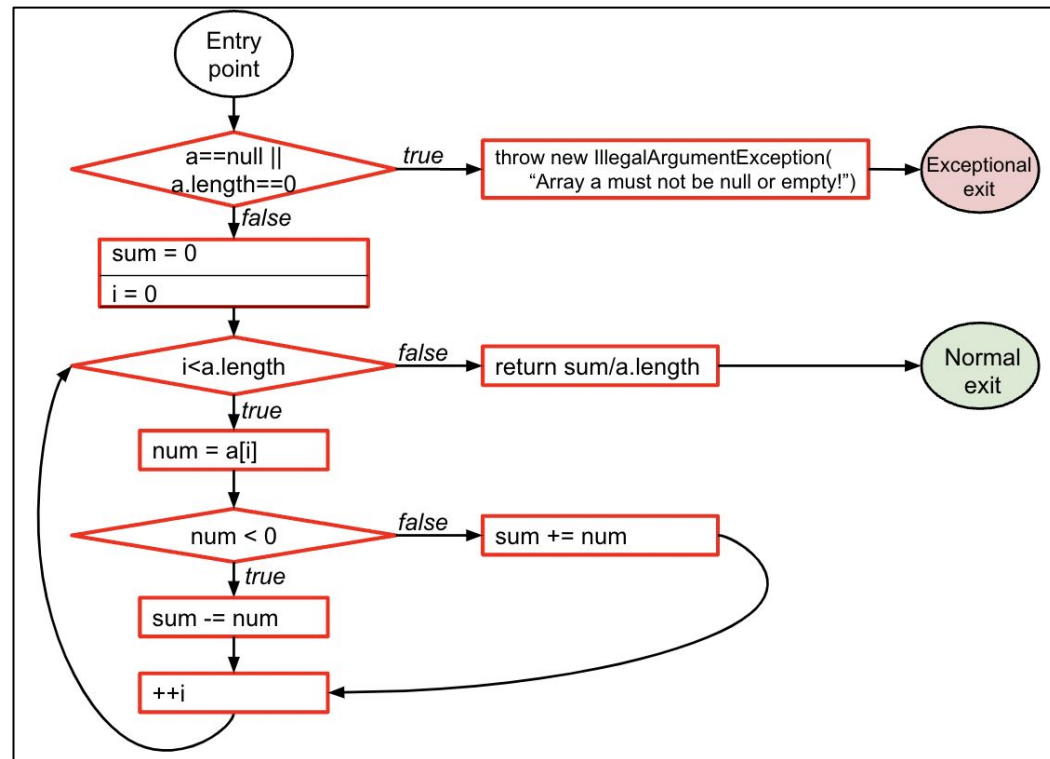


Statement coverage



Statement coverage

- **Every statement** in the program must be **executed at least once**.
- Given the control-flow graph (CFG), this is equivalent to node coverage.



Condition coverage vs. decision coverage

Terminology

- **Condition:** a boolean expression that cannot be decomposed into simpler boolean expressions (atomic).
- **Decision:** a boolean expression that is composed of conditions, using 0 or more logical connectors (a decision with 0 logical connectors is a condition).
- **Example:** if $(a \mid b) \{ \dots \}$
 - a and b are *conditions*.
 - The boolean expression $a \mid b$ is a *decision*.

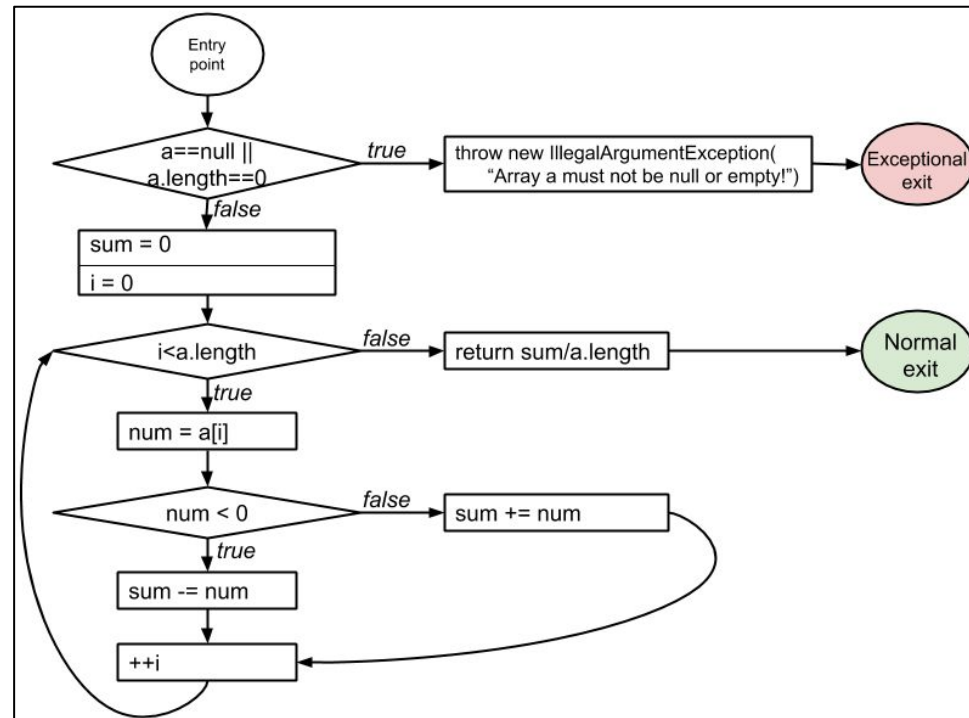
Condition coverage vs. decision coverage

Terminology

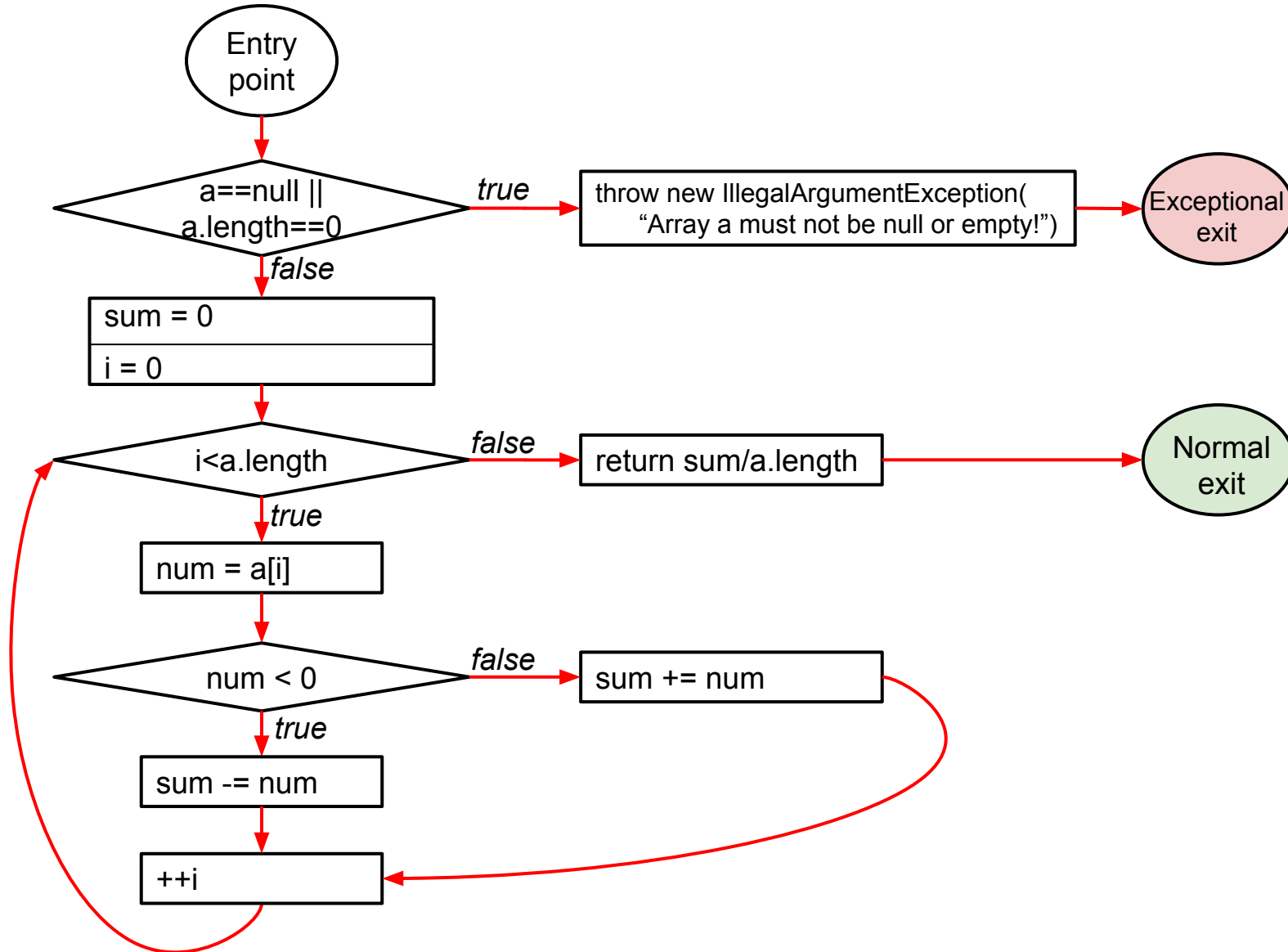
- **Condition:** a boolean expression that cannot be decomposed into simpler boolean expressions (atomic).
- **Decision:** a boolean expression that is composed of conditions, using 0 or more logical connectors (a decision with 0 logical connectors is a condition).
- **Example:** if $(a \mid b) \{ \dots \}$
 - a and b are *conditions*.
 - The boolean expression $a \mid b$ is a *decision*.

Decision coverage

- **Every decision in the program must take on all possible outcomes (true/false) at least once.**

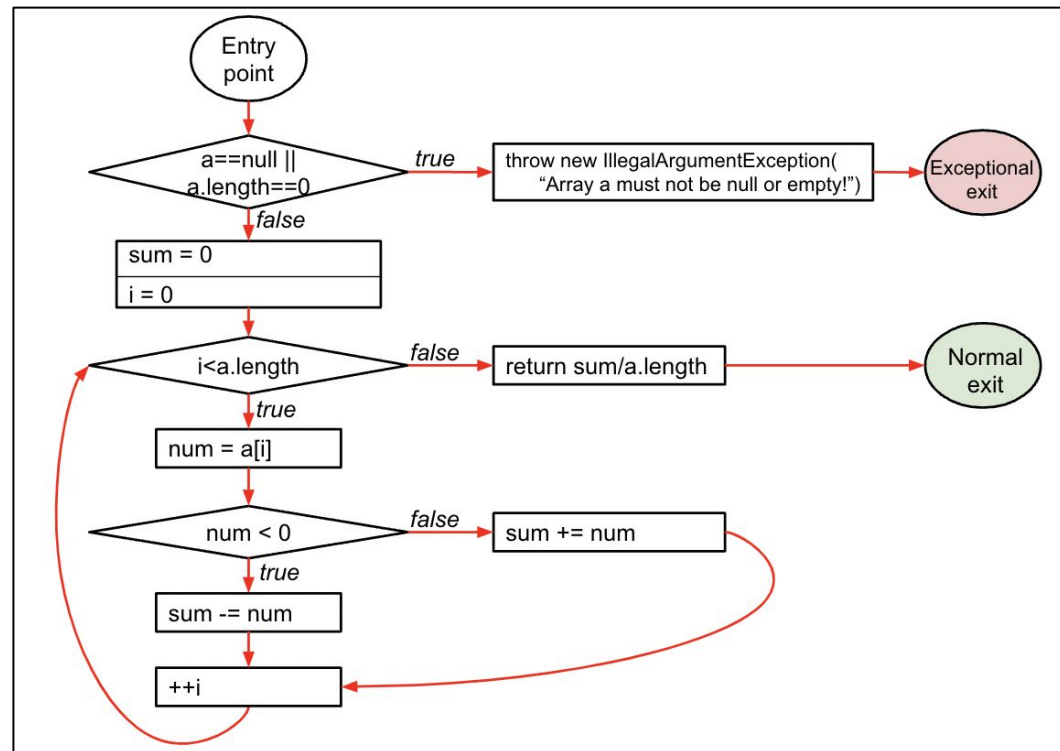


Decision coverage



Decision coverage

- **Every decision** in the program must take on **all possible outcomes** (true/false) **at least once**.
- Given the CFG, this is equivalent to edge coverage.



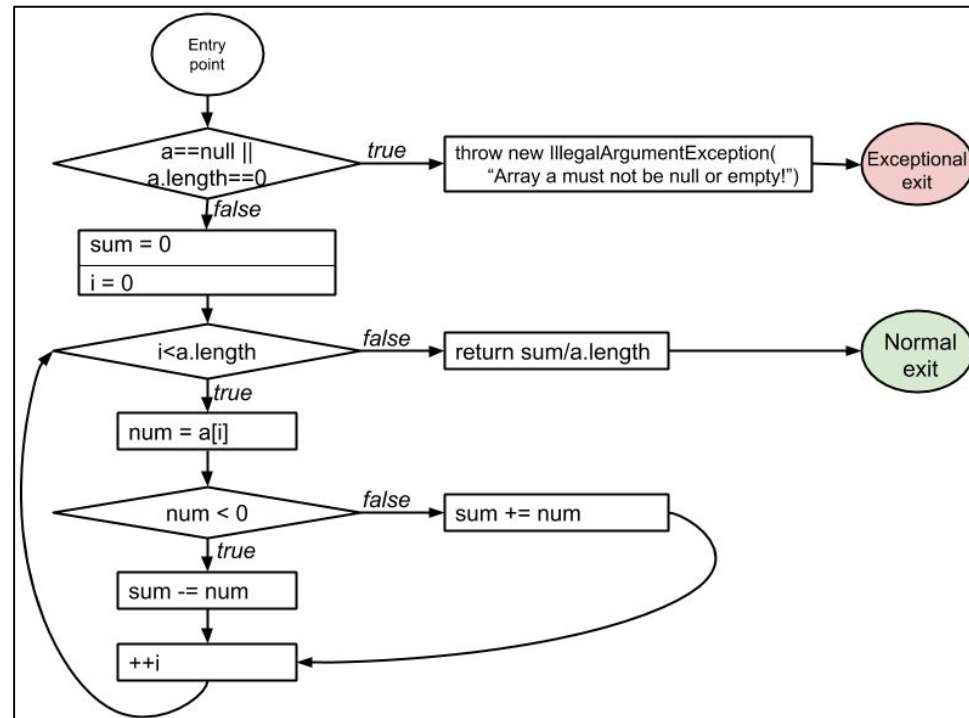
Condition coverage vs. decision coverage

Terminology

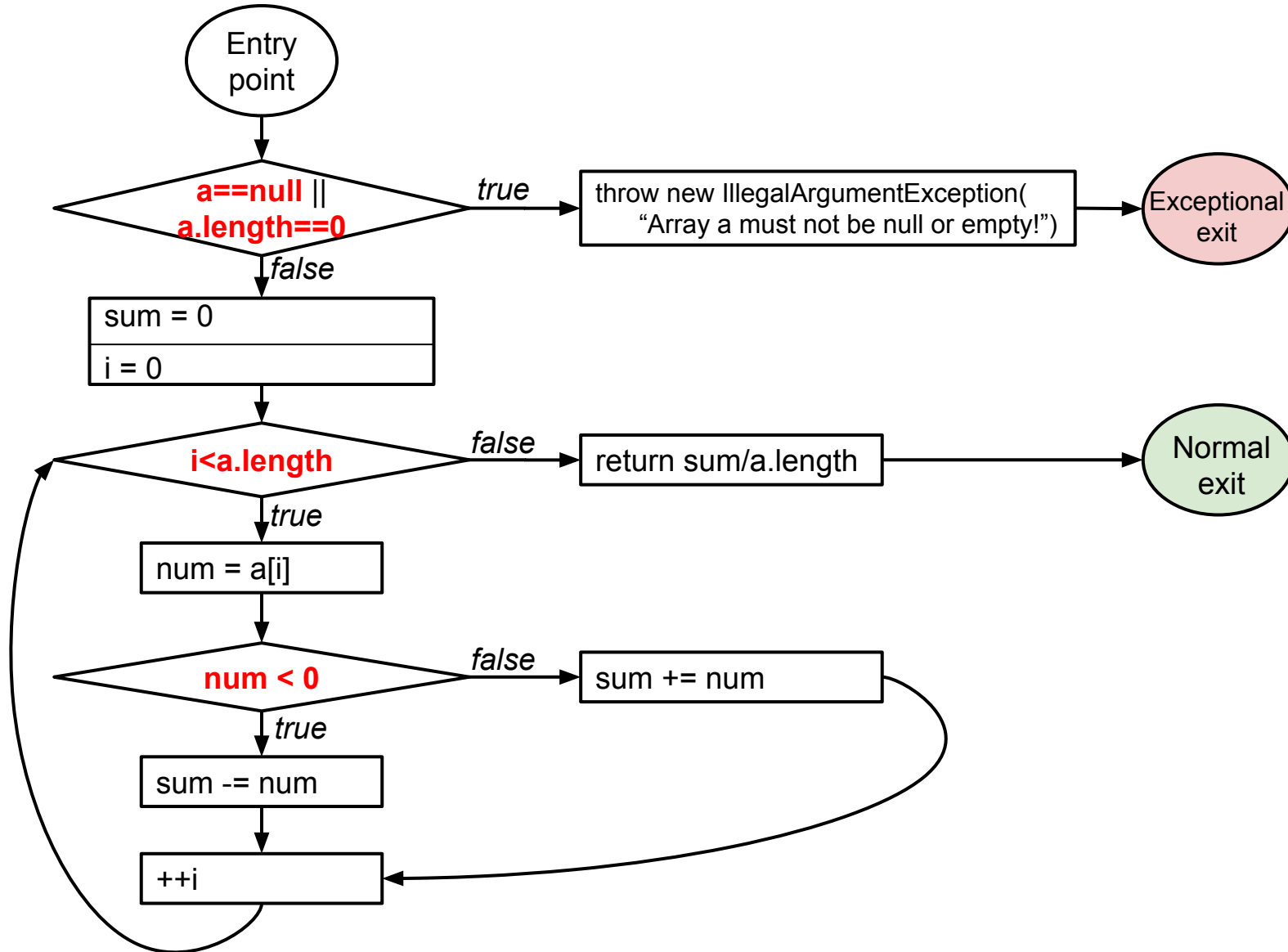
- **Condition:** a boolean expression that cannot be decomposed into simpler boolean expressions (atomic).
- **Decision:** a boolean expression that is composed of conditions, using 0 or more logical connectors (a decision with 0 logical connectors is a condition).
- **Example:** if $(a \mid b) \{ \dots \}$
 - a and b are *conditions*.
 - The boolean expression $a \mid b$ is a *decision*.

Condition coverage

- **Every condition** in the program must take on **all possible outcomes** (true/false) **at least once**.

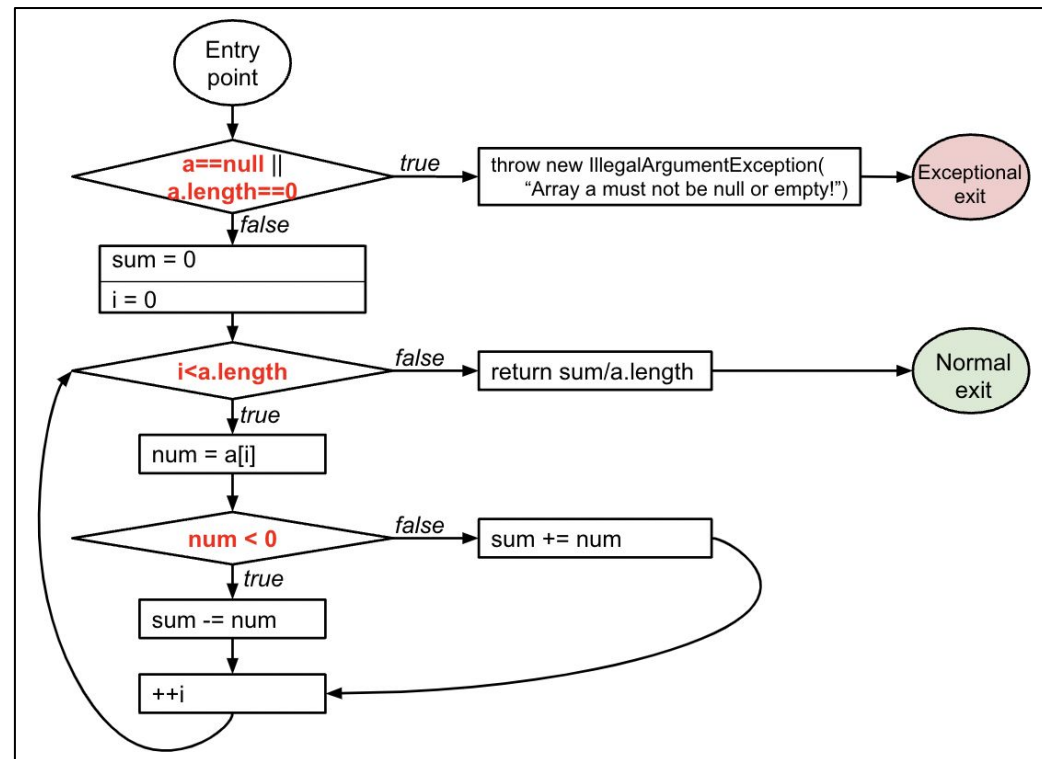


Condition coverage



Condition coverage

- **Every condition in the program must take on all possible outcomes (true/false) at least once.**





Structural code coverage: subsumption

Given two coverage criteria A and B,

A subsumes B iff **satisfying A implies satisfying B**

- Subsumption relationships:
 1. Does statement coverage subsume decision coverage?
 2. Does decision coverage subsume statement coverage?
 3. Does decision coverage subsume condition coverage?
 4. Does condition coverage subsume decision coverage?

<https://forms.gle/be8GgfACsyFRA26K8>

Modified Condition/Decision Coverage (MC/DC)

MCDC: Modified condition and decision coverage

- **Every decision** in the program must take on **all possible outcomes** (true/false) **at least once**
- **Every condition** in the program must take on **all possible outcomes** (true/false) **at least once**
- **Each condition** in a decision has been shown to **independently affect** that decision's **outcome**.

(A condition is shown to independently affect a decision's outcome by: varying just that condition while holding fixed all other possible conditions.)

Required for safety critical systems (DO-178B/C)

MC/DC: an example

if (a | b)

a	b	Outcome
0	0	0
0	1	1
1	0	1
1	1	1

MCDC

- **Decision** coverage
- **Condition** coverage
- **Each condition** shown to **independently affect outcome**

Which tests (combinations of a and b) satisfy MCDC?

MC/DC: an example

if (a | b)

a	b	Outcome
0	0	0
0	1	1
1	0	1
1	1	1

MCDC

- **Decision** coverage
- **Condition** coverage
- **Each condition** shown to **independently affect outcome**

MCDC is still cheaper than testing all possible combinations.

MC/DC: another example

```
if (a || b)
```

a	b	Outcome
0	0	0
0	1	1
1	0	1
1	1	1

MCDC

- **Decision** coverage
- **Condition** coverage
- **Each condition** shown to **independently affect outcome**

Why is this example different?

MC/DC: another example

```
if (a || b)
```

a	b	Outcome
0	0	0
0	1	1
1	--	1
1	--	1

MCDC

- **Decision** coverage
- **Condition** coverage
- **Each condition** shown to **independently affect outcome**

Short-circuiting operators may not evaluate all conditions.

MC/DC: yet another example

```
if (!a) ... if (a || b)
```

a	b	Outcome
0	0	0
0	1	1
1	0	1
1	1	1

MCDC

- **Decision** coverage
- **Condition** coverage
- **Each condition** shown to **independently affect outcome**

What about this example?

MC/DC: another example

```
if (!a) ... if (a || b)
```

a	b	Outcome
0	0	0
0	1	1
X	X	X
X	X	X

MCDC

- **Decision** coverage
- **Condition** coverage
- **Each condition** shown to **independently affect outcome**

Not all combinations of conditions may be possible.

MCDC: complex expressions



Provide an MCDC-adequate test suite for:

1. $a \mid b \mid c$
2. $a \ \& \ b \ \& \ c$

<https://forms.gle/6otG8qCjBVeVWZpUA>

Structural code coverage: summary

Classes in this File	Line Coverage	Branch Coverage	Complexity
Avg	100% 10/10	100% 8/8	6

```
1 package avg;
2
3 4 public class Avg {
4
5     /*
6      * Compute the average of the absolute values of an array of doubles
7      */
8     public double avgAbs(double ... numbers) {
9         // We expect the array to be non-null and non-empty
10 4         if (numbers == null || numbers.length == 0) {
11 2             throw new IllegalArgumentException("Array numbers must not be null or empty!");
12         }
13
14 2         double sum = 0;
15 8         for (int i=0; i<numbers.length; ++i) {
16 6             double d = numbers[i];
17 6             if (d < 0) {
18 2                 sum -= d;
19             } else {
20 4                 sum += d;
21             }
22         }
23 2         return sum/numbers.length;
24     }
25 }
```

- Code coverage is easy to compute.
- Code coverage has an intuitive interpretation.
- Code coverage in industry: [Code coverage at Google](#)
- Code coverage itself is not sufficient!