# Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing

Gregory M. Kapfhammer[†]

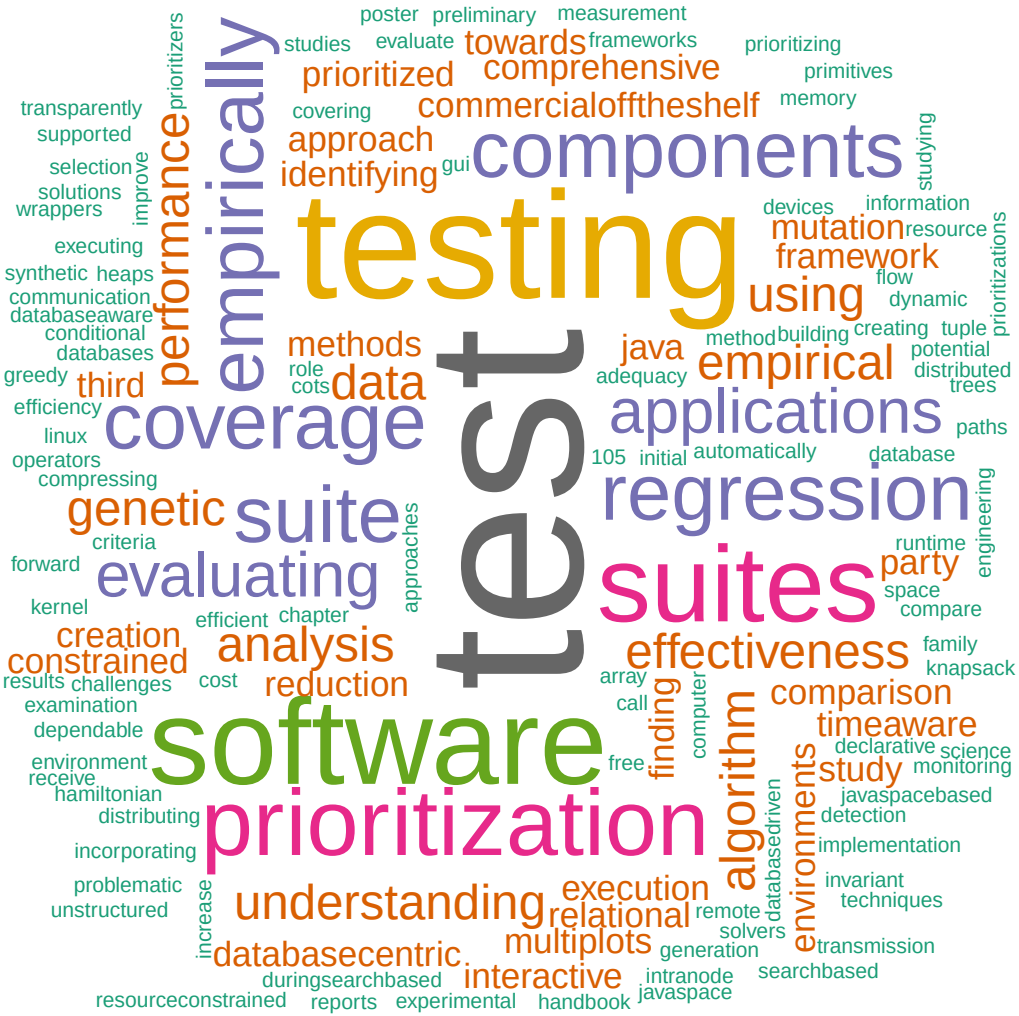
Department of Computer Science
Allegheny College
http://www.cs.allegheny.edu/~gkapfham/

University of Delhi – May 9, 2012

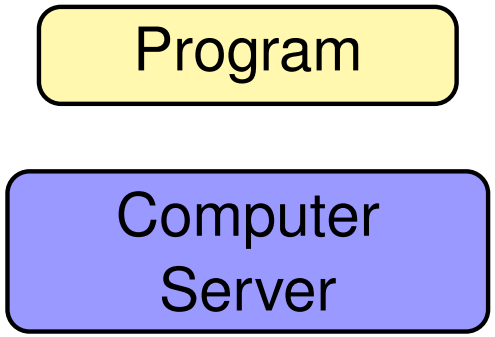
[†] Joint with Jonathan Miller Kauffman (Allegheny College)


ALLEGHENY COLLEGE

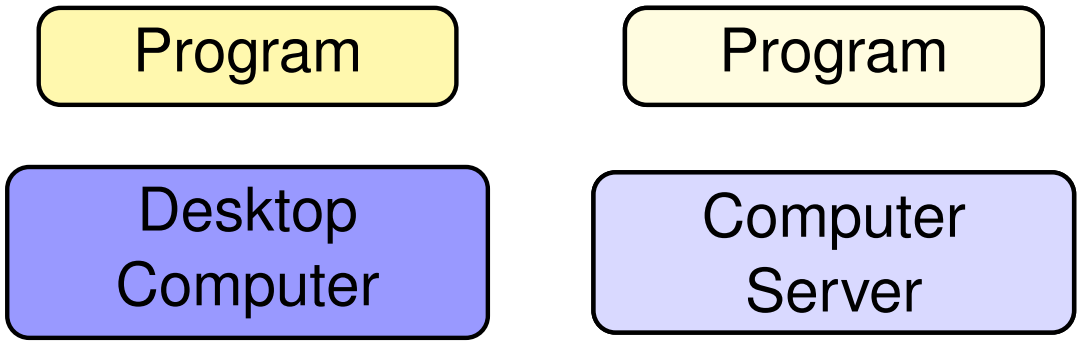
# Presenter Introduction: Gregory M. Kapfhammer



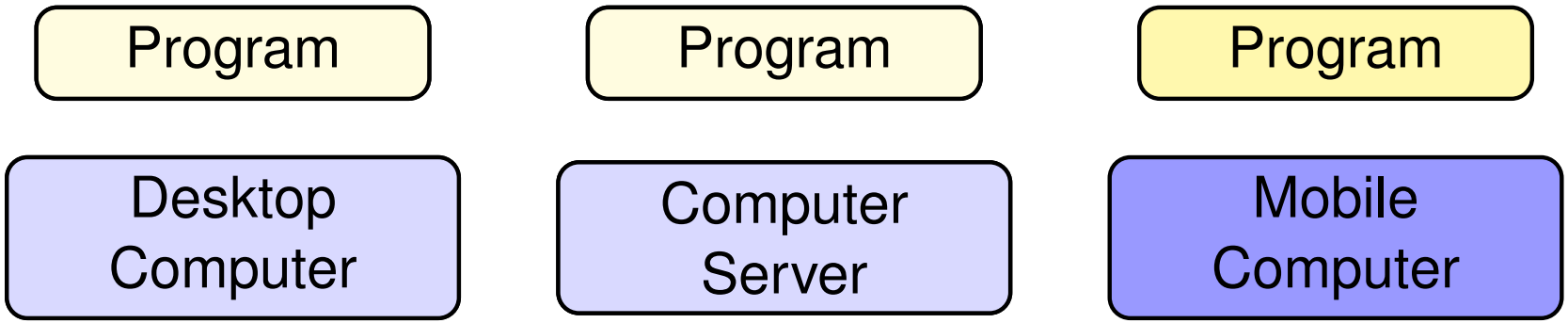
# Software is Everywhere



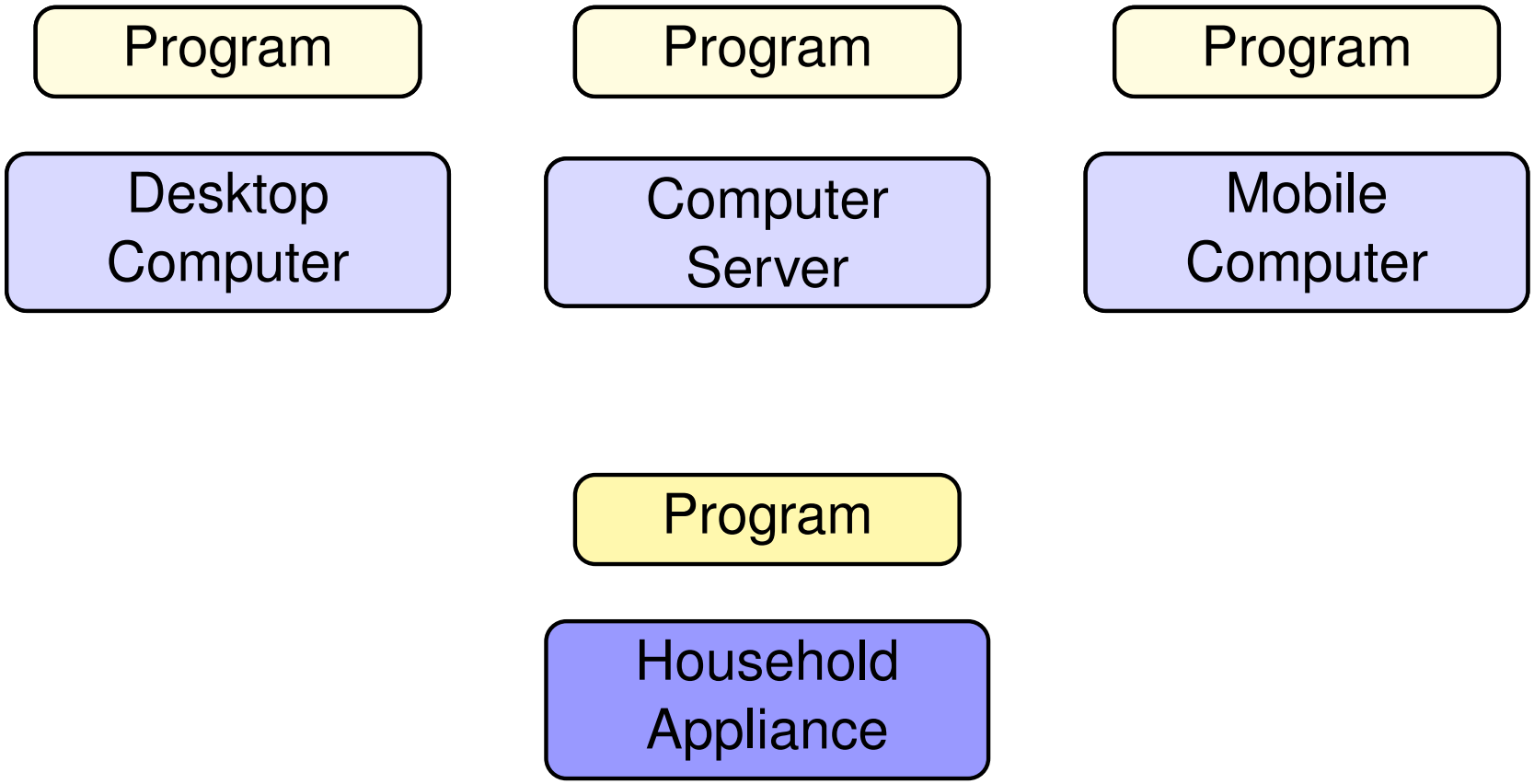
# Software is Everywhere



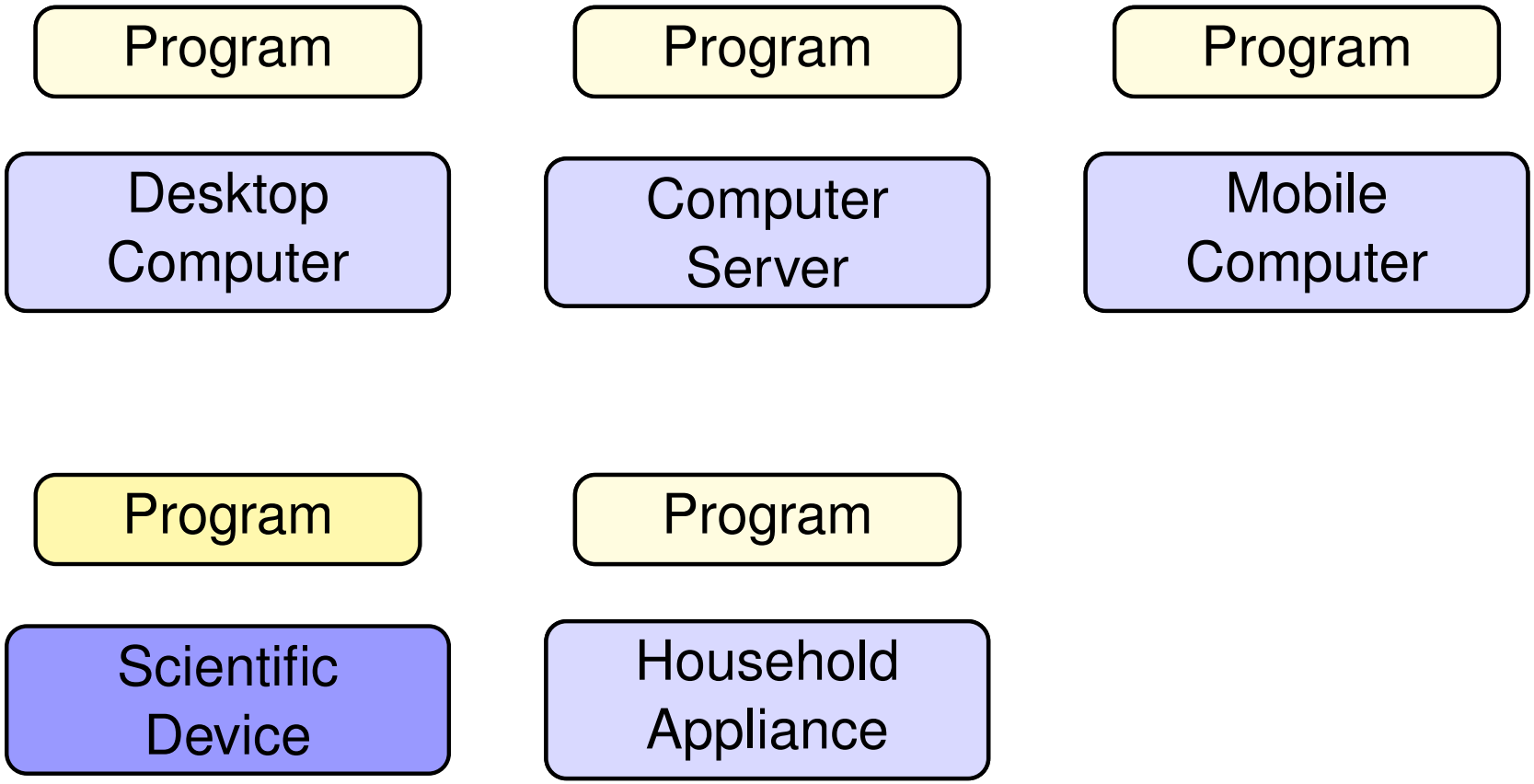
# Software is Everywhere



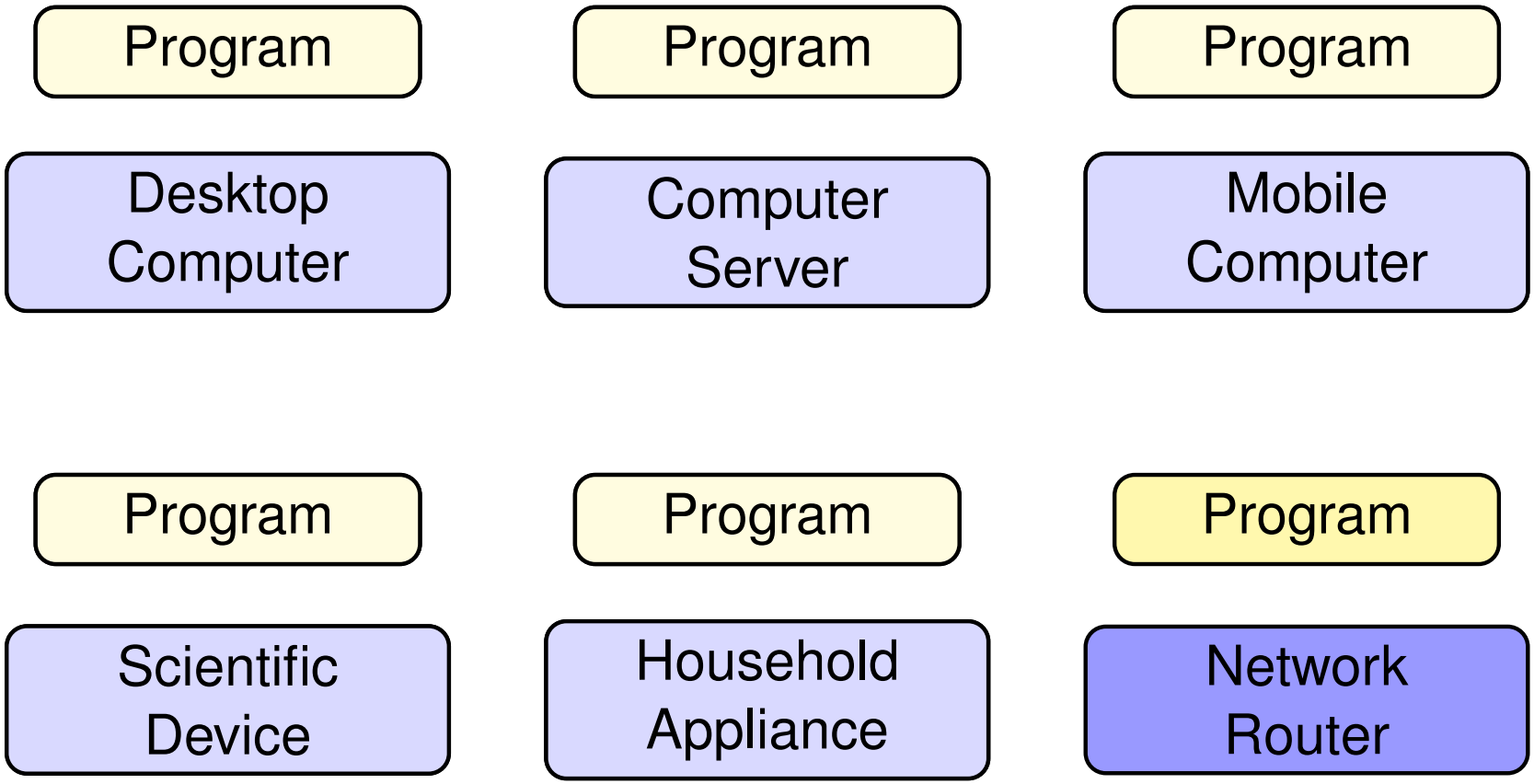
# Software is Everywhere



# Software is Everywhere



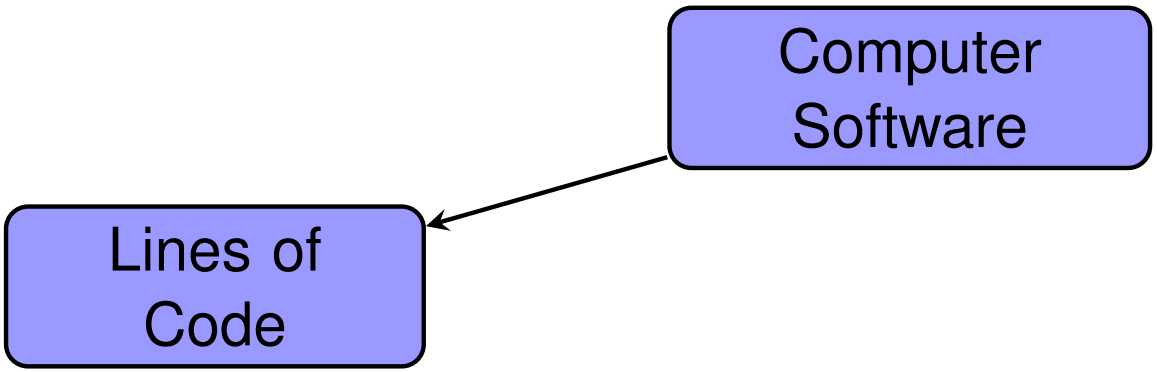
# Software is Everywhere



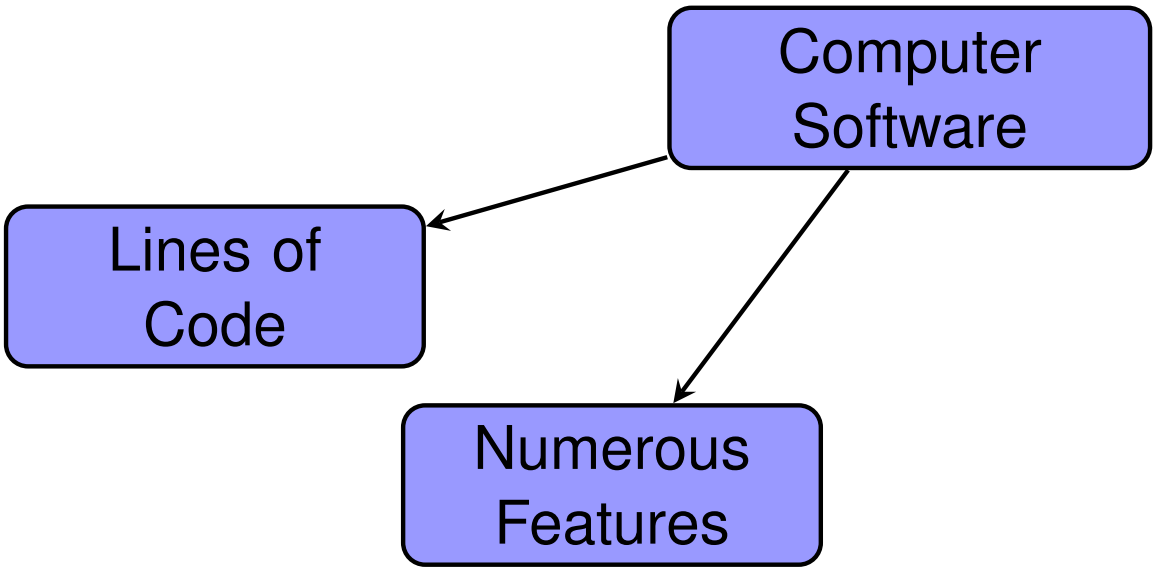
# Software is Complex

Computer Software

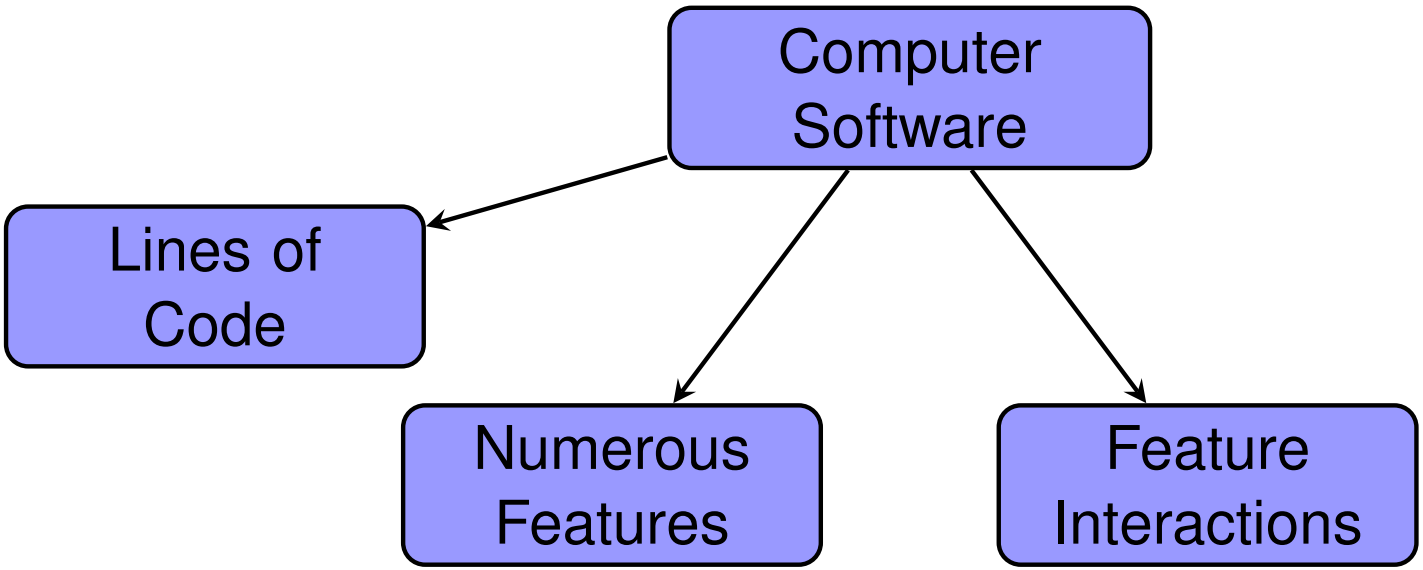
# Software is Complex



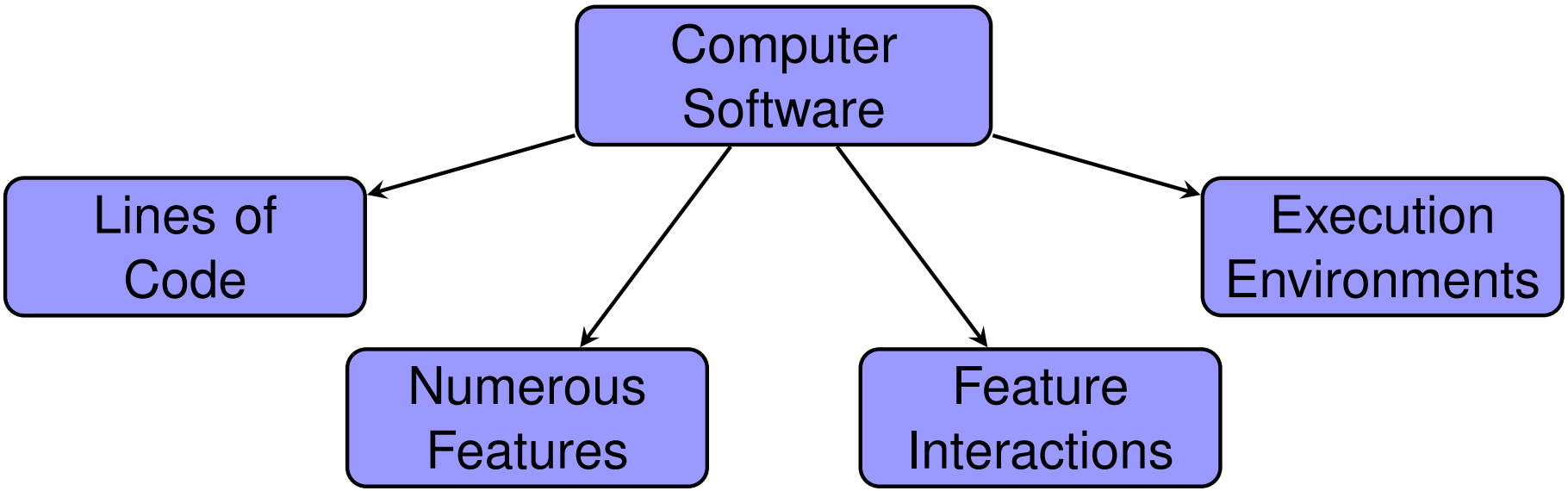
# Software is Complex



# Software is Complex

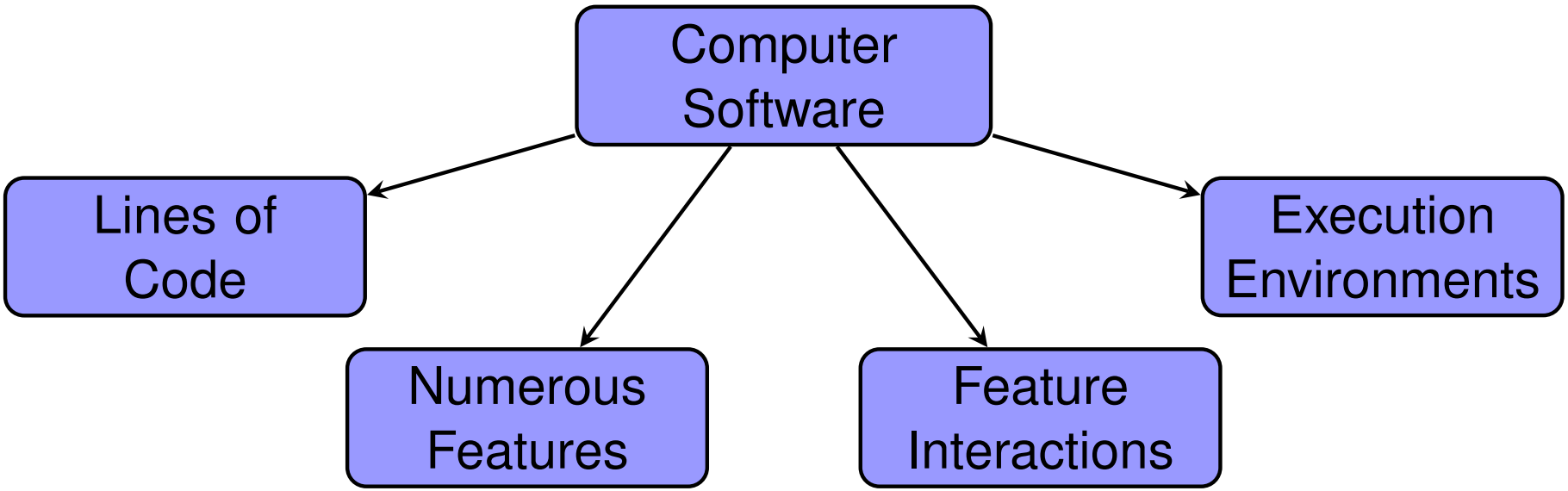


# Software is Complex

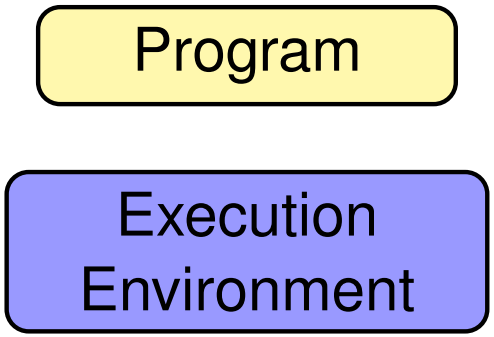


# Software is Complex

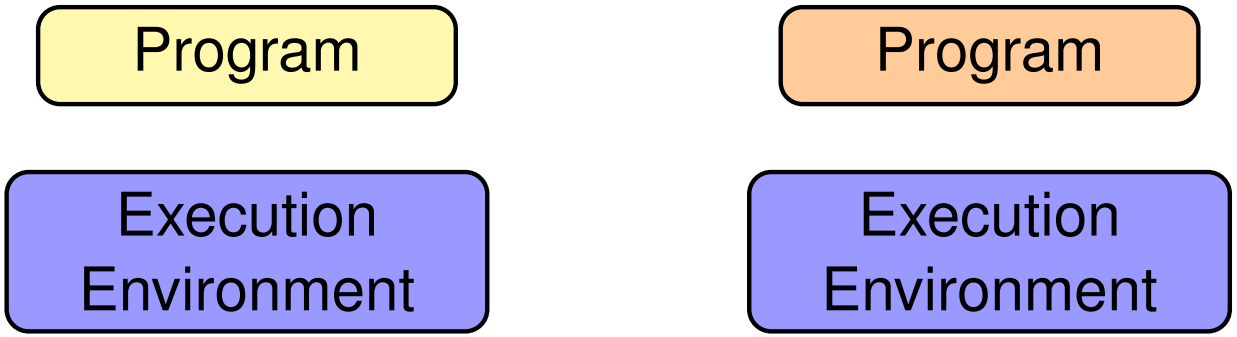
Software entities are more complex for their size than perhaps any other human construct - Frederick P. Brooks, Jr.



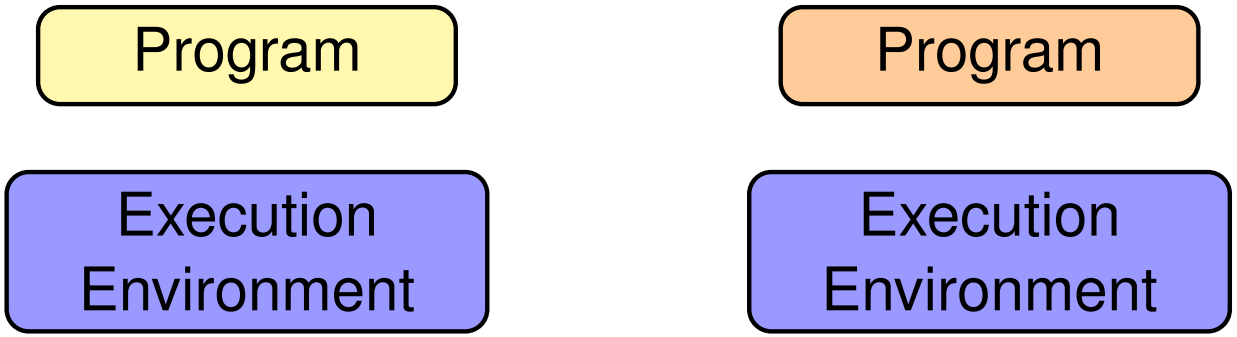
# Software is Evolving



# Software is Evolving

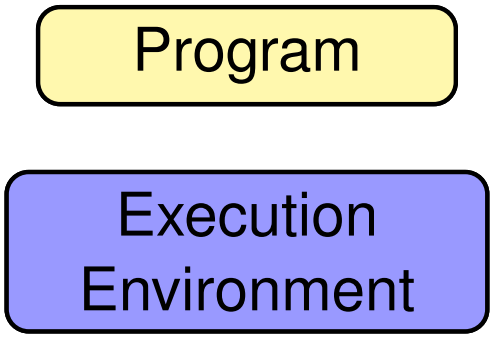


# Software is Evolving

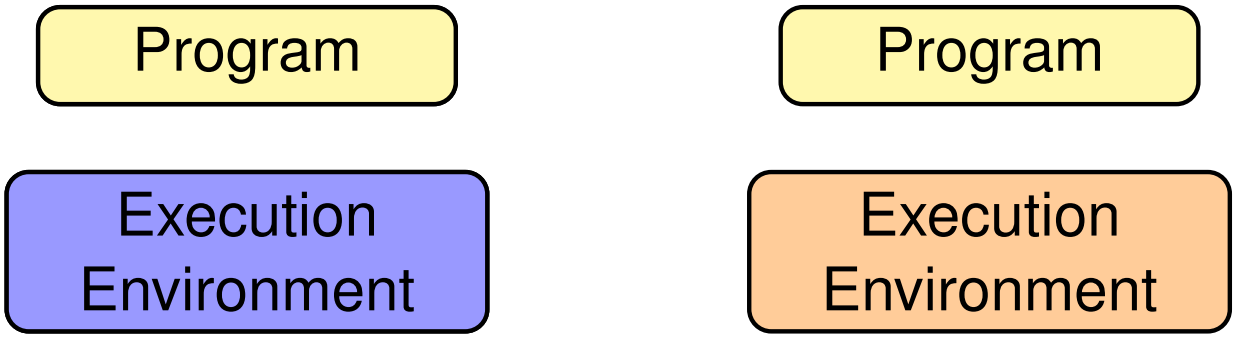



**Program Changed** because of the addition of a new feature or the correction of a defect

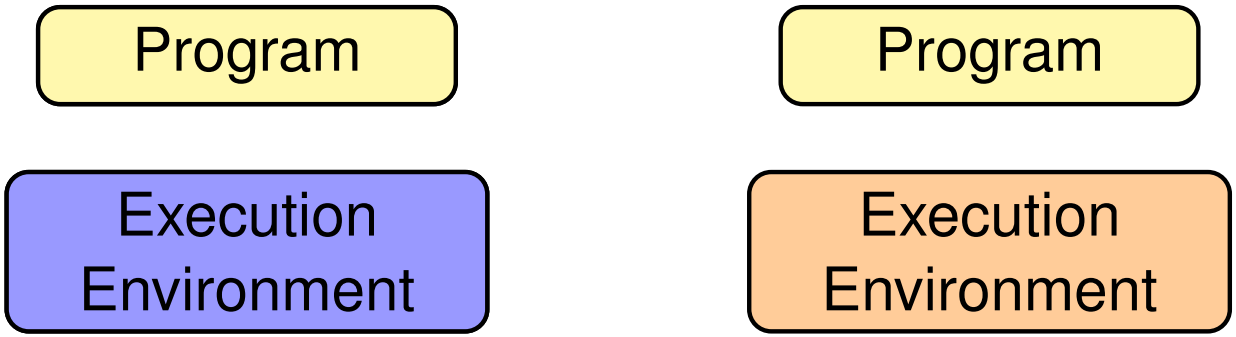
# Software is Evolving



# Software is Evolving



# Software is Evolving

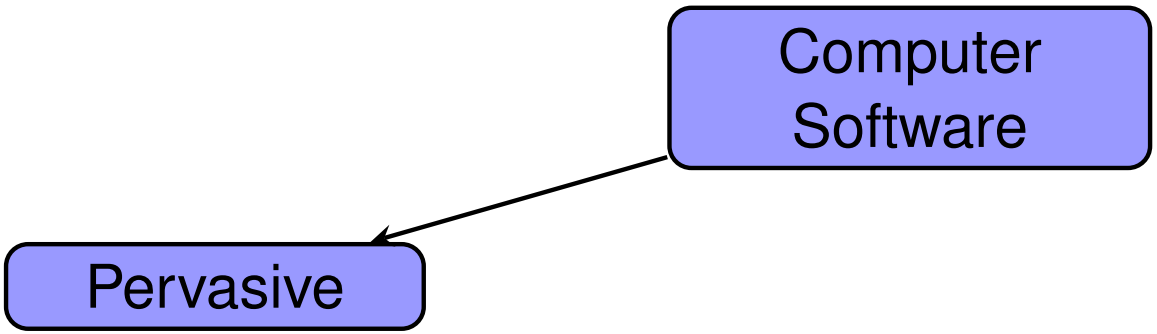



**Execution Environment Changed** due to an upgrade in a kernel, device driver, or virtual machine

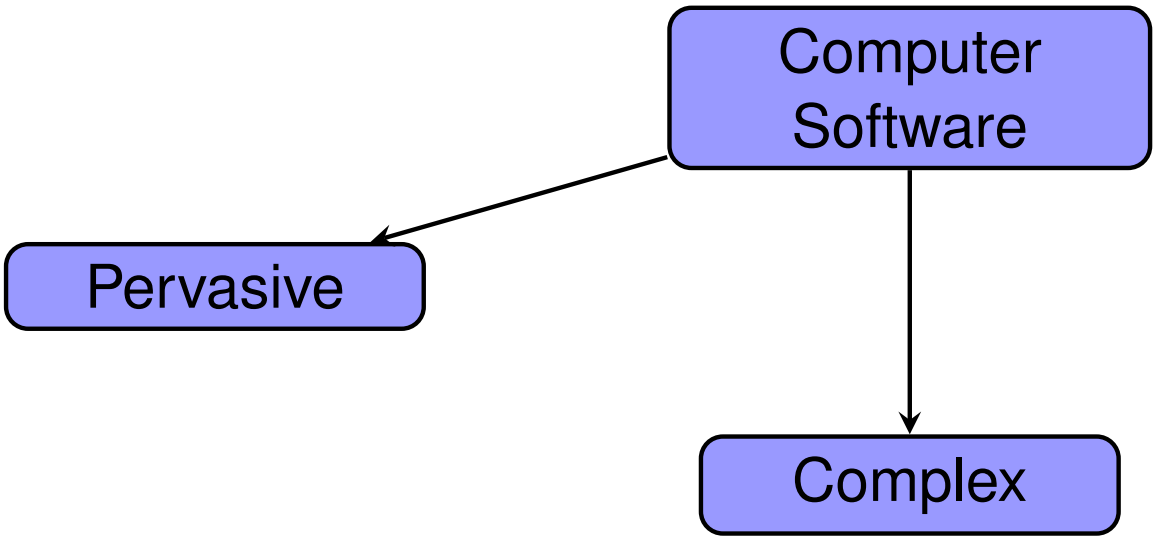
# Regression Testing to the Rescue

Computer Software

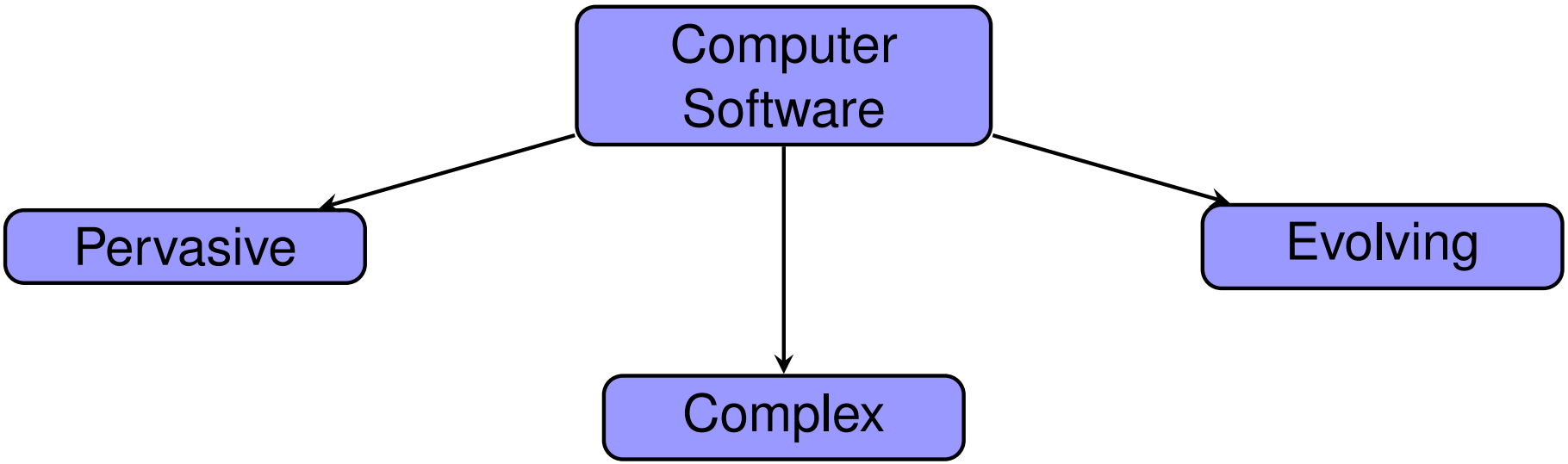
# Regression Testing to the Rescue



# Regression Testing to the Rescue

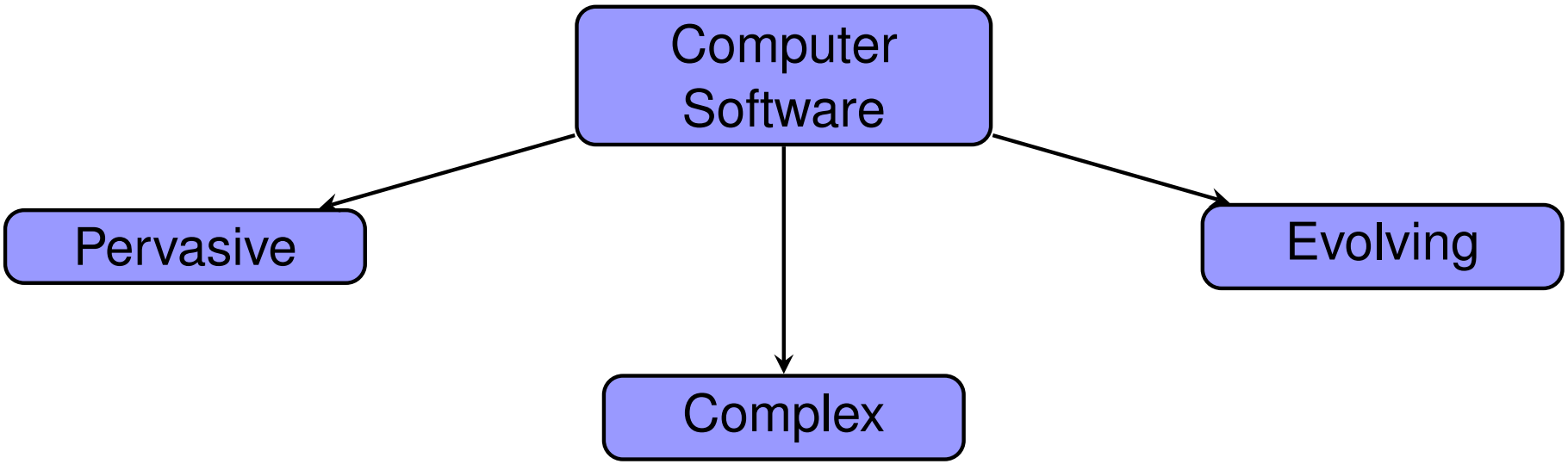


# Regression Testing to the Rescue

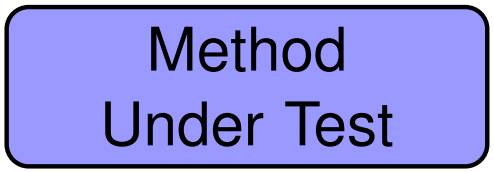


# Regression Testing to the Rescue

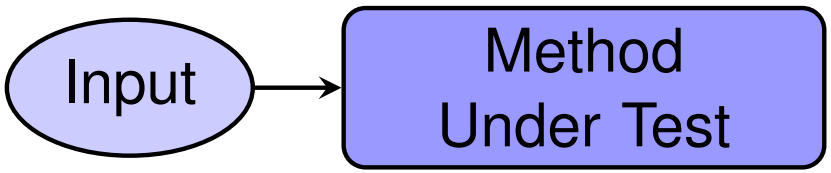
**Regression Testing** supports the efficient construction of pervasive software that is complex and rapidly evolving



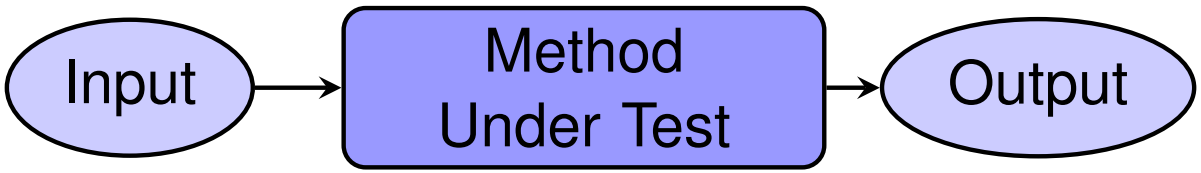
# What is a Test Case?



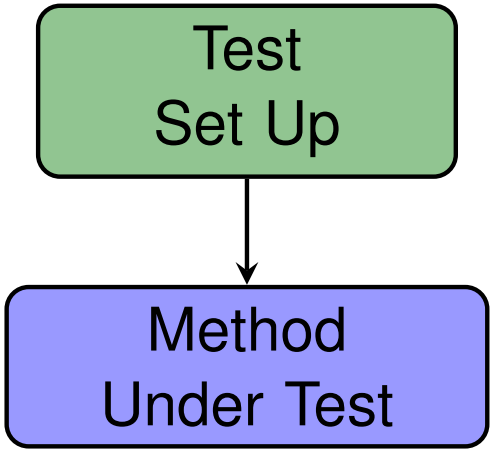
# What is a Test Case?



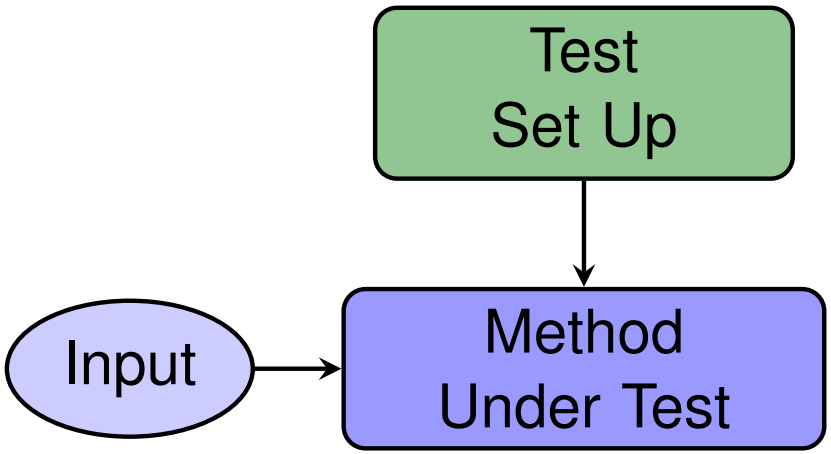
# What is a Test Case?



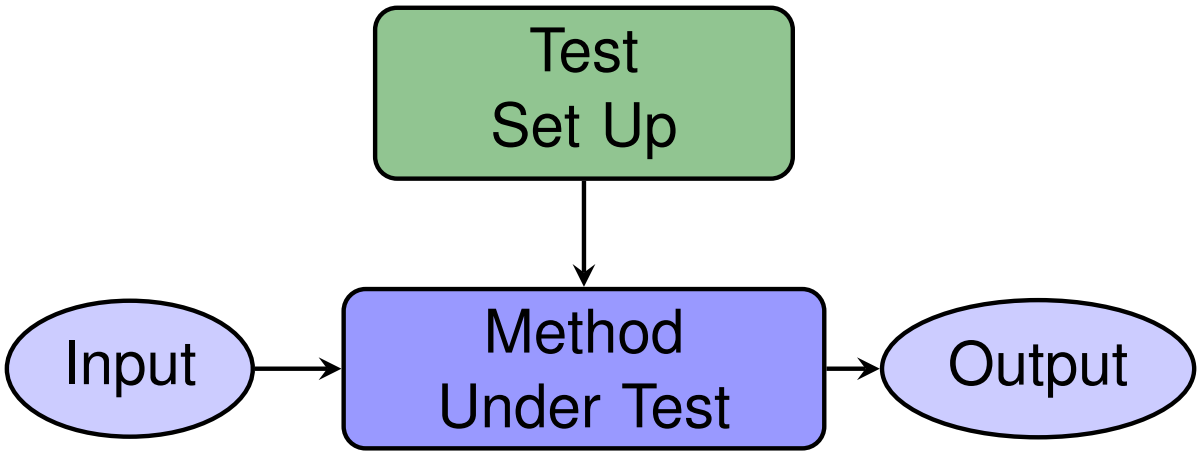
# What is a Test Case?



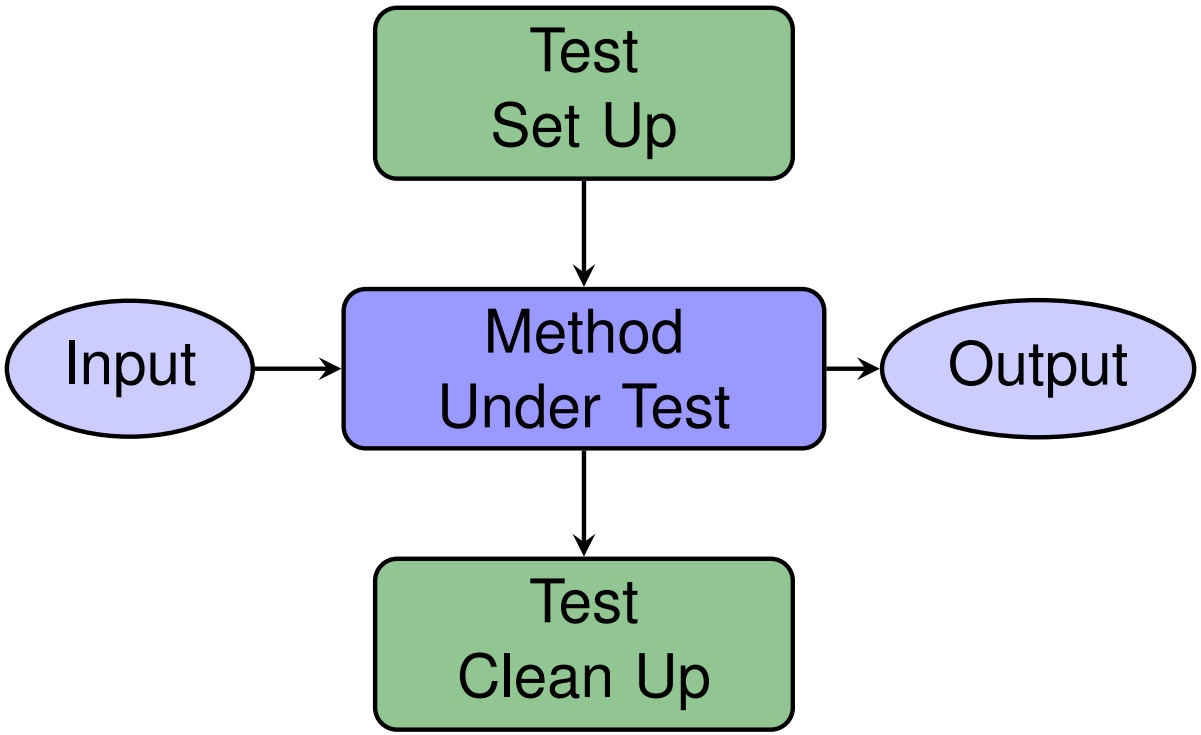
# What is a Test Case?



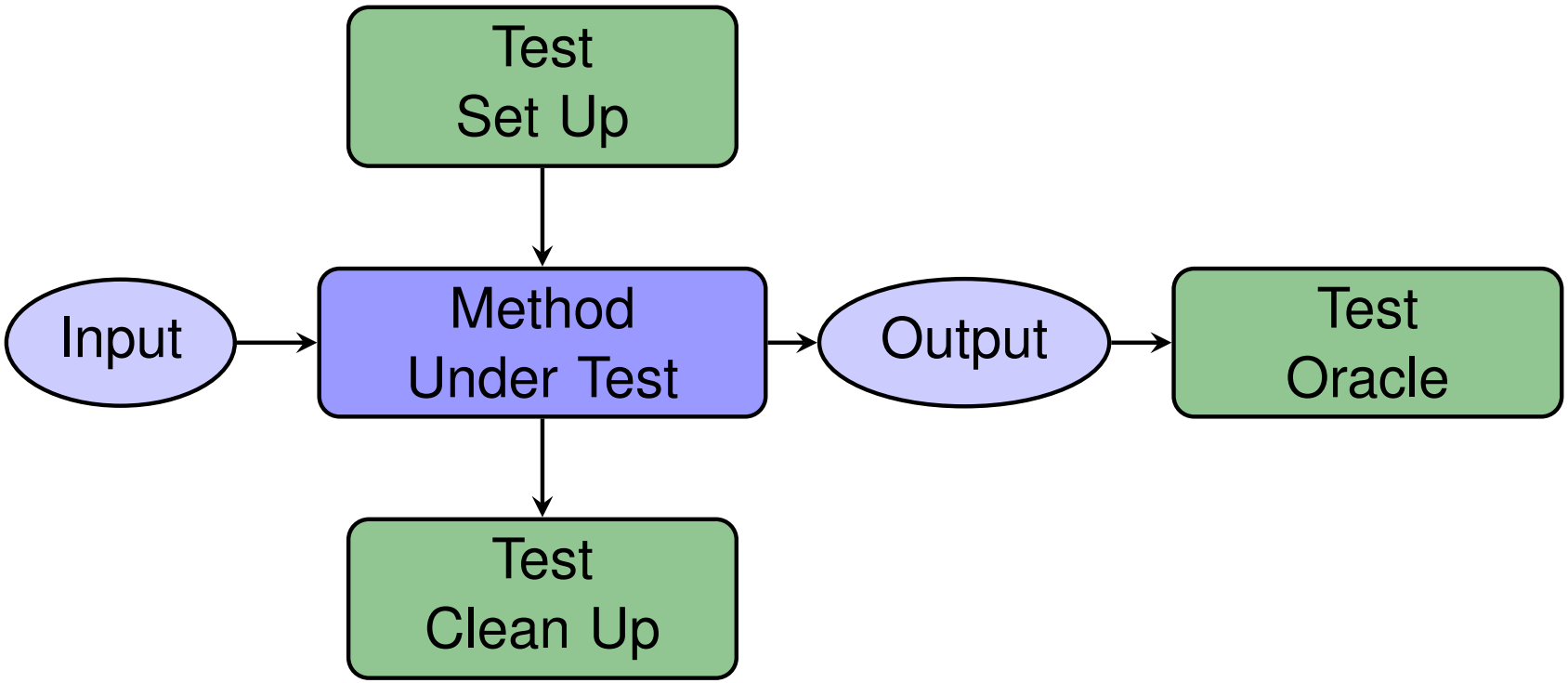
# What is a Test Case?



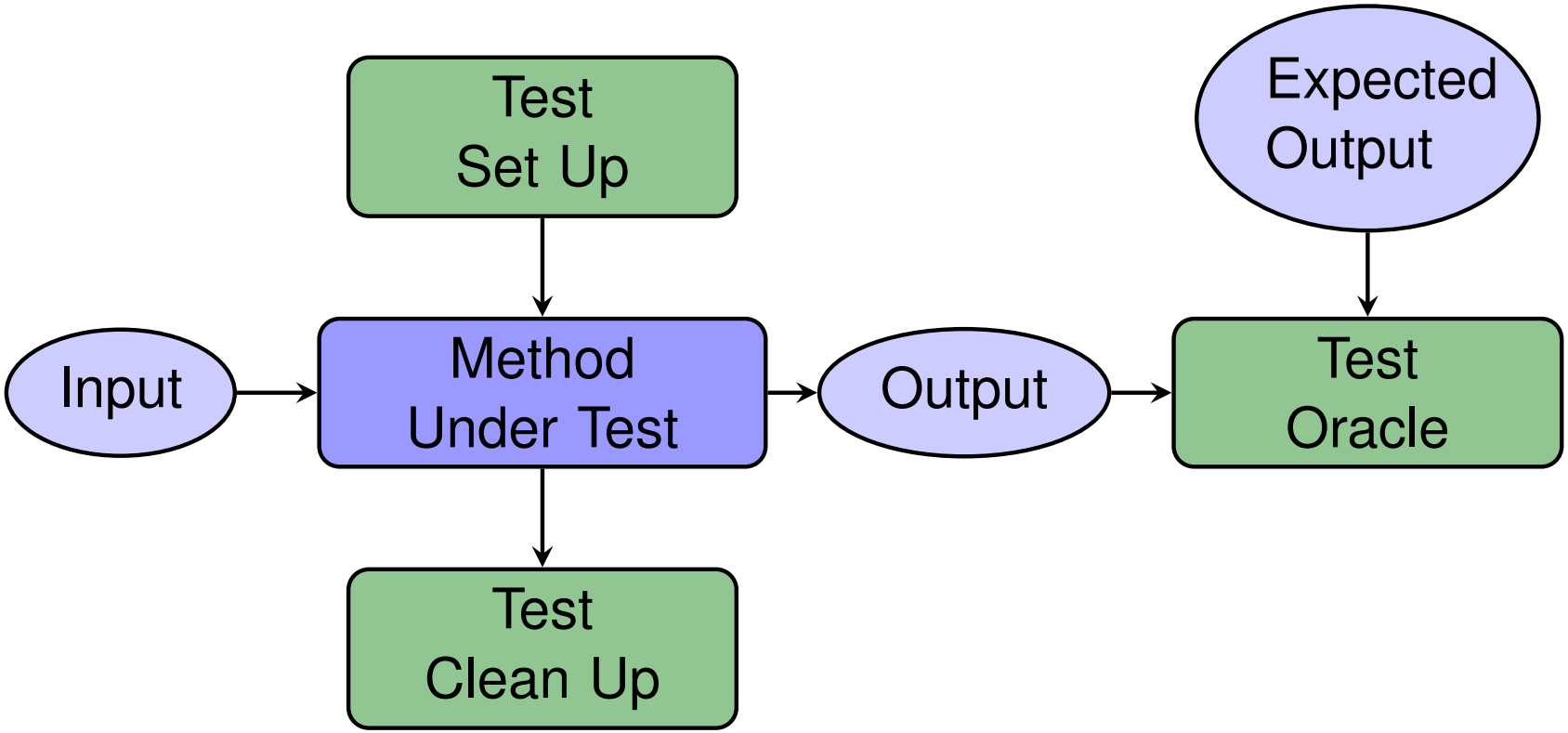
# What is a Test Case?



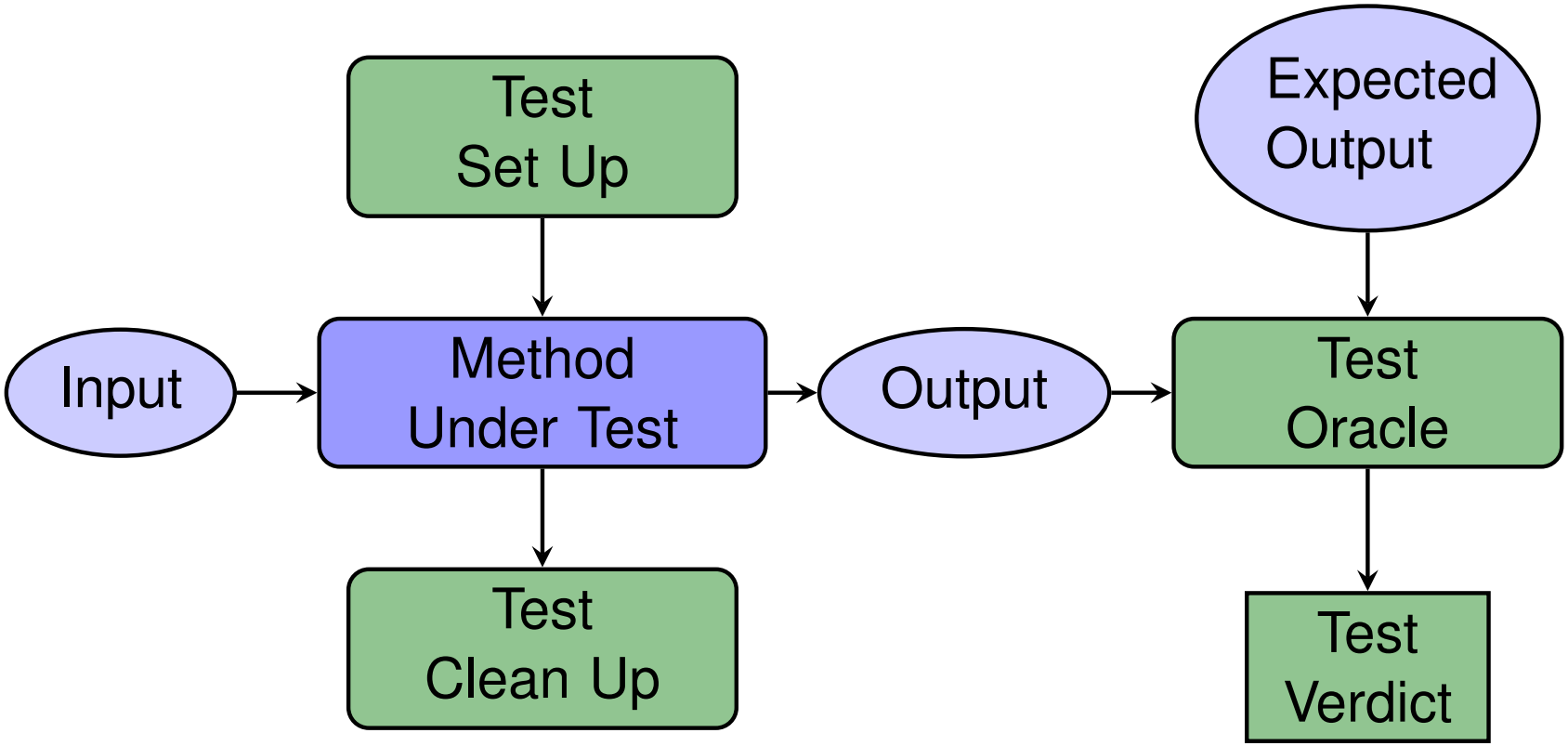
# What is a Test Case?



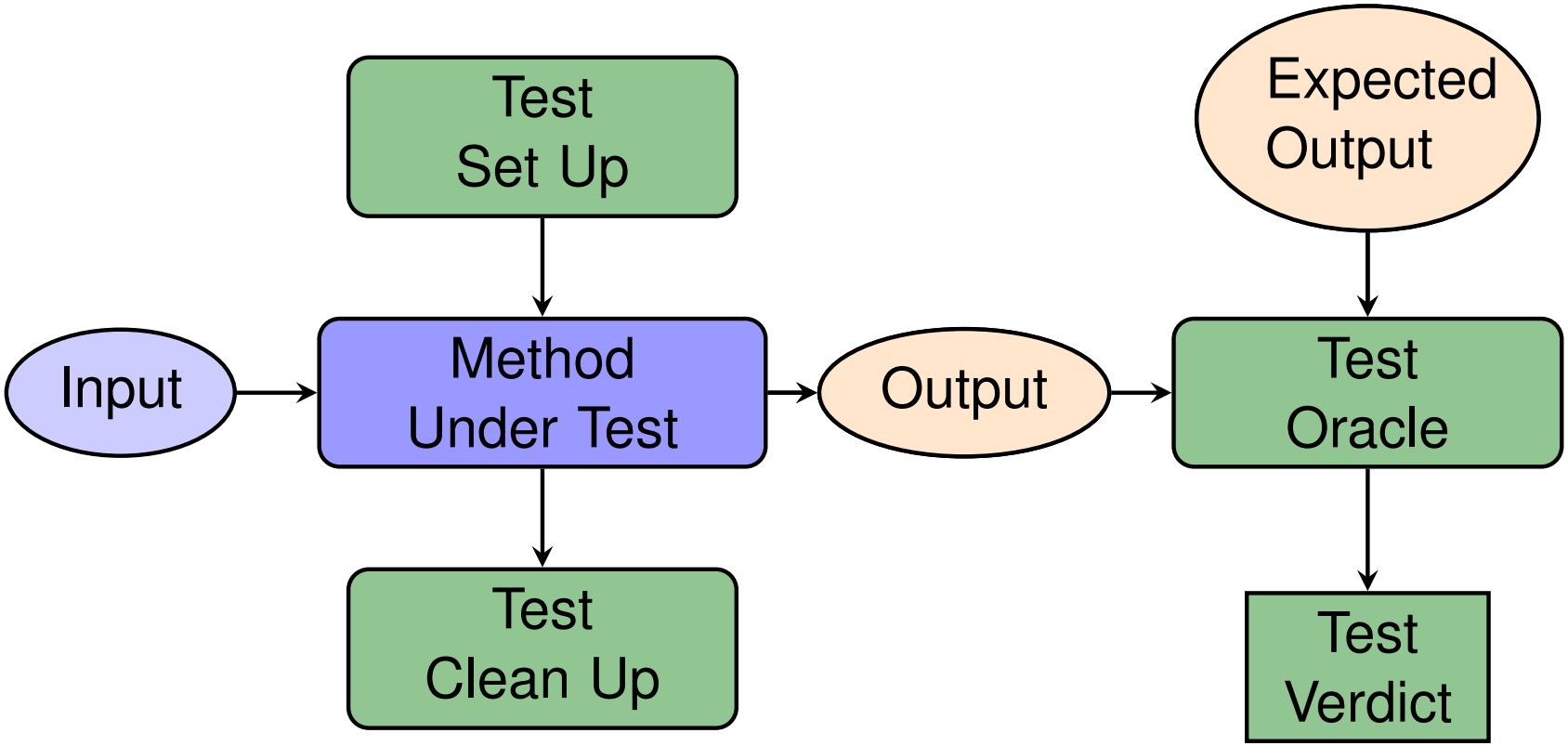
# What is a Test Case?



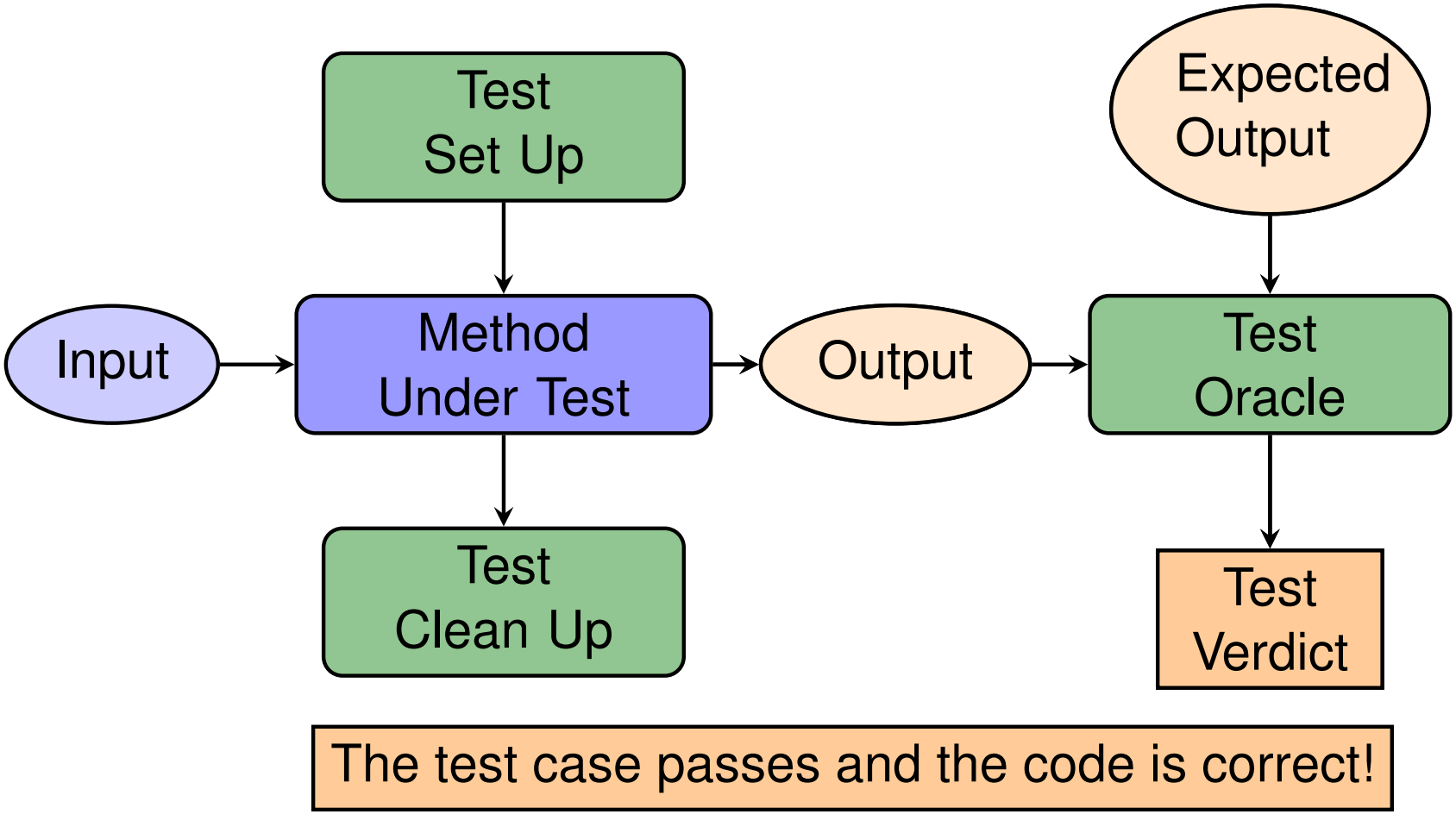
# What is a Test Case?



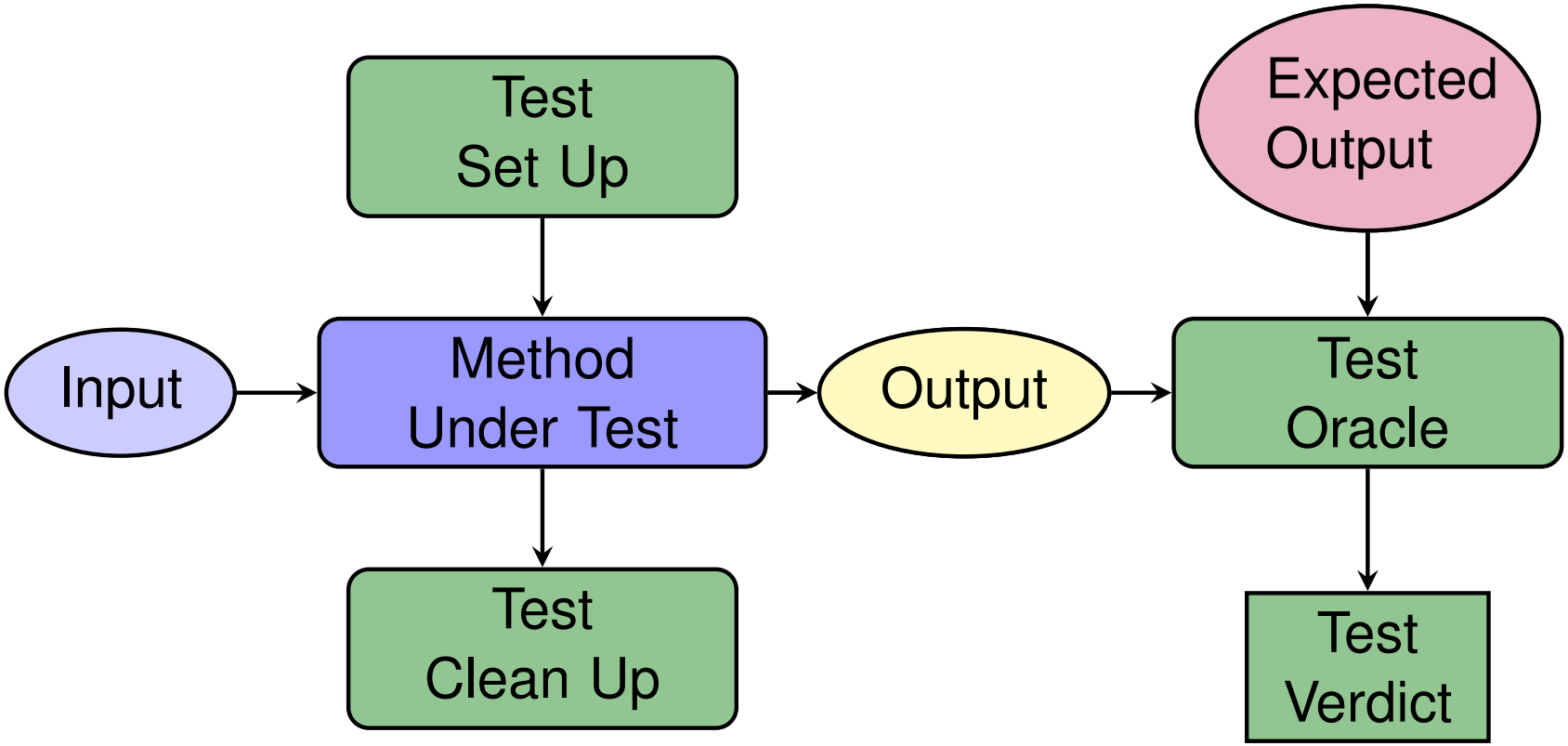
# What is a Test Case?



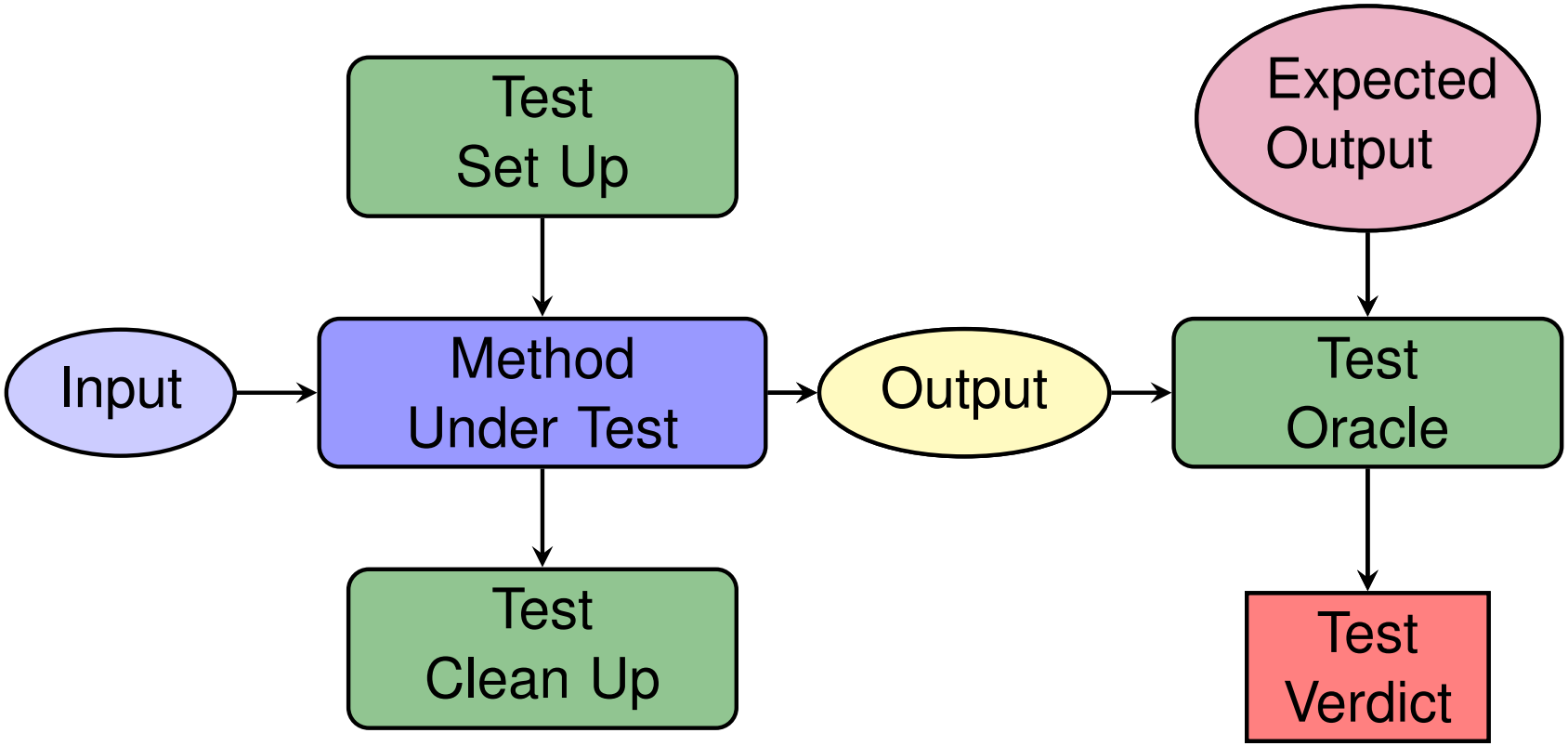
# What is a Test Case?



# What is a Test Case?



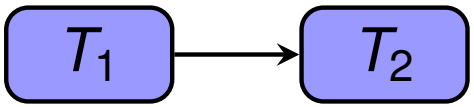
# What is a Test Case?



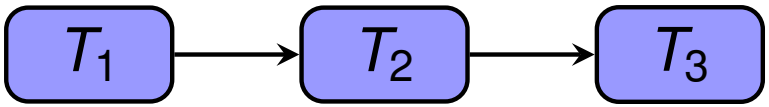
# What is a Test Suite?

$T_1$

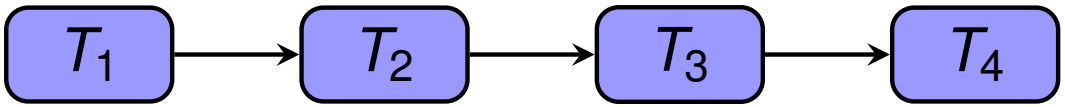
# What is a Test Suite?



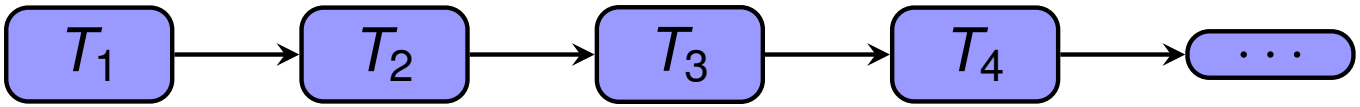
# What is a Test Suite?



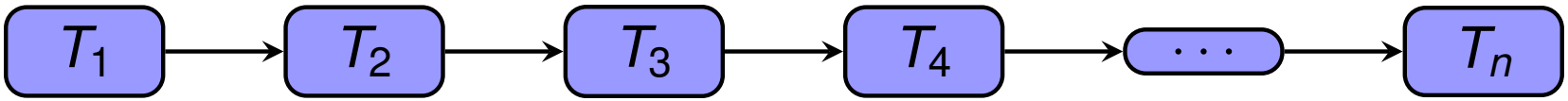
# What is a Test Suite?



# What is a Test Suite?

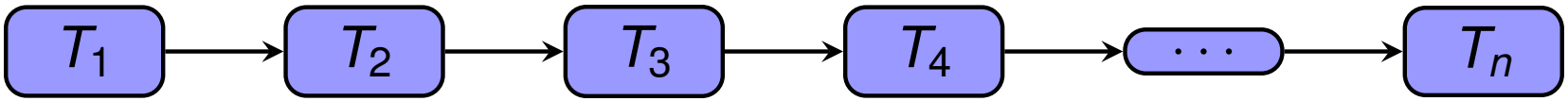


# What is a Test Suite?



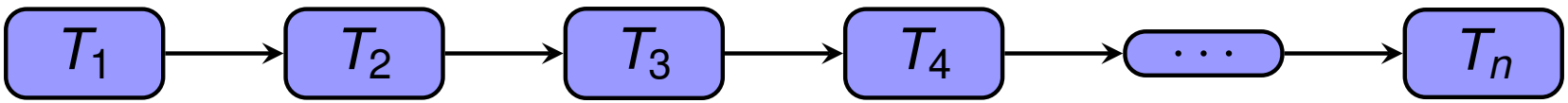
# What is a Test Suite?

Organize the Test Cases into a Test Suite



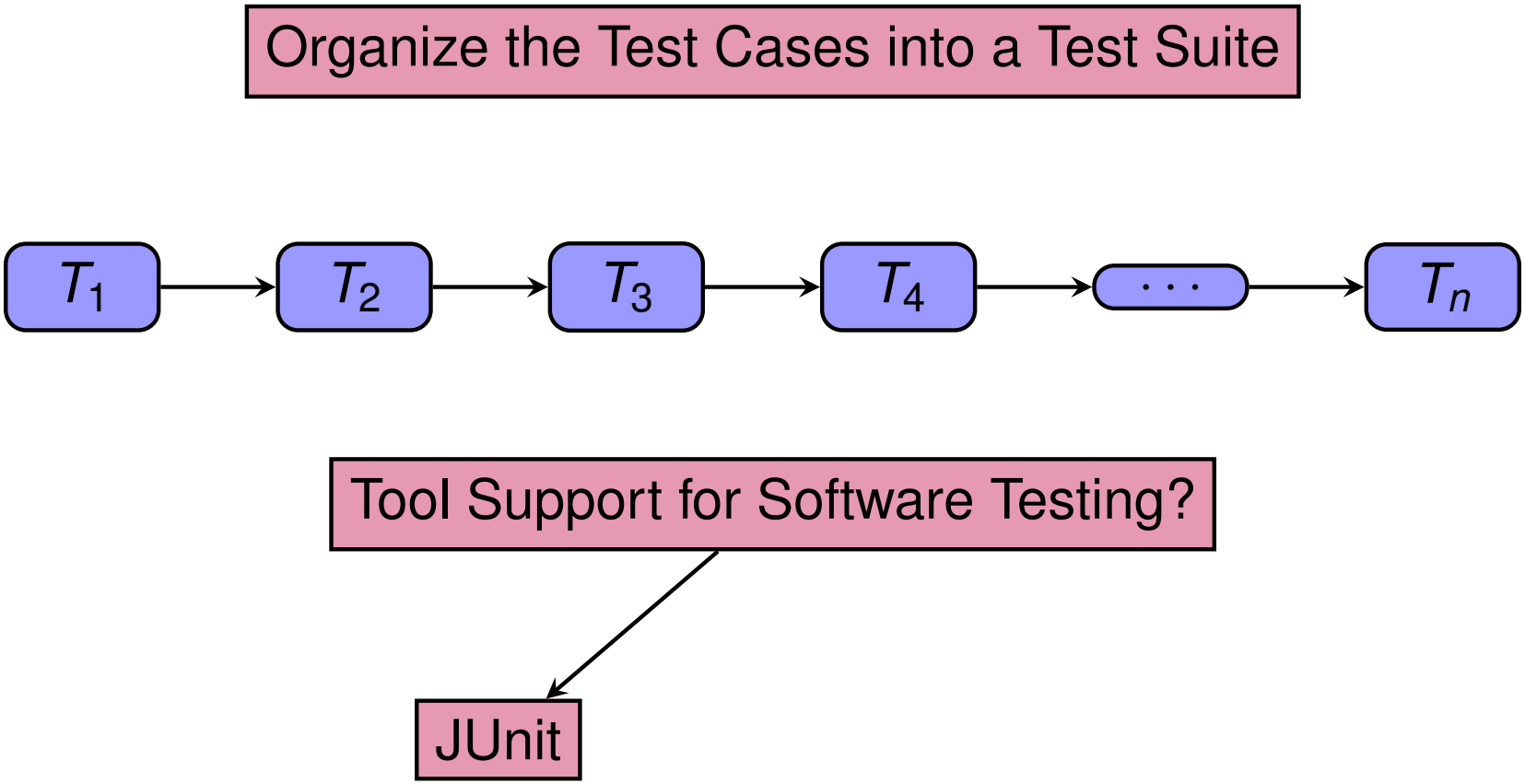
# What is a Test Suite?

Organize the Test Cases into a Test Suite

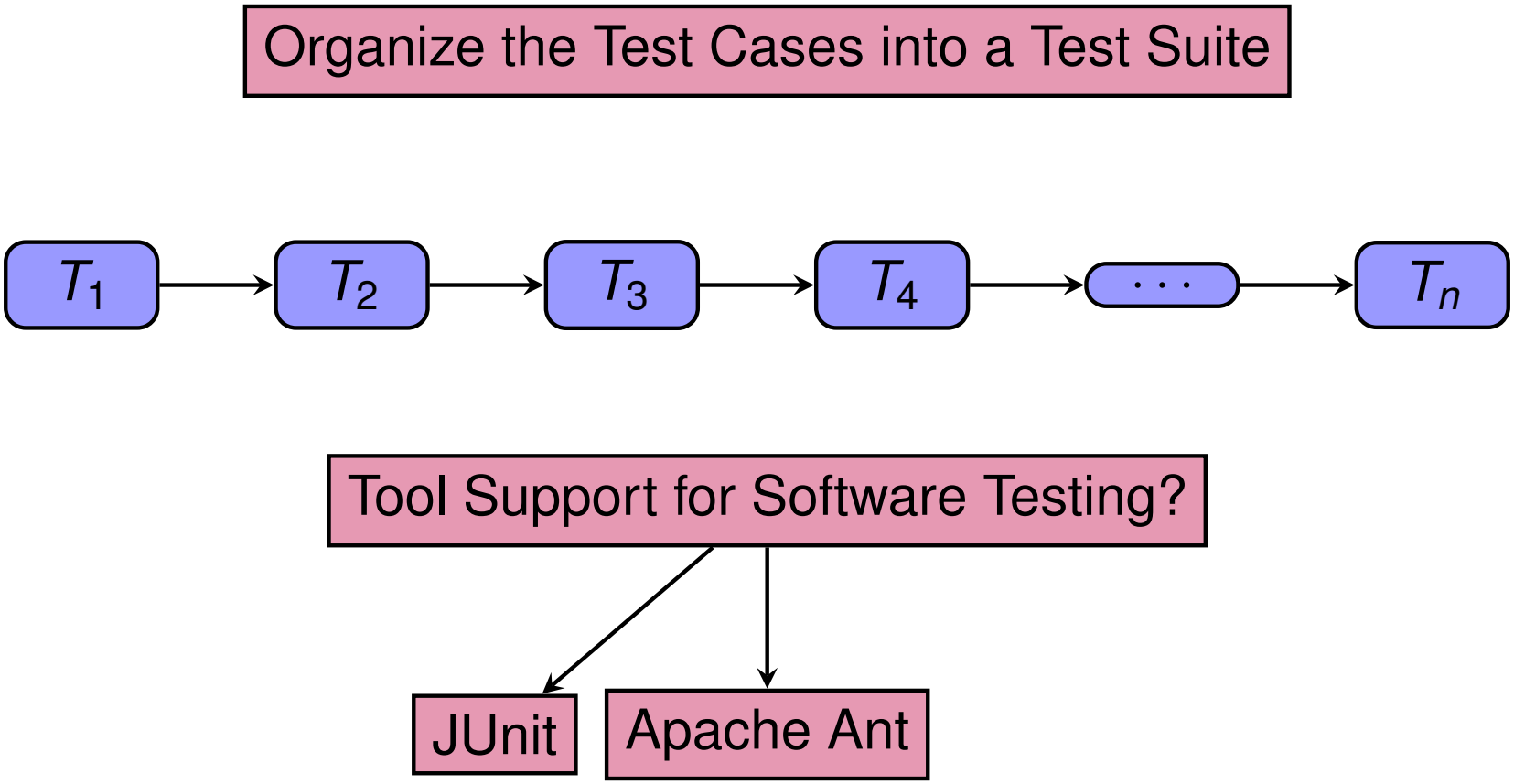



Tool Support for Software Testing?

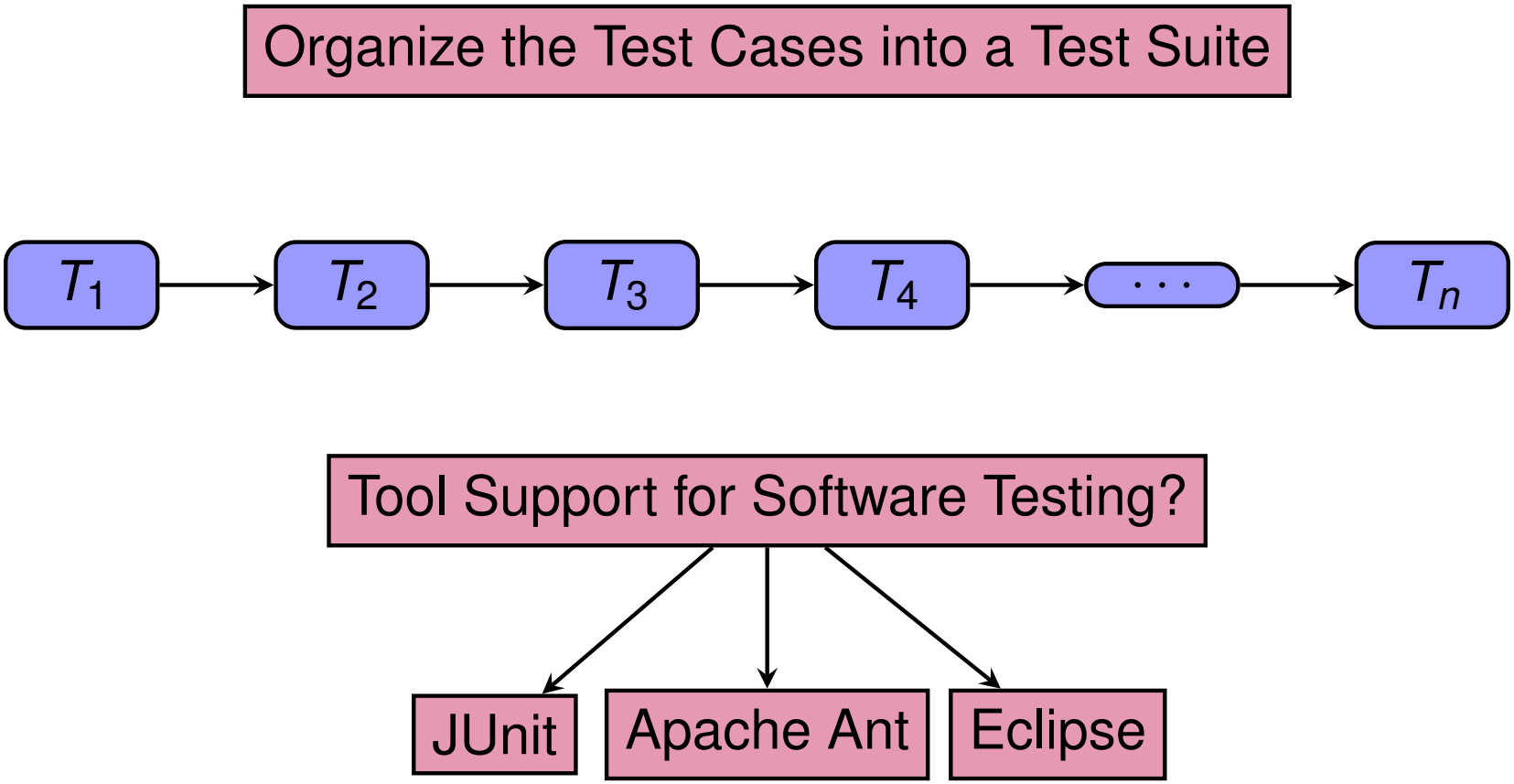
# What is a Test Suite?



# What is a Test Suite?

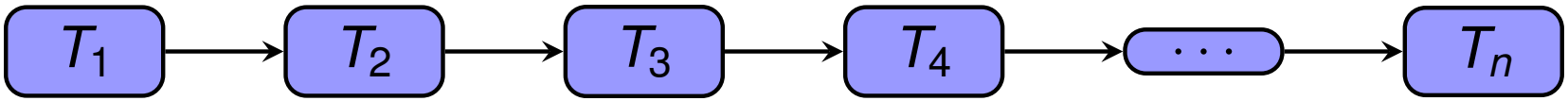


# What is a Test Suite?

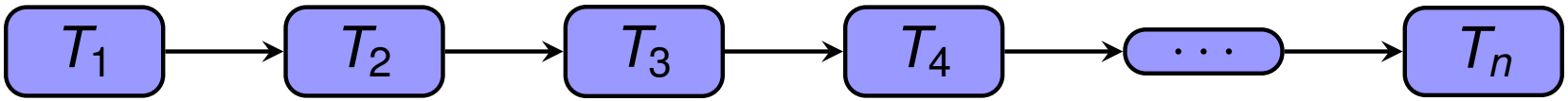


# Test Suite Management

Organize the Test Cases into a Test Suite



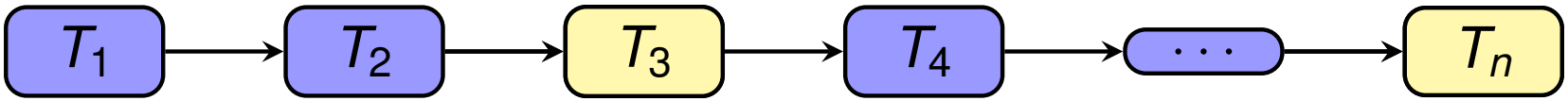
# Test Suite Management




Regression Testing Technique

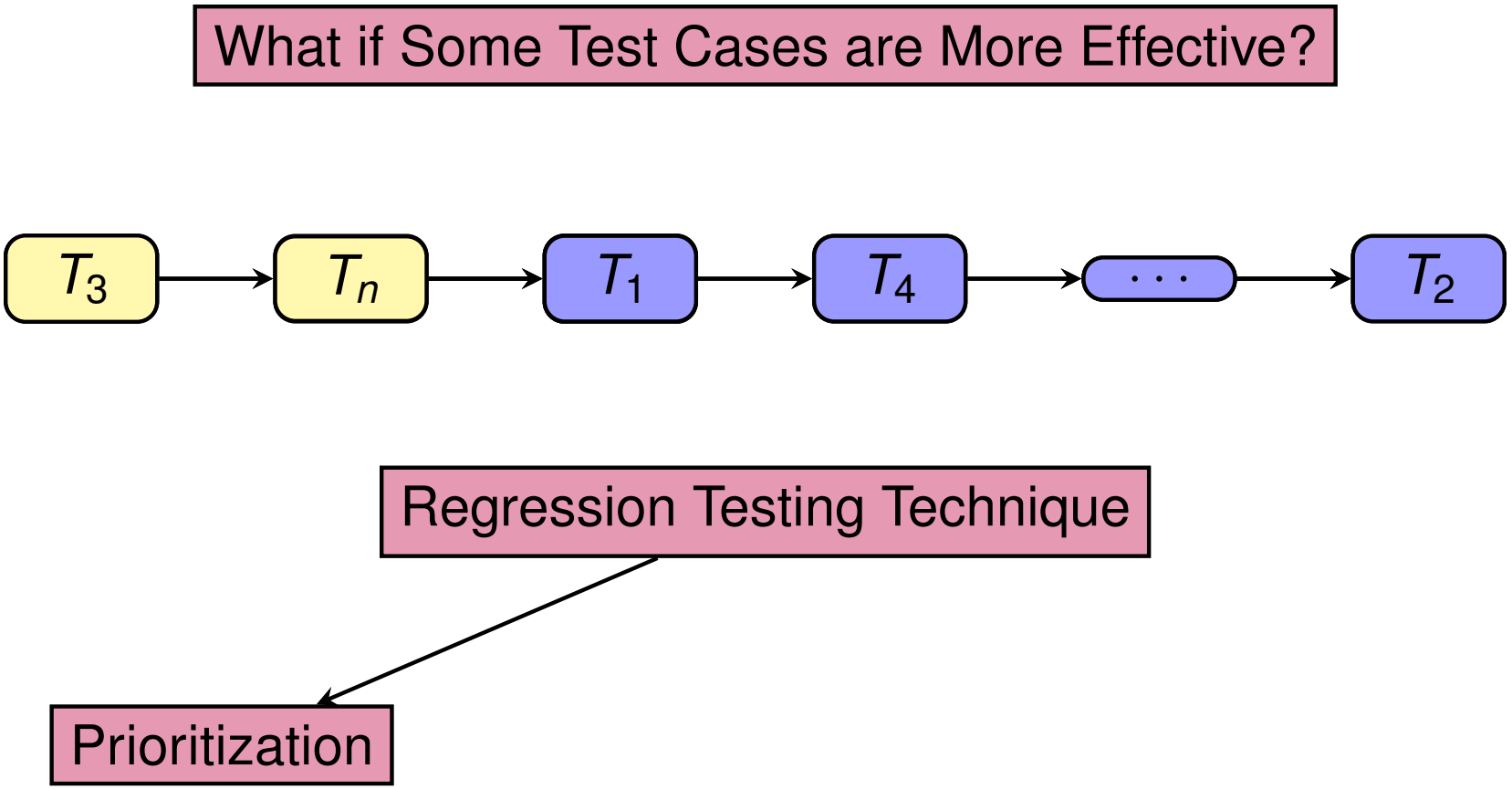
# Test Suite Management

What if Some Test Cases are More Effective?

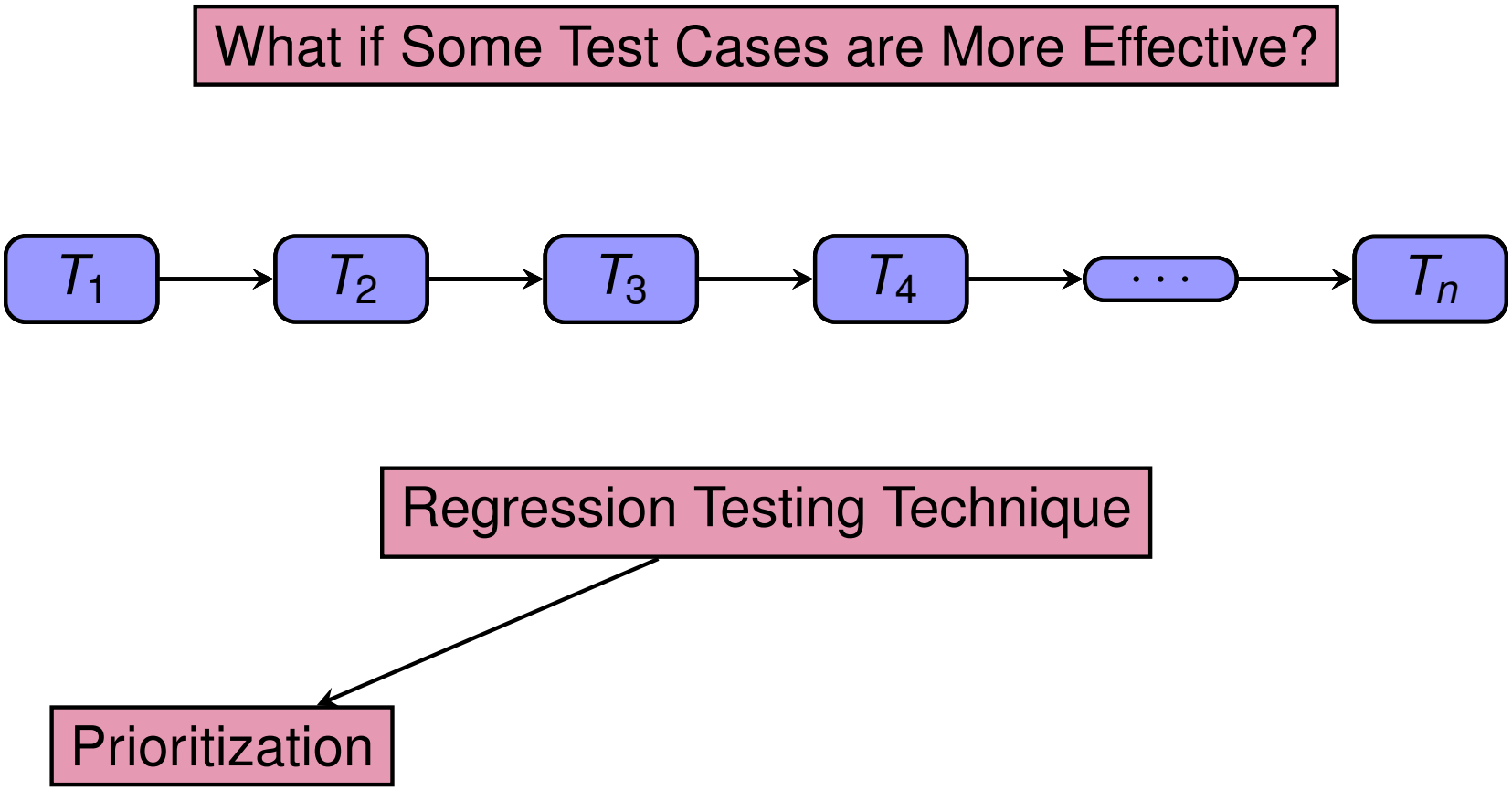



Regression Testing Technique

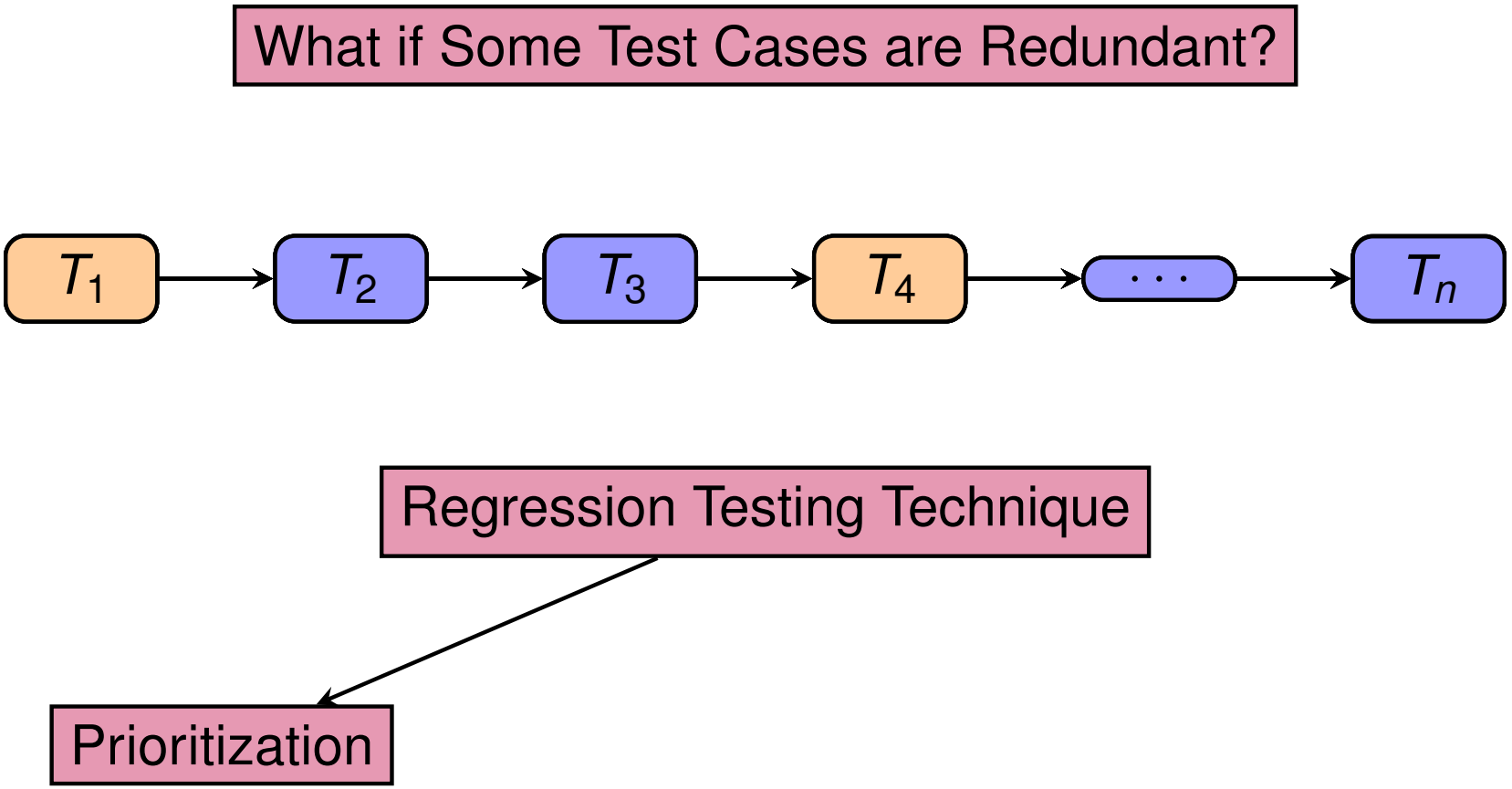
# Test Suite Management



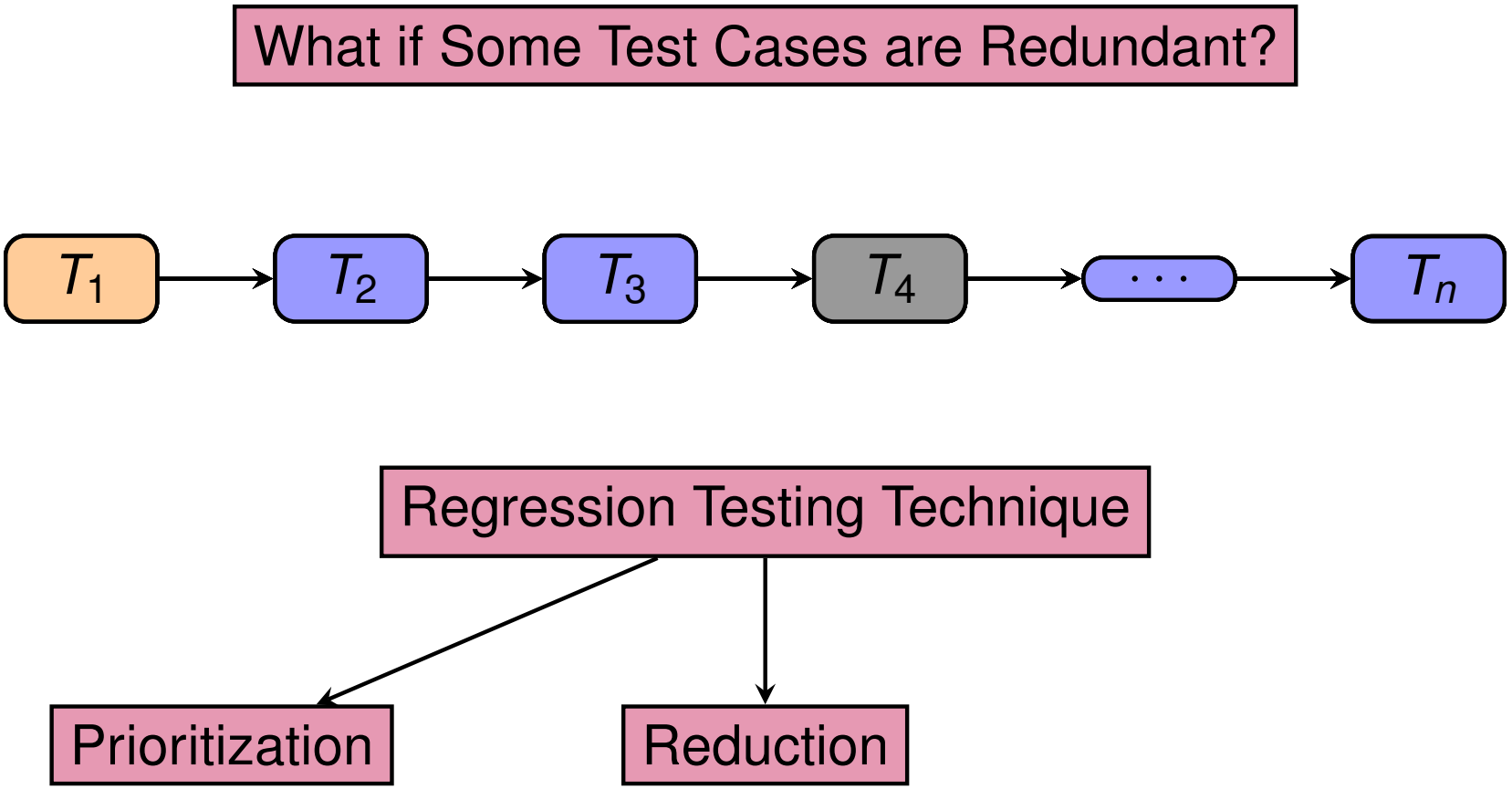
# Test Suite Management



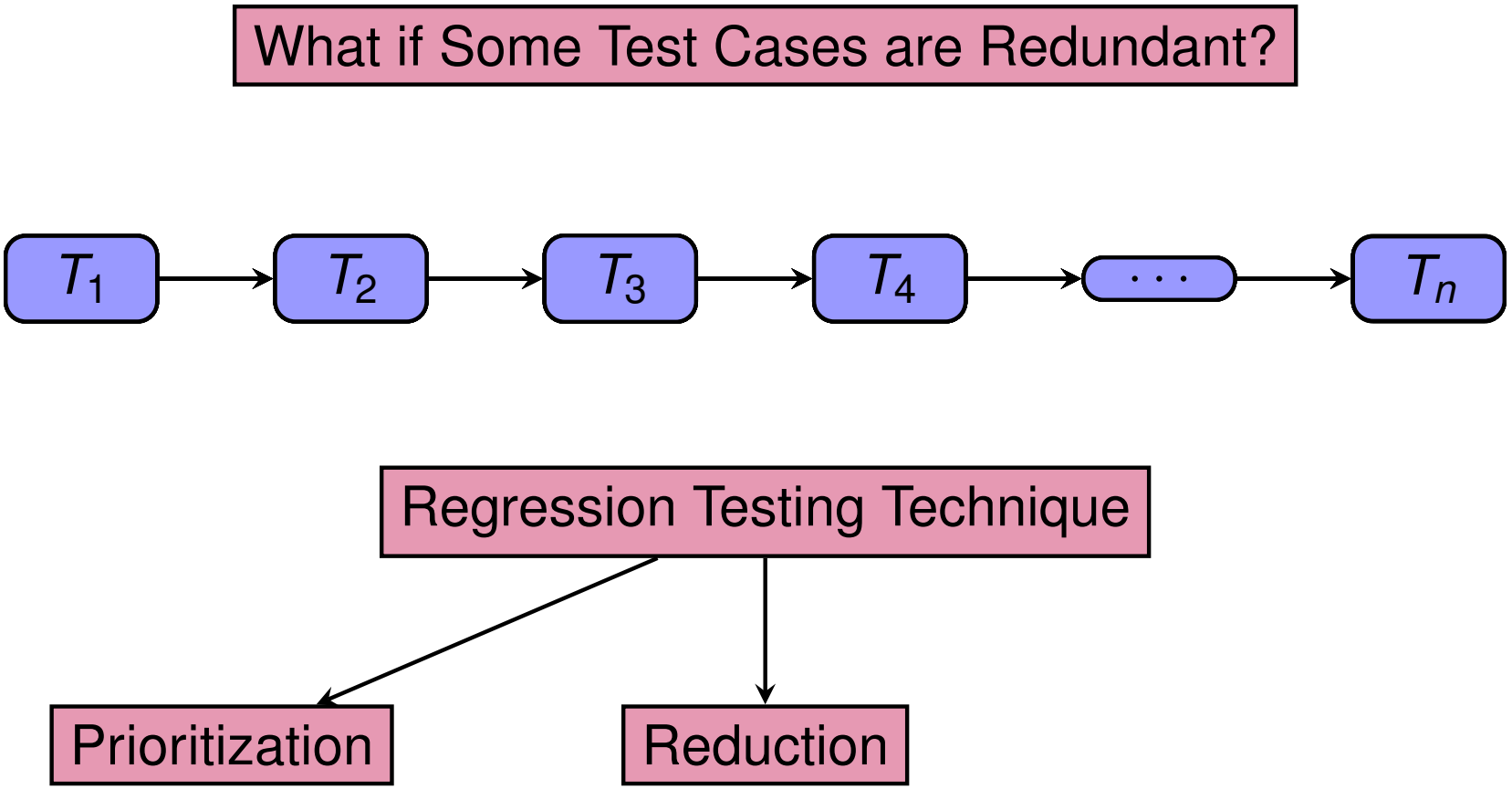
# Test Suite Management



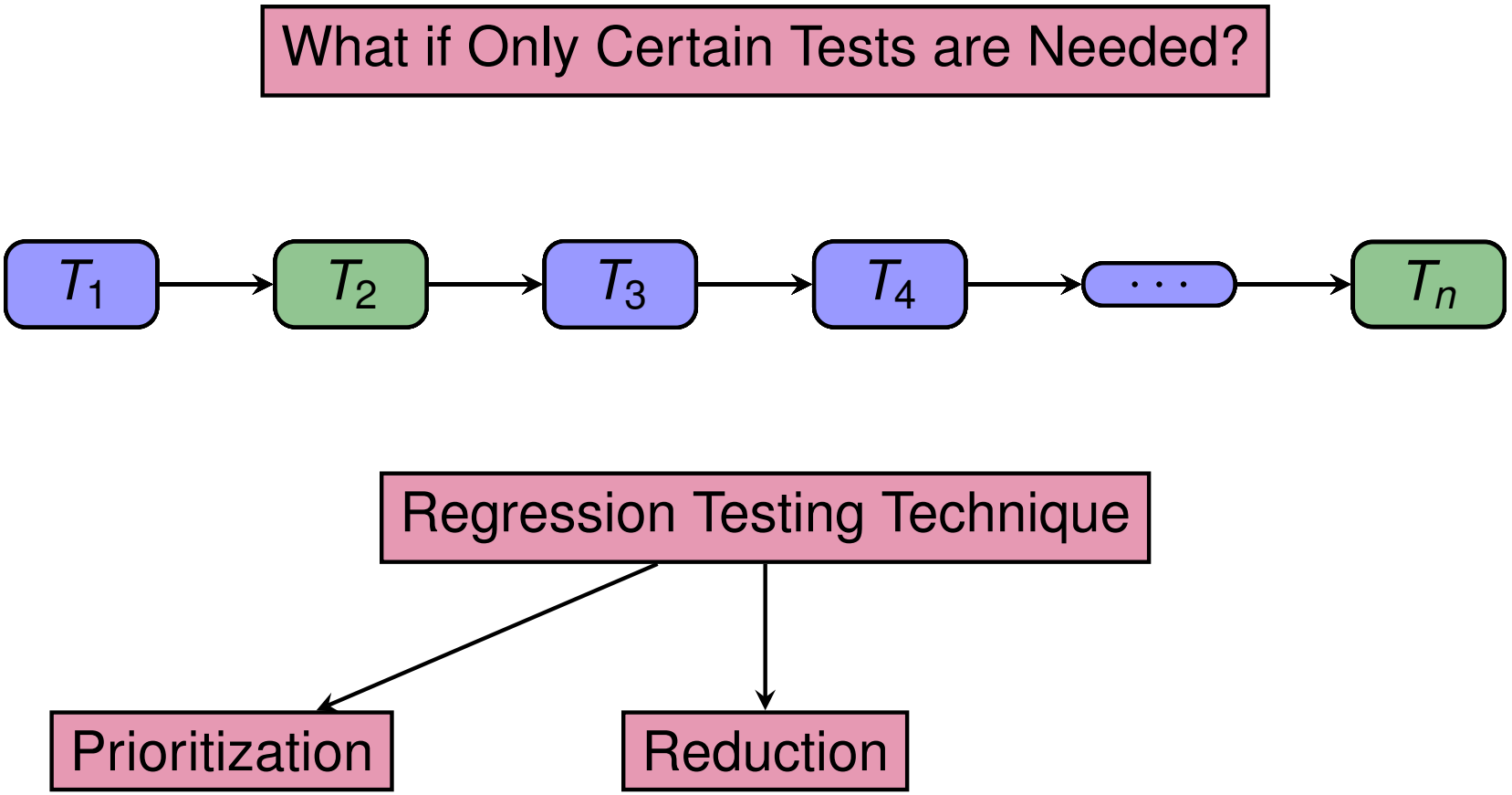
# Test Suite Management



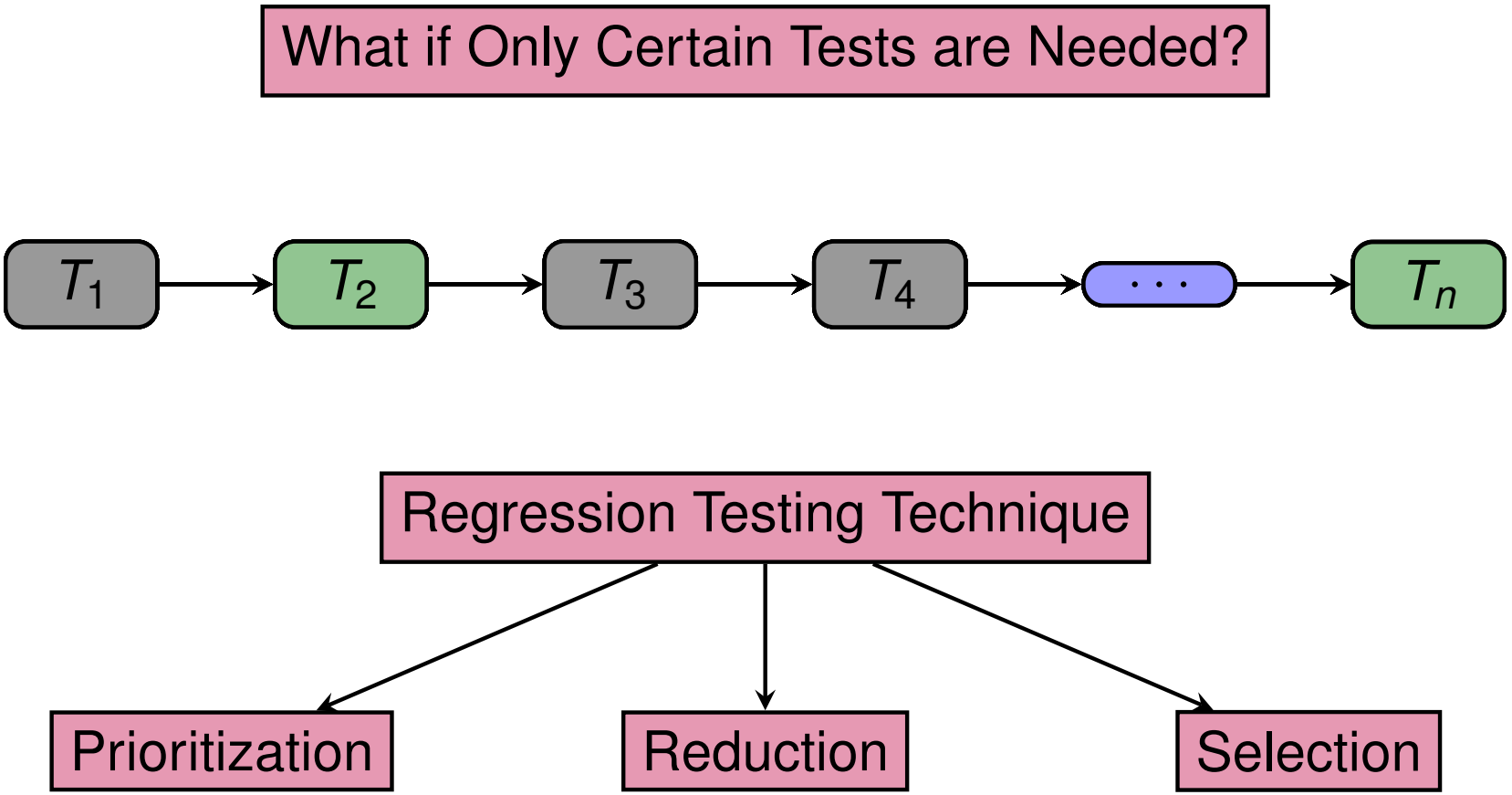
# Test Suite Management



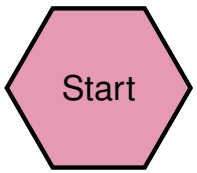
# Test Suite Management



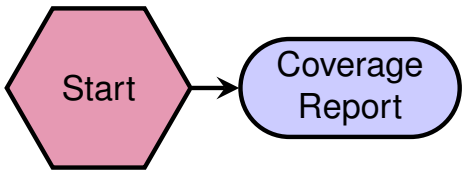
# Test Suite Management



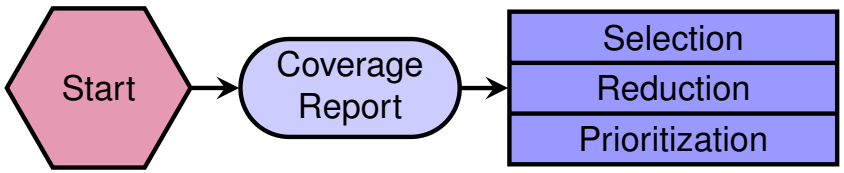
# Model of Regression Testing



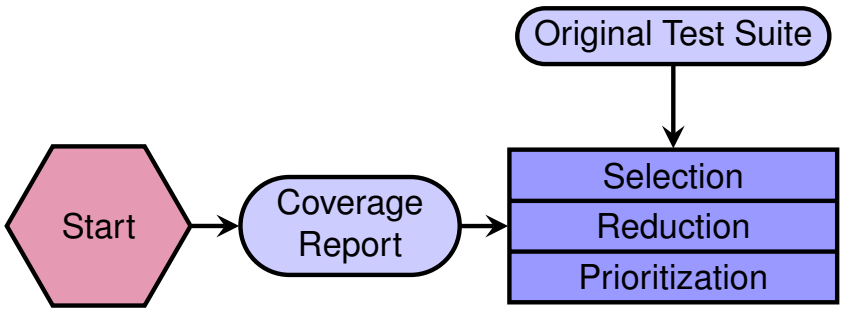
# Model of Regression Testing



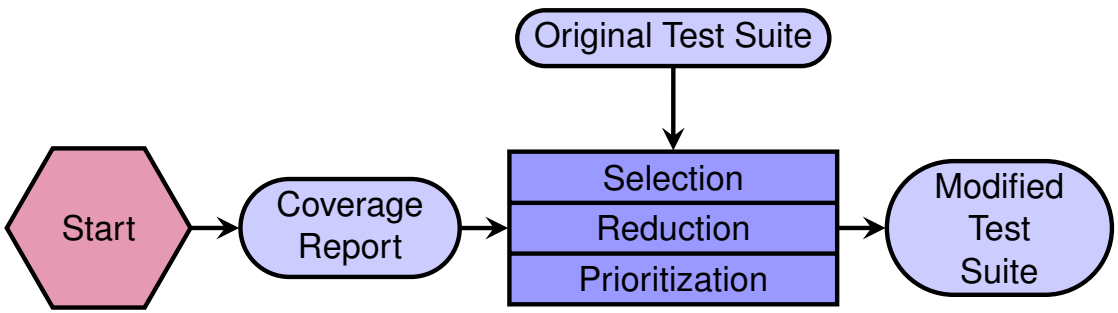
# Model of Regression Testing



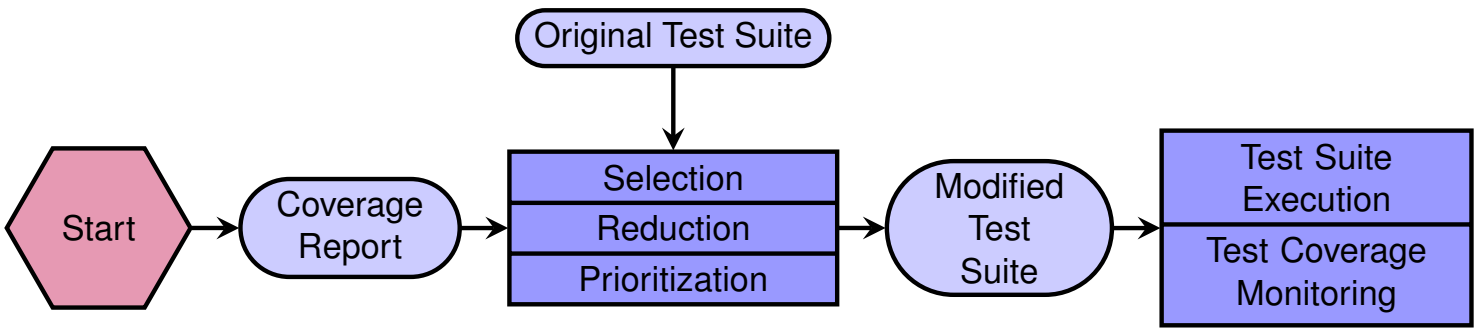
# Model of Regression Testing



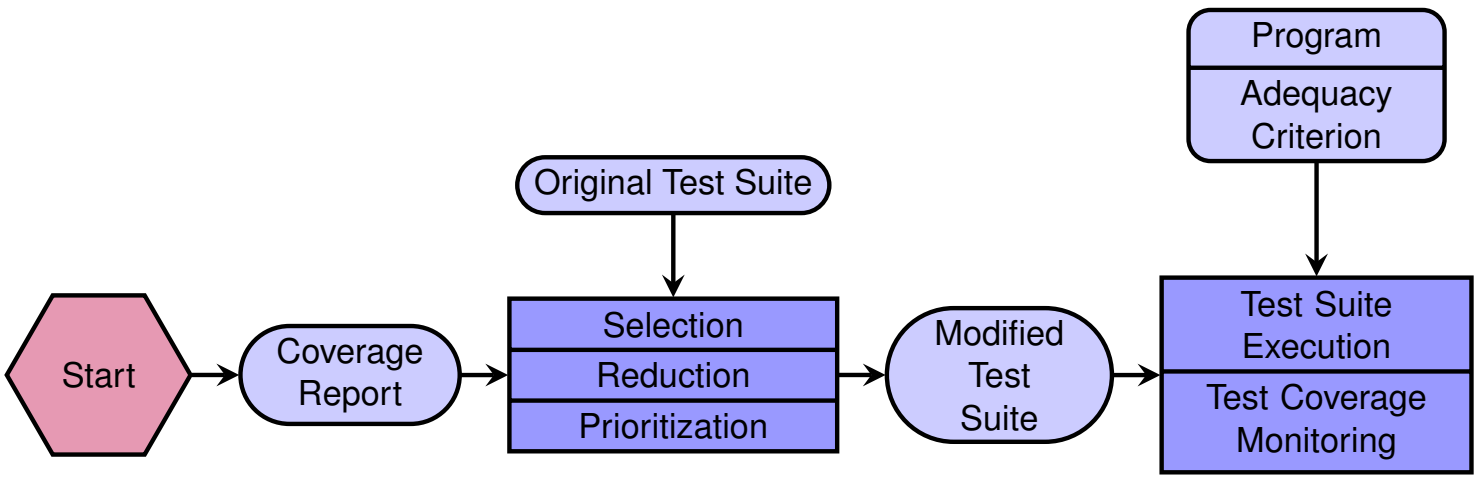
# Model of Regression Testing



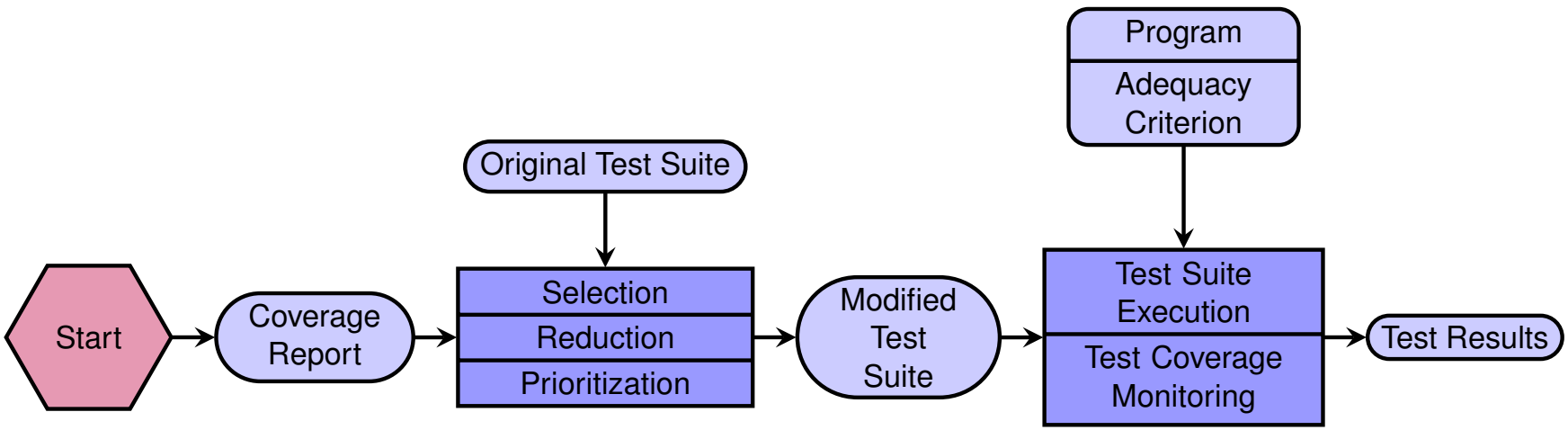
# Model of Regression Testing



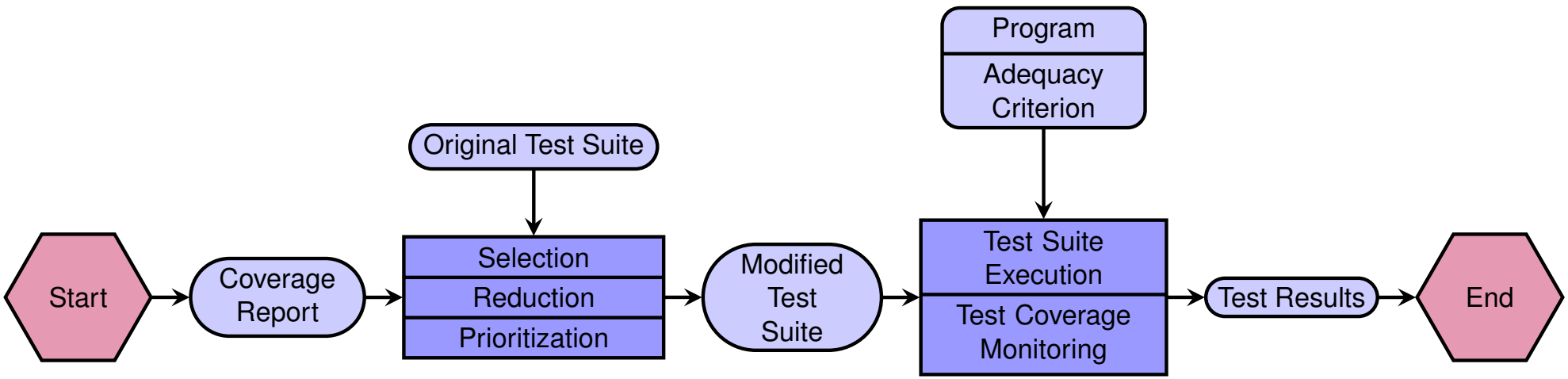
# Model of Regression Testing



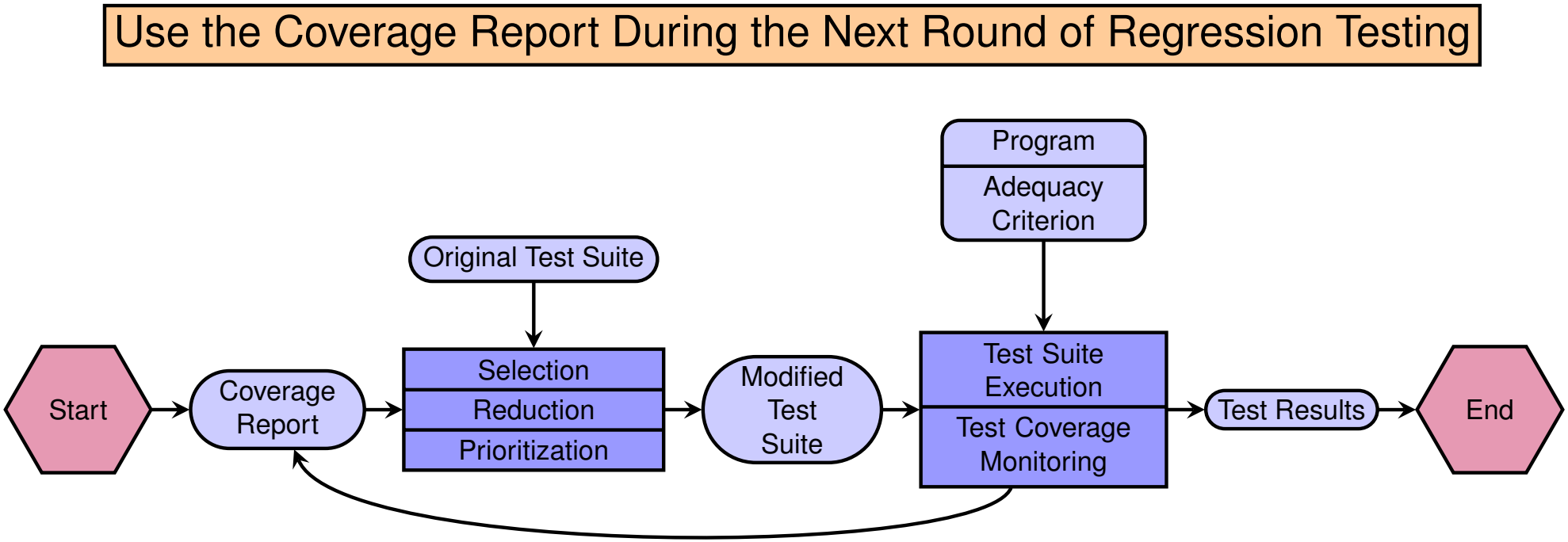
# Model of Regression Testing



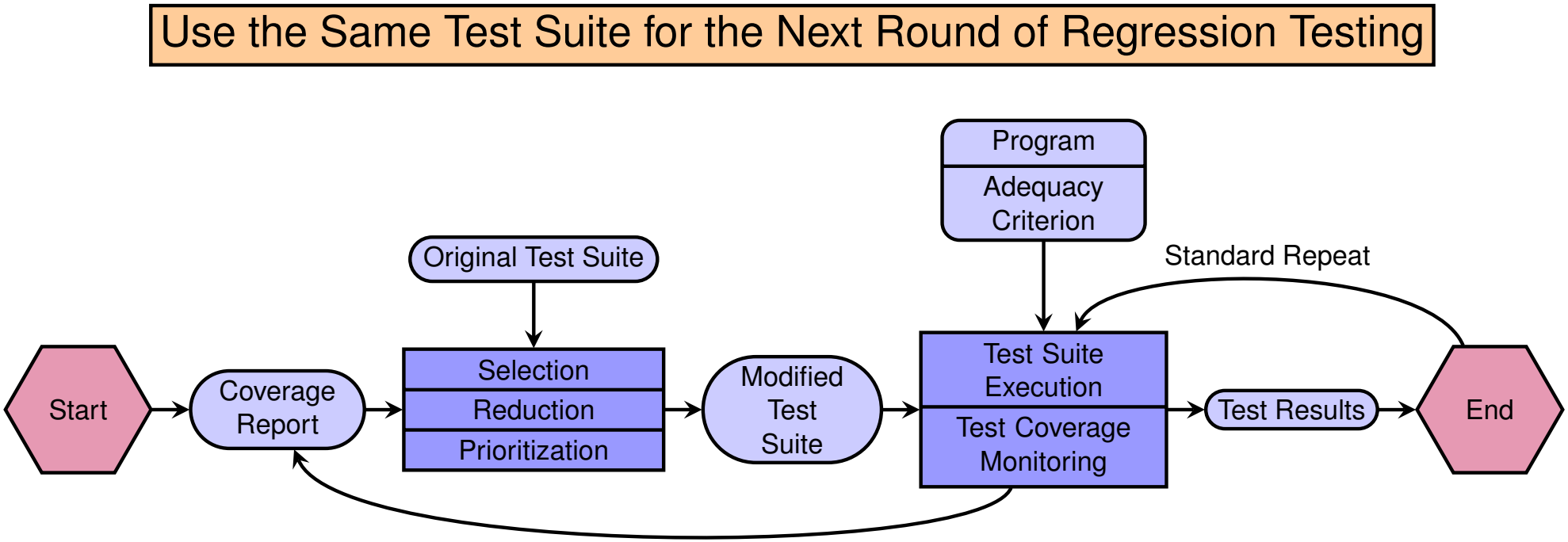
# Model of Regression Testing



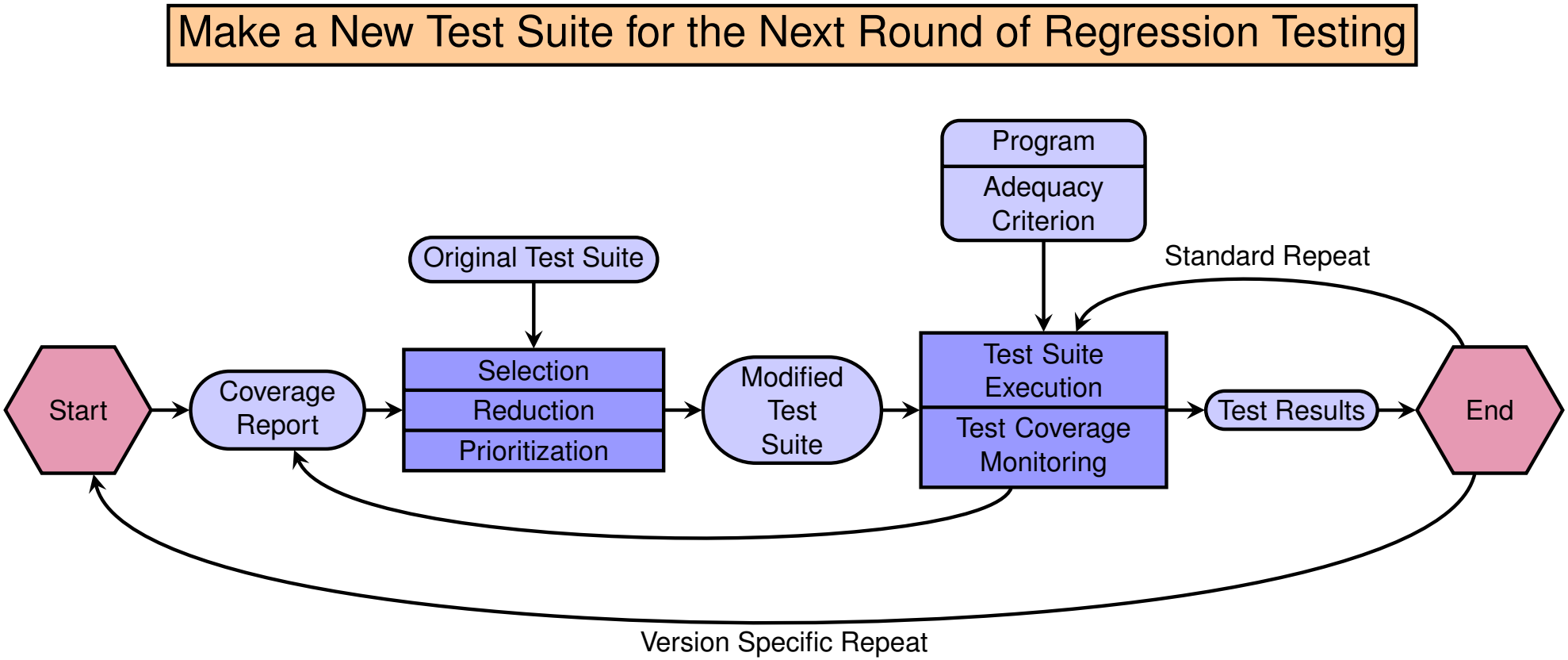
# Model of Regression Testing



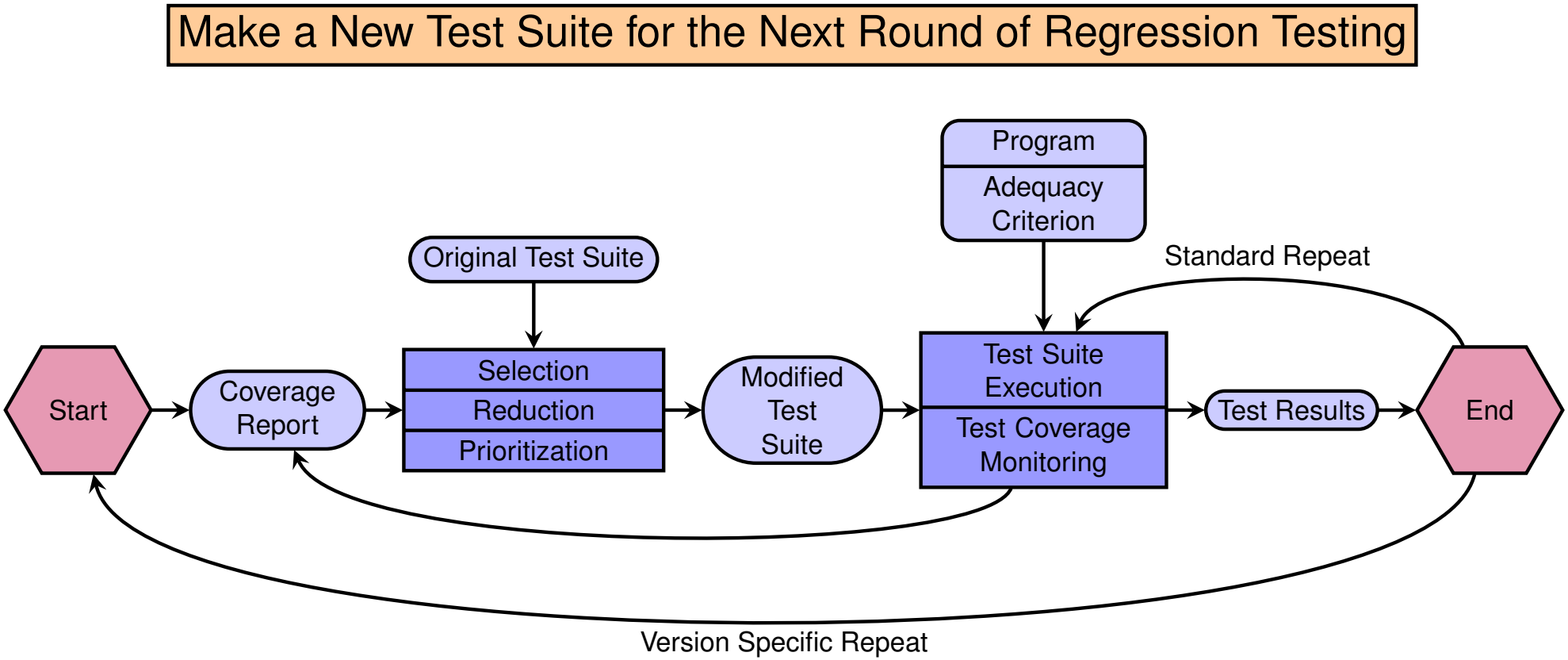
# Model of Regression Testing



# Model of Regression Testing



# Model of Regression Testing



# Test Suite Adequacy

$T_1$ $T_2$

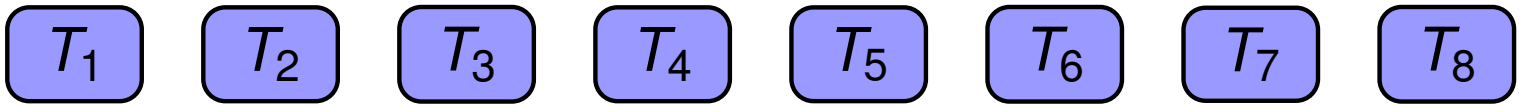
# Test Suite Adequacy

$T_1$ $T_2$ $T_3$ $T_4$

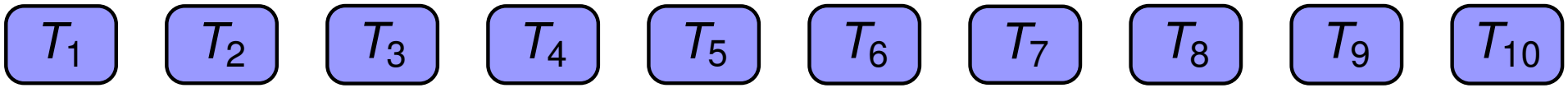
# Test Suite Adequacy

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$

# Test Suite Adequacy



# Test Suite Adequacy



# Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

# Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

$R_1$ $R_2$

# Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

$R_1$ $R_2$ $R_3$ $R_4$

# Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$

# Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$

# Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$ $R_9$ $R_{10}$

# Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$ $R_9$ $R_{10}$ $R_{11}$ $R_{12}$

# Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

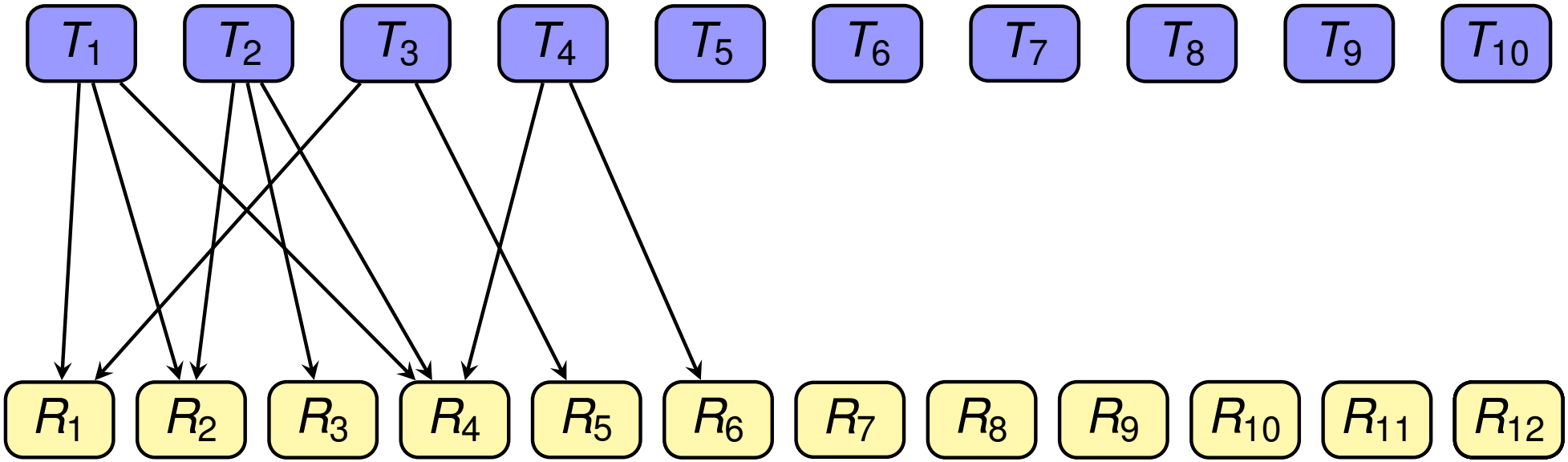
$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$ $R_9$ $R_{10}$ $R_{11}$ $R_{12}$

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Test Suite Adequacy

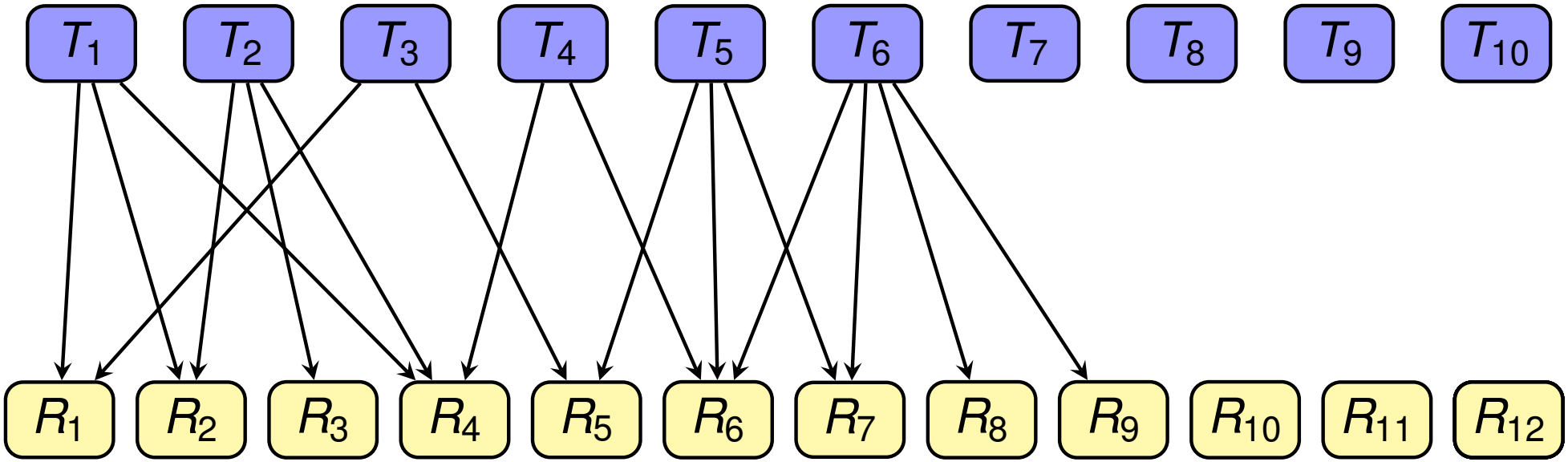
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$




Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Test Suite Adequacy

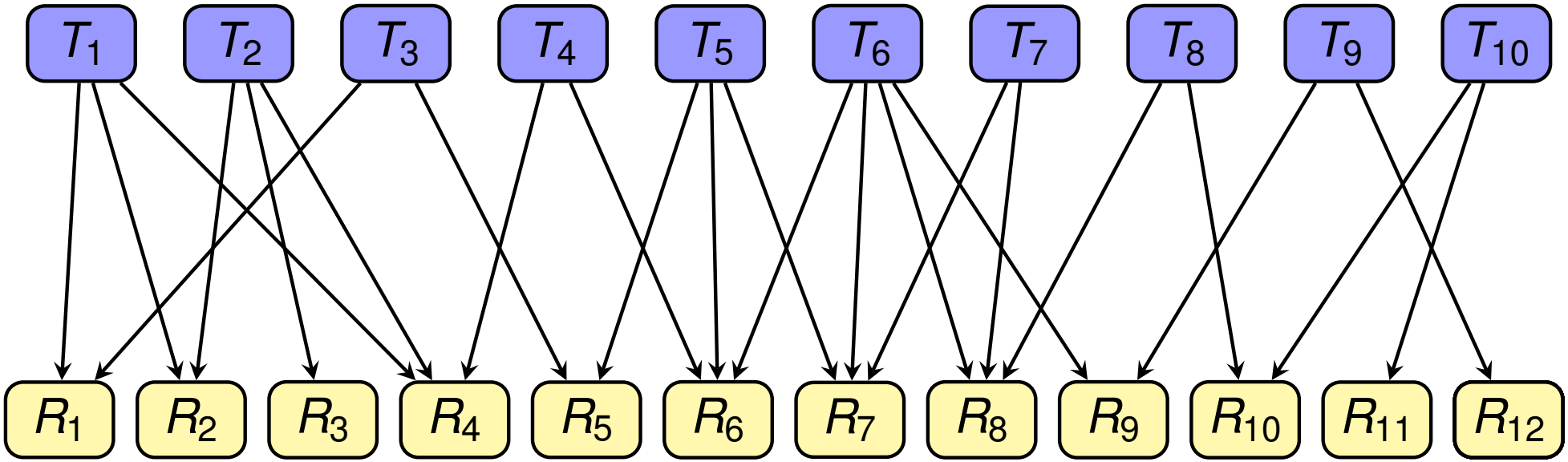
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$




Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

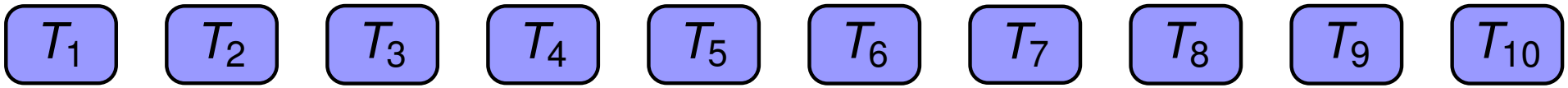
# Test Suite Adequacy

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$




Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Test Suite Execution



# Test Suite Execution

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

# Test Suite Execution

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

JUnit Test Automation Framework

# Test Suite Execution

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

JUnit Test Automation Framework

Run Test Case

# Test Suite Execution

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

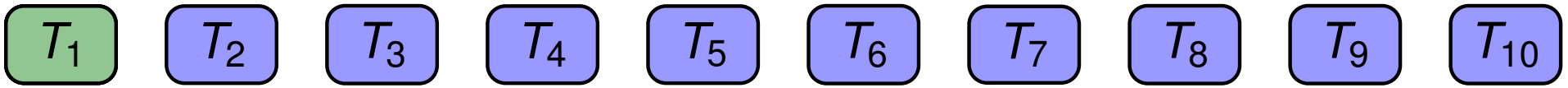
$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

JUnit Test Automation Framework

Passing Test Case: $O_A = O_E$

# Test Suite Execution

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$




JUnit Test Automation Framework

# Test Suite Execution

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

JUnit Test Automation Framework

# Test Suite Execution

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

JUnit Test Automation Framework

Run Test Case

# Test Suite Execution

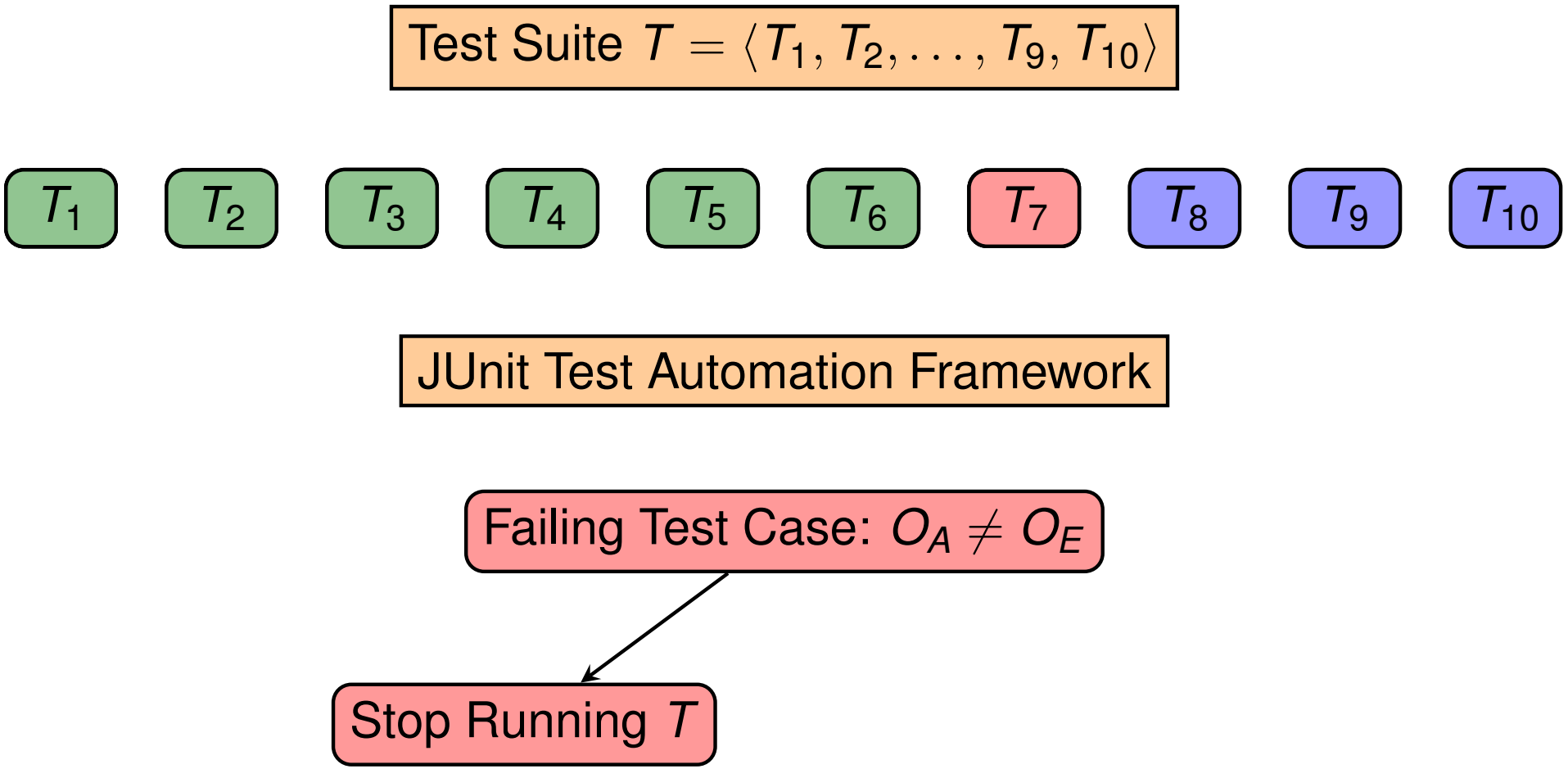
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

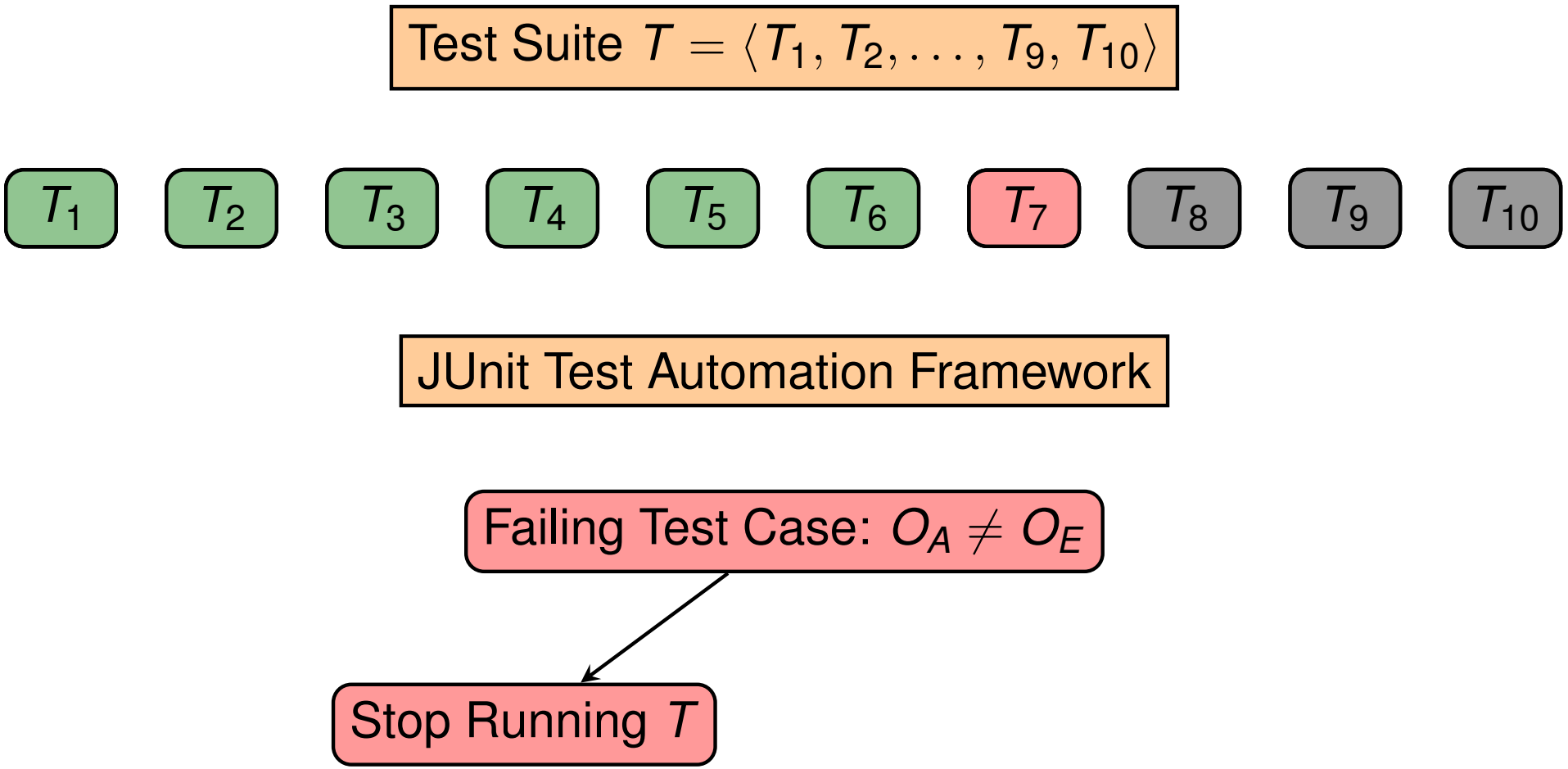
JUnit Test Automation Framework

Failing Test Case: $O_A \neq O_E$

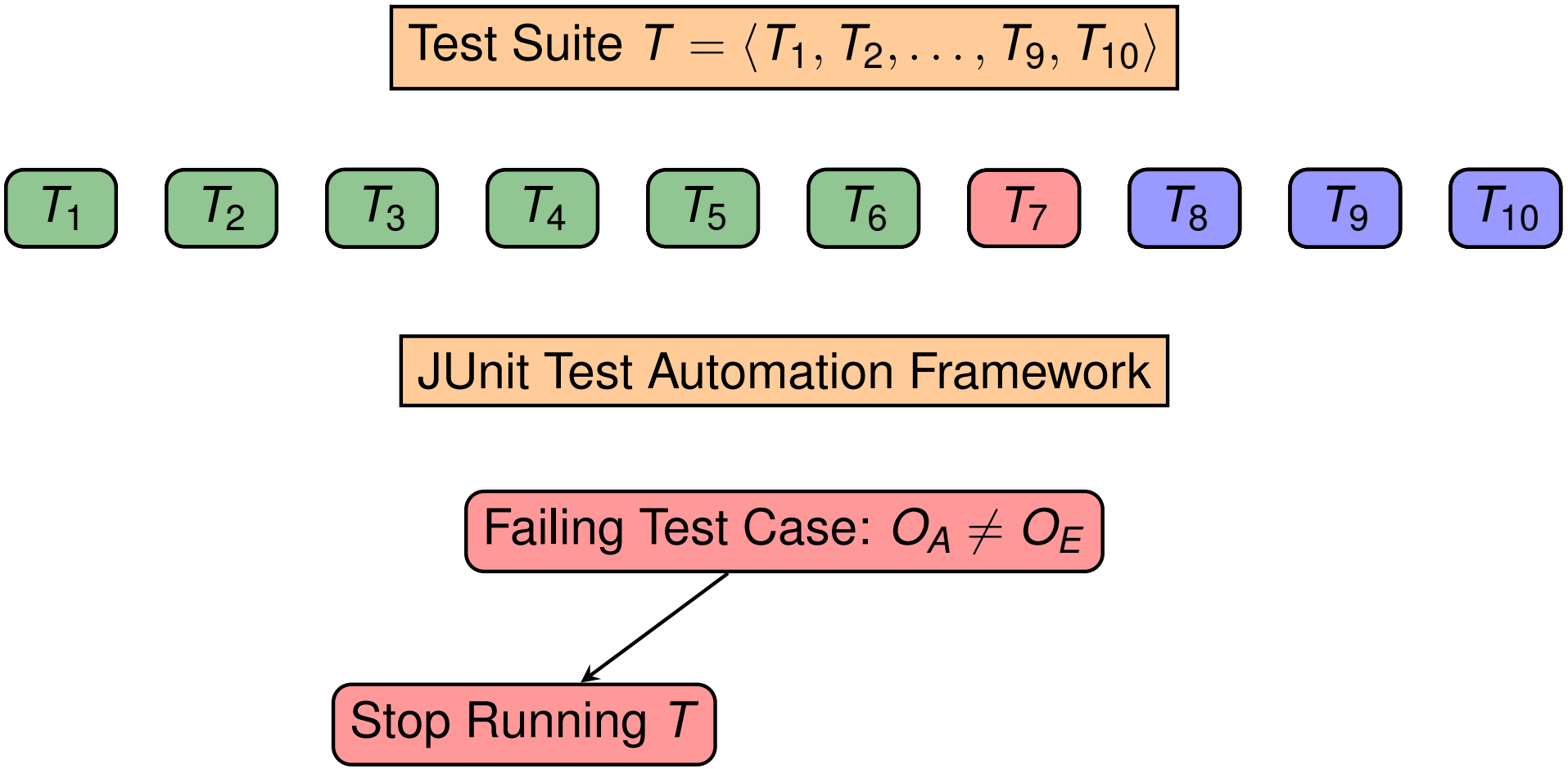
# Test Suite Execution



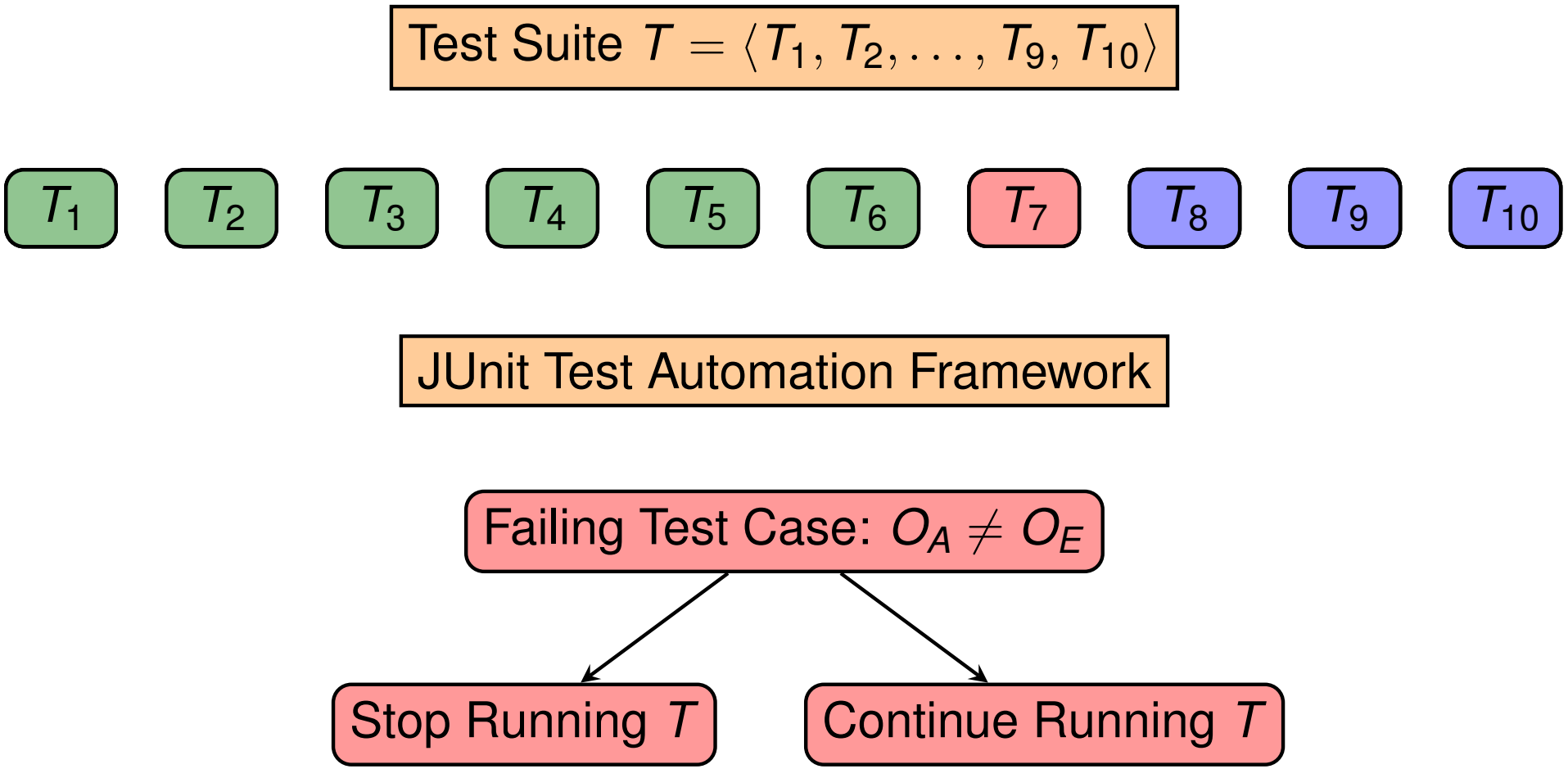
# Test Suite Execution



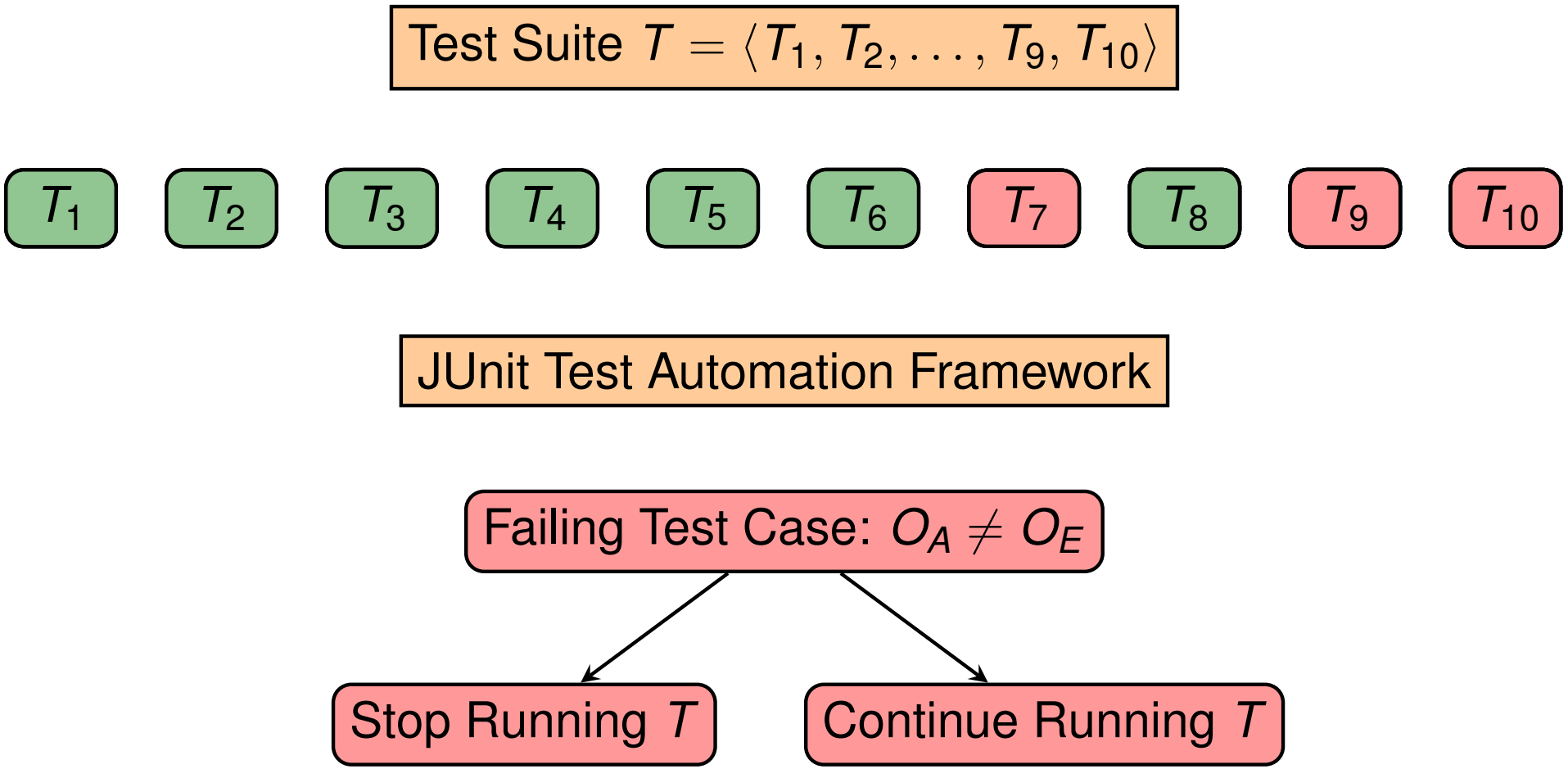
# Test Suite Execution



# Test Suite Execution



# Test Suite Execution



# Test Coverage Monitoring

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

# Test Coverage Monitoring

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

# Test Coverage Monitoring

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

JUnit Test Automation Framework
Cobertura Test Coverage Monitor
Proteja Test Suite Manager

# Test Coverage Monitoring

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

JUnit Test Automation Framework
Cobertura Test Coverage Monitor
Proteja Test Suite Manager

Run Test Case

Collect Per-Test Case Coverage

# Test Coverage Monitoring

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$ $R_9$ $R_{10}$ $R_{11}$ $R_{12}$

# Test Coverage Monitoring

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

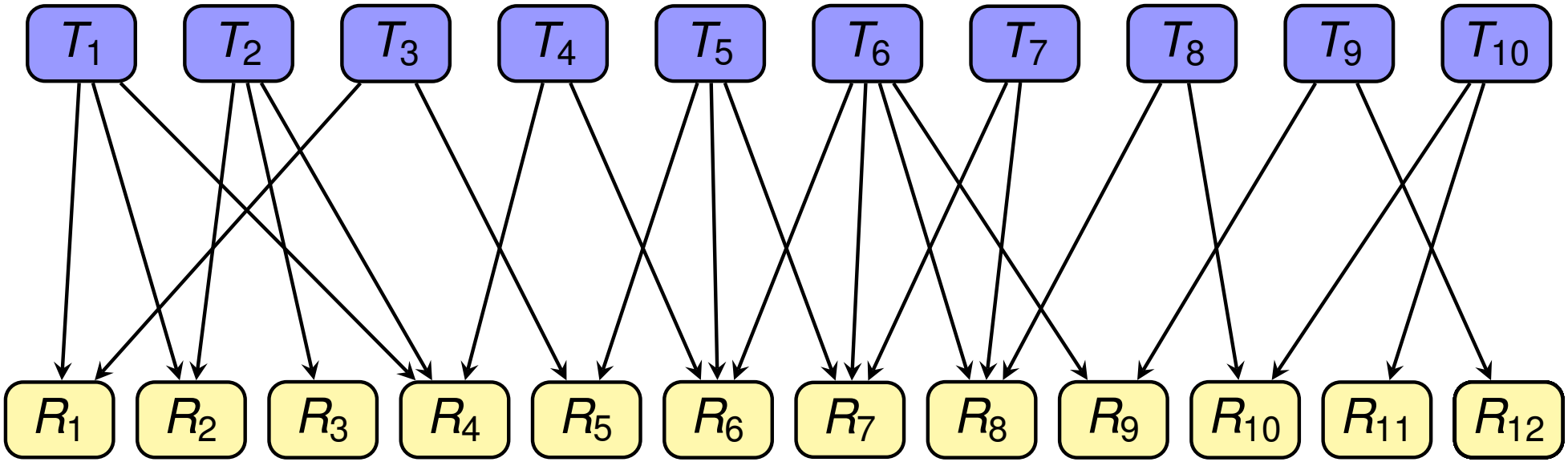
$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$ $R_9$ $R_{10}$ $R_{11}$ $R_{12}$

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Test Coverage Monitoring

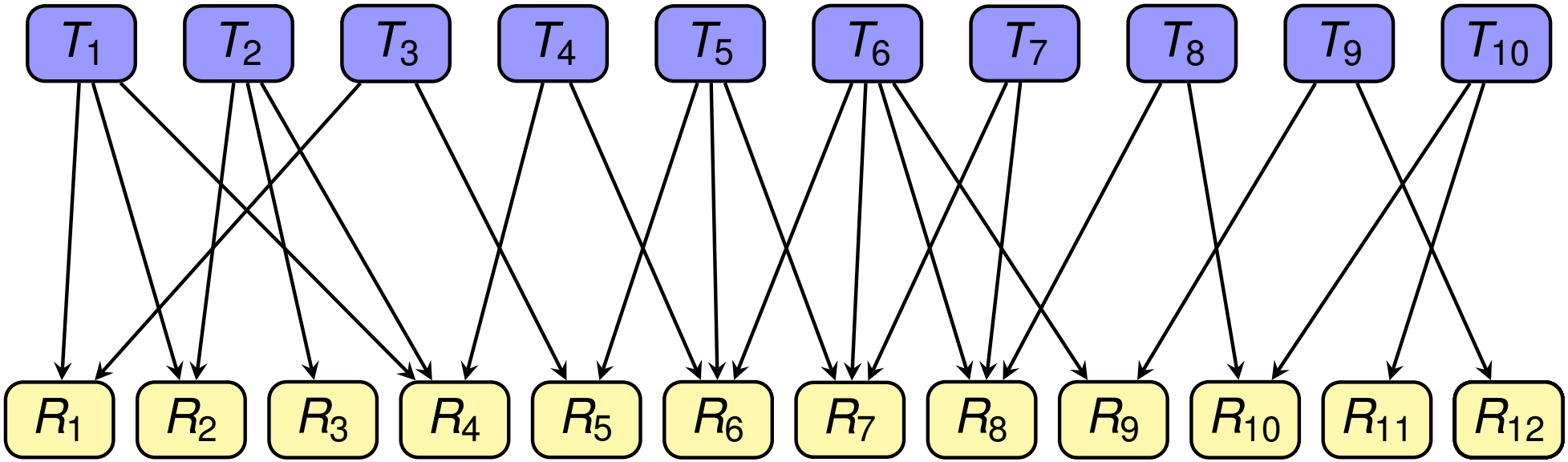
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$




Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Test Coverage Monitoring

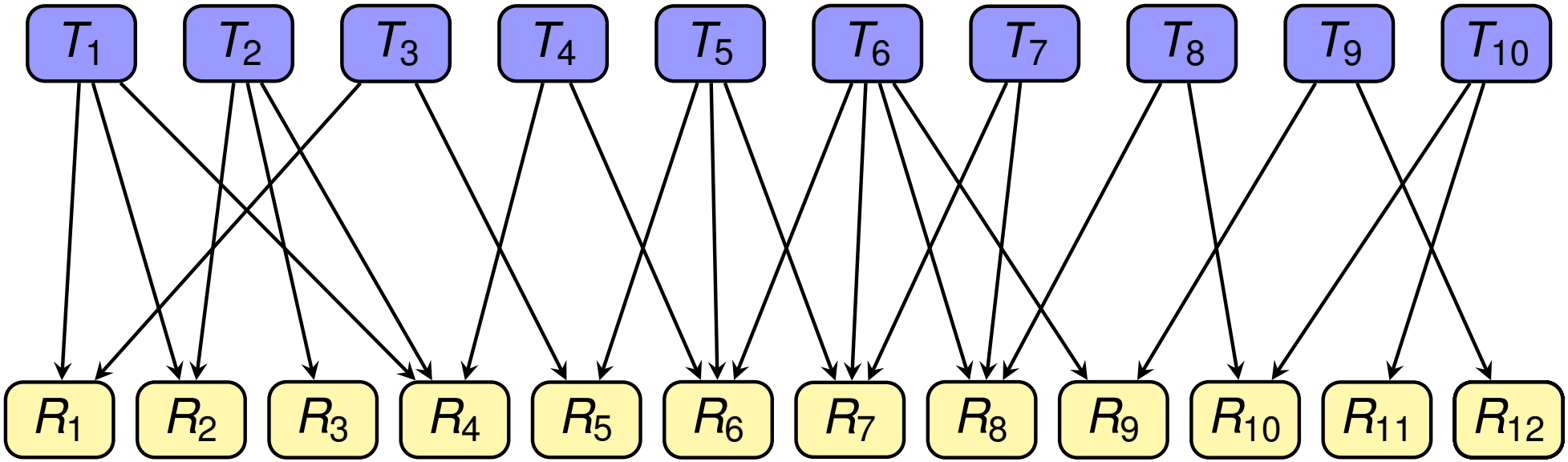
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$




Requirements Set $R$ for ... Statement Coverage

# Test Coverage Monitoring

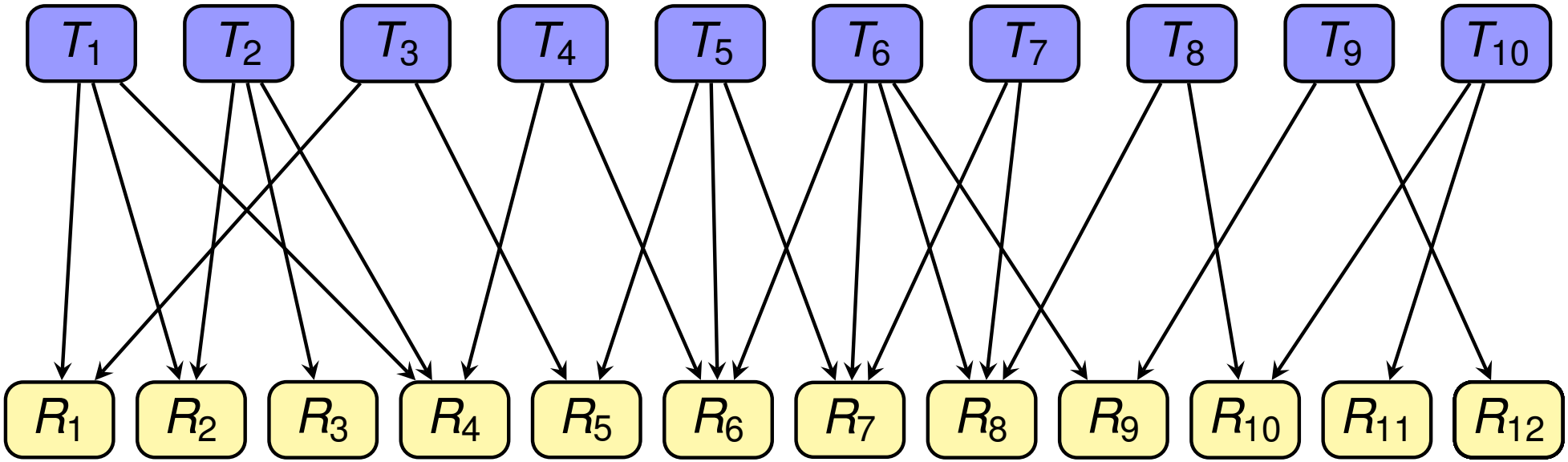
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$




Requirements Set $R$ for ... Mutation Coverage

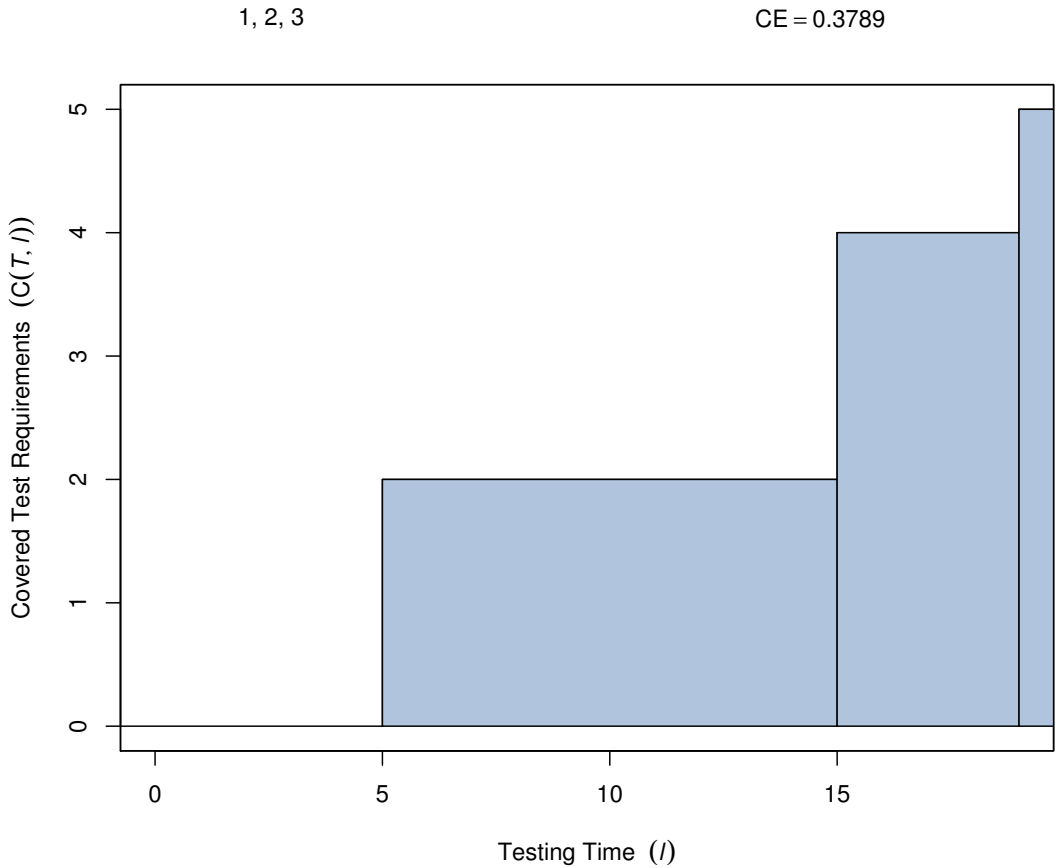
# Test Coverage Monitoring

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$



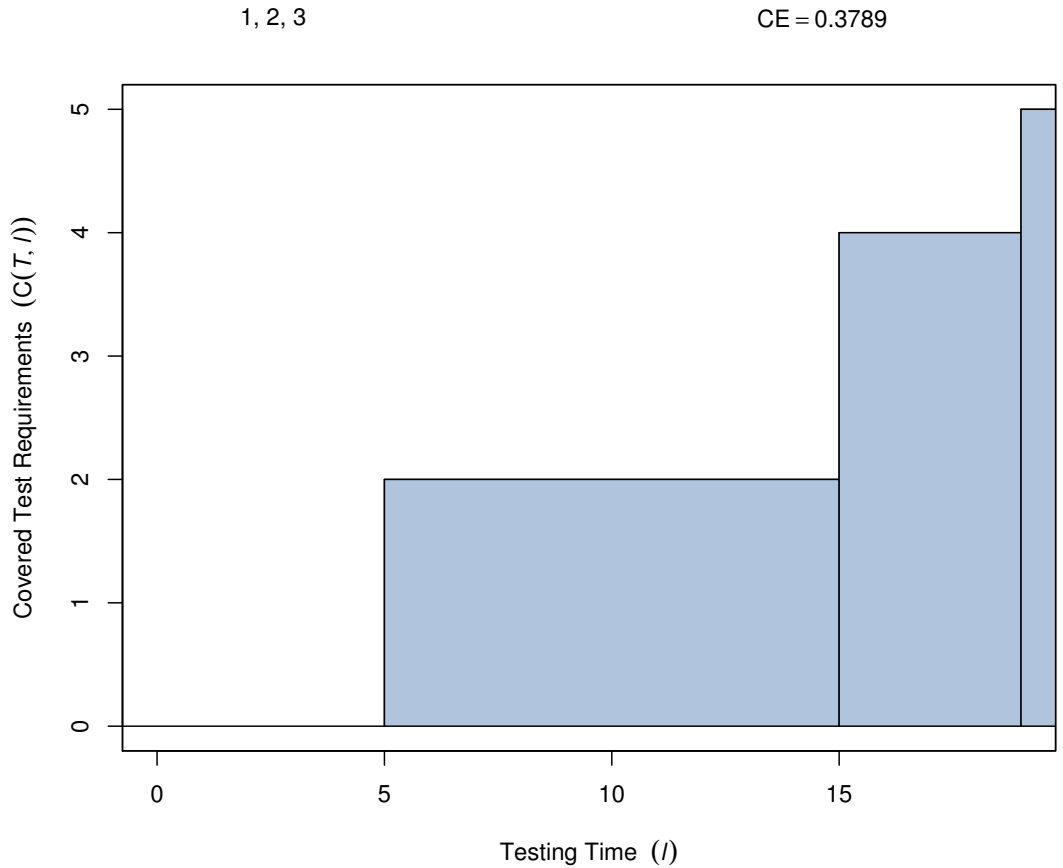

Requirements Set $R$ for ... Definition-Use Coverage

# Importance of Test Suite Prioritization




Prioritize to **increase** the CE of a test suite $CE = \frac{\text{Actual}}{\text{Ideal}} \in [0, 1]$

# Importance of Test Suite Prioritization

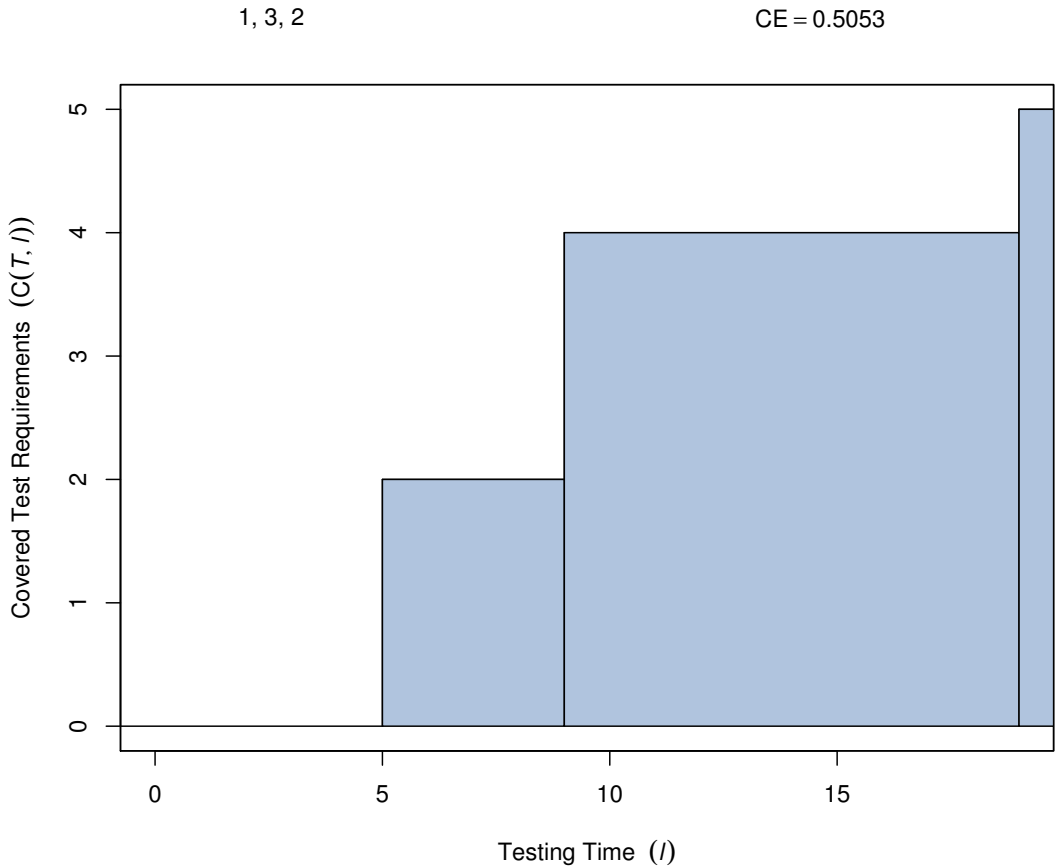



## Test Orderings

1,2,3

**Original ordering** exhibits poor effectiveness score - **CE = 0.3789**

# Importance of Test Suite Prioritization



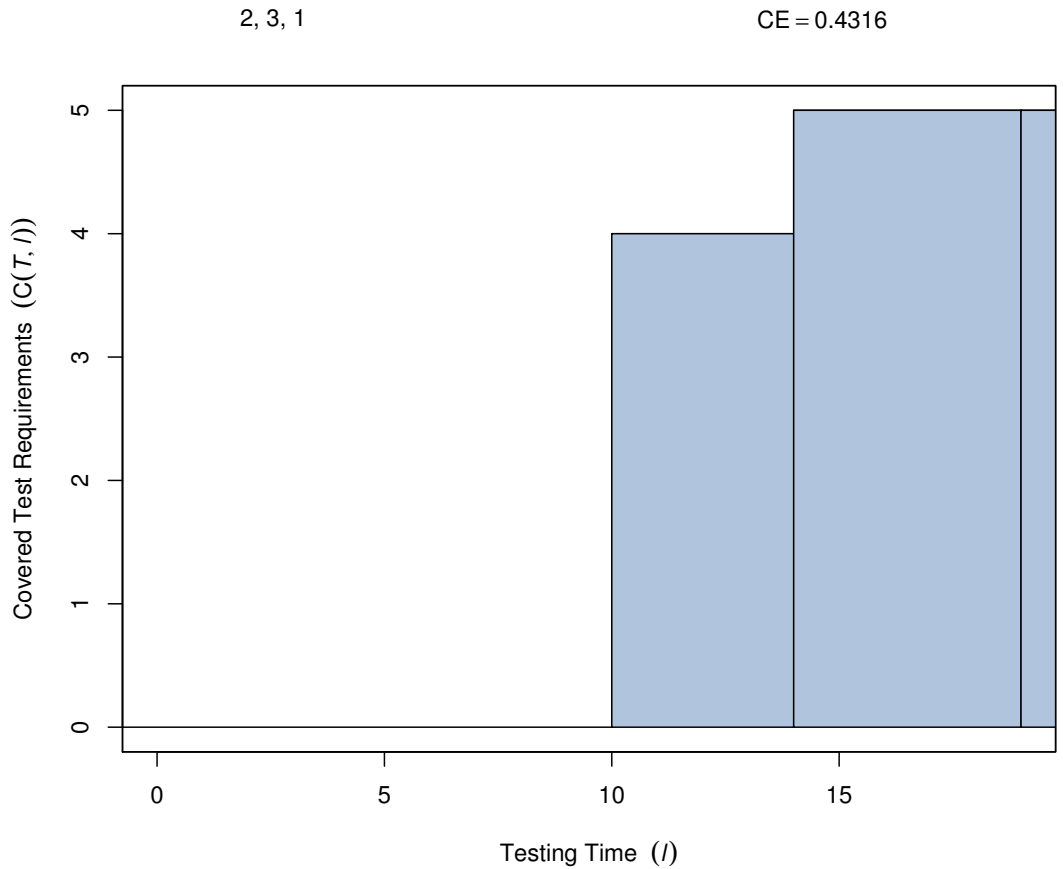

## Test Orderings

1,2,3

1,3,2

**Different ordering** improves the effectiveness score - **CE = 0.5053**

# Importance of Test Suite Prioritization




## Test Orderings

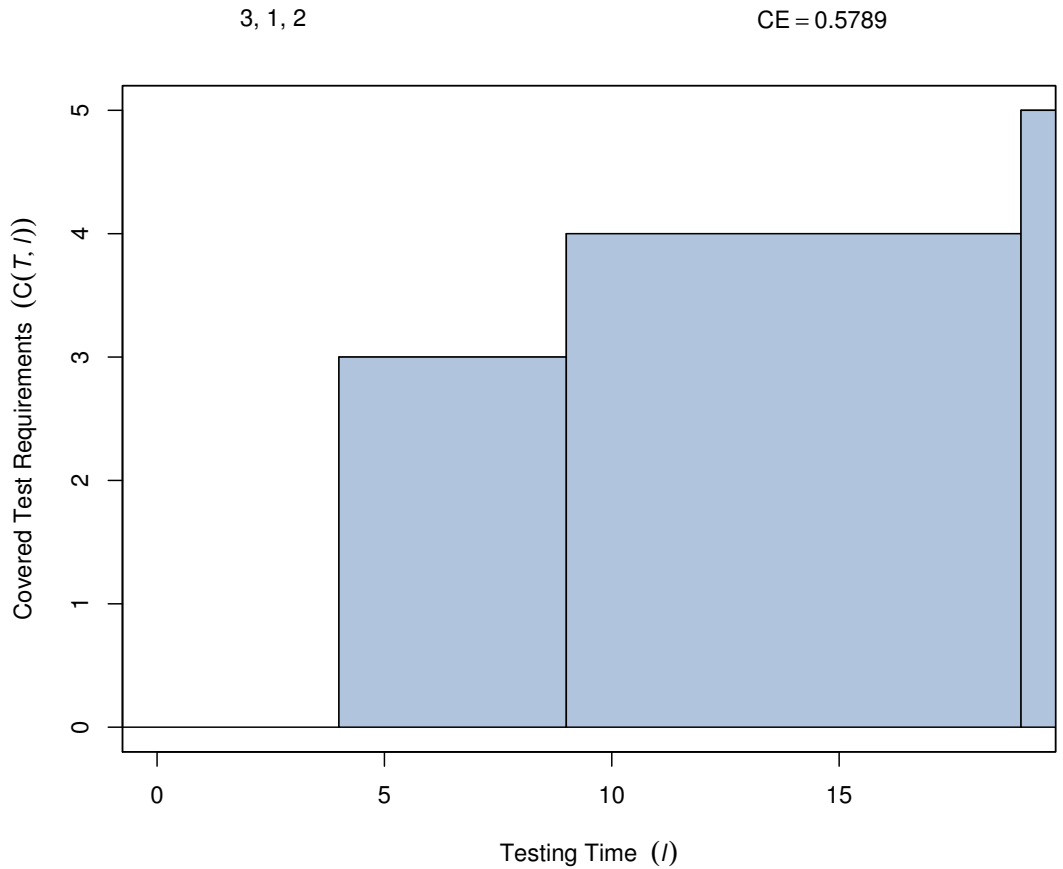
1,2,3

1,3,2

2,3,1

**Some orderings** have less improved scores - **CE = 0.4316**

# Importance of Test Suite Prioritization

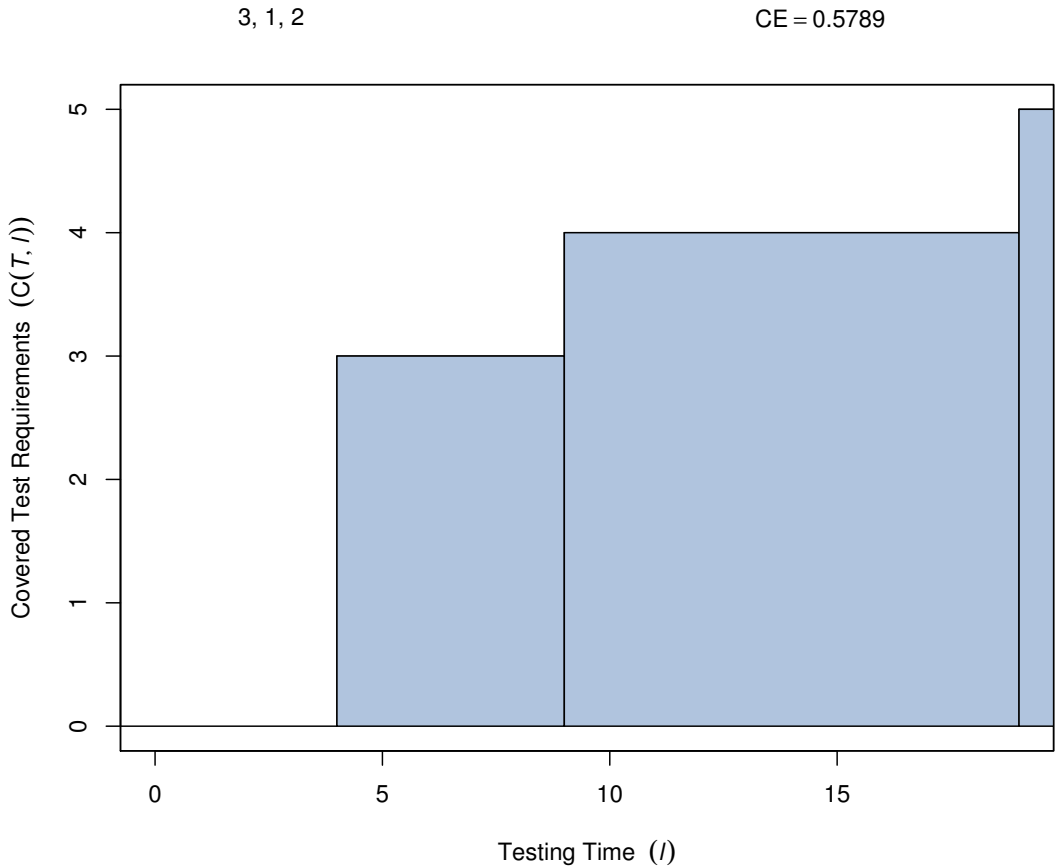



## Test Orderings

- 1,2,3
- 1,3,2
- 2,3,1
- 3,1,2

**Best ordering** shows a higher effectiveness scores - **CE = 0.5789**

# Importance of Test Suite Prioritization

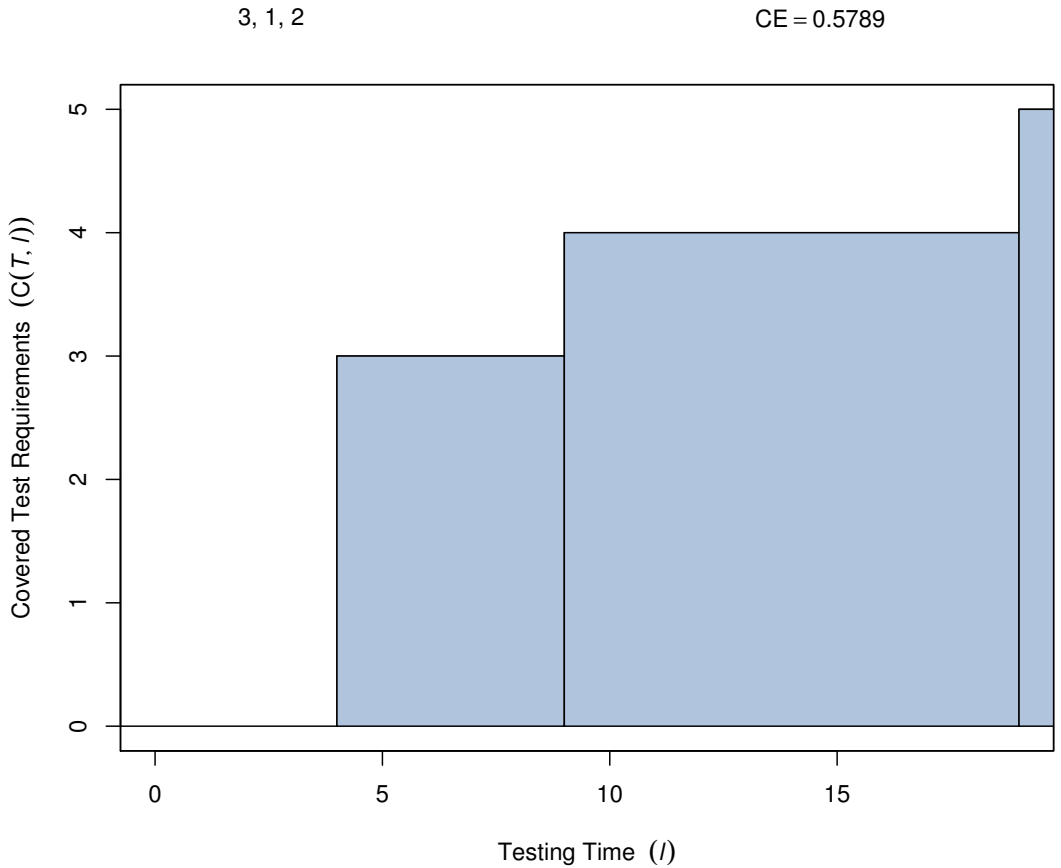



## Test Orderings

- 1,2,3
- 1,3,2
- 2,3,1
- 3,1,2

**Greedy methods** often produce high-effectiveness orderings

# Importance of Test Suite Prioritization




## Test Orderings

1,2,3

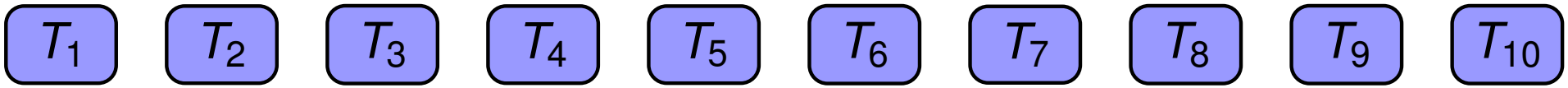
1,3,2

2,3,1

3,1,2

**Search-based techniques** may have some desirable characteristics

# Greedy Algorithms



# Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

# Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$ $R_9$ $R_{10}$ $R_{11}$ $R_{12}$

# Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

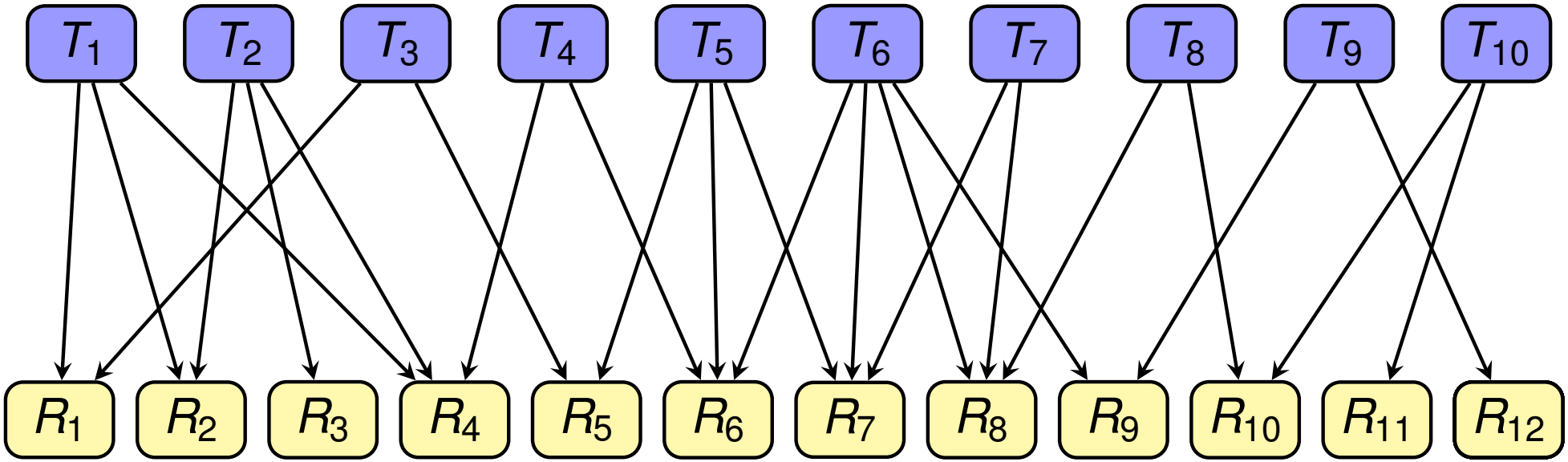
$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$ $R_9$ $R_{10}$ $R_{11}$ $R_{12}$

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

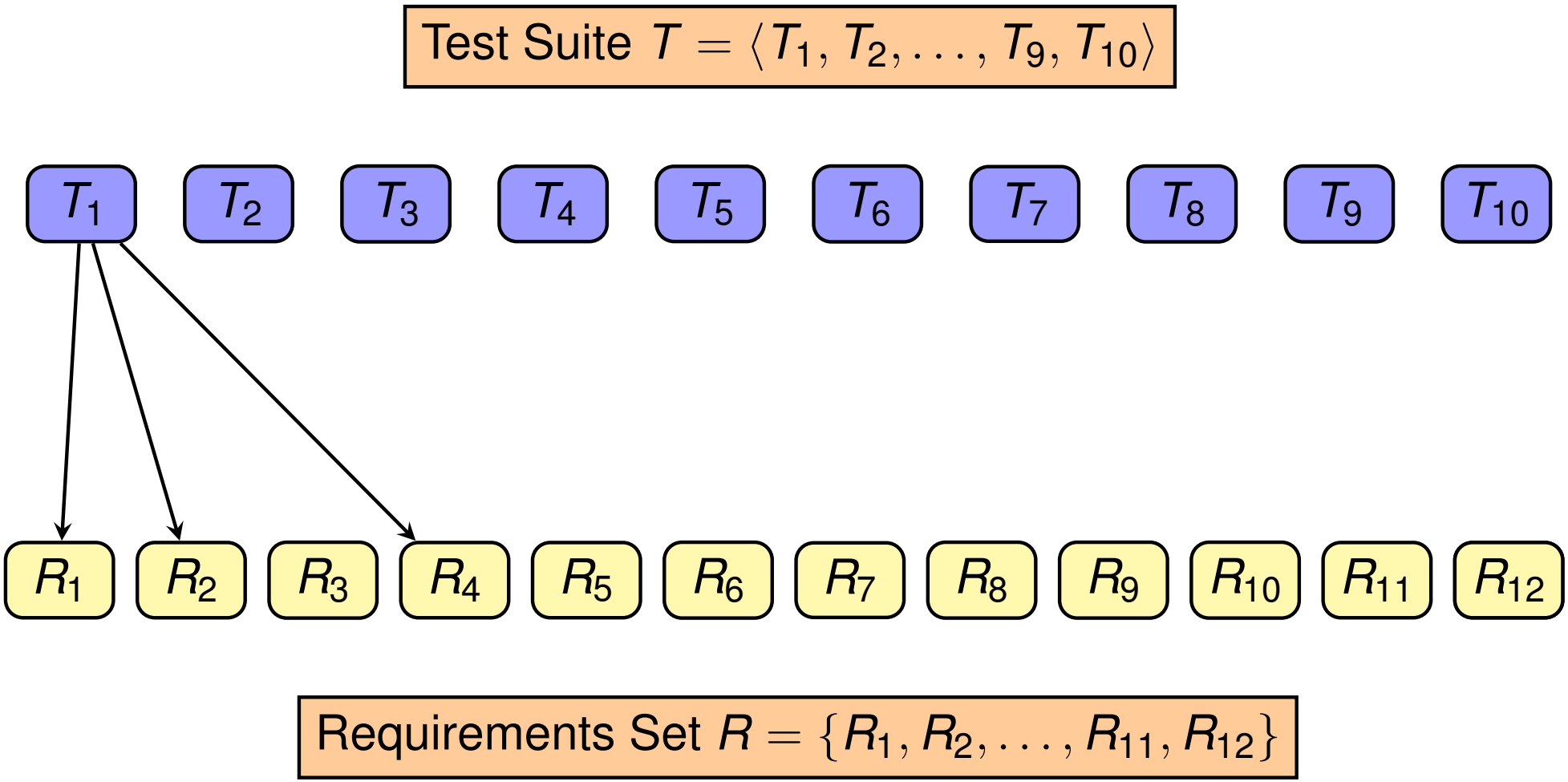
# Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$



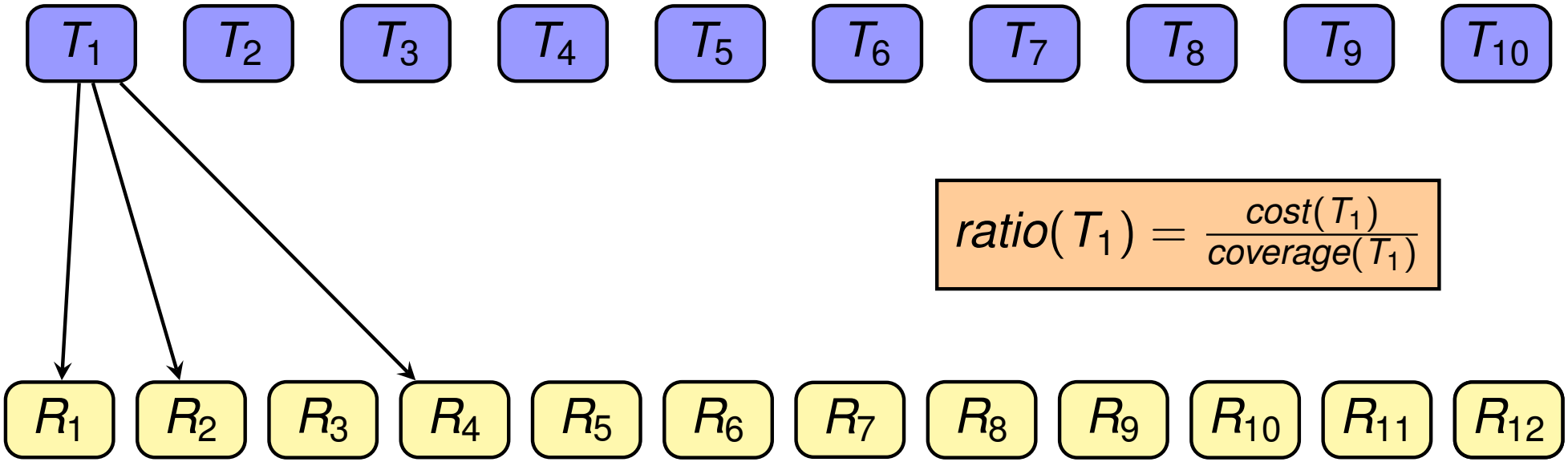

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms



# Greedy Algorithms

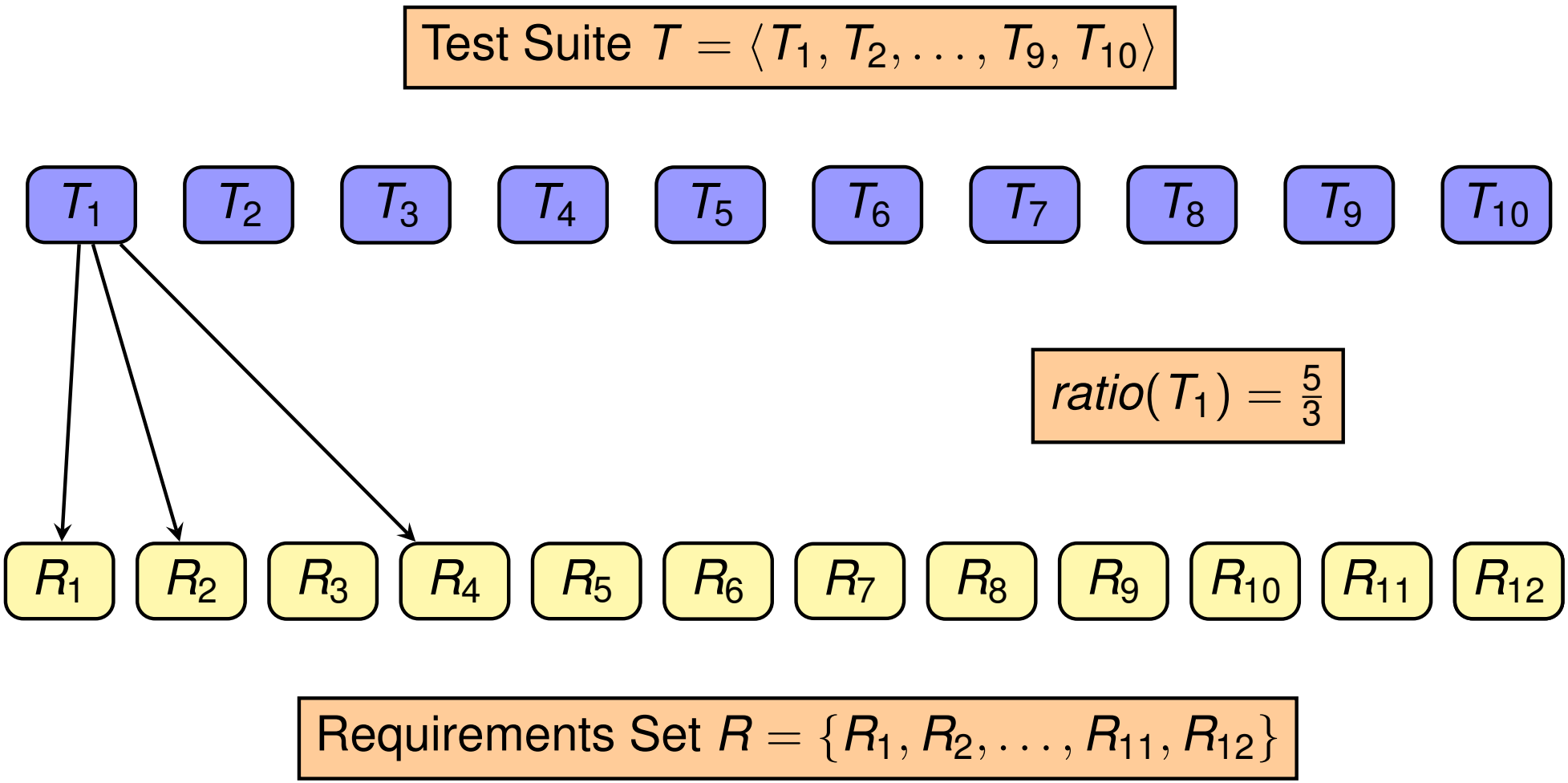
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$



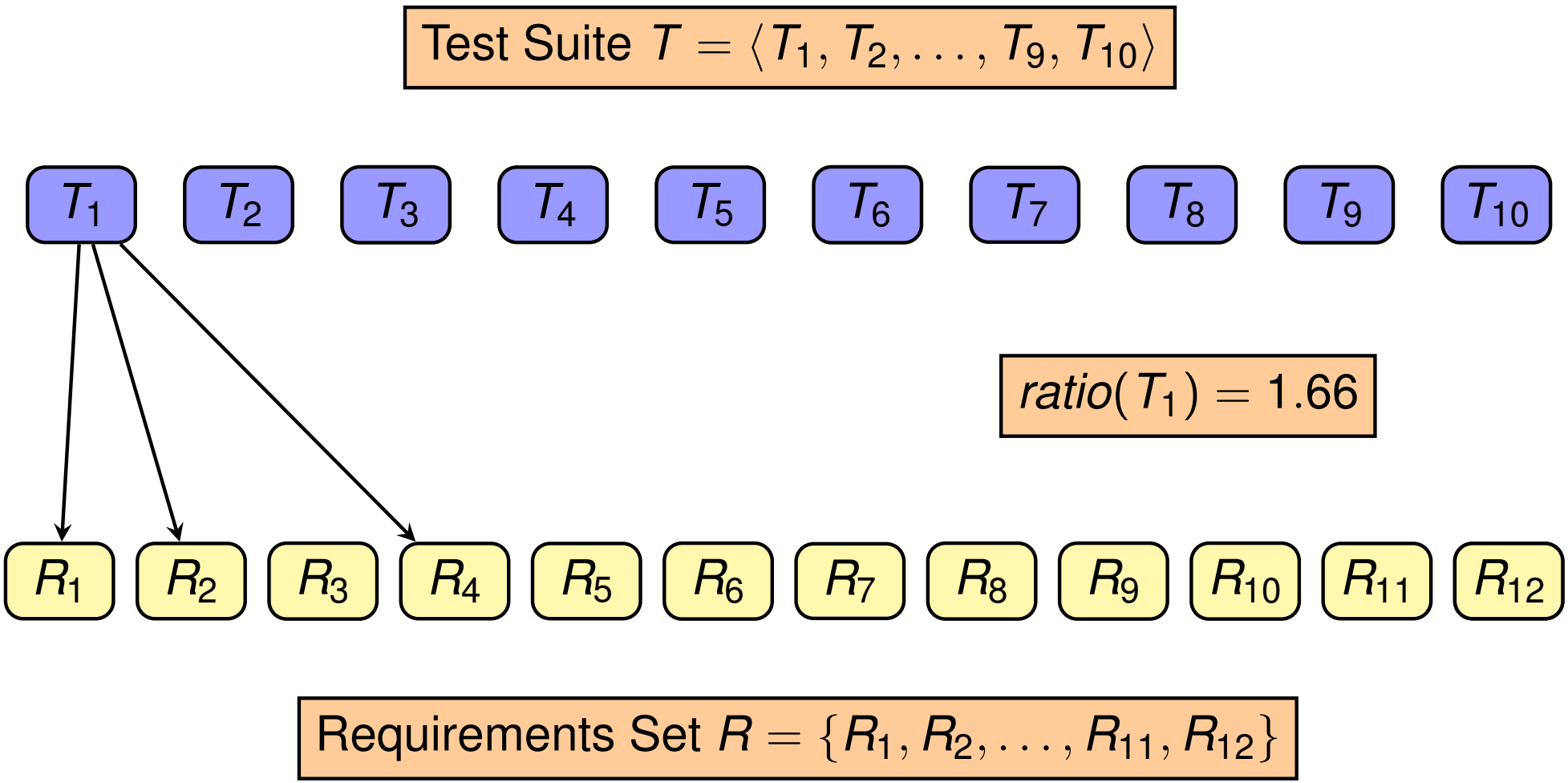

$$ratio(T_1) = \frac{cost(T_1)}{coverage(T_1)}$$

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms



# Greedy Algorithms



# Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

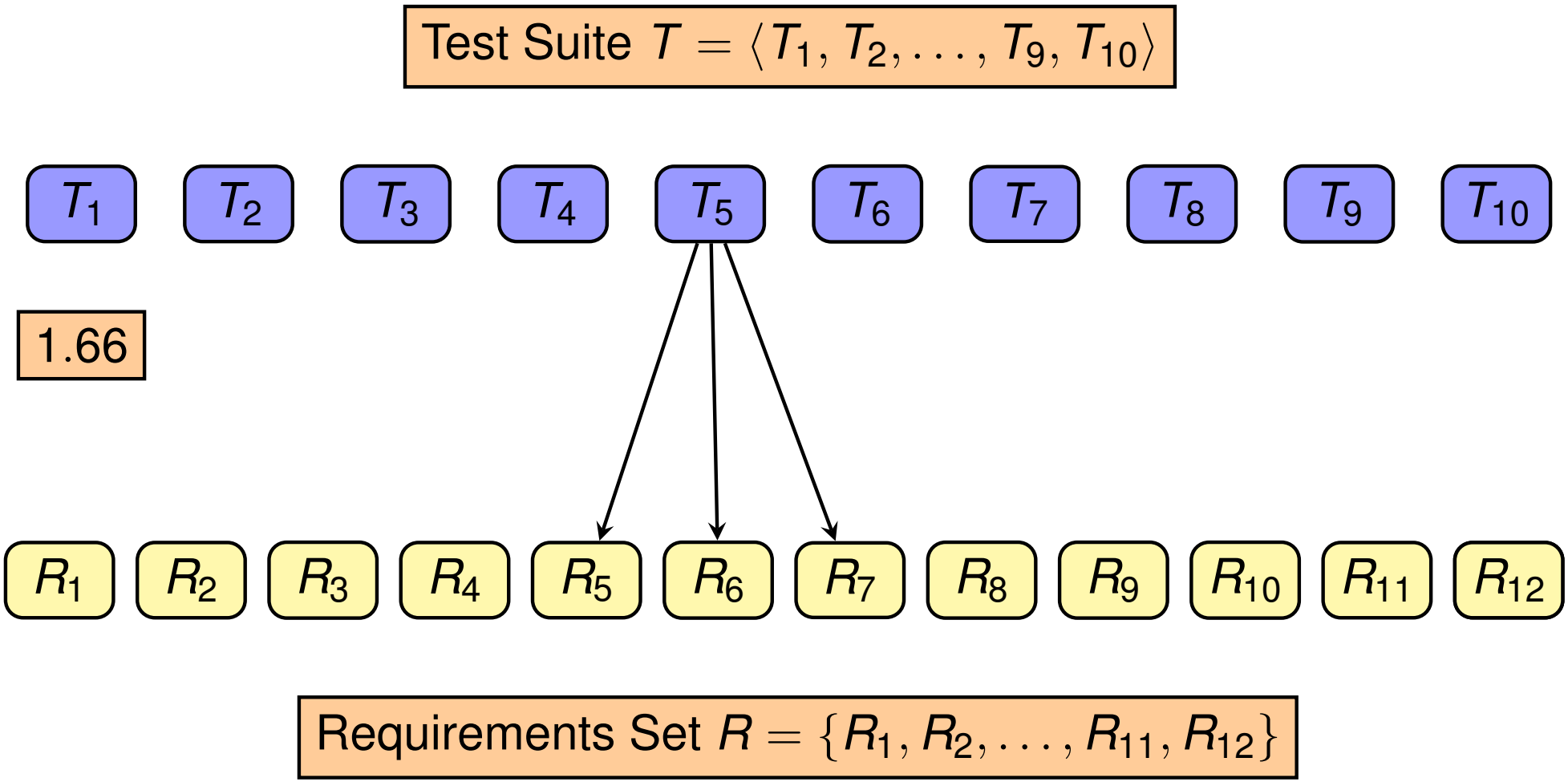
$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

1.66

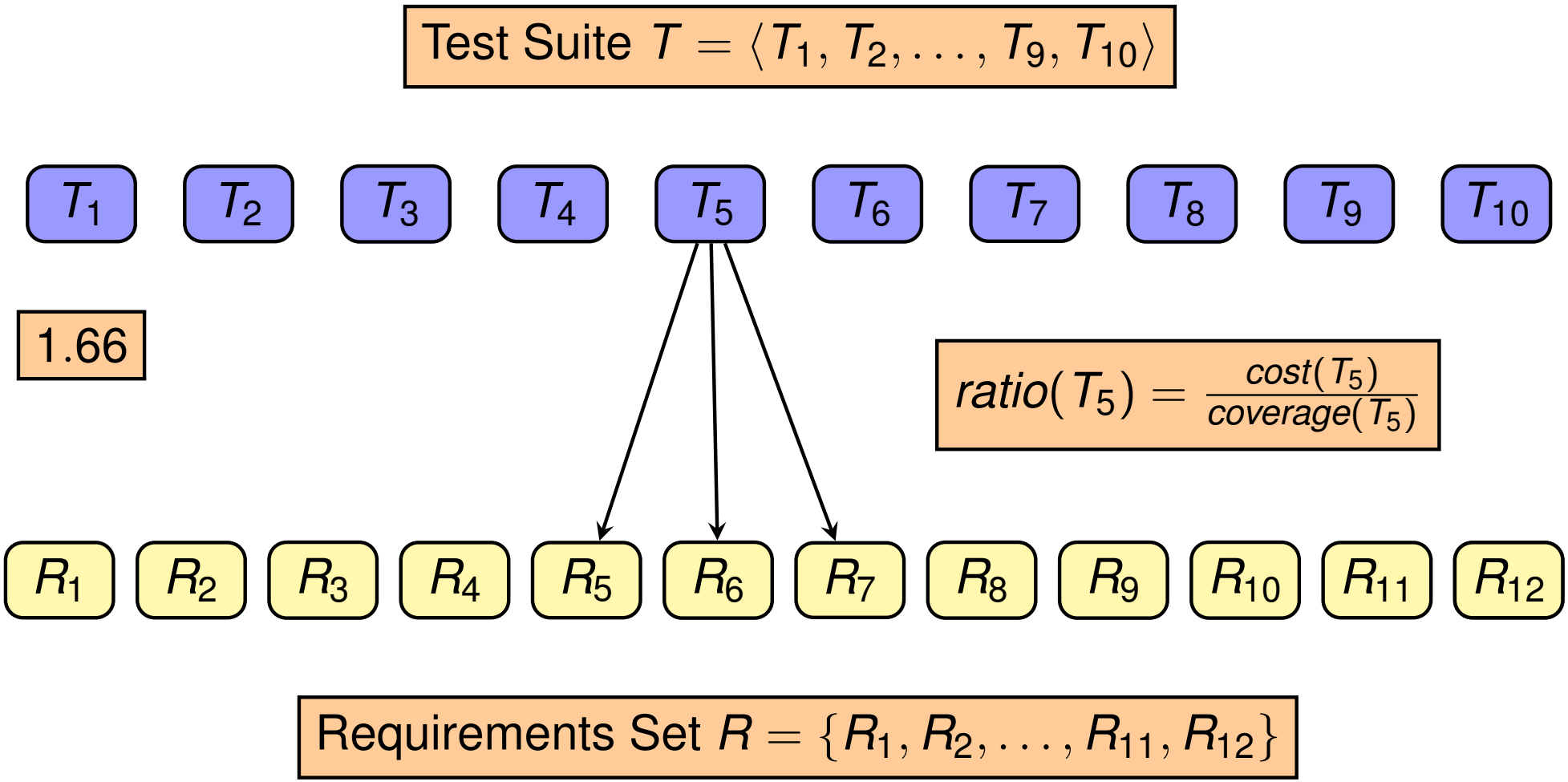
$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$ $R_9$ $R_{10}$ $R_{11}$ $R_{12}$

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

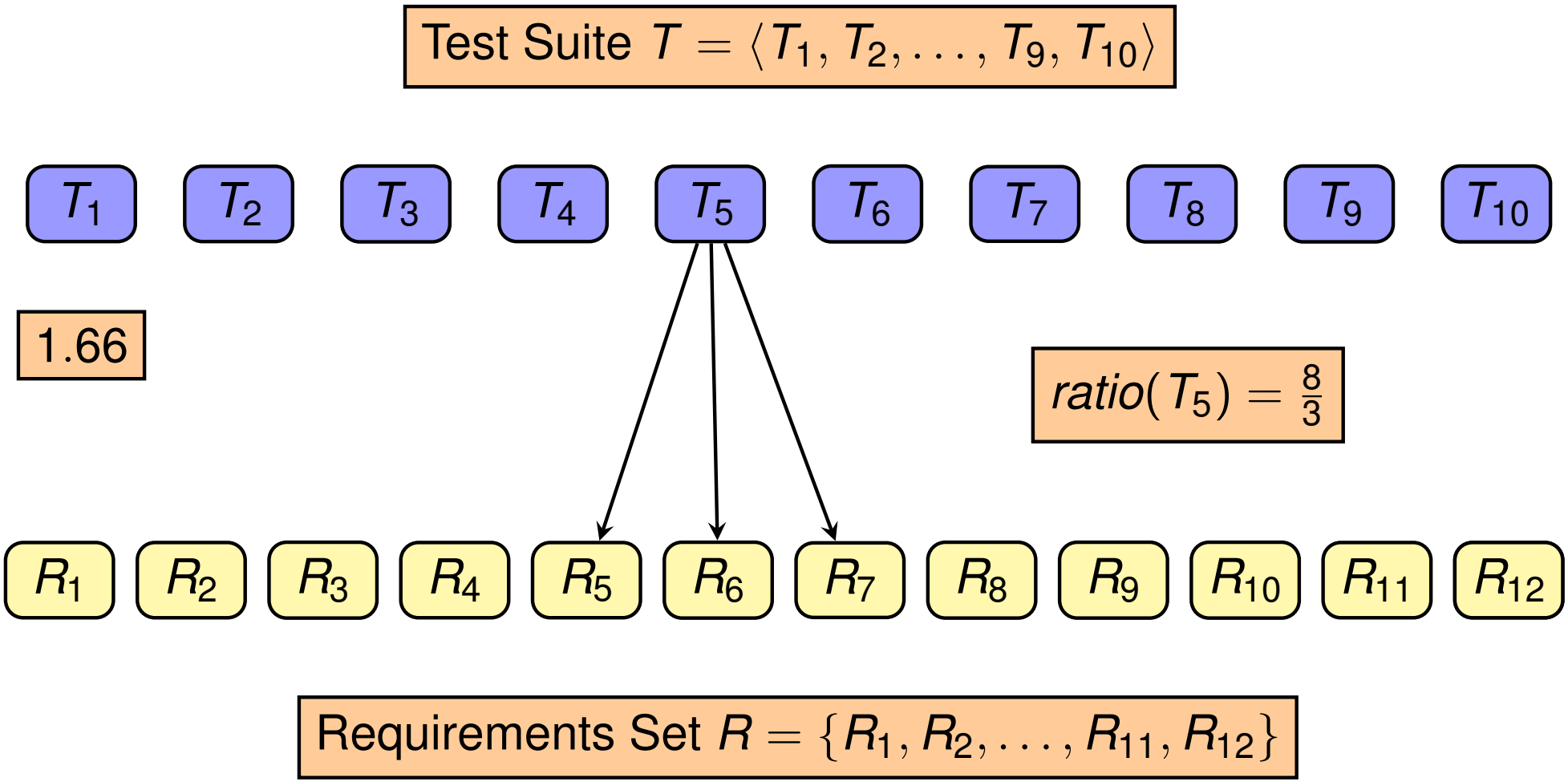
# Greedy Algorithms



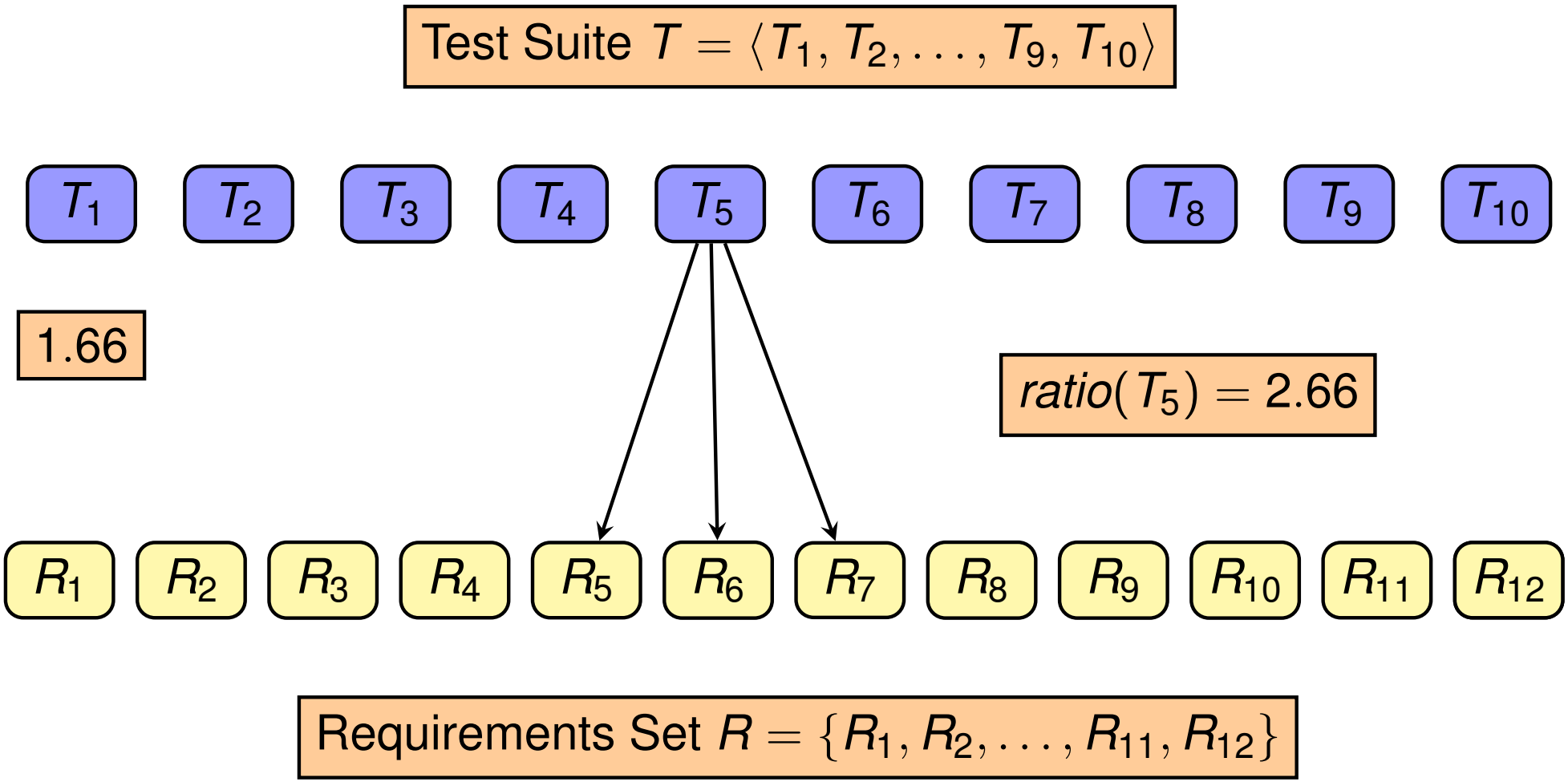
# Greedy Algorithms



# Greedy Algorithms



# Greedy Algorithms



# Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

1.66

2.66

$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$ $R_9$ $R_{10}$ $R_{11}$ $R_{12}$

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

1.66

2.66

$ratio(T_1) < ratio(T_5)$
Prefer $T_1$ over $T_5$

$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$ $R_9$ $R_{10}$ $R_{11}$ $R_{12}$

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms

Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_7$ $T_8$ $T_9$ $T_{10}$

1.66

2.66

Proceed incrementally, picking the test case with the lowest *ratio* value for the uncovered requirements

$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$ $R_9$ $R_{10}$ $R_{11}$ $R_{12}$

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms

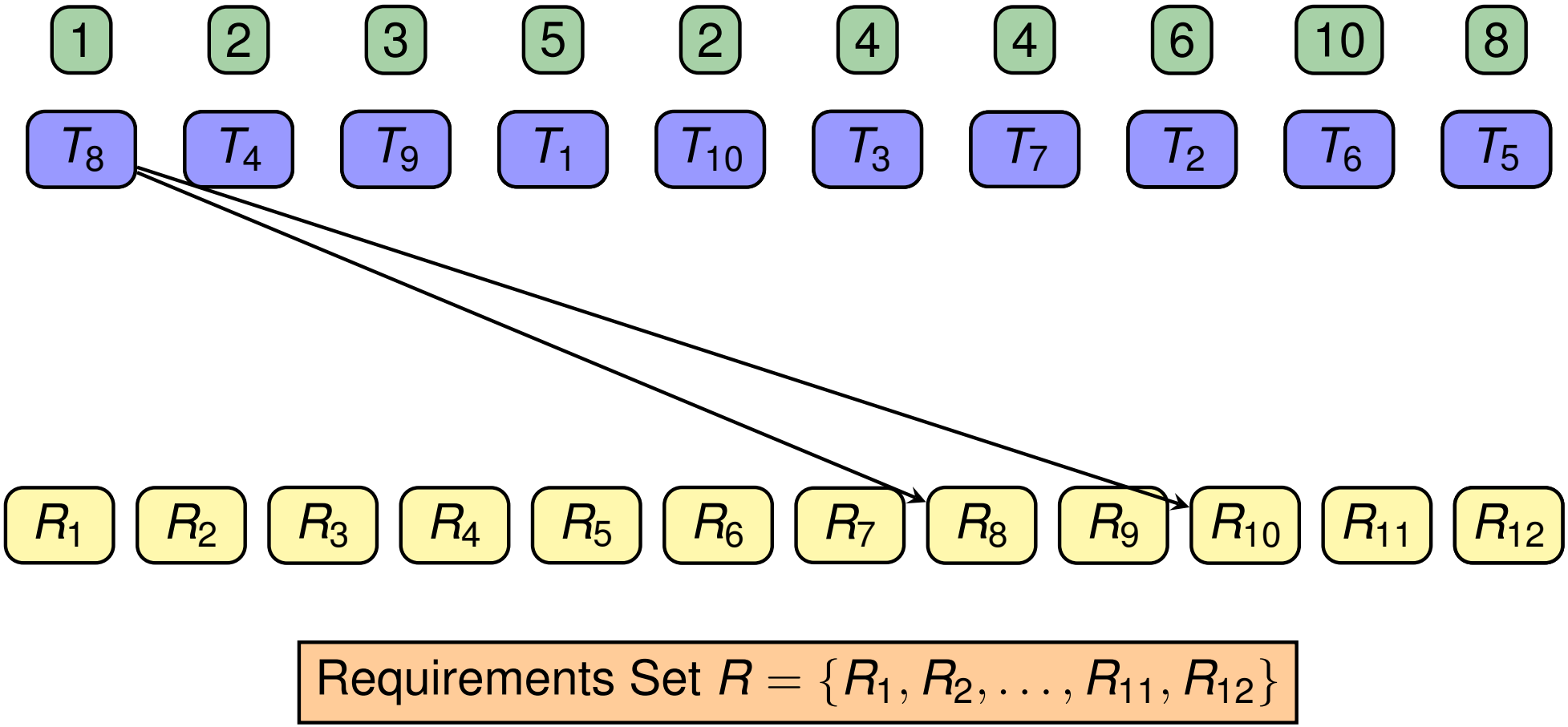
Test Suite $T = \langle T_8, T_4, T_9, T_1, T_{10}, T_3, T_7, T_2, T_6, T_5 \rangle$

$T_8$ $T_4$ $T_9$ $T_1$ $T_{10}$ $T_3$ $T_7$ $T_2$ $T_6$ $T_5$

$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$ $R_9$ $R_{10}$ $R_{11}$ $R_{12}$

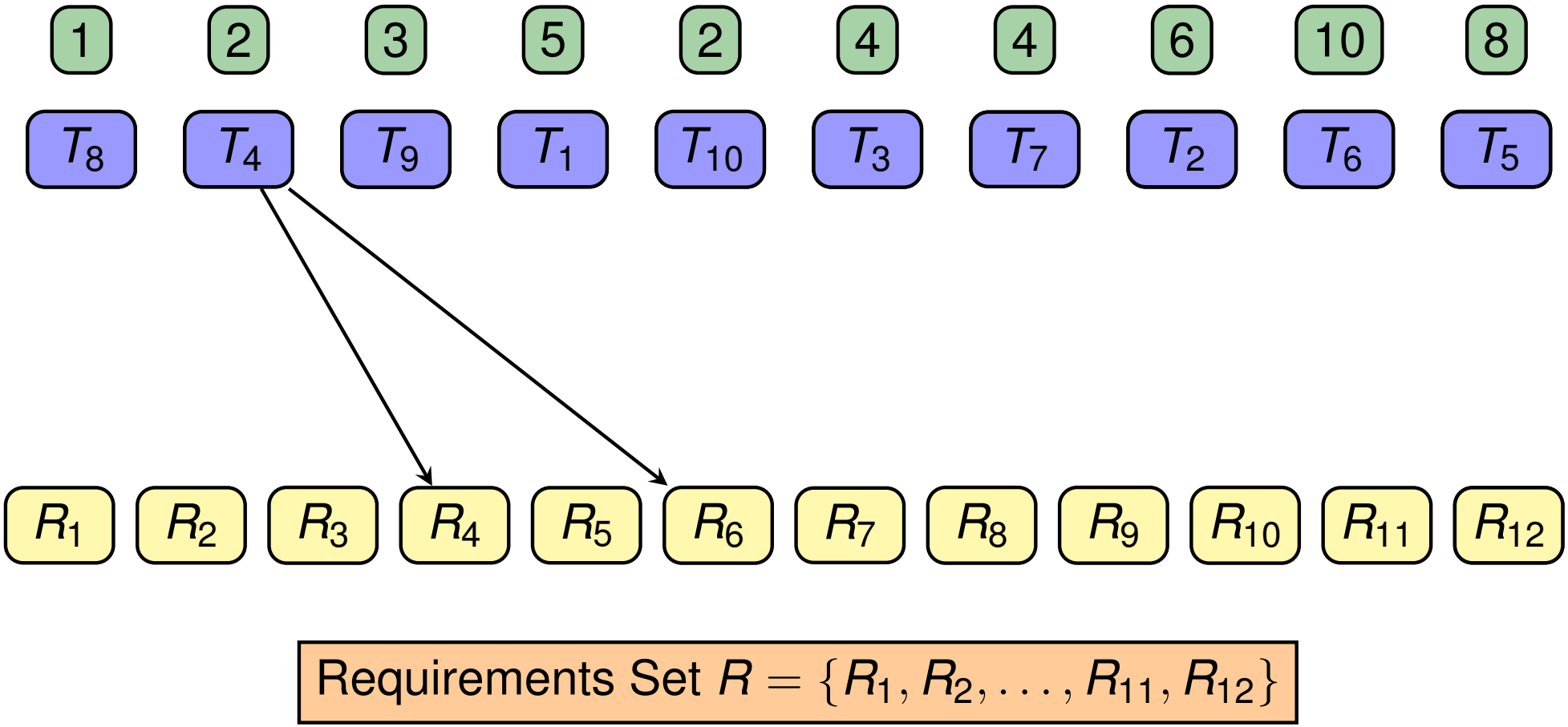
Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms



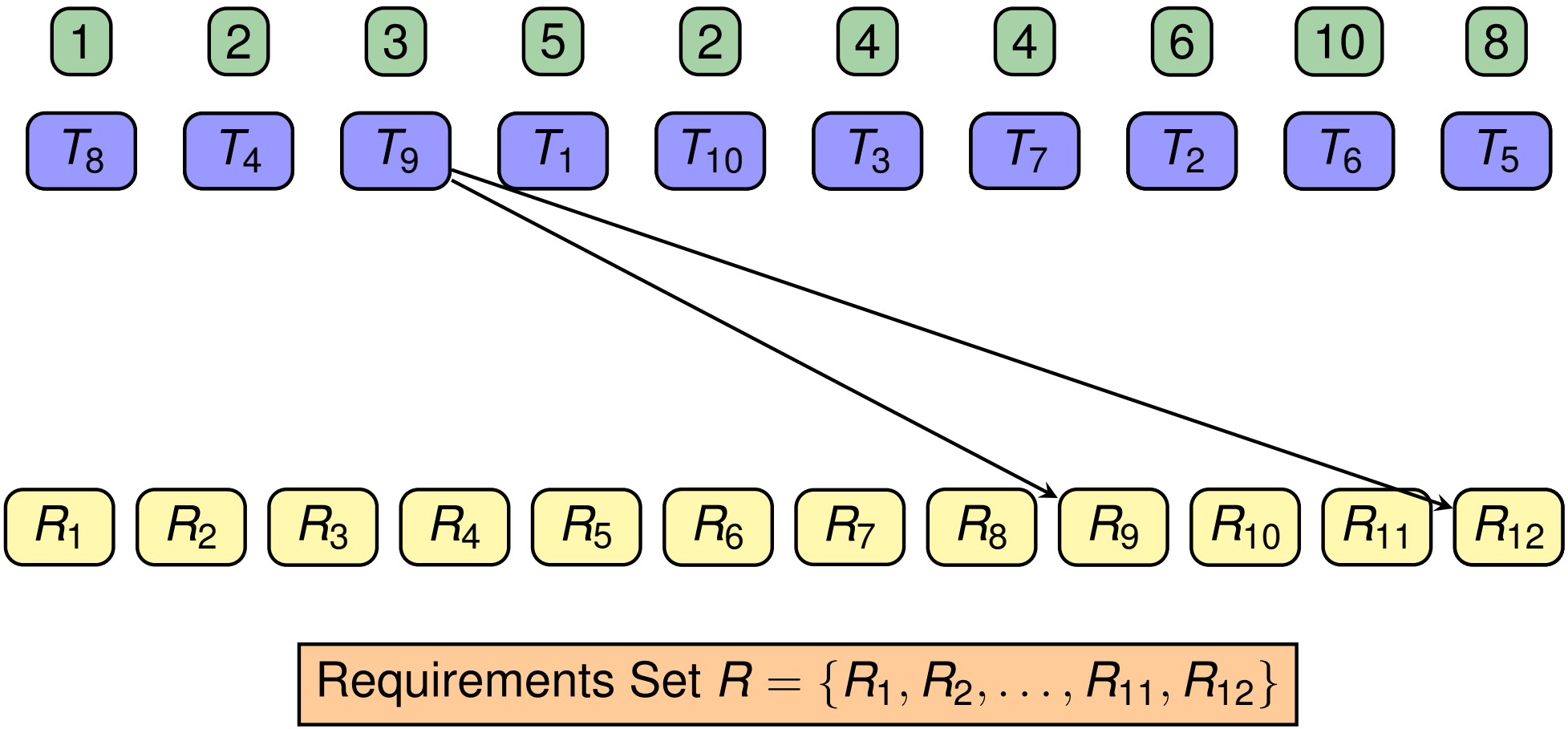

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms



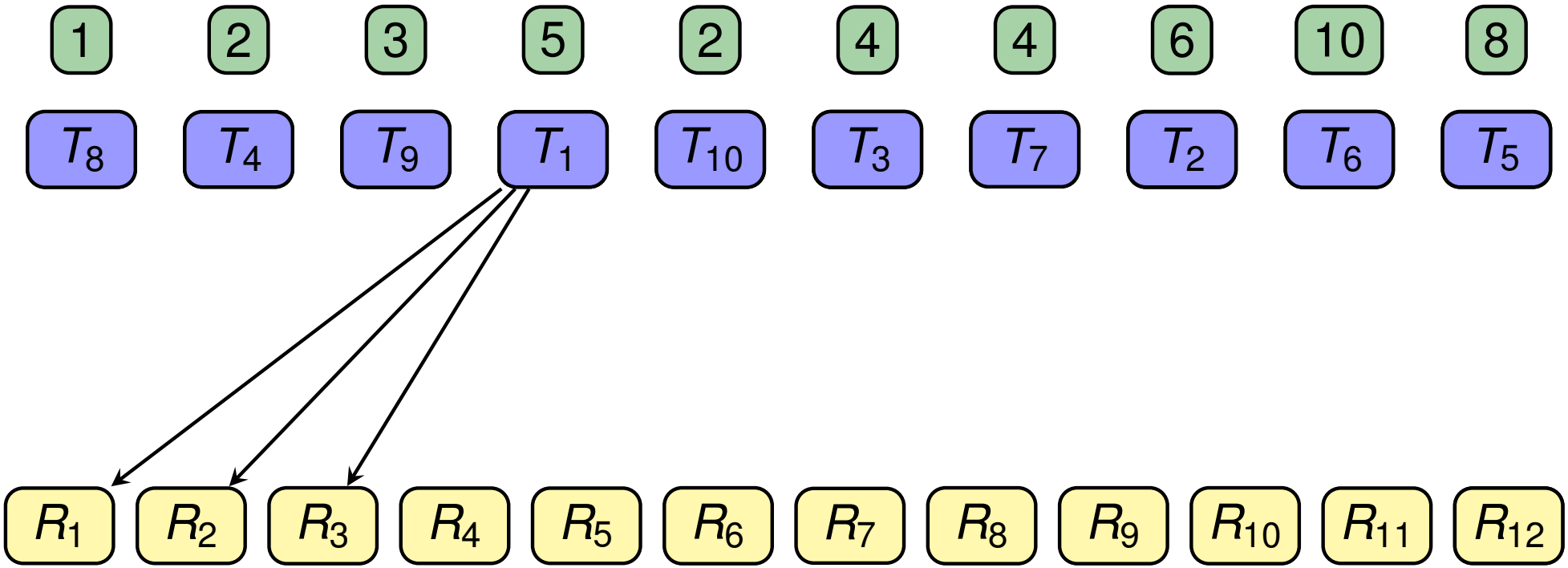

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms



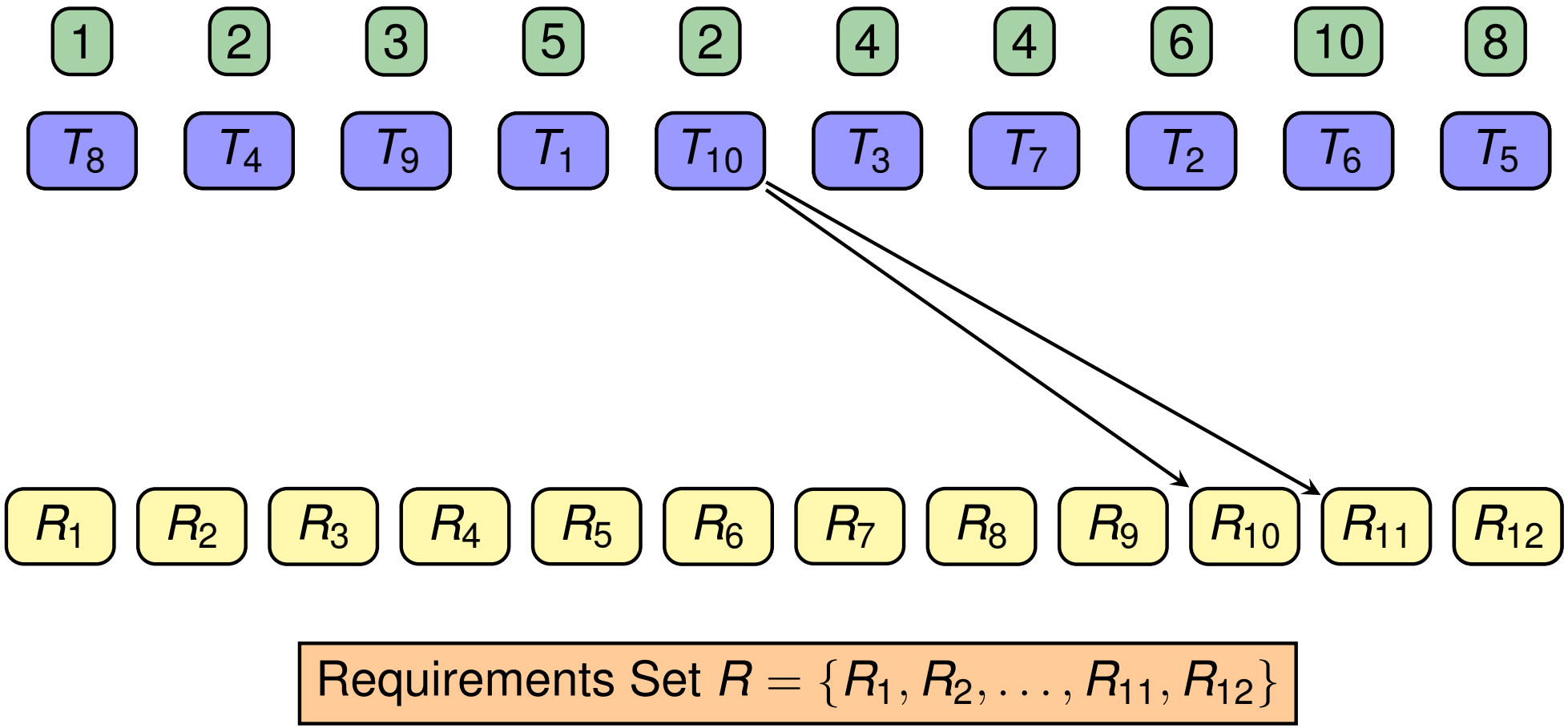

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms



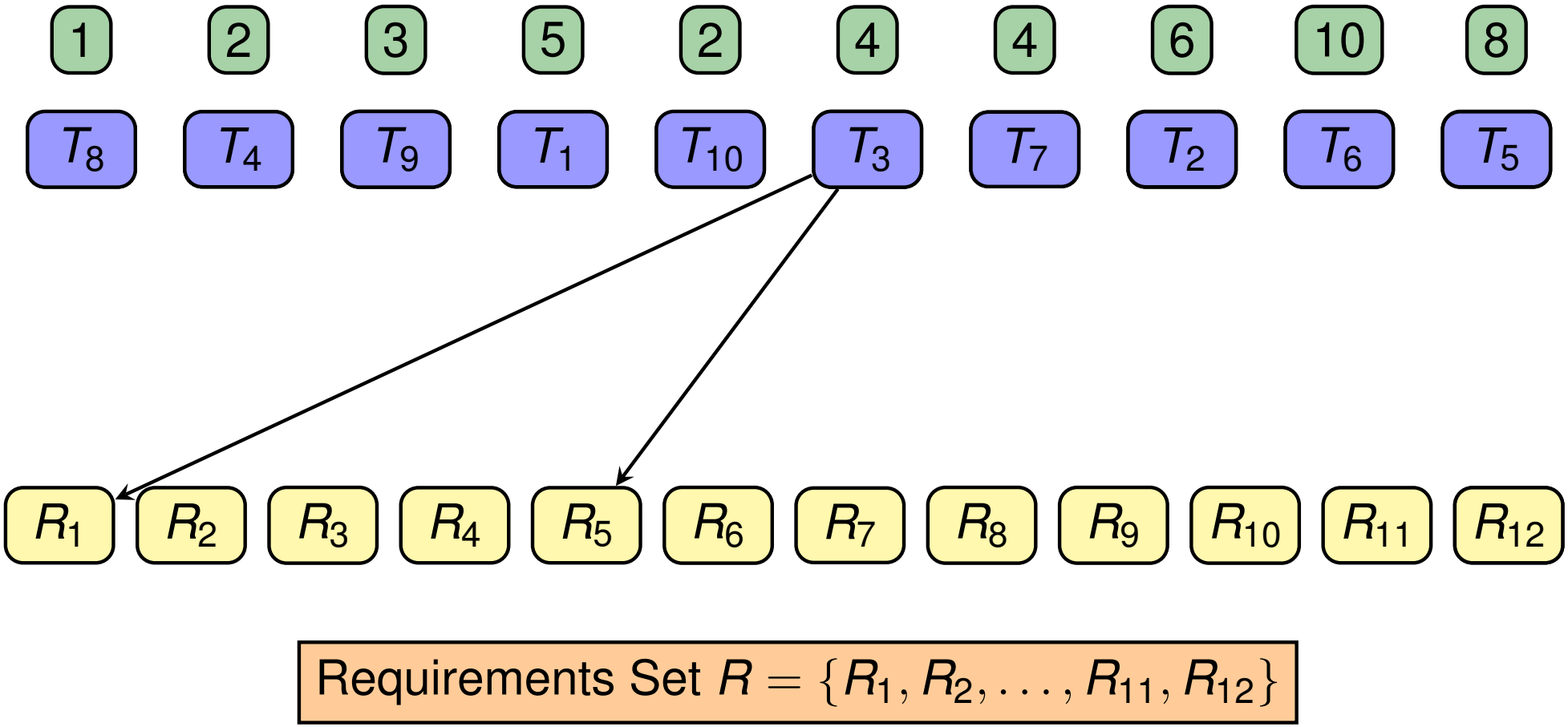

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms



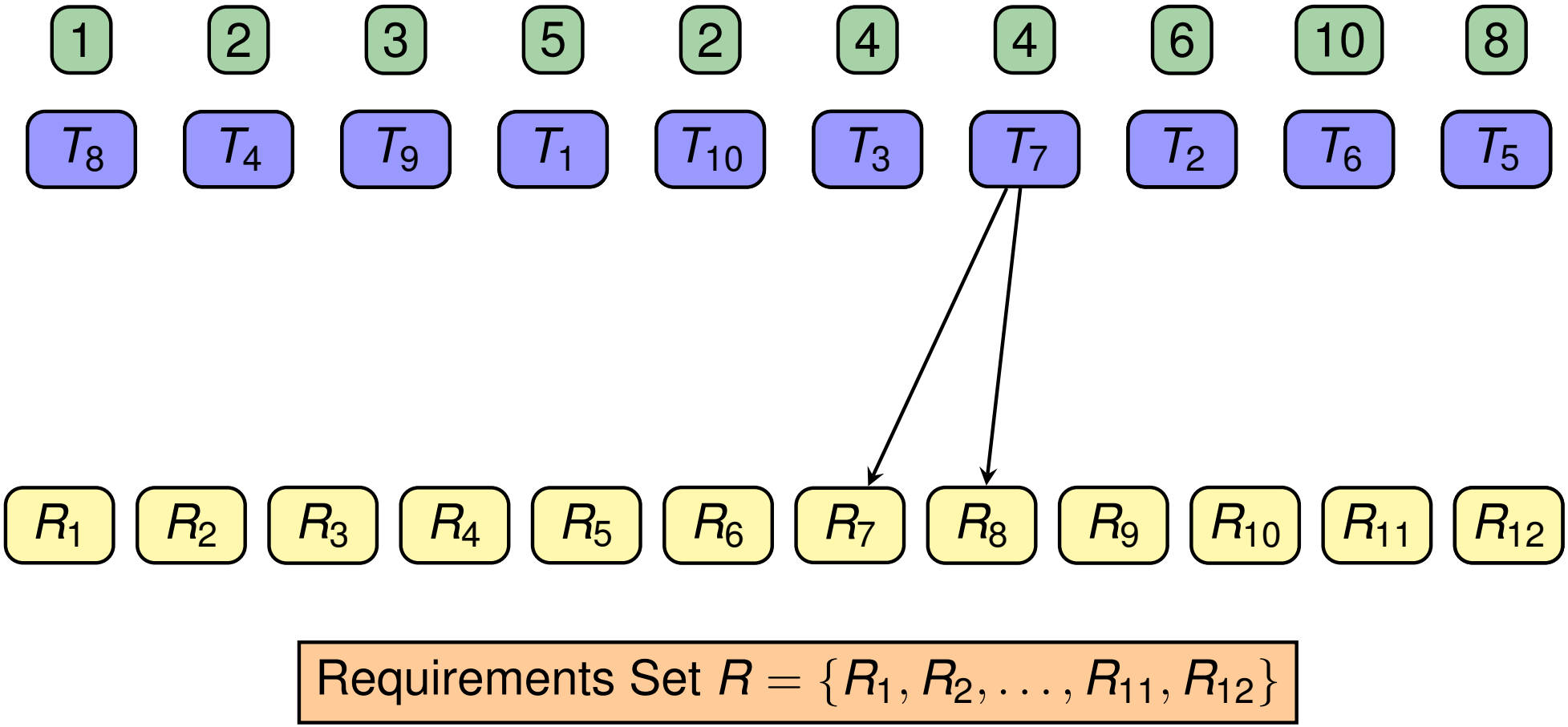

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms



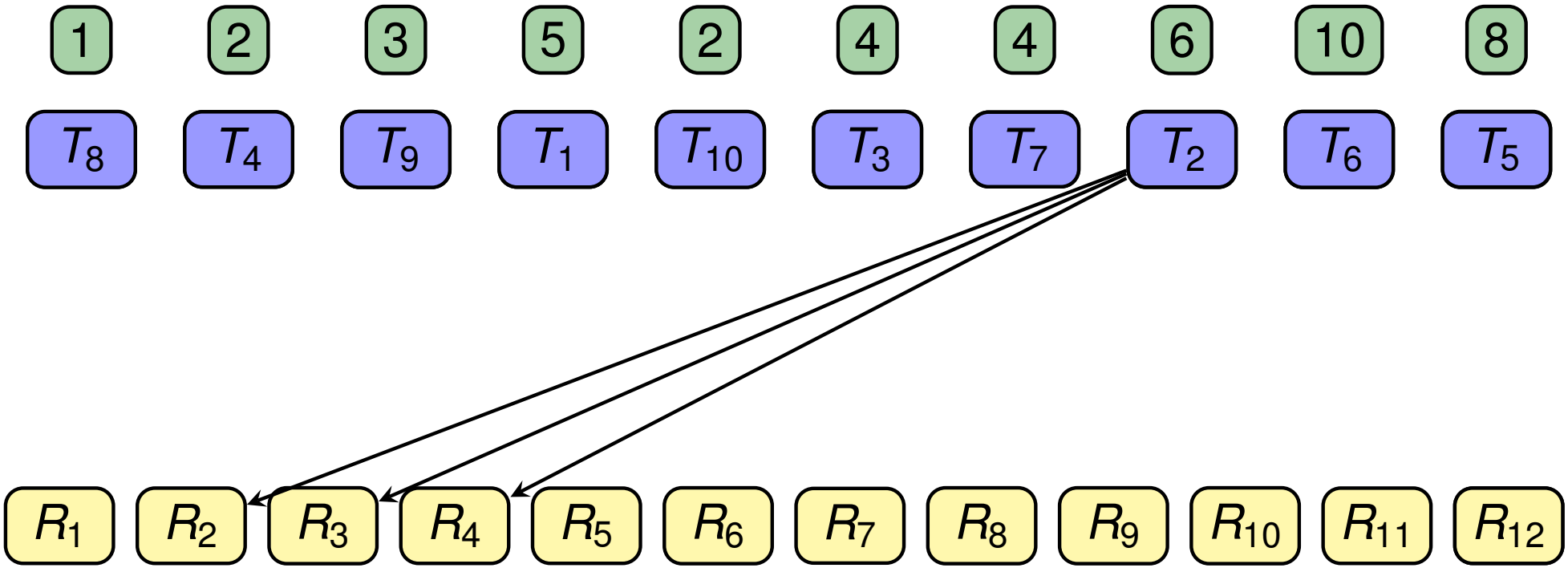

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms



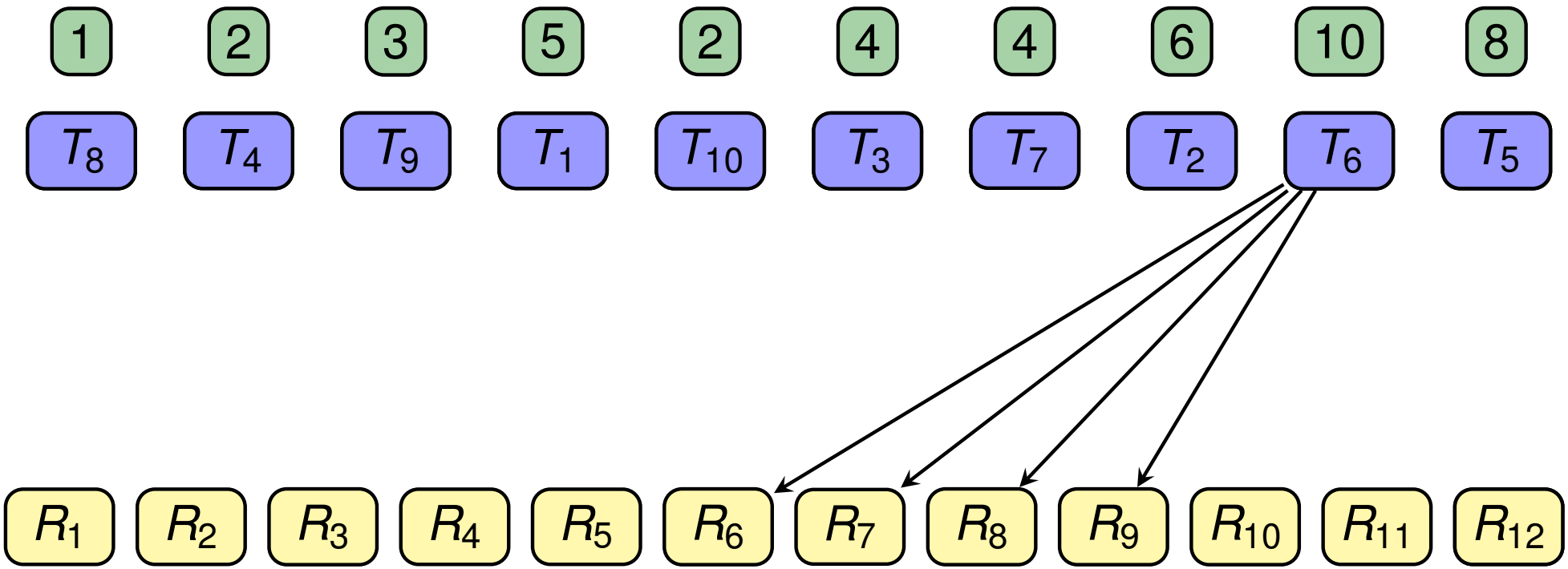

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms



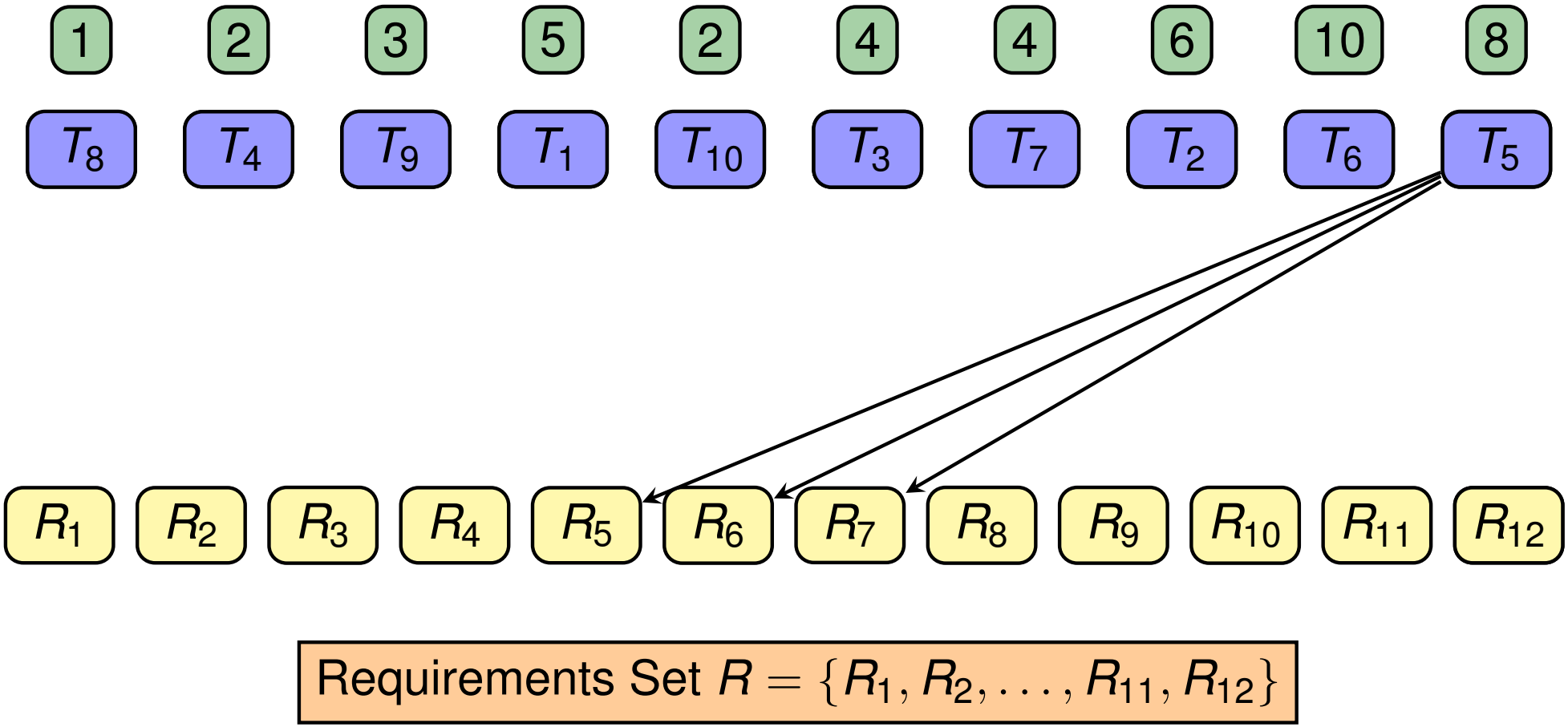

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms




Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms




Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms

Test Suite $T = \langle T_8, T_4, T_9, T_1, T_{10}, T_3, T_7 \rangle$

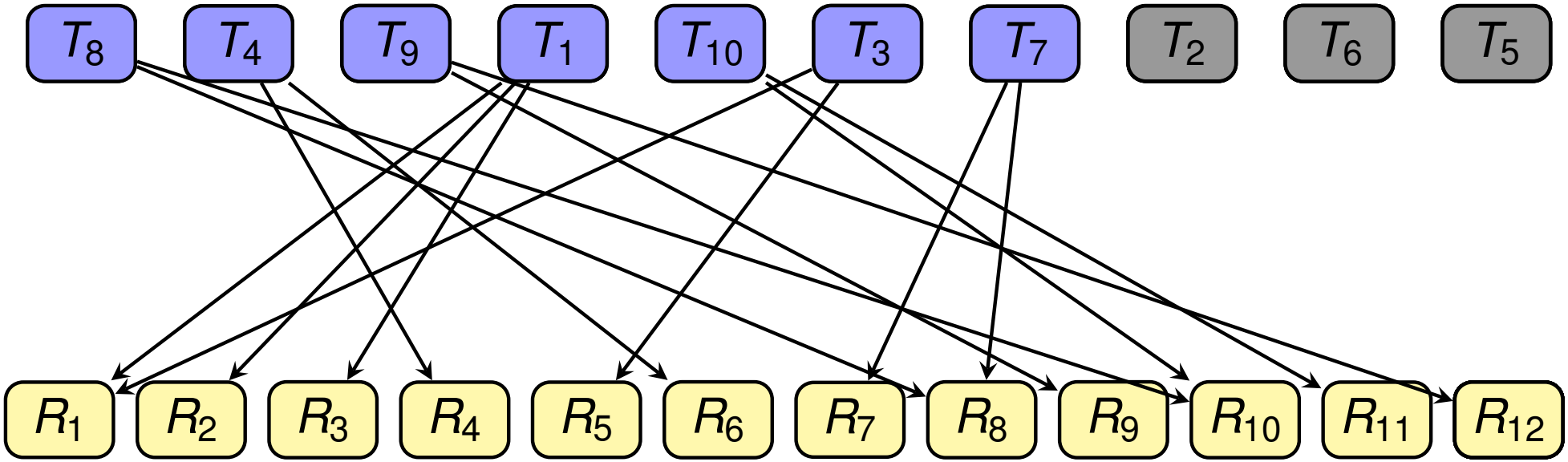
$T_8$ $T_4$ $T_9$ $T_1$ $T_{10}$ $T_3$ $T_7$ $T_2$ $T_6$ $T_5$

$R_1$ $R_2$ $R_3$ $R_4$ $R_5$ $R_6$ $R_7$ $R_8$ $R_9$ $R_{10}$ $R_{11}$ $R_{12}$

Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Greedy Algorithms

Test Suite $T = \langle T_8, T_4, T_9, T_1, T_{10}, T_3, T_7 \rangle$

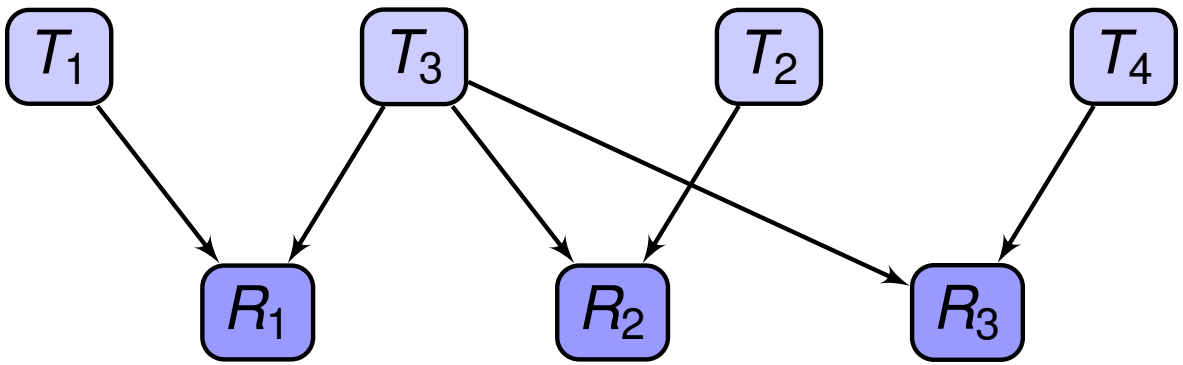



Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

# Limitations of Greedy Algorithms

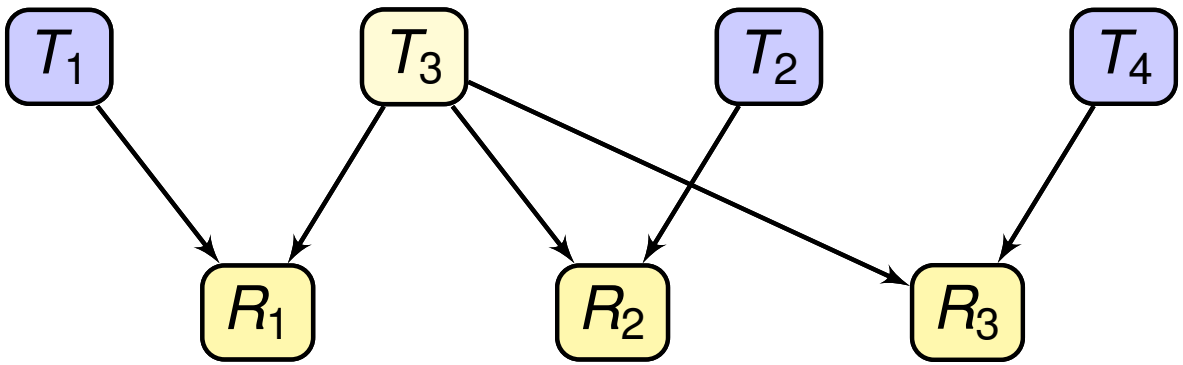

Possible configuration of the **coverage report**

# Limitations of Greedy Algorithms



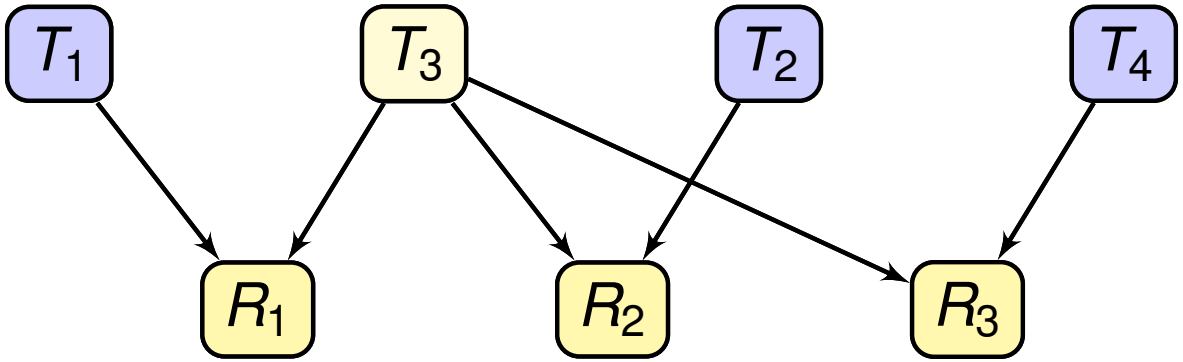

Possible configuration of the **coverage report**

# Limitations of Greedy Algorithms




Possible configuration of the **coverage report**

# Limitations of Greedy Algorithms



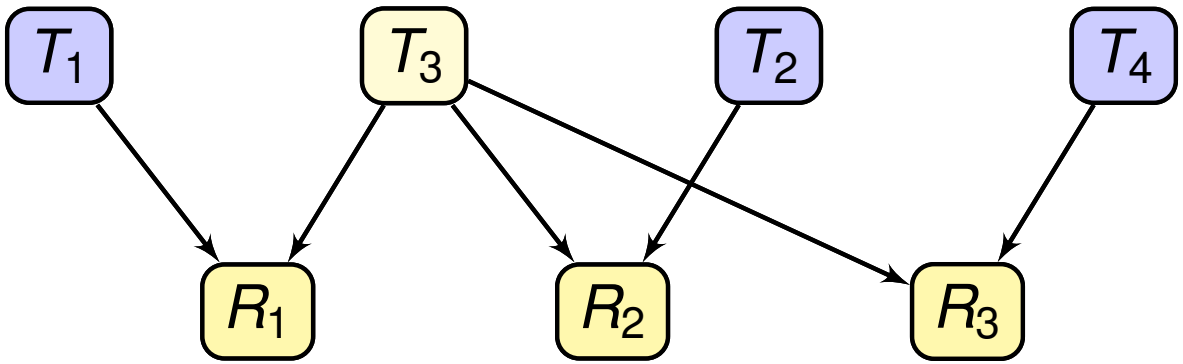

$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

**Execution time** of the test cases may mislead greedy

# Limitations of Greedy Algorithms




$time(T_1) = 1$

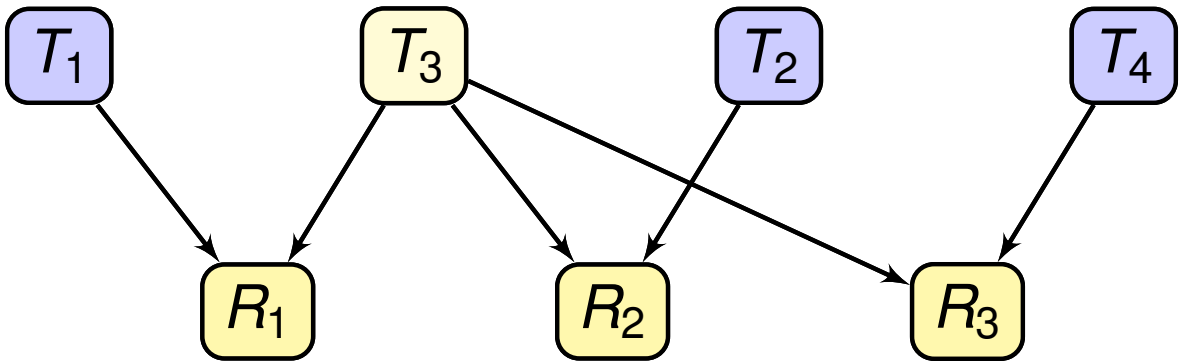
$time(T_2) = 1$

$time(T_4) = 1$

$time(T_3) = 2.45$

**Execution time** of the test cases may mislead greedy

# Limitations of Greedy Algorithms

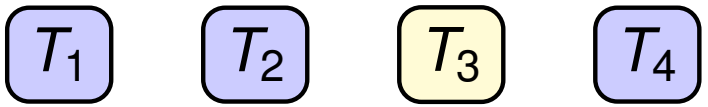



$time(T_1) = 1$

$time(T_2) = 1$

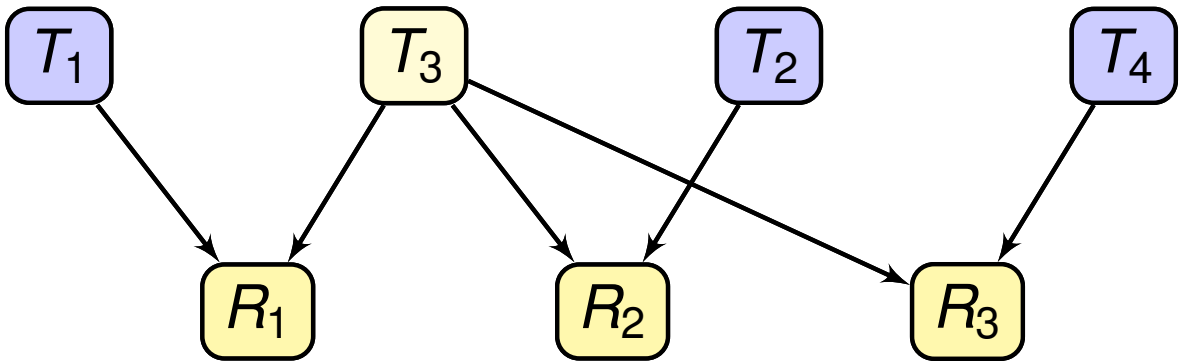
$time(T_4) = 1$

$time(T_3) = 2.45$




**Original** ordering has low effectiveness score

# Limitations of Greedy Algorithms

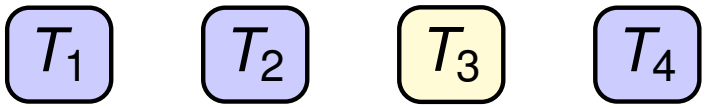



$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

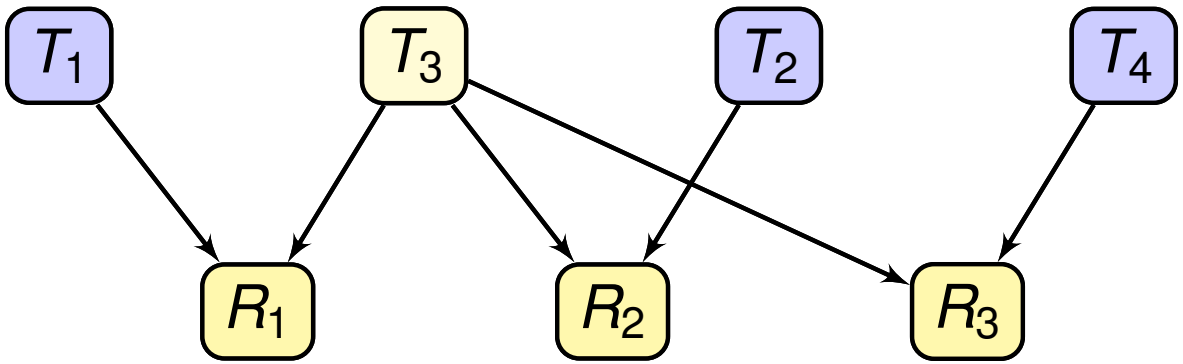
$time(T_3) = 2.45$




$CE(T) = 0.54$

**Original** ordering has low effectiveness score

# Limitations of Greedy Algorithms

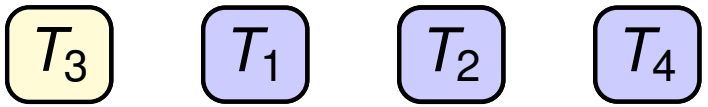



$time(T_1) = 1$

$time(T_2) = 1$

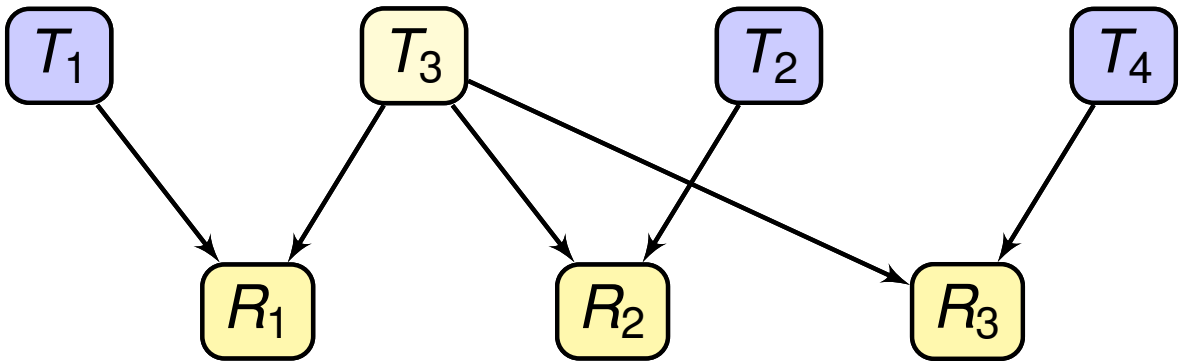
$time(T_4) = 1$

$time(T_3) = 2.45$




**Greedy** method constructs suite with marginal improvement

# Limitations of Greedy Algorithms

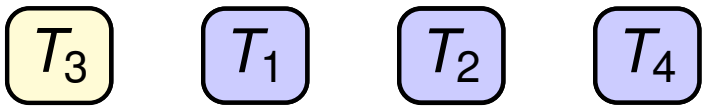



$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

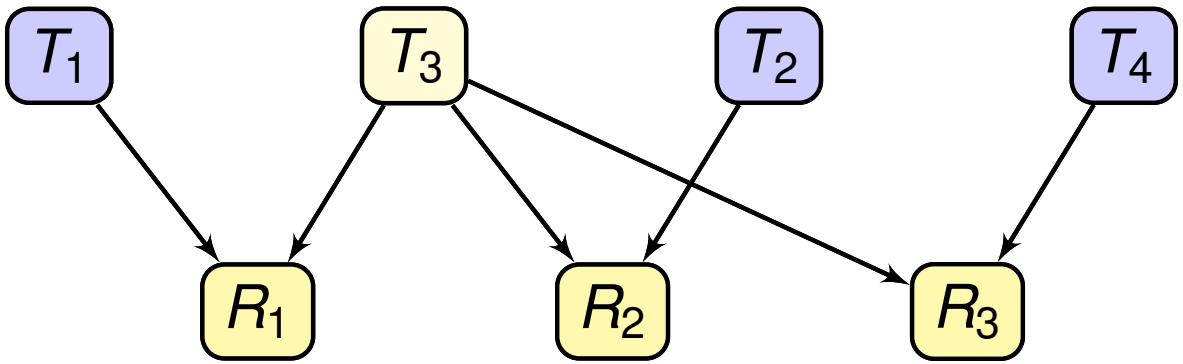
$time(T_3) = 2.45$




$CE(T') = 0.55$

**Greedy** method constructs suite with marginal improvement

# Limitations of Greedy Algorithms

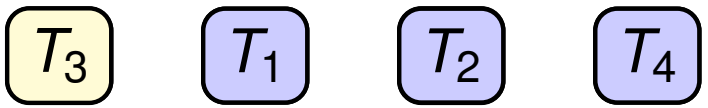



$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

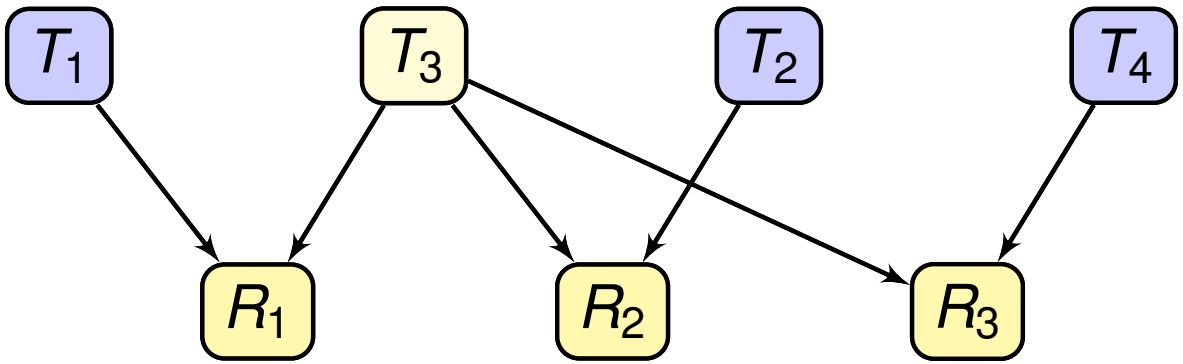
$time(T_3) = 2.45$




$CE(T') = 0.55$

**Greedy** can exhibit high run-times (Jiang et al. ASE 2009)

# Limitations of Greedy Algorithms

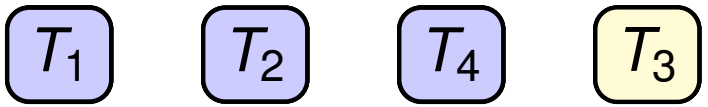



$time(T_1) = 1$

$time(T_2) = 1$

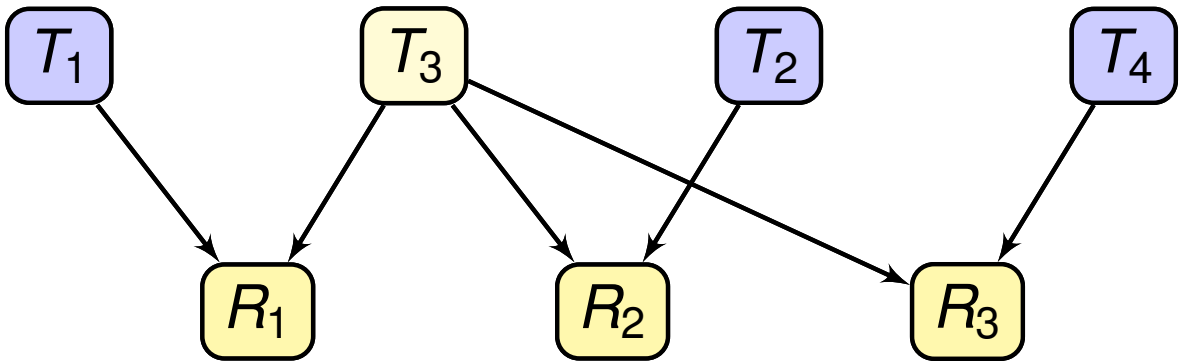
$time(T_4) = 1$

$time(T_3) = 2.45$




**Genetic** may find **better** orderings (Conrad et al. GECCO 2010)

# Limitations of Greedy Algorithms

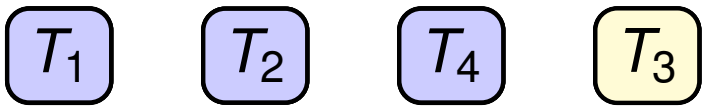



$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

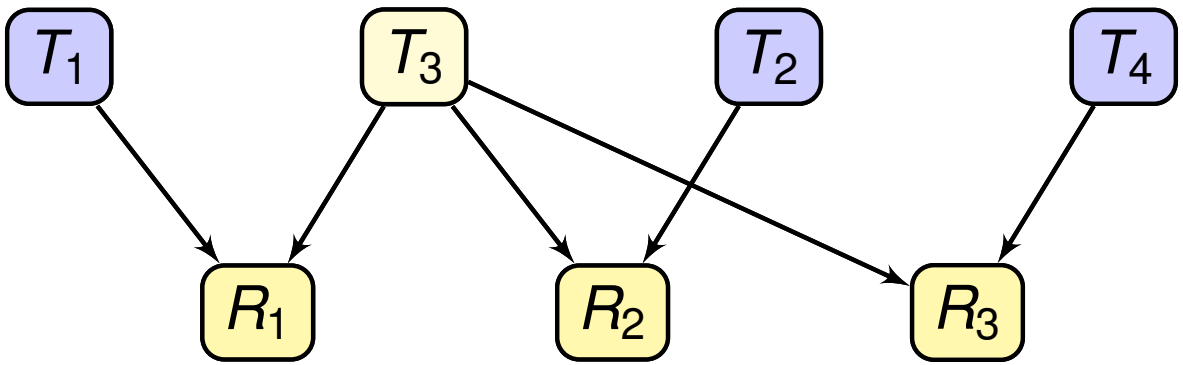
$time(T_3) = 2.45$




$CE(T') = 0.63$

**Genetic** may find **better** orderings (Conrad et al. GECCO 2010)

# Limitations of Greedy Algorithms

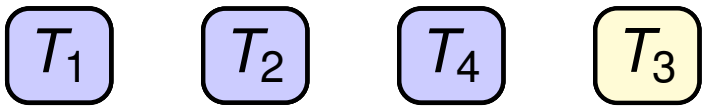



$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

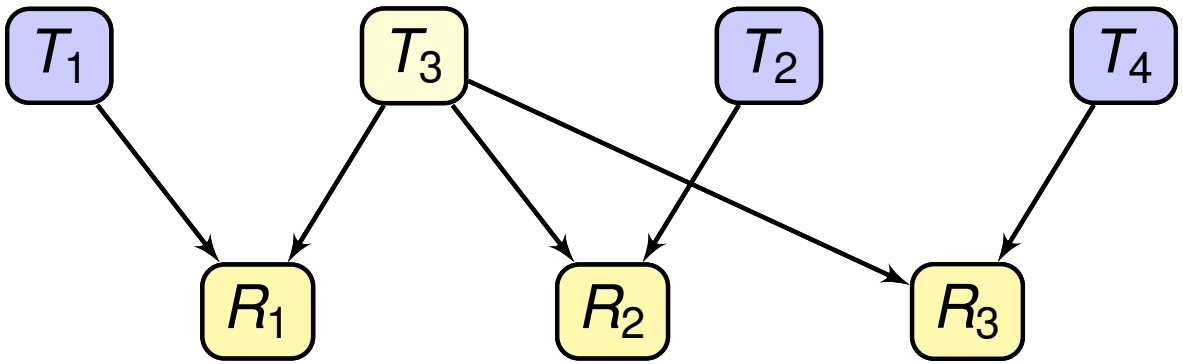
$time(T_3) = 2.45$




$CE(T') = 0.63$

**Search-based** algorithms are amenable to **parallelization**

# Limitations of Greedy Algorithms

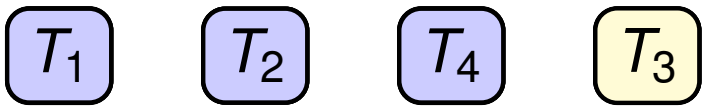



$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

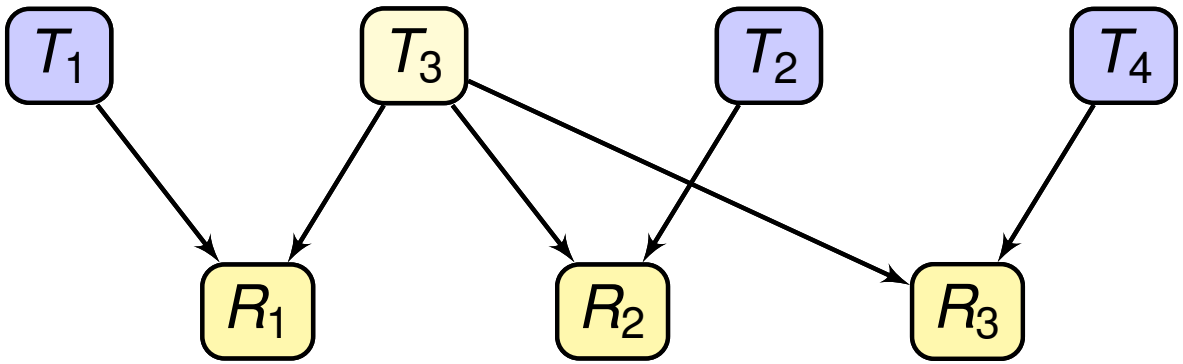
$time(T_3) = 2.45$




$CE(T') = 0.63$

**Search-based** algorithms support **“human in the loop”**

# Limitations of Greedy Algorithms

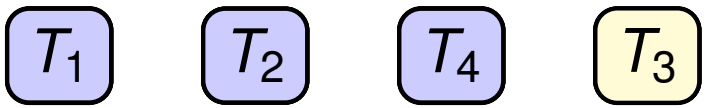



$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

$time(T_3) = 2.45$




$CE(T') = 0.63$

**Search-based** algorithms construct **diverse** test orderings

# Hill Climbing Algorithm

Explore the “neighborhood” of test suites from a starting point

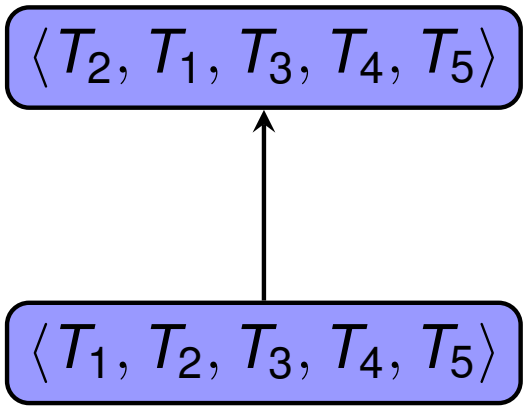
# Hill Climbing Algorithm

Explore the “neighborhood” of test suites from a starting point

$\langle T_1, T_2, T_3, T_4, T_5 \rangle$

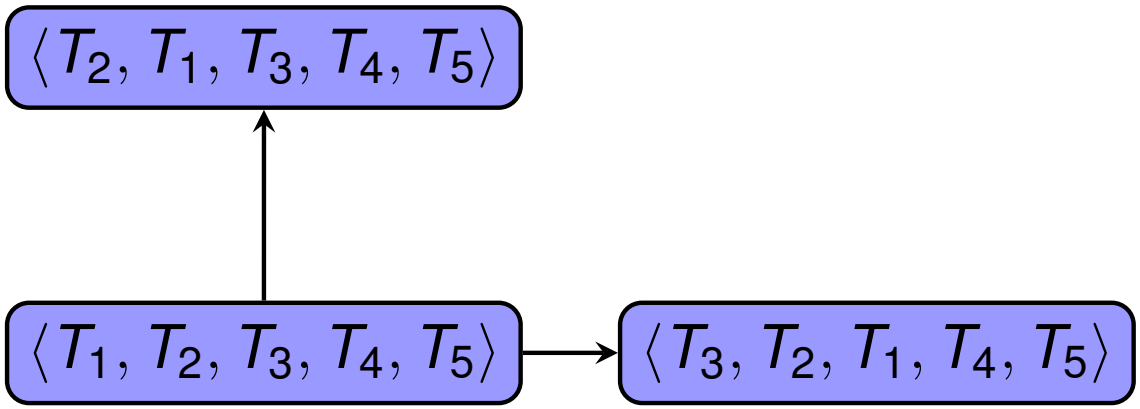
# Hill Climbing Algorithm

Explore the “neighborhood” of test suites from a starting point



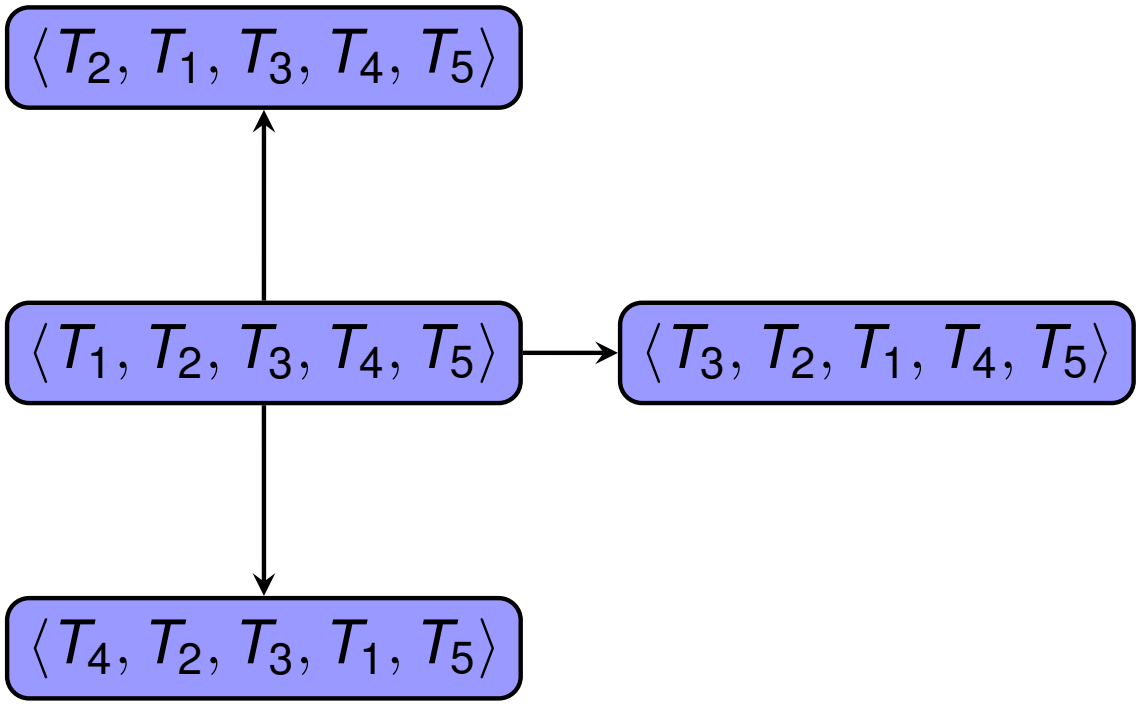
# Hill Climbing Algorithm

Explore the “neighborhood” of test suites from a starting point



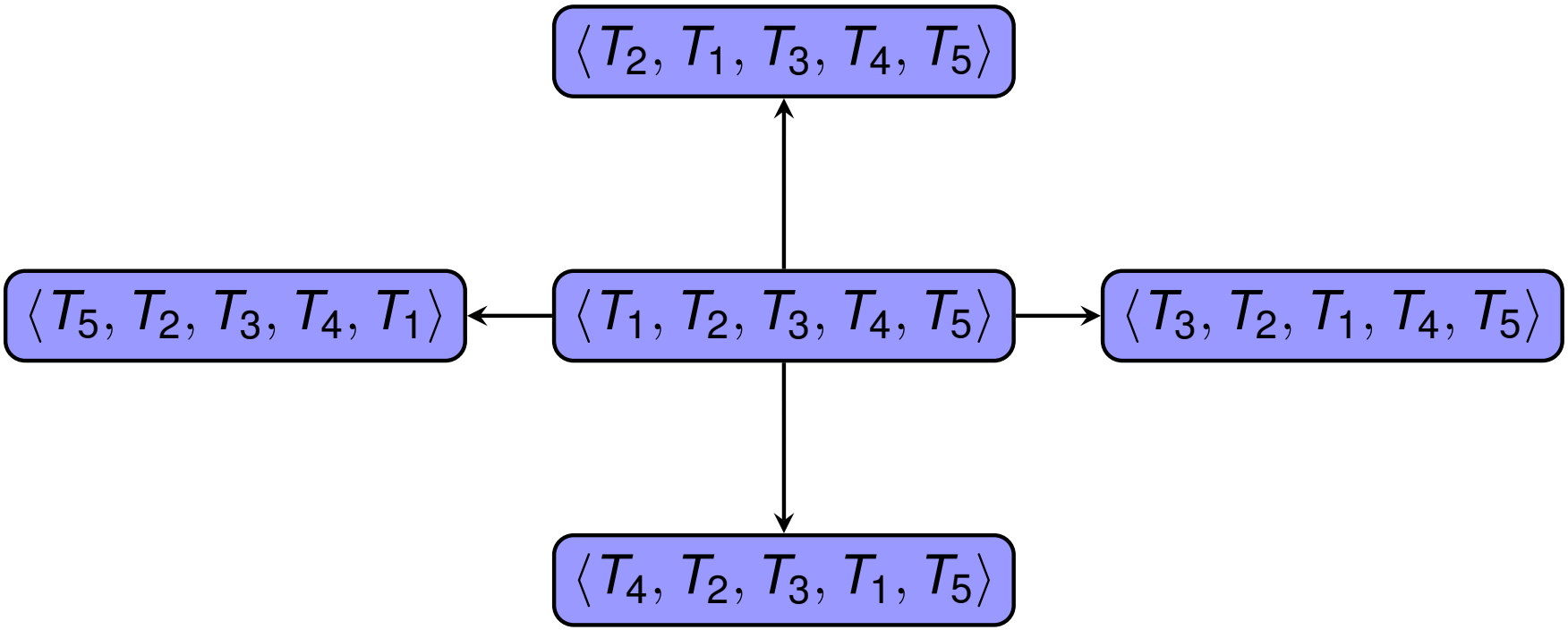
# Hill Climbing Algorithm

Explore the “neighborhood” of test suites from a starting point



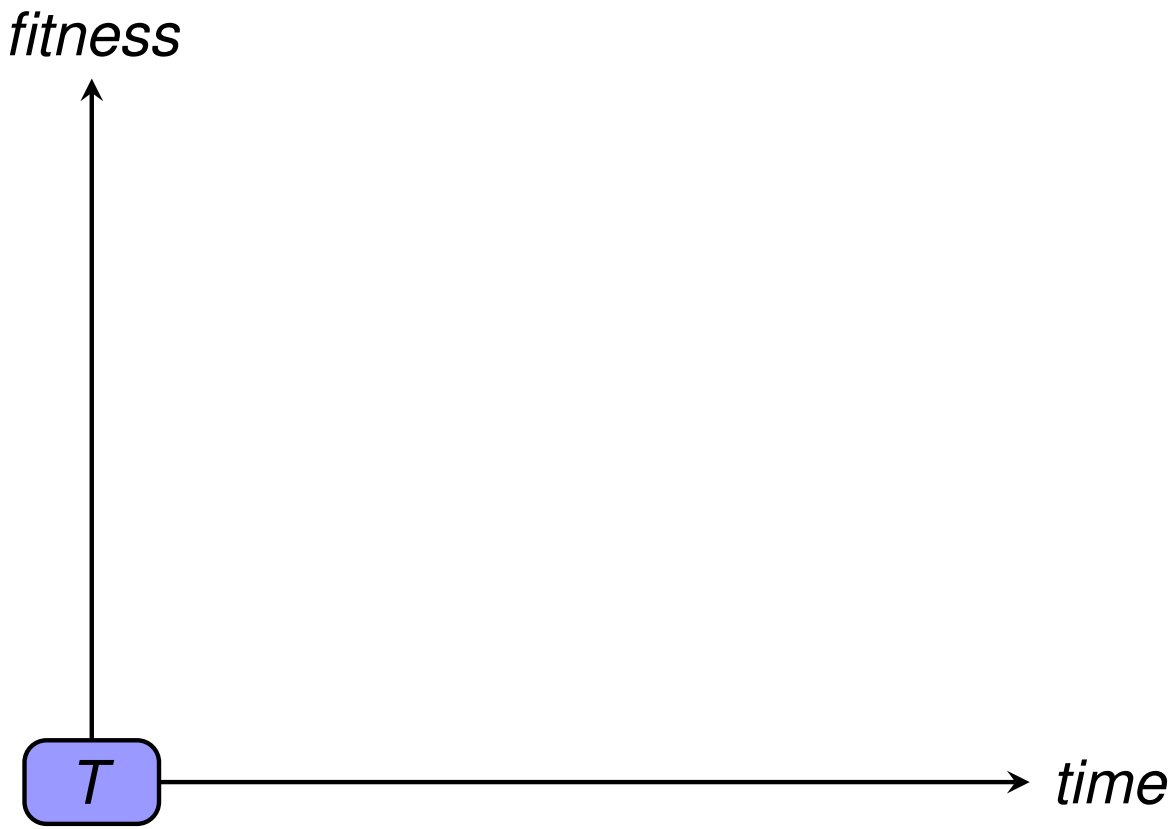
# Hill Climbing Algorithm

Explore the “neighborhood” of test suites from a starting point

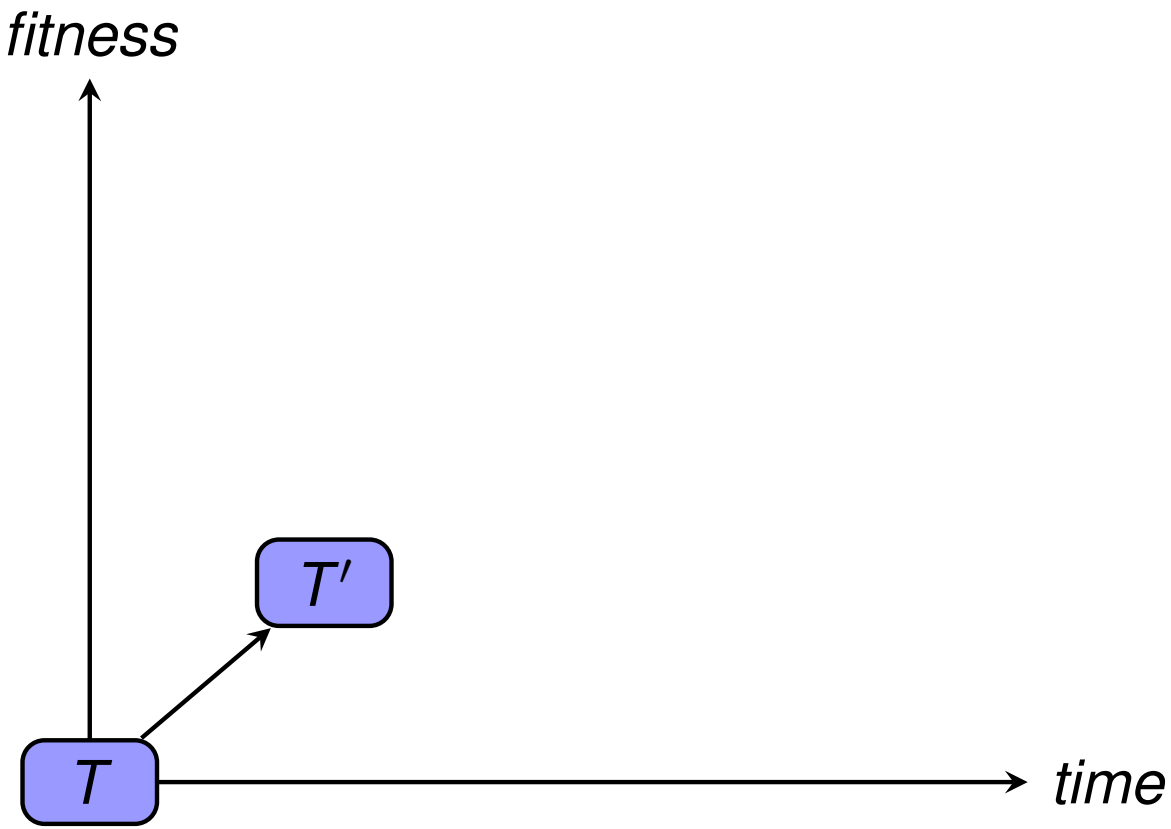


# Hill Climbing Algorithm

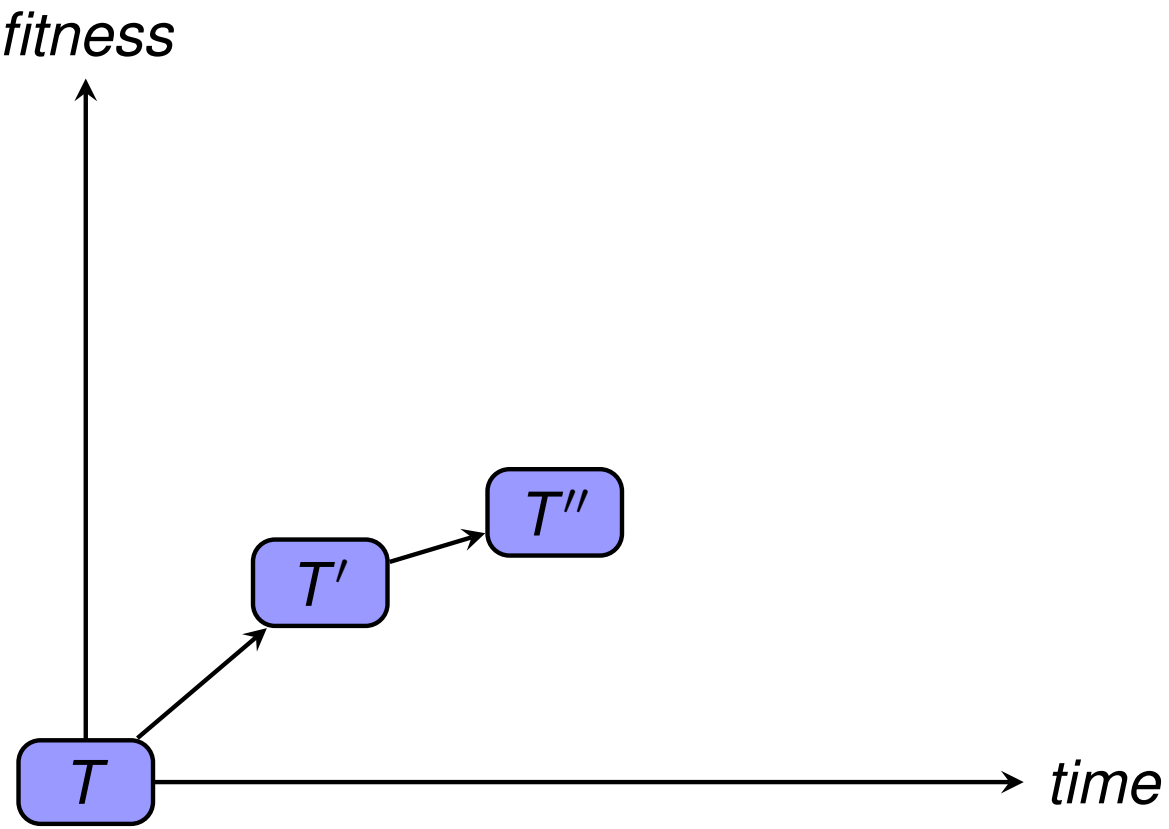
# Hill Climbing Algorithm



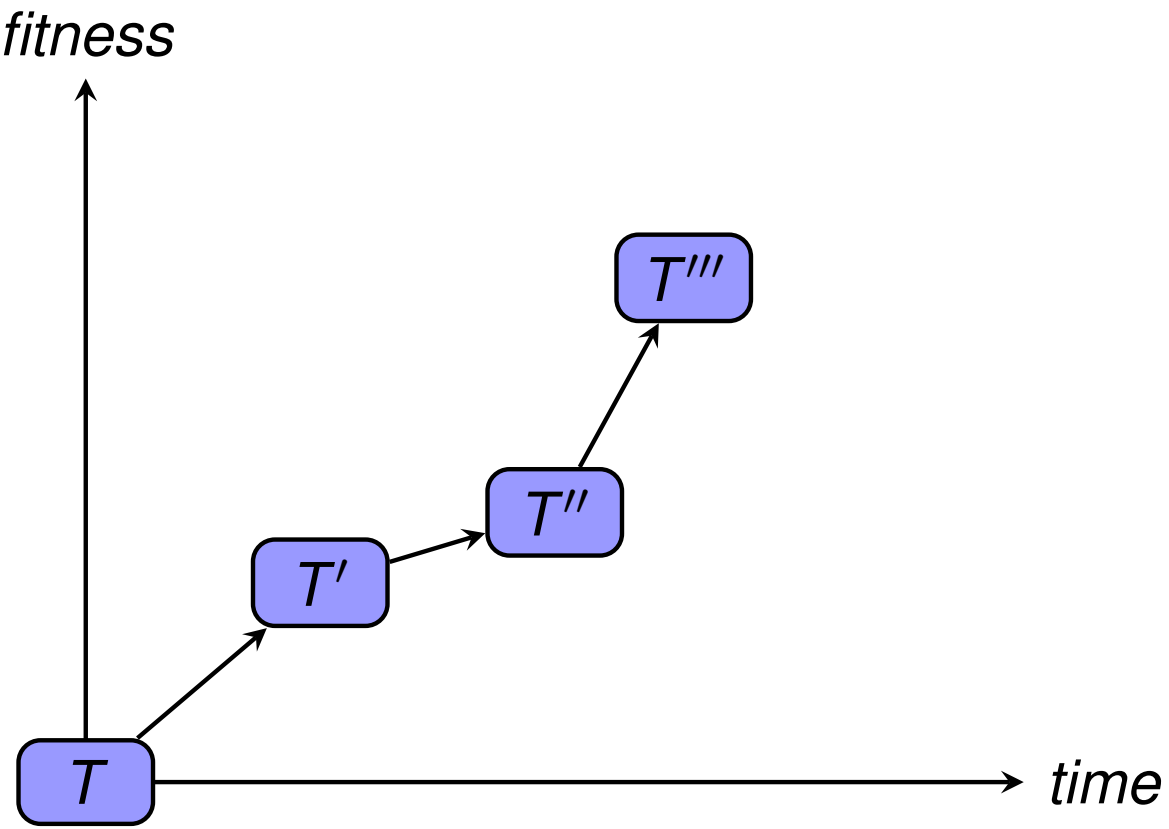
# Hill Climbing Algorithm



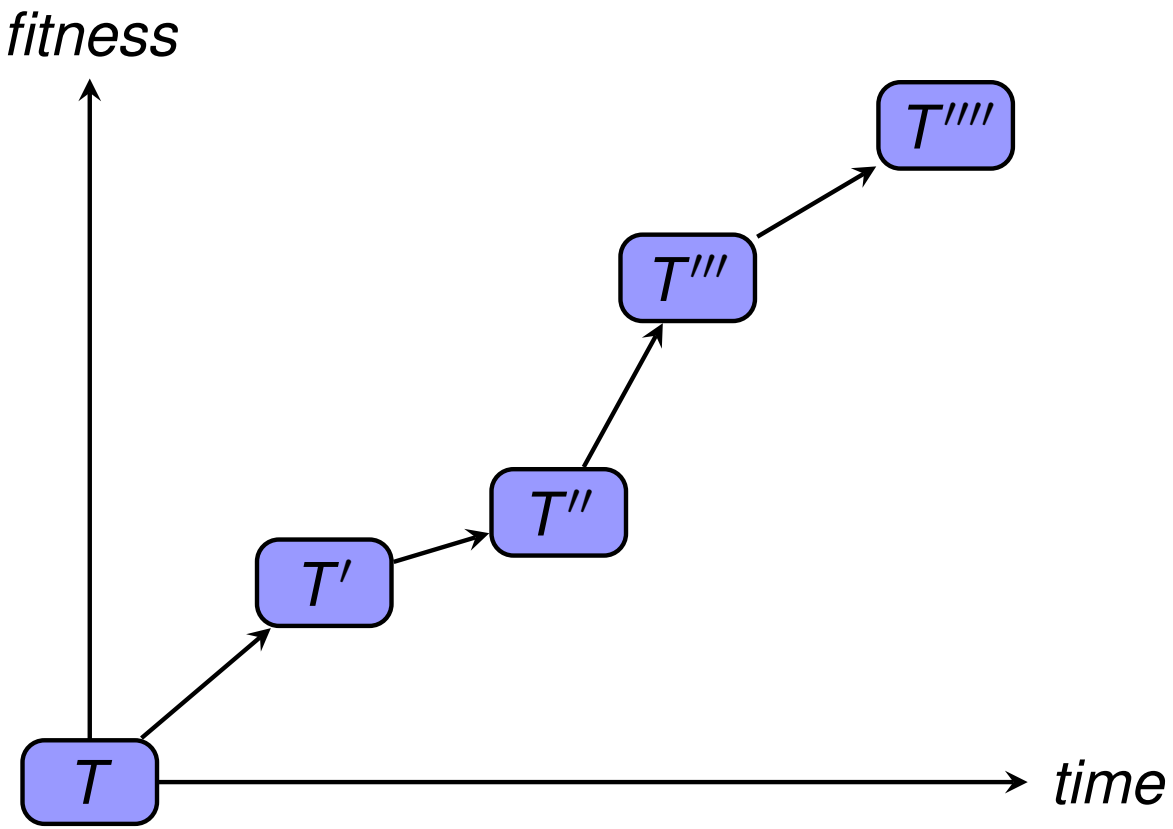
# Hill Climbing Algorithm



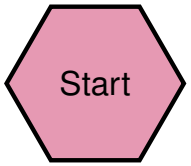
# Hill Climbing Algorithm



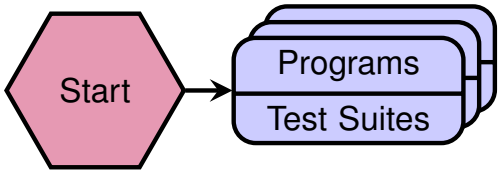
# Hill Climbing Algorithm



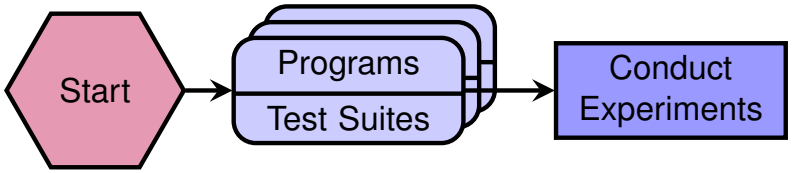
# How Do I Evaluate Regression Testing Methods?



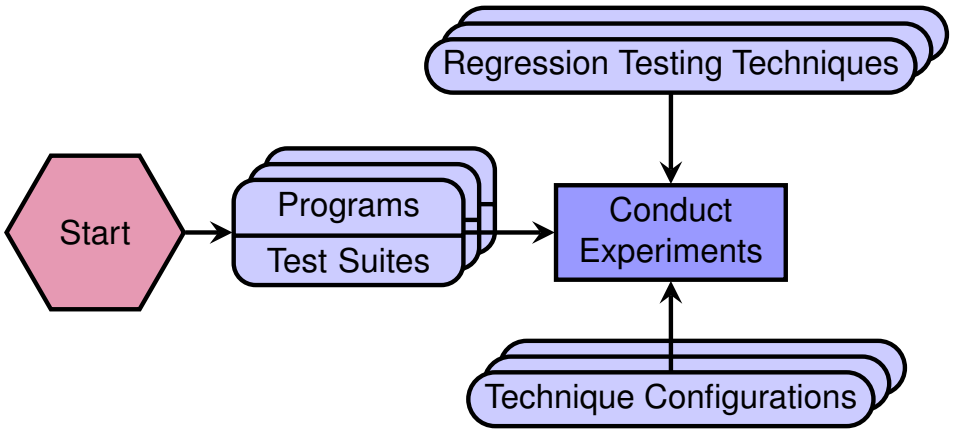
# How Do I Evaluate Regression Testing Methods?



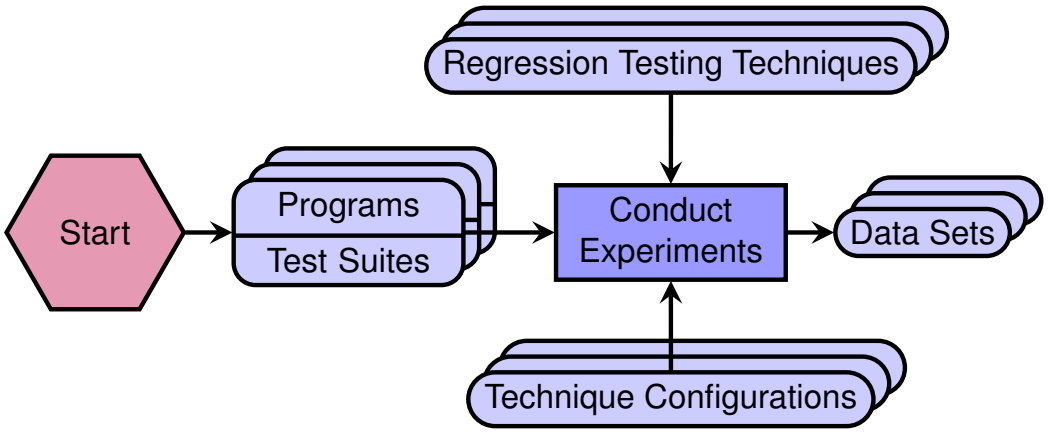
# How Do I Evaluate Regression Testing Methods?



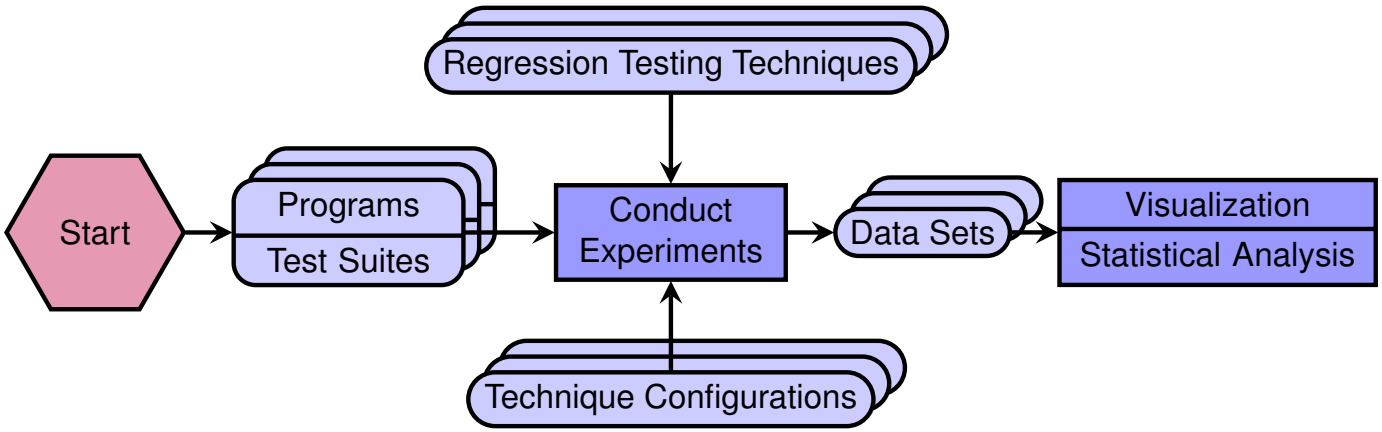
# How Do I Evaluate Regression Testing Methods?



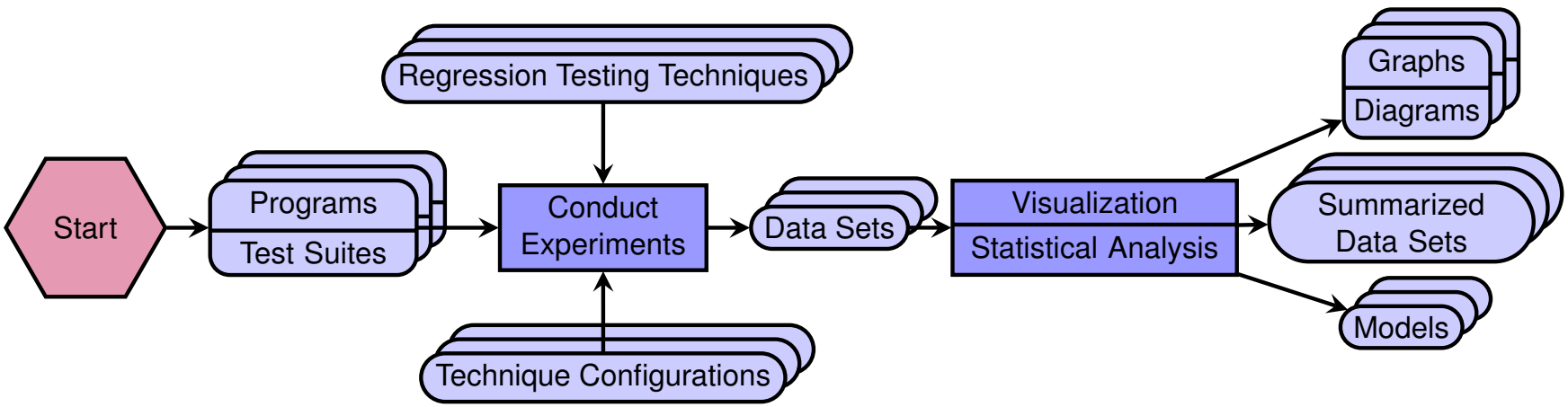
# How Do I Evaluate Regression Testing Methods?



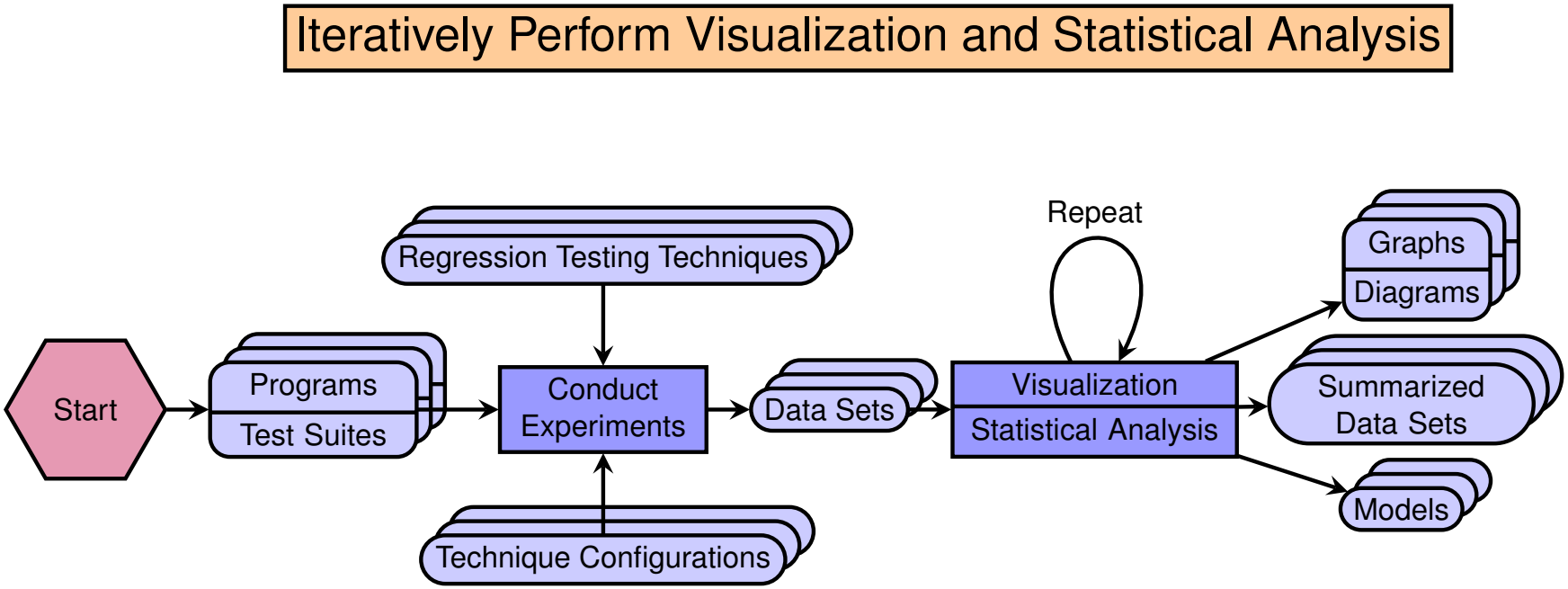
# How Do I Evaluate Regression Testing Methods?



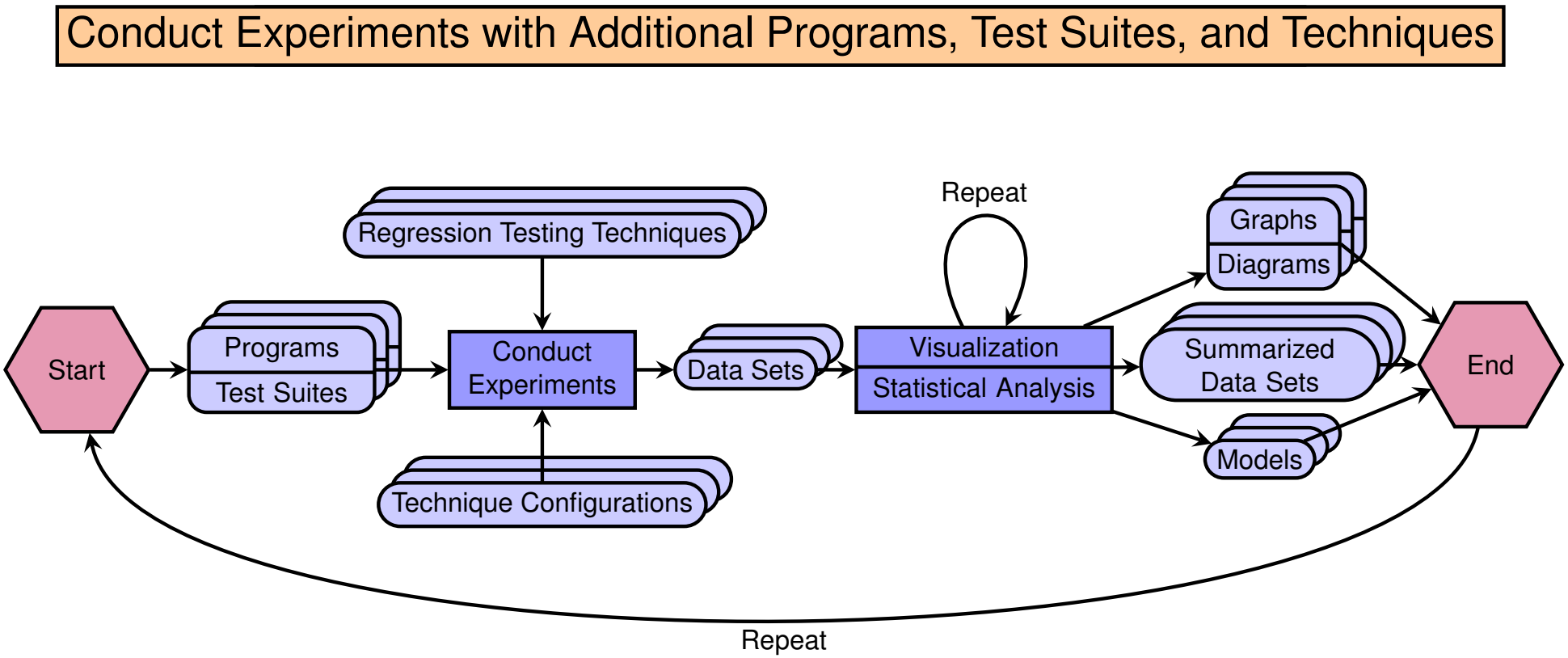
# How Do I Evaluate Regression Testing Methods?



# How Do I Evaluate Regression Testing Methods?

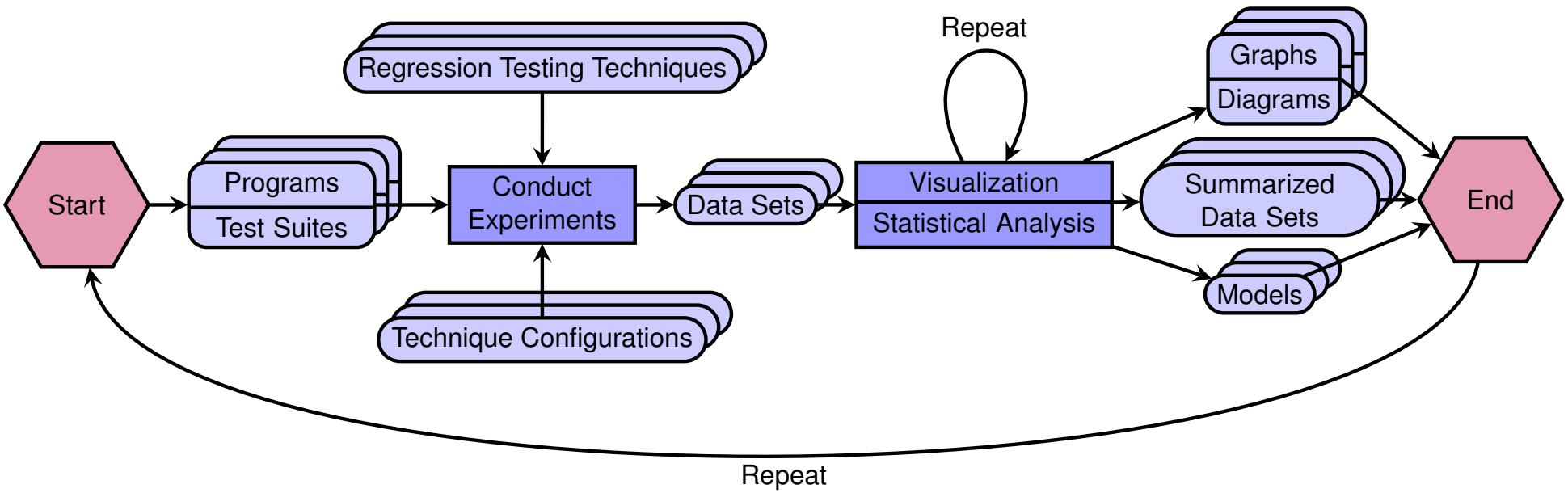


# How Do I Evaluate Regression Testing Methods?



# How Do I Evaluate Regression Testing Methods?

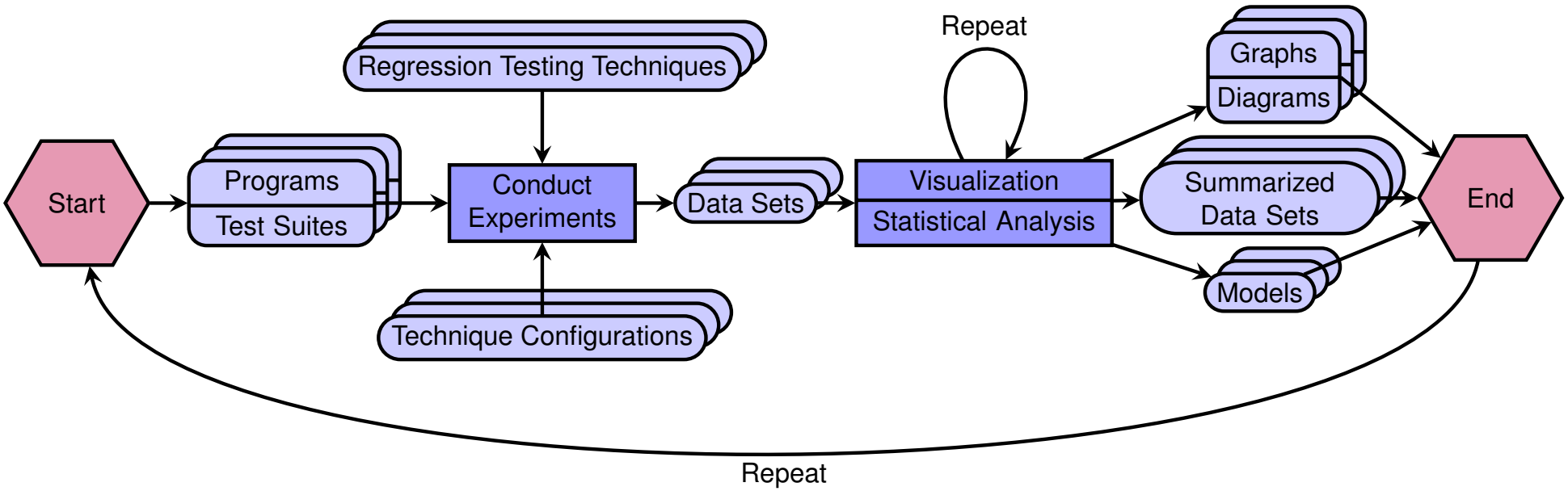
Conduct Experiments with Additional Programs, Test Suites, and Techniques




Our tools support all of these tasks!

# How Do I Evaluate Regression Testing Methods?

Conduct Experiments with Additional Programs, Test Suites, and Techniques




Greedy, Hill Climbing, Random, Adaptive Random, Simulated Annealing, Genetic

# Case Study Applications

| Application | Program Size | | | | |
|---|---|---|---|---|---|
| | Lines | Methods | Classes | Faults | Test Cases |
| CommissionEmployee | 34 | 18 | 2 | 95 | 15 |
| Point | 125 | 26 | 3 | 44 | 13 |
| DataStructures | 189 | 57 | 8 | 324 | 106 |
| Employee | 192 | 52 | 7 | 84 | 14 |
| LoopFinder | 193 | 26 | 4 | 34 | 13 |
| Sudoku | 231 | 58 | 4 | 414 | 25 |
| JDepend | 1,462 | 282 | 35 | 2,659 | 39 |
| Reduction and Prioritization | 2,050 | 211 | 19 | 1,412 | 38 |
| Barbecue | 2,501 | 422 | 59 | 18,312 | 140 |
| JodaTime | 12,687 | 3,644 | 223 | 20,894 | 206 |
| CommonsMath | 20,763 | 4,185 | 556 | 6,077 | 268 |
| Total | 40,427 | 8,981 | 920 | 50,349 | 877 |
| Average | 3,675 | 816 | 84 | 4,577 | 80 |

# Empirical Results: Greedy and Search-Based

| Prioritizer | NAPSC | Runtime (sec) | Application |
|---|---|---|---|
| GRD | 0.98 | 0.22 | Sudoku |
| GRD | 0.36 | 0.38 | ReductionAndPrioritization |
| GRD | 0.71 | 33.88 | JodaTime |
| HC_SA_FS | 0.98 | 0.01 | Sudoku |
| HC_SA_FS | 0.36 | 0.04 | ReductionAndPrioritization |
| HC_SA_FS | 0.69 | 7.88 | JodaTime |

# Empirical Results: Greedy and Search-Based

| Prioritizer | NAPSC | Runtime (sec) | Application |
|---|---|---|---|
| GRD | 0.98 | 0.22 | Sudoku |
| GRD | 0.36 | 0.38 | ReductionAndPrioritization |
| GRD | 0.71 | 33.88 | JodaTime |
| HC_SA_FS | 0.98 | 0.01 | Sudoku |
| HC_SA_FS | 0.36 | 0.04 | ReductionAndPrioritization |
| HC_SA_FS | 0.69 | 7.88 | JodaTime |

# Empirical Results: Greedy and Search-Based

| Prioritizer | NAPSC | Runtime (sec) | Application |
|---|---|---|---|
| GRD | 0.98 | 0.22 | Sudoku |
| GRD | 0.36 | 0.38 | ReductionAndPrioritization |
| GRD | 0.71 | 33.88 | JodaTime |
| HC_SA_FS | 0.98 | 0.01 | Sudoku |
| HC_SA_FS | 0.36 | 0.04 | ReductionAndPrioritization |
| HC_SA_FS | 0.69 | 7.88 | JodaTime |

# Empirical Results: Greedy and Search-Based

| Prioritizer | NAPSC | Runtime (sec) | Application |
|---|---|---|---|
| GRD | 0.98 | 0.22 | Sudoku |
| GRD | 0.36 | 0.38 | ReductionAndPrioritization |
| GRD | 0.71 | 33.88 | JodaTime |
| HC_SA_FS | 0.98 | 0.01 | Sudoku |
| HC_SA_FS | 0.36 | 0.04 | ReductionAndPrioritization |
| HC_SA_FS | 0.69 | 7.88 | JodaTime |

# Empirical Results: Greedy and Search-Based

| Prioritizer | NAPSC | Runtime (sec) | Application |
|---|---|---|---|
| GRD | 0.98 | 0.22 | Sudoku |
| GRD | 0.36 | 0.38 | ReductionAndPrioritization |
| GRD | 0.71 | 33.88 | JodaTime |
| HC_SA_FS | 0.98 | 0.01 | Sudoku |
| HC_SA_FS | 0.36 | 0.04 | ReductionAndPrioritization |
| HC_SA_FS | 0.69 | 7.88 | JodaTime |

# Empirical Results: Greedy and Search-Based

| Prioritizer | NAPSC | Runtime (sec) | Application |
|---|---|---|---|
| GRD | 0.98 | 0.22 | Sudoku |
| GRD | 0.36 | 0.38 | ReductionAndPrioritization |
| GRD | 0.71 | 33.88 | JodaTime |
| HC_SA_FS | 0.98 | 0.01 | Sudoku |
| HC_SA_FS | 0.36 | 0.04 | ReductionAndPrioritization |
| HC_SA_FS | 0.69 | 7.88 | JodaTime |

# Empirical Results: Greedy and Search-Based

| Prioritizer | NAPSC | Runtime (sec) | Application |
|---|---|---|---|
| GRD | 0.98 | 0.22 | Sudoku |
| GRD | 0.36 | 0.38 | ReductionAndPrioritization |
| GRD | 0.71 | 33.88 | JodaTime |
| HC_SA_FS | 0.98 | 0.01 | Sudoku |
| HC_SA_FS | 0.36 | 0.04 | ReductionAndPrioritization |
| HC_SA_FS | 0.69 | 7.88 | JodaTime |

Greedy and hill climbing produce comparable orderings

# Empirical Results: Greedy and Search-Based

| Prioritizer | NAPSC | Runtime (sec) | Application |
|---|---|---|---|
| GRD | 0.98 | 0.22 | Sudoku |
| GRD | 0.36 | 0.38 | ReductionAndPrioritization |
| GRD | 0.71 | 33.88 | JodaTime |
| HC_SA_FS | 0.98 | 0.01 | Sudoku |
| HC_SA_FS | 0.36 | 0.04 | ReductionAndPrioritization |
| HC_SA_FS | 0.69 | 7.88 | JodaTime |

However, hill climbing is slightly more efficient than greedy

# Empirical Results: Greedy and Search-Based

| Prioritizer | NAPSC | Runtime (sec) | Application |
|---|---|---|---|
| GRD | 0.98 | 0.22 | Sudoku |
| GRD | 0.36 | 0.38 | ReductionAndPrioritization |
| GRD | 0.71 | 33.88 | JodaTime |
| HC_SA_FS | 0.98 | 0.01 | Sudoku |
| HC_SA_FS | 0.36 | 0.04 | ReductionAndPrioritization |
| HC_SA_FS | 0.69 | 7.88 | JodaTime |

Greedy produces a slightly better ordering than hill climbing

# Empirical Results: Greedy and Search-Based

| Prioritizer | NAPSC | Runtime (sec) | Application |
|---|---|---|---|
| GRD | 0.98 | 0.22 | Sudoku |
| GRD | 0.36 | 0.38 | ReductionAndPrioritization |
| GRD | 0.71 | 33.88 | JodaTime |
| HC_SA_FS | 0.98 | 0.01 | Sudoku |
| HC_SA_FS | 0.36 | 0.04 | ReductionAndPrioritization |
| HC_SA_FS | 0.69 | 7.88 | JodaTime |

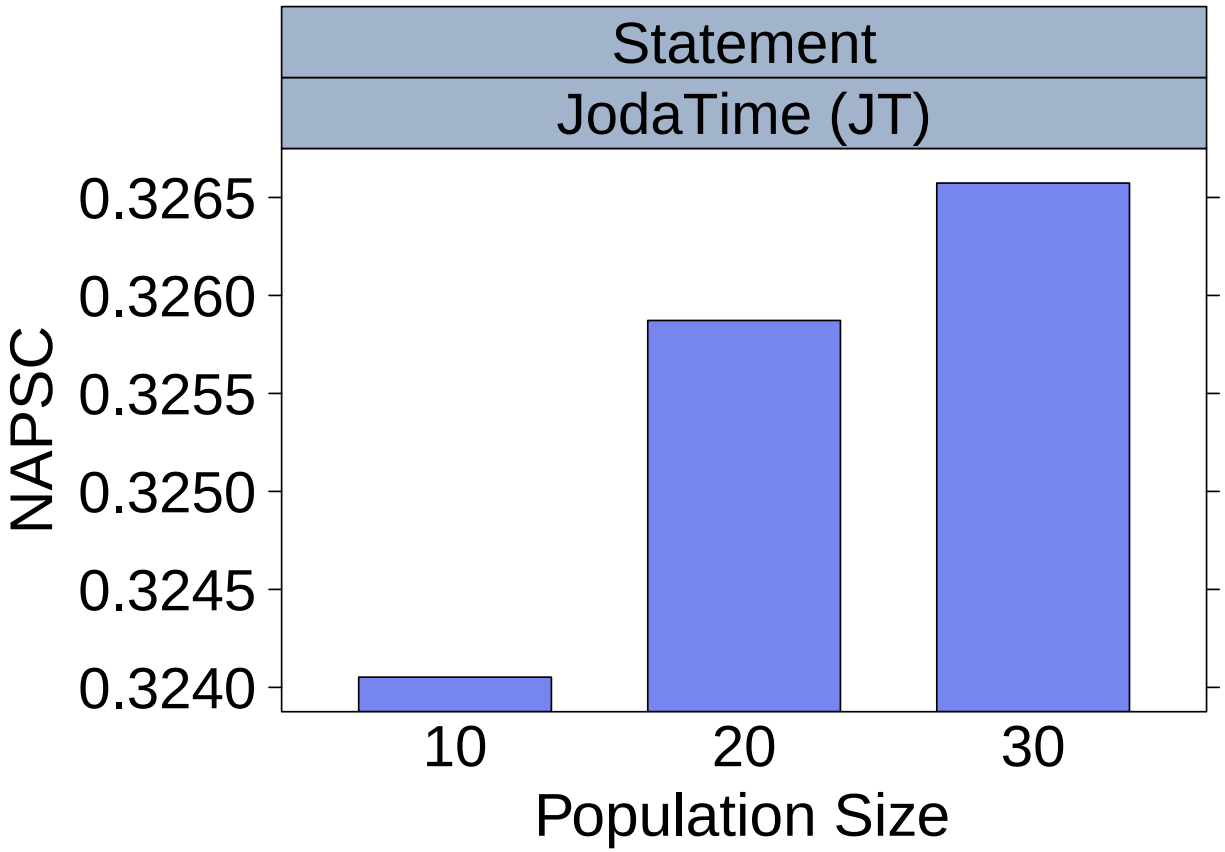
But the hill climbing algorithm executes over four times faster!

# Empirical Results: Greedy and Search-Based

| Prioritizer | NAPSC | Runtime (sec) | Application |
|---|---|---|---|
| GRD | 0.98 | 0.22 | Sudoku |
| GRD | 0.36 | 0.38 | ReductionAndPrioritization |
| GRD | 0.71 | 33.88 | JodaTime |
| HC_SA_FS | 0.98 | 0.01 | Sudoku |
| HC_SA_FS | 0.36 | 0.04 | ReductionAndPrioritization |
| HC_SA_FS | 0.69 | 7.88 | JodaTime |

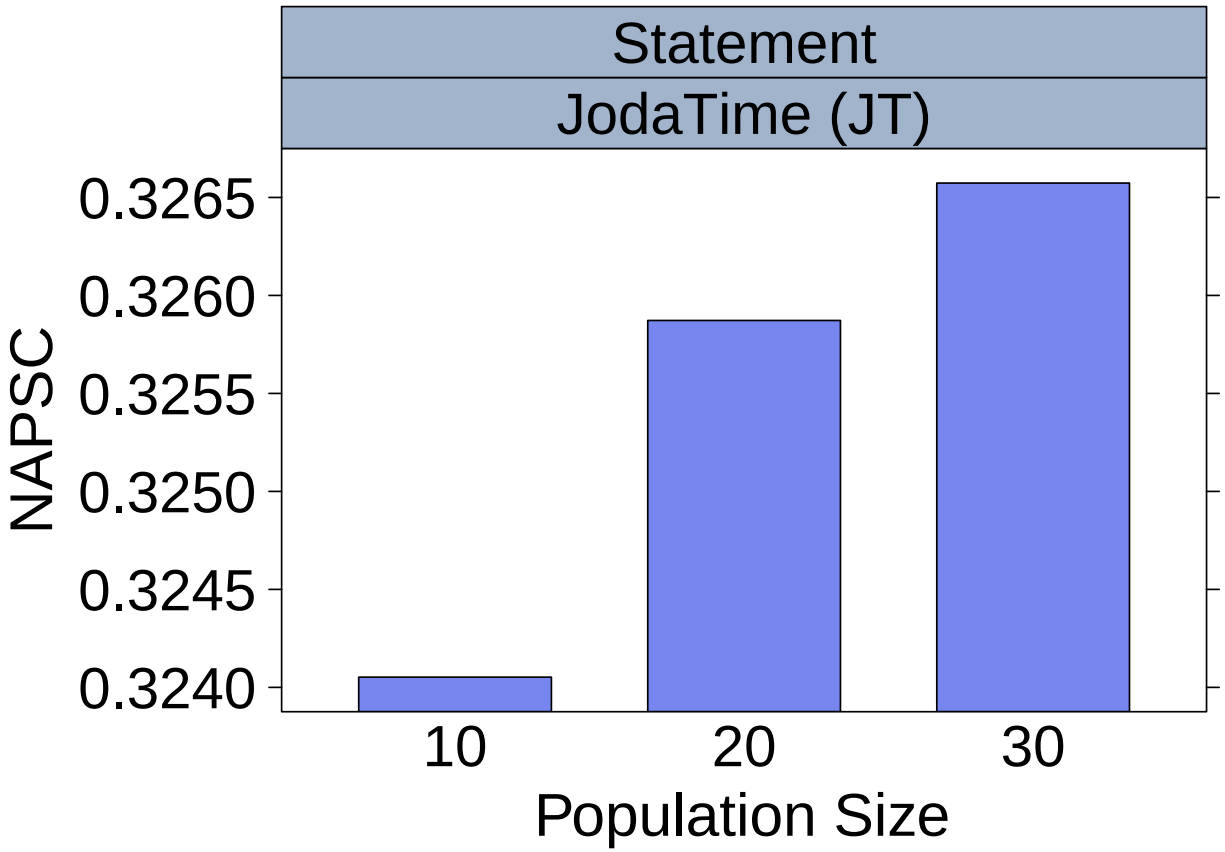
A small NAPSC increase may result in a large runtime increase

# Empirical Results: Random and Adaptive Random



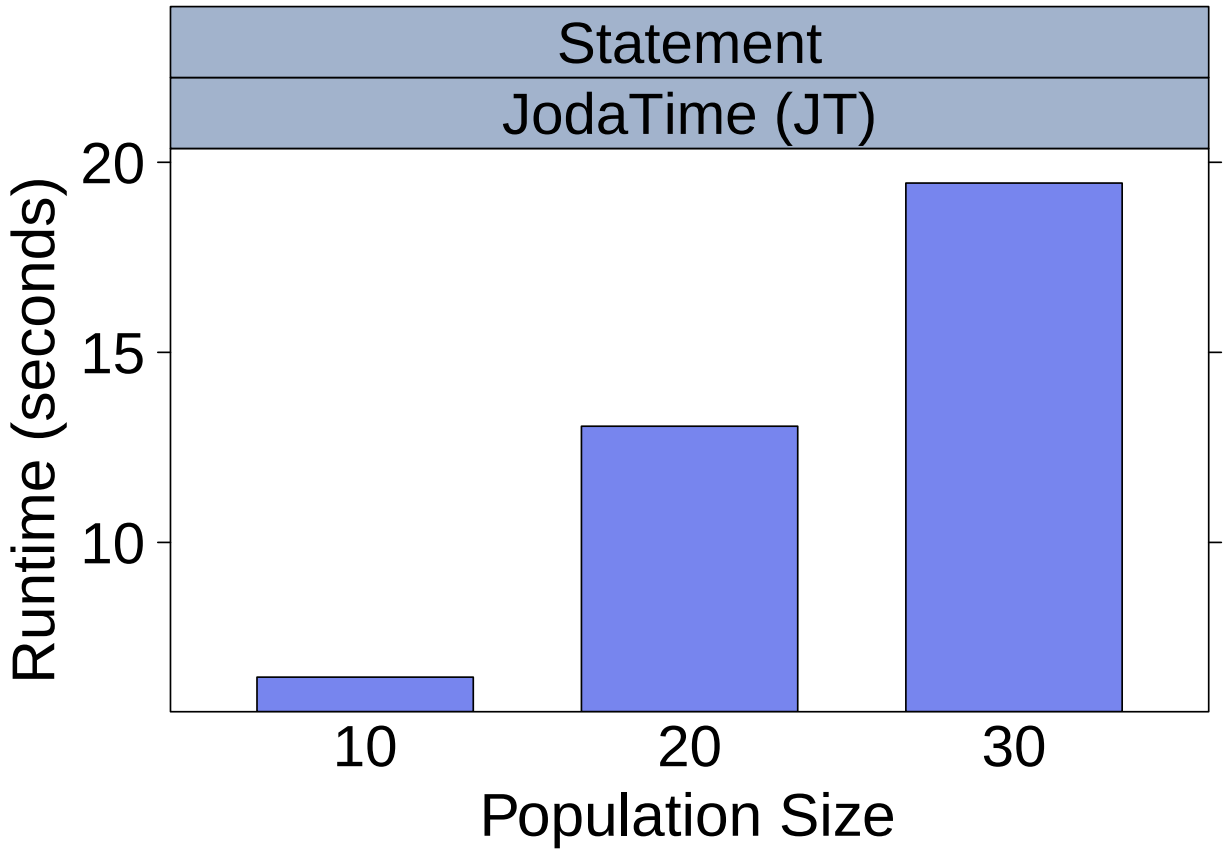

Original NAPSC Score: 0.2784

# Empirical Results: Random and Adaptive Random



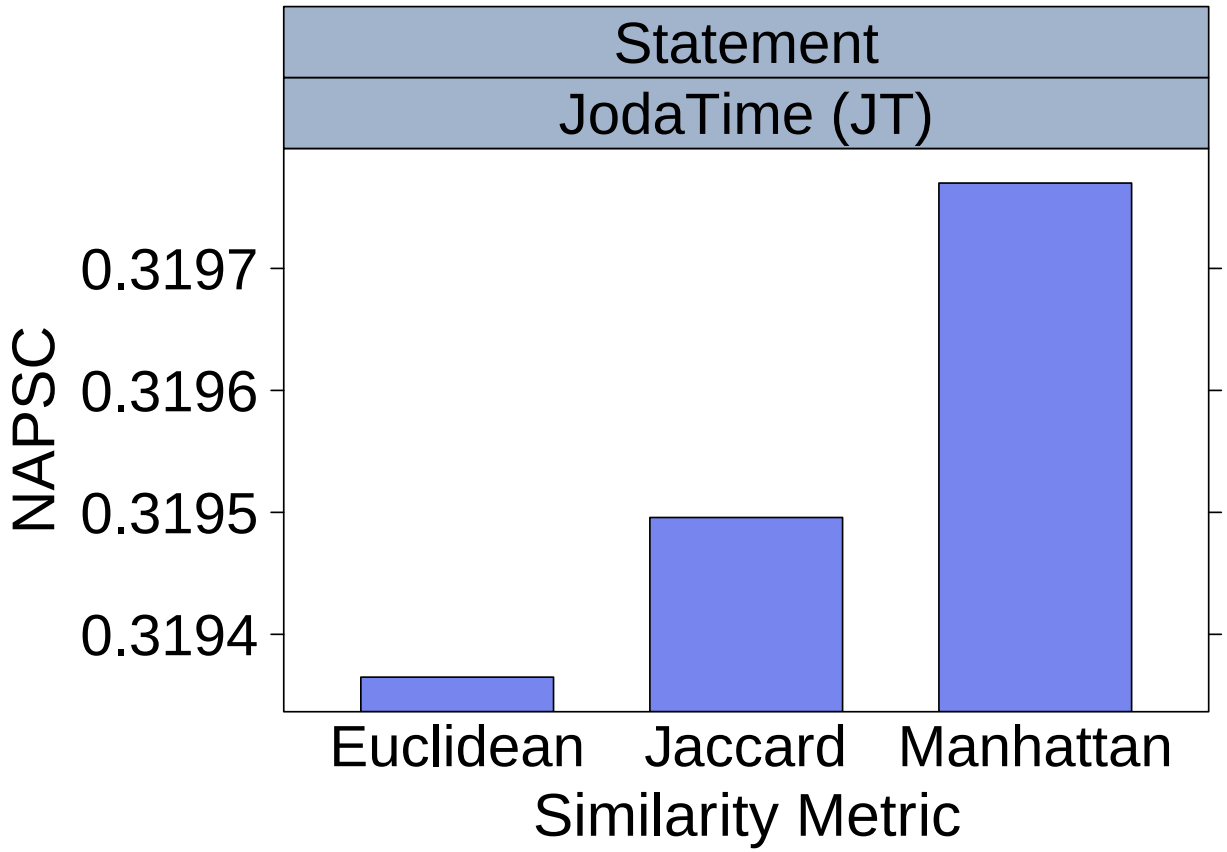

Negligible NAPSC increase as population size increases

# Empirical Results: Random and Adaptive Random



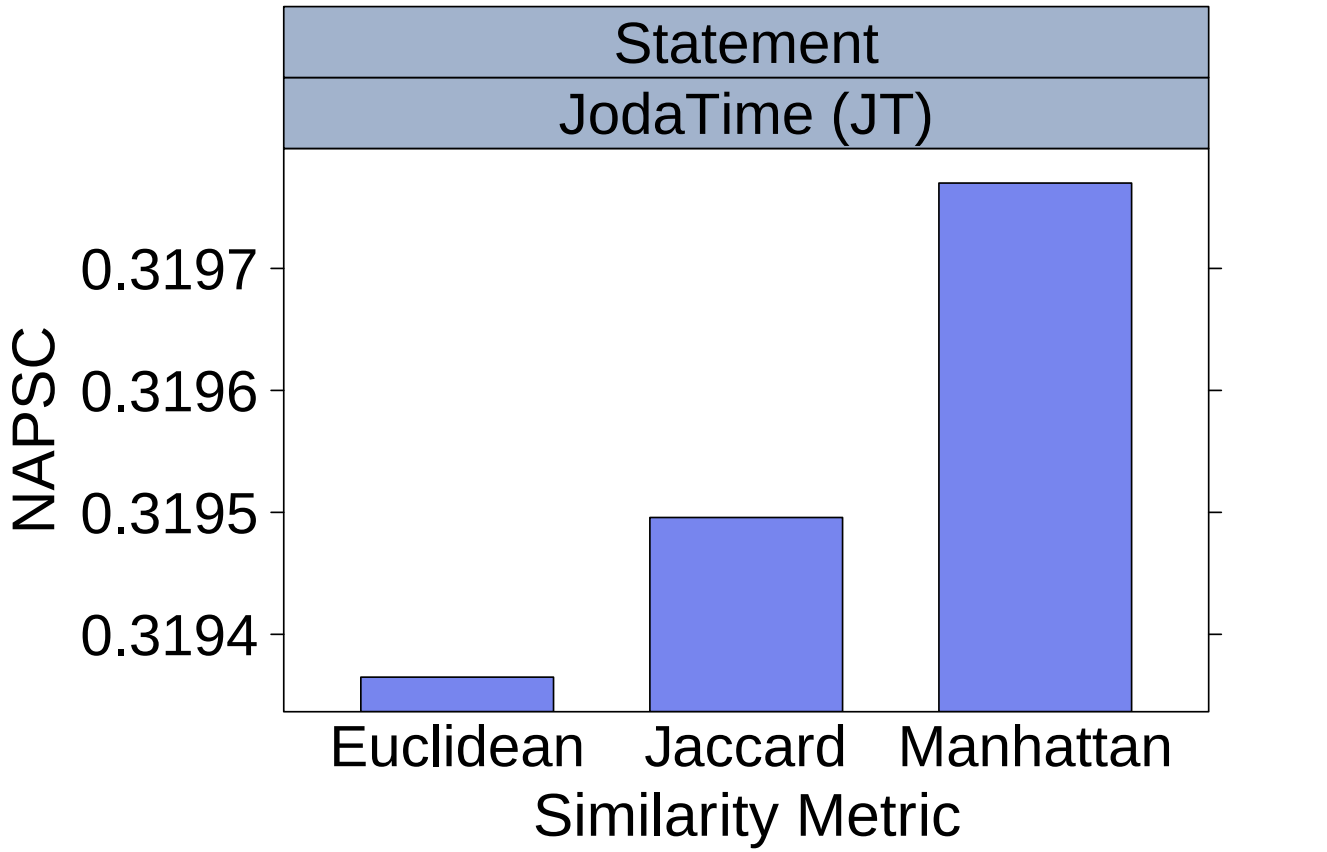

Increases in runtime are more marked

# Empirical Results: Random and Adaptive Random



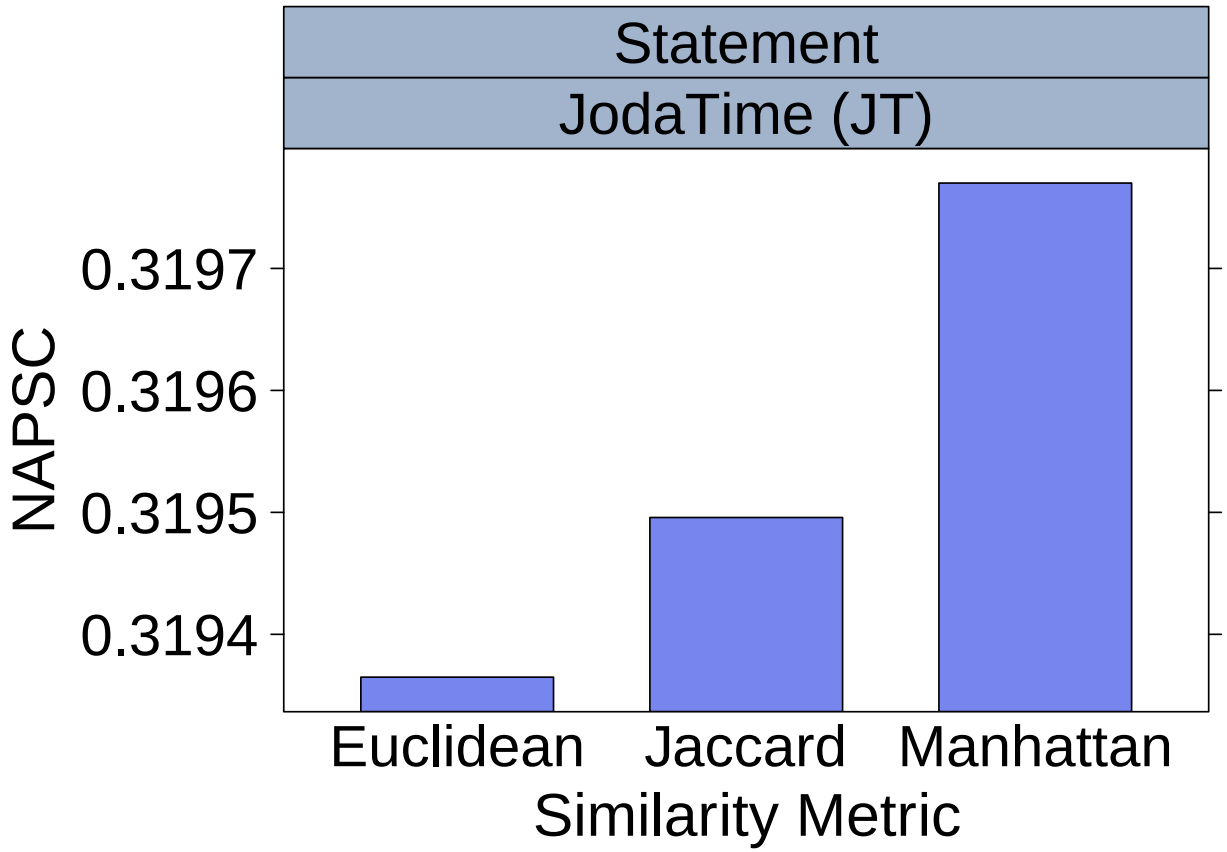

Original NAPSC Score: 0.2784

# Empirical Results: Random and Adaptive Random



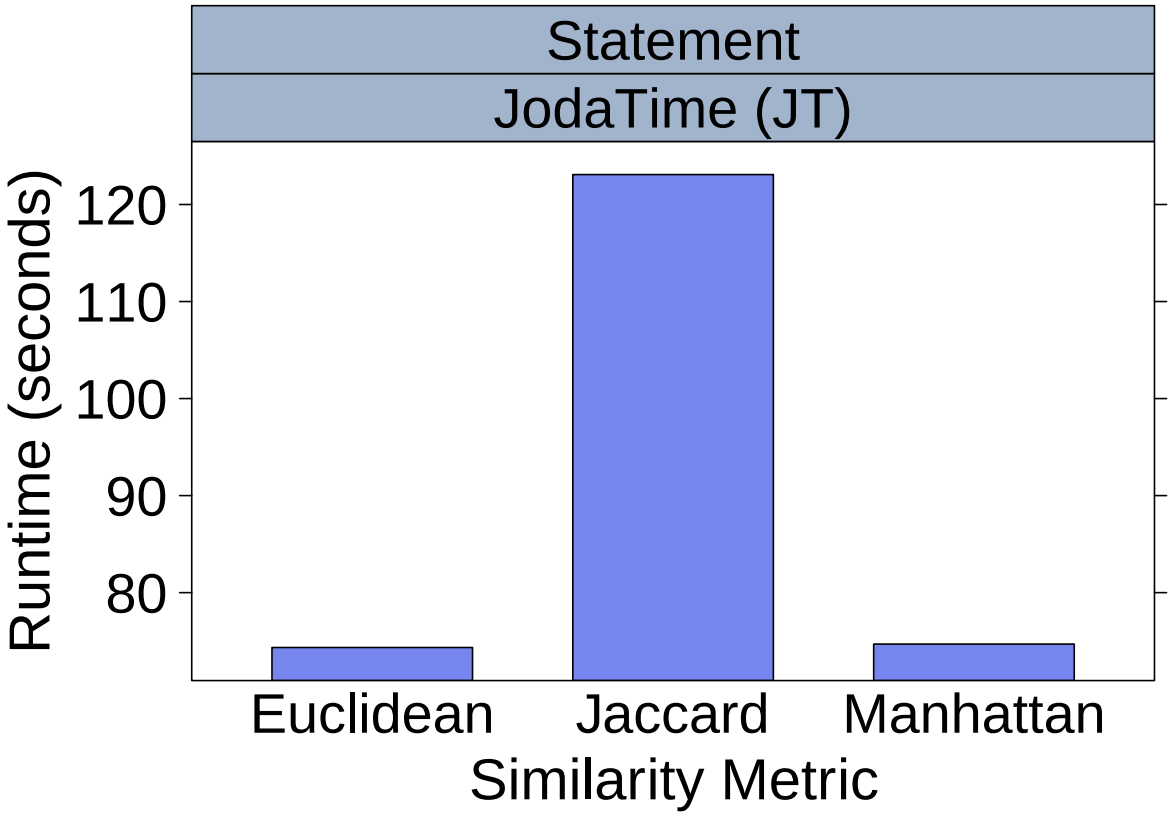

NAPSC changes little as similarity metric is varied

# Empirical Results: Random and Adaptive Random



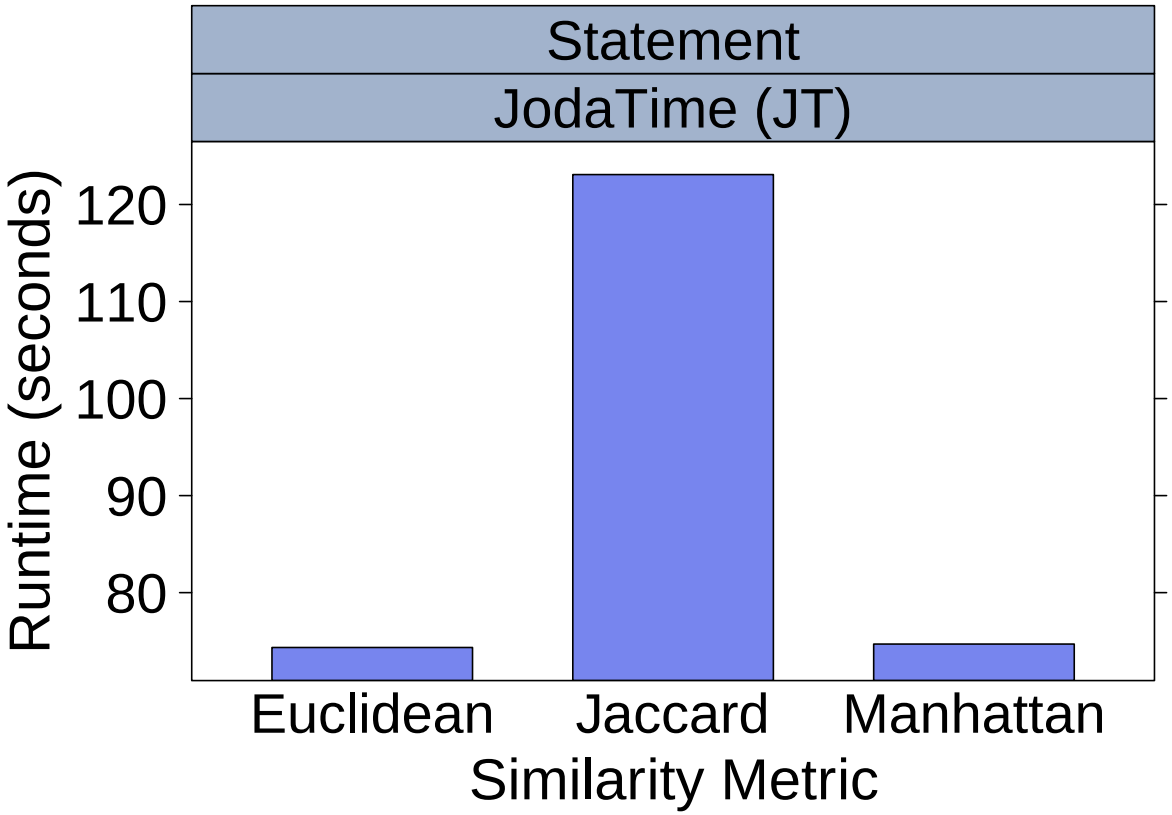

Scores are comparable to those produced by random (0.3240 - 0.3265)

# Empirical Results: Random and Adaptive Random




Adaptive random executes more slowly than random

# Empirical Results: Random and Adaptive Random




Choose random because it produces comparable NAPSC scores in less time

# The Future of Regression Testing

**Research**

- Reproducible research by releasing software tools and data
- Integrate many existing algorithms into a single framework
- Develop new forums for publishing important results
  - Software Quality Journal special issue
  - International Workshop on Regression Testing

**Practice**

- Encourage the use of coarse-grained information
- Try to apply existing tools to industrial programs
- Participate in community events; publish experience reports

# The Future of Regression Testing

**Research**

- Reproducible research by releasing software tools and data
- Integrate many existing algorithms into a single framework
- Develop new forums for publishing important results
  - Software Quality Journal special issue
  - International Workshop on Regression Testing

**Practice**

- Encourage the use of coarse-grained information
- Try to apply existing tools to industrial programs
- Participate in community events; publish experience reports

# Conclusions and Future Work

## Concluding Remarks

- Comprehensive framework for regression testing
- Interesting empirical results demonstrate trade-offs
- Free/open source tools are available for download
  - `http://proteja.googlecode.com`
  - `http://modificare.googlecode.com`

## Future Work

- Add new algorithms for regression testing
- Conduct experiments with more case study applications
- Further develop statistically meaningful empirical results

# Conclusions and Future Work

## Concluding Remarks

- Comprehensive framework for regression testing
- Interesting empirical results demonstrate trade-offs
- Free/open source tools are available for download
  - `http://proteja.googlecode.com`
  - `http://modificare.googlecode.com`

## Future Work

- Add new algorithms for regression testing
- Conduct experiments with more case study applications
- Further develop statistically meaningful empirical results

# Software Quality Improvement through Repeated Test Execution: An Exploration of the Present and Future of Regression Testing

Gregory M. Kapfhammer

http://www.cs.allegheny.edu/~gkapfham/

Thank you for your attention!
Contact me with questions and/or comments!


ALLEGHENY COLLEGE