Regression Testing: Theoretical Underpinnings, Practical Techniques, and Empirical Insights

Gregory M. Kapfhammer* and Jonathan Miller Kauffman†

Department of Computer Science
Allegheny College
*http://www.cs.allegheny.edu/~gkapfham/
†https://sites.google.com/a/allegheny.edu/jonathan-kauffman/

15th International Conference on Software Engineering and Applications
December 14 - 16, 2011

ALLEGHENY COLLEGE

Accessing the Presentation

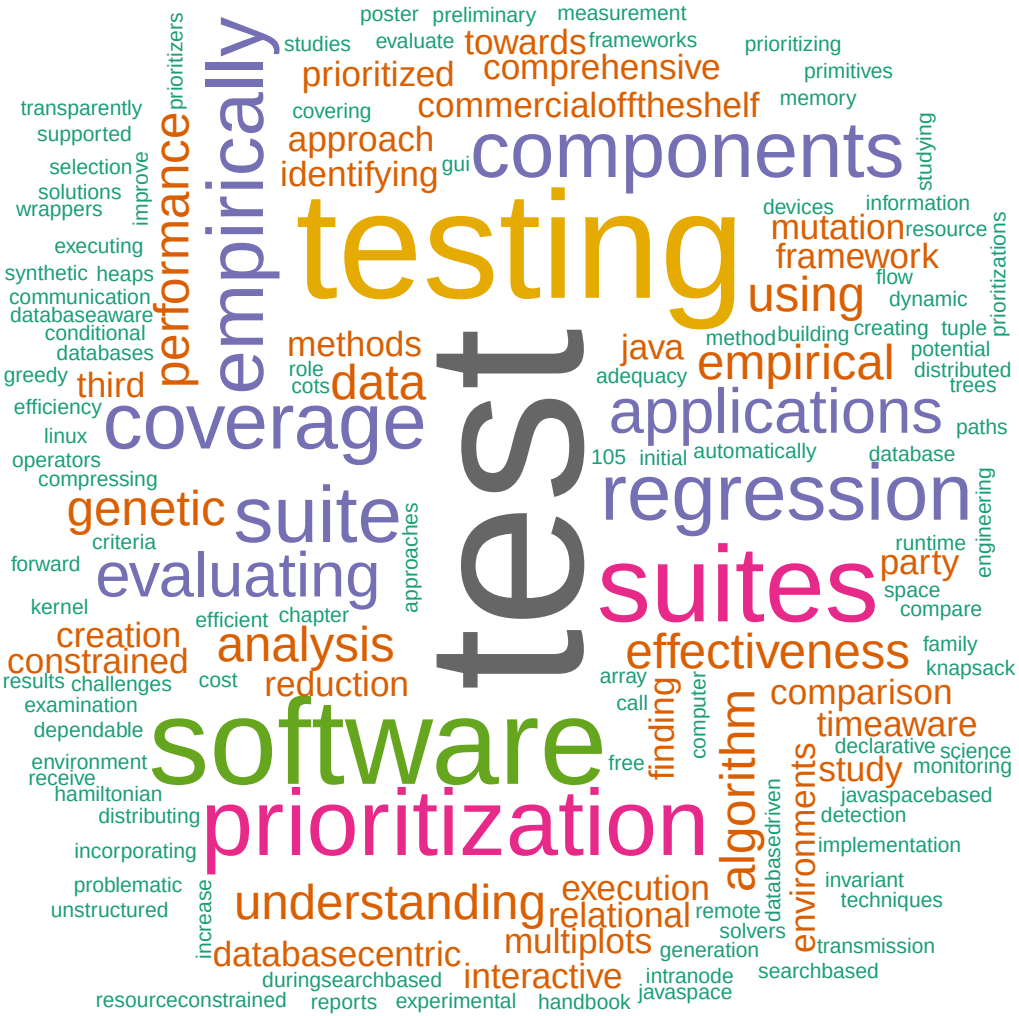
Scan this QR Code with your smartphone!

... or, visit this Web site:

`http://www.cs.allegheny.edu/~gkapfham/sea2011.pdf`

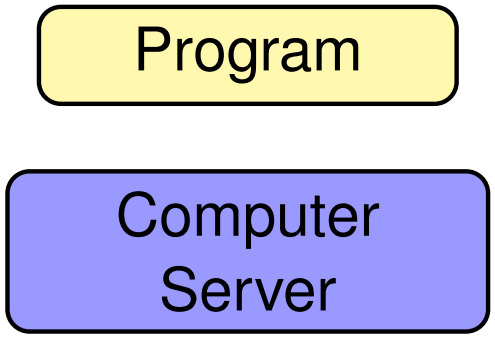
... or, ask us for a USB drive!

Presenter Introduction: Gregory M. Kapfhammer

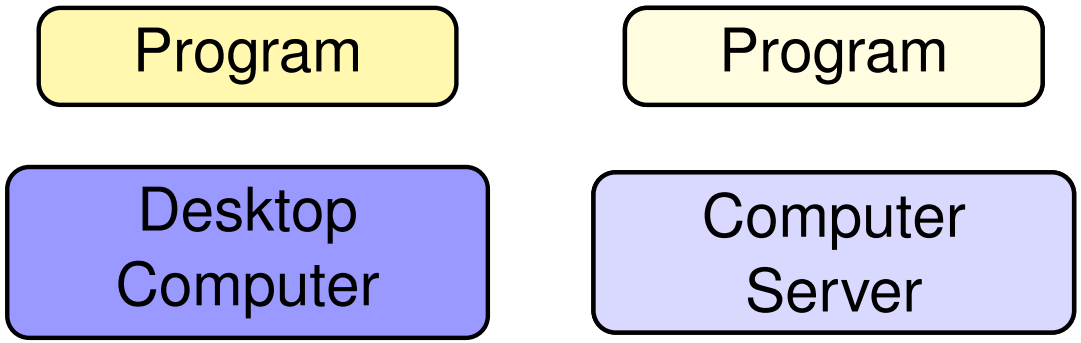


Presenter Introduction: Jonathan Miller Kauffman

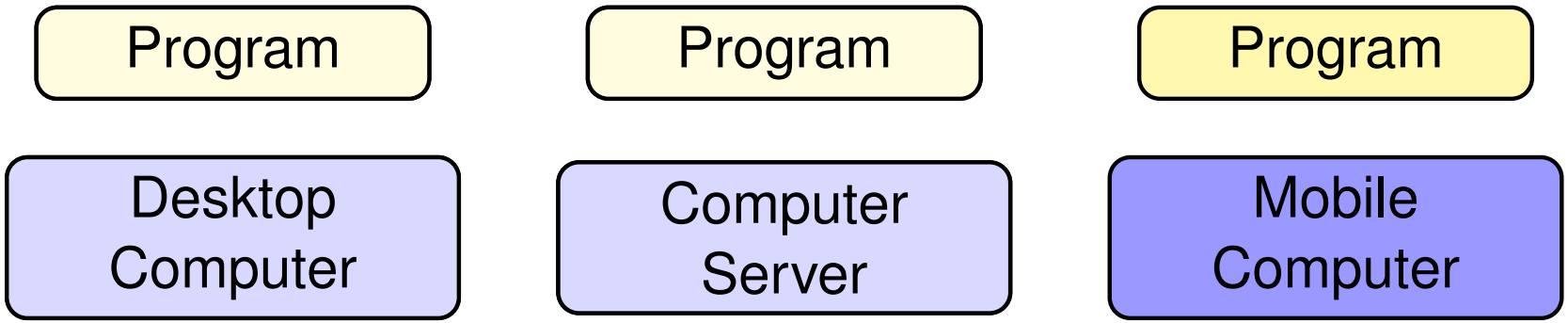
Software is Everywhere



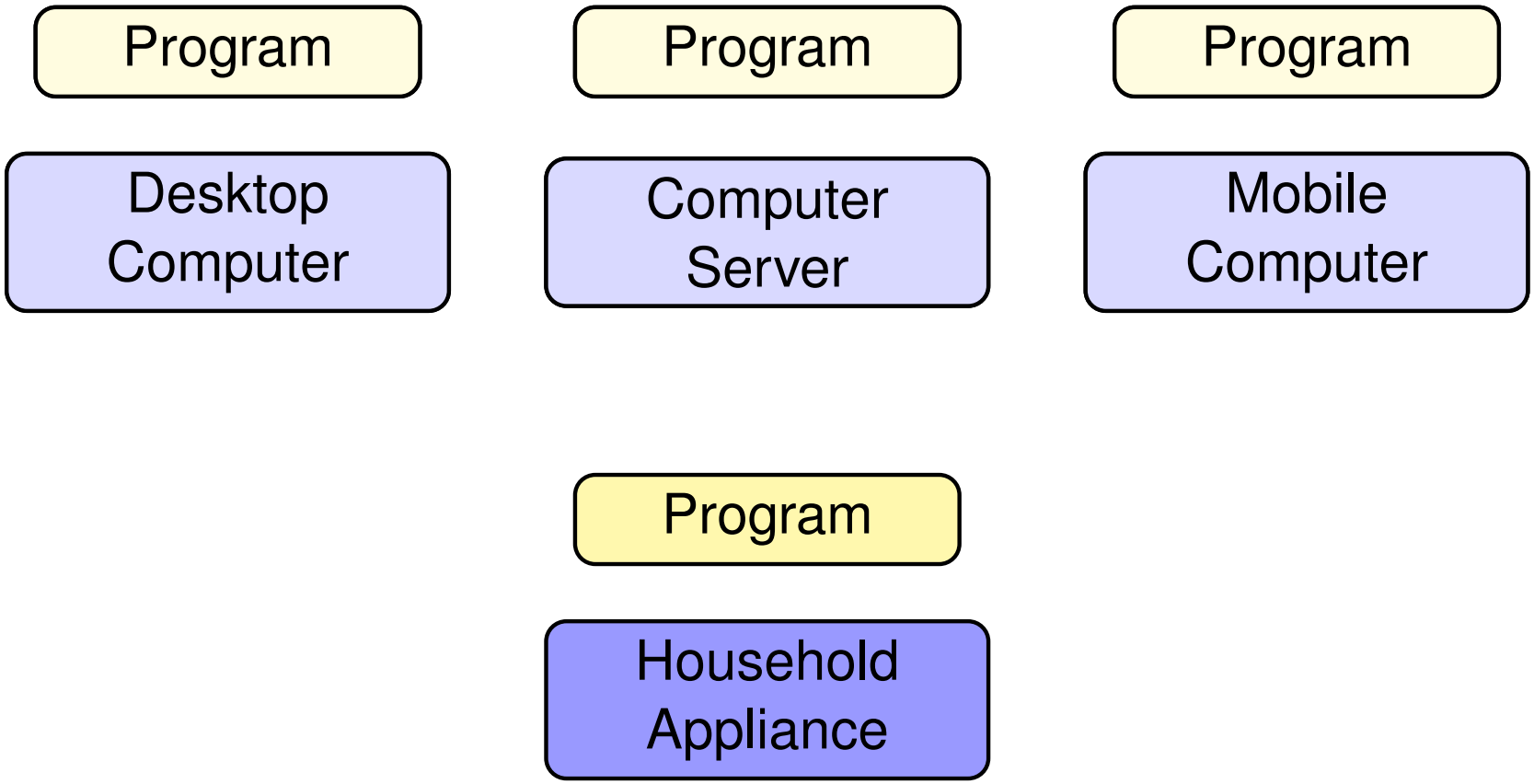
Software is Everywhere



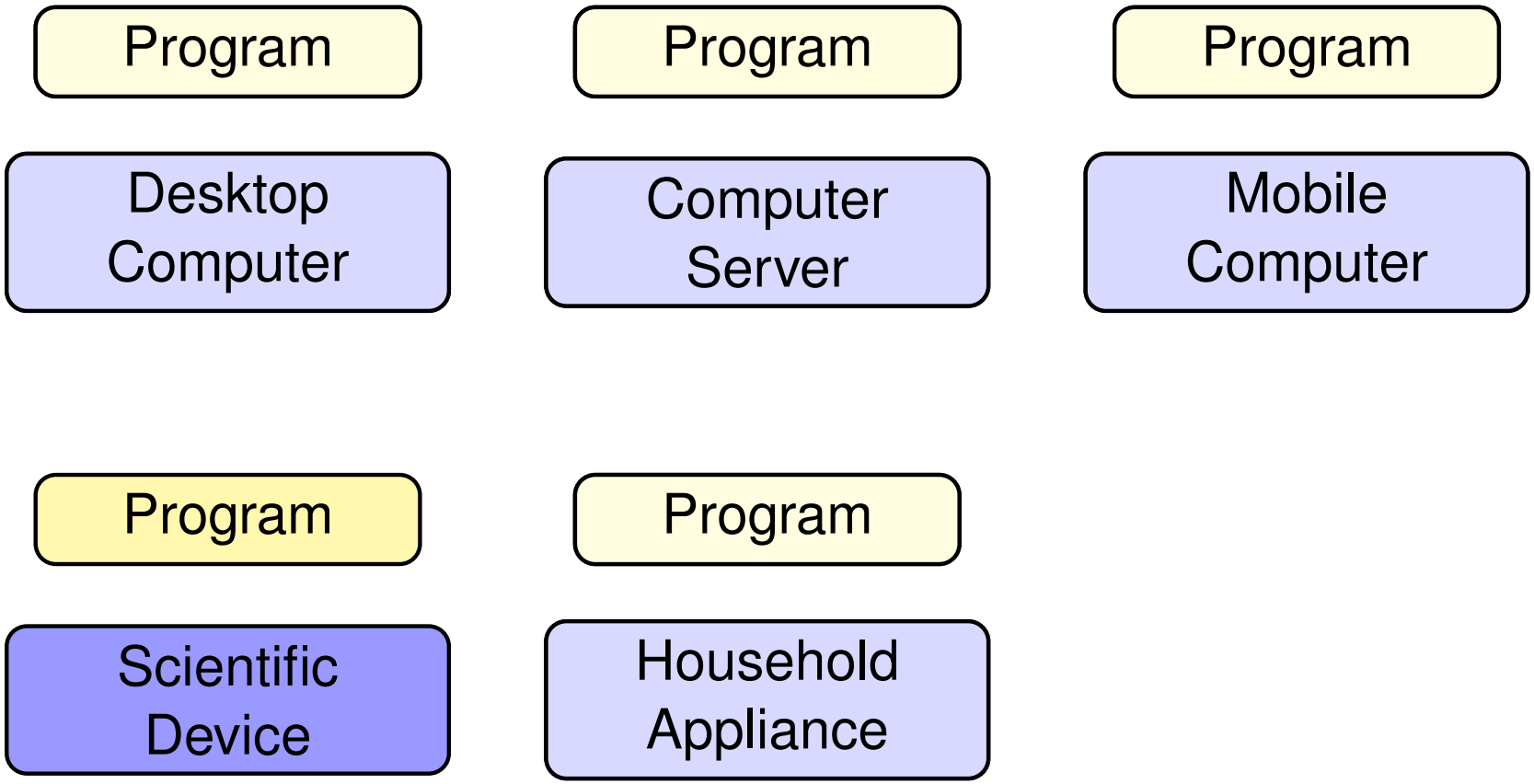
Software is Everywhere



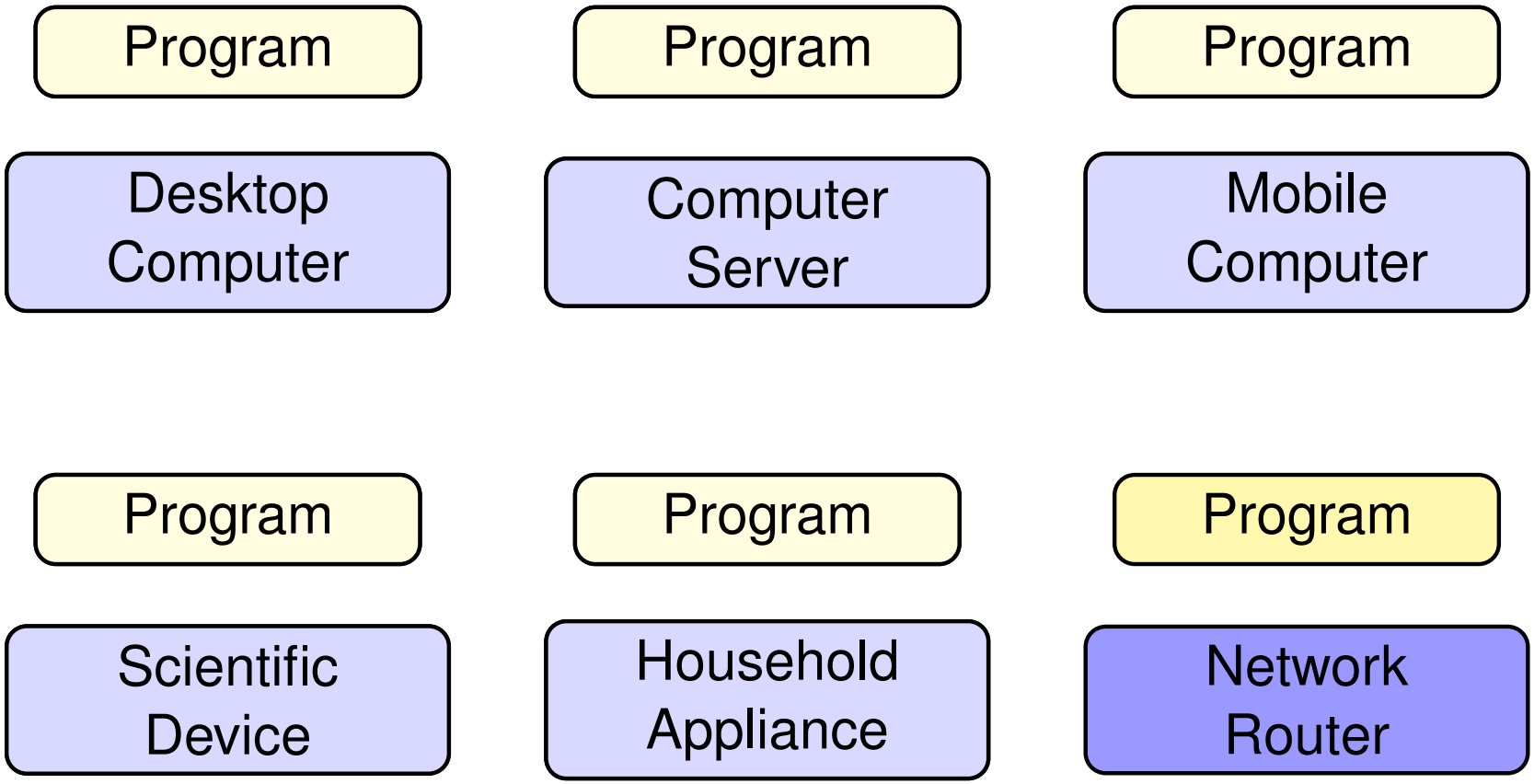
Software is Everywhere



Software is Everywhere



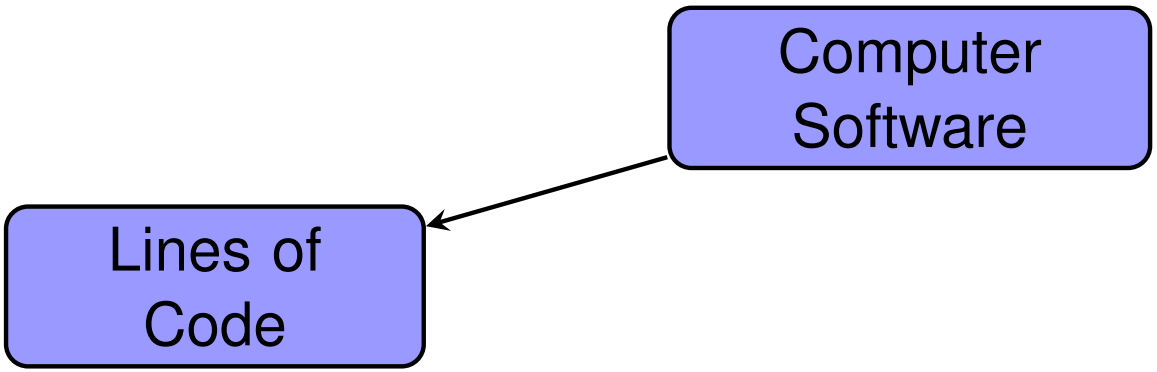
Software is Everywhere



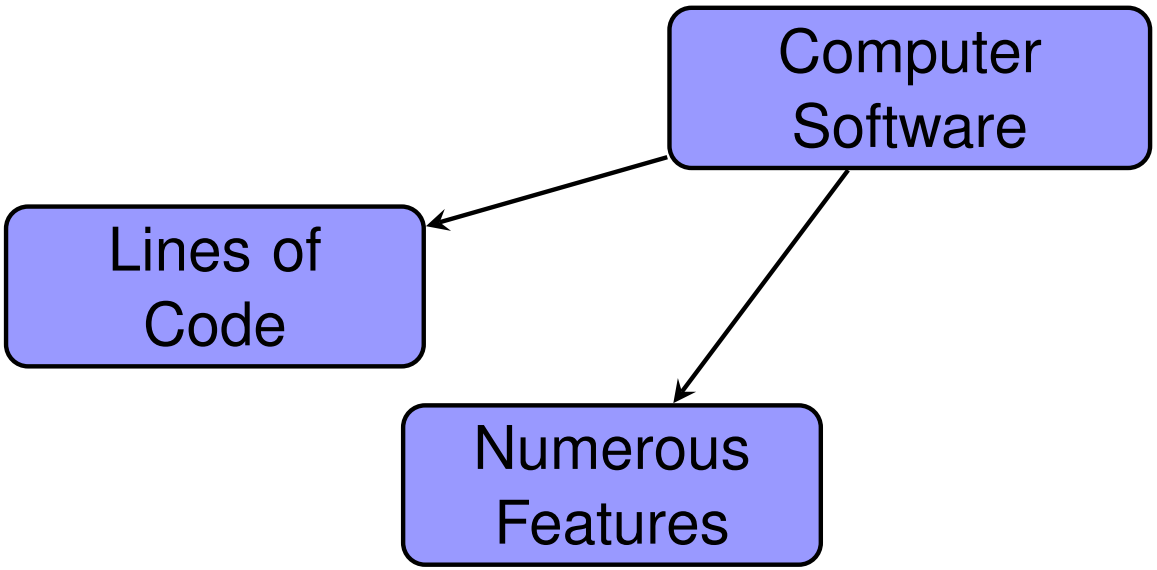
Software is Complex

Computer Software

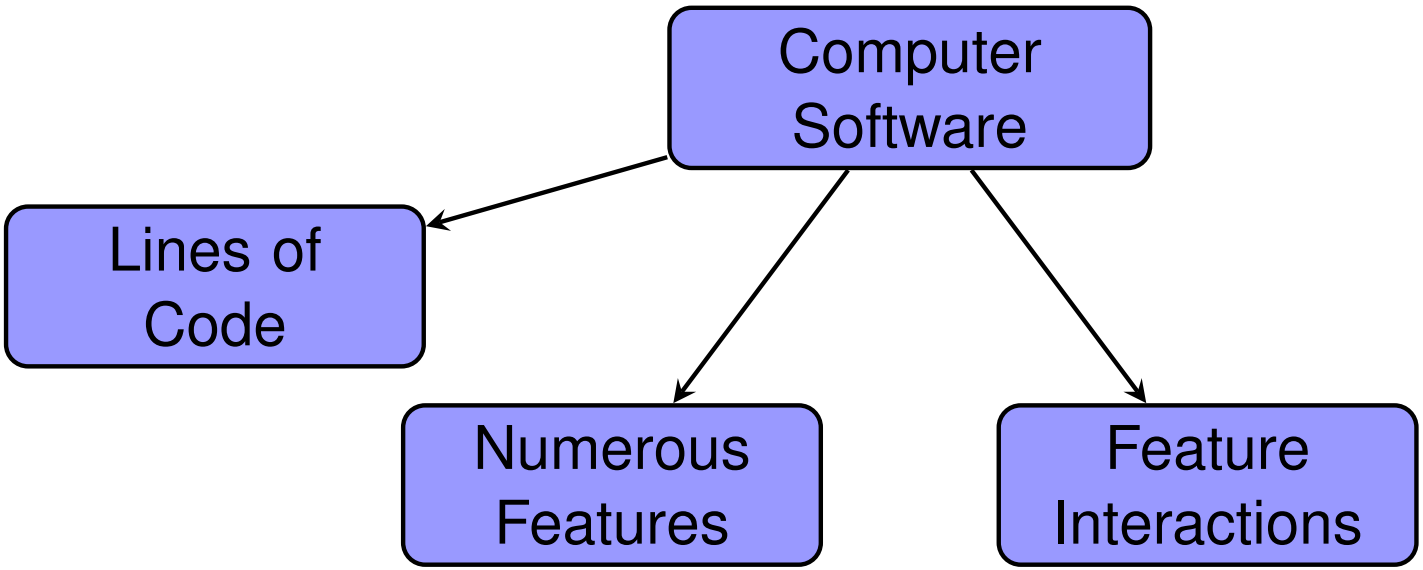
Software is Complex



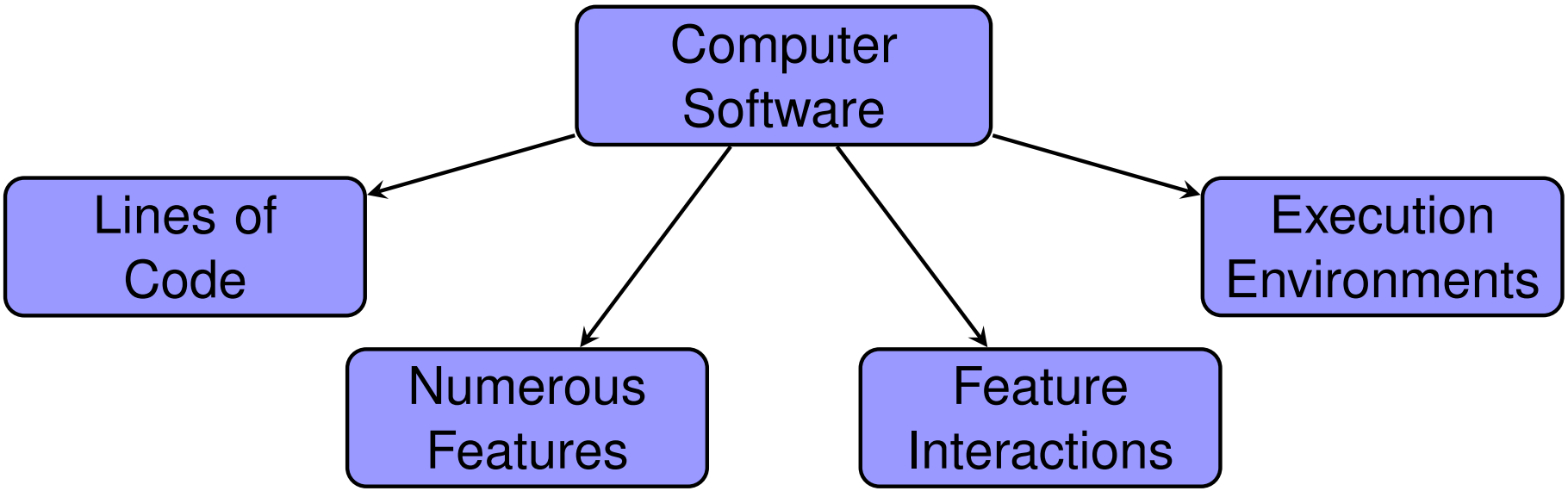
Software is Complex



Software is Complex

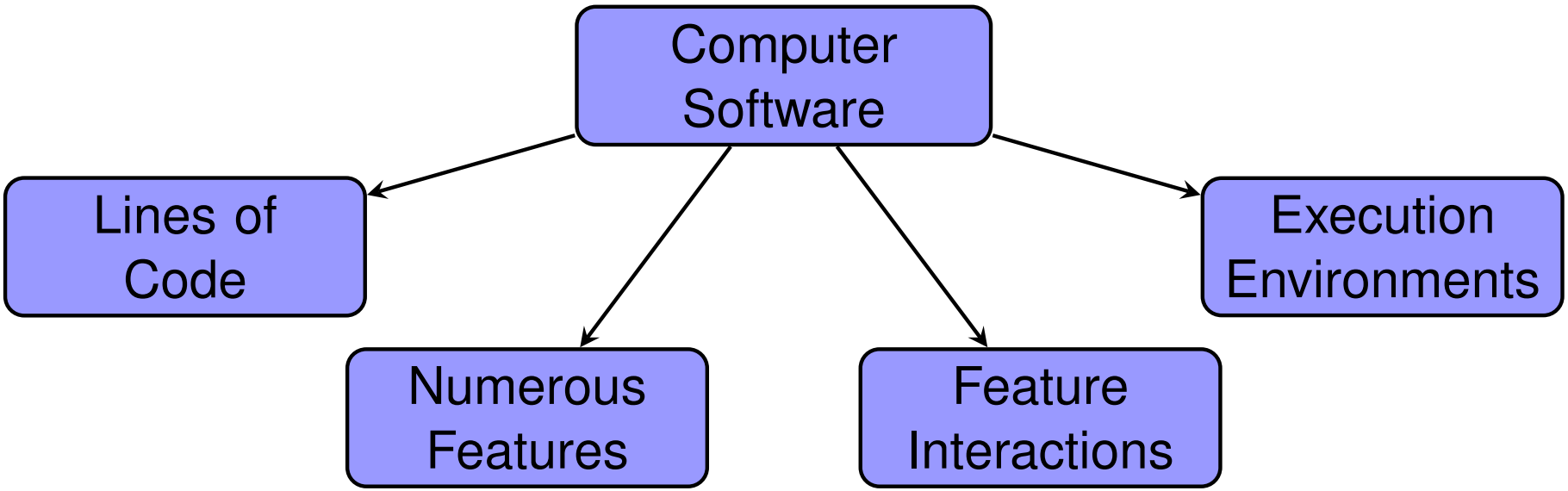


Software is Complex

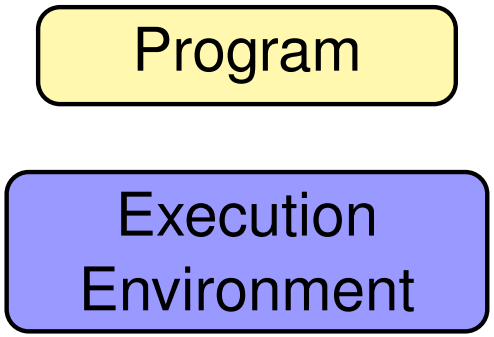


Software is Complex

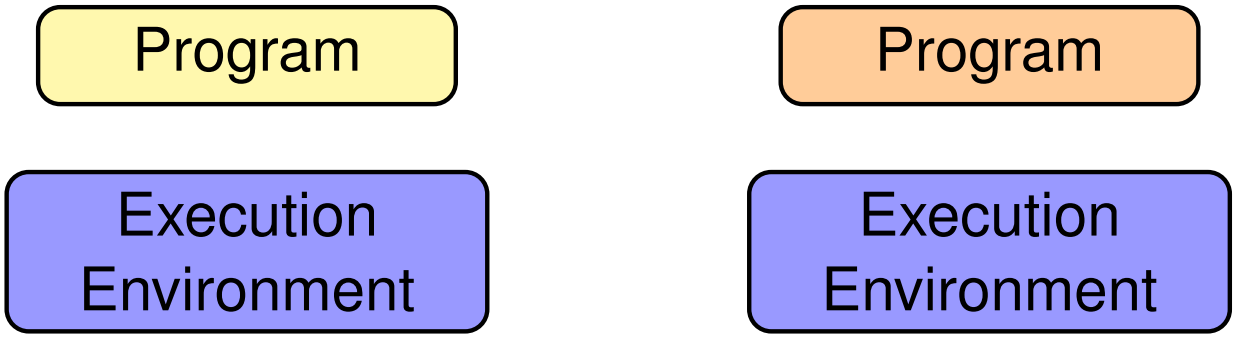
Software entities are more complex for their size than perhaps any other human construct - Frederick P. Brooks, Jr.



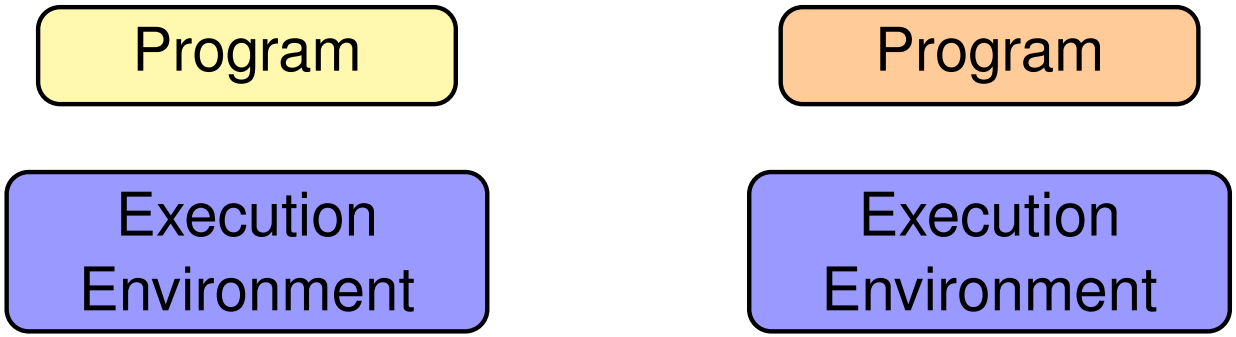
Software is Evolving



Software is Evolving

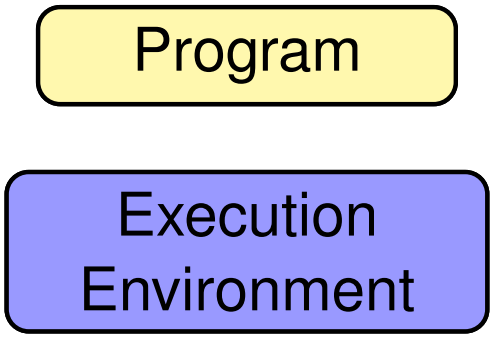


Software is Evolving

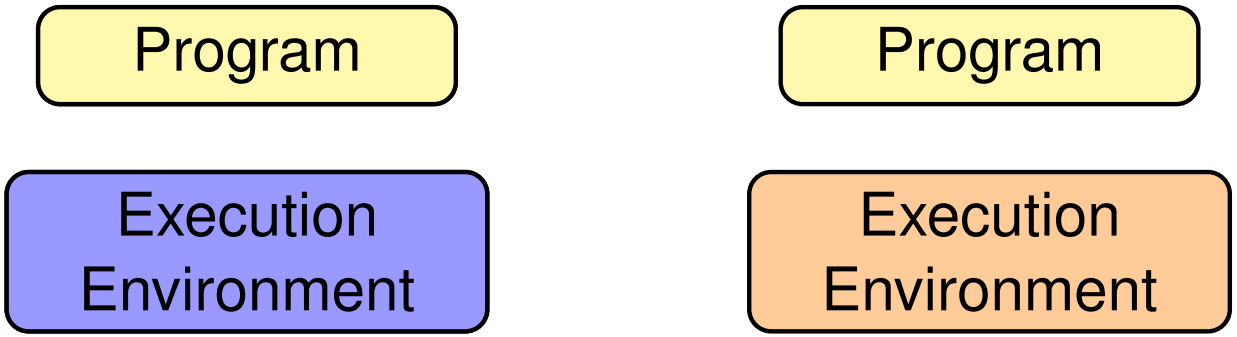


Program Changed because of the addition of a new feature or the correction of a defect

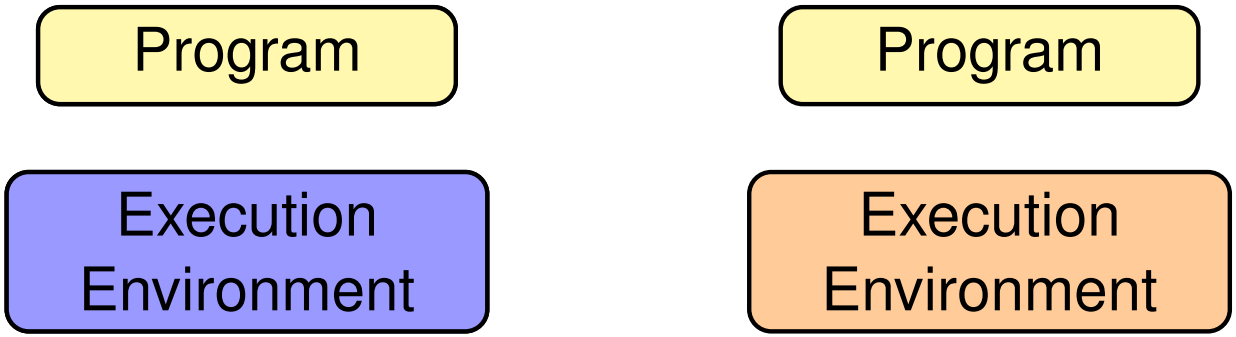
Software is Evolving



Software is Evolving



Software is Evolving

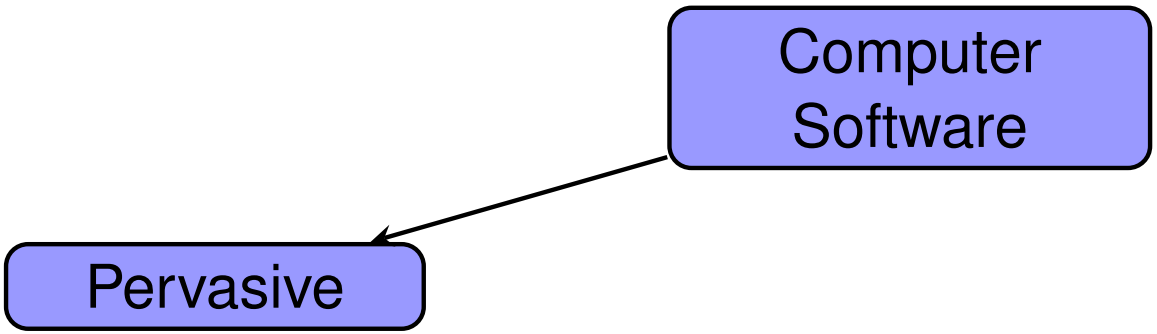


Execution Environment Changed due to an upgrade in a kernel, device driver, or virtual machine

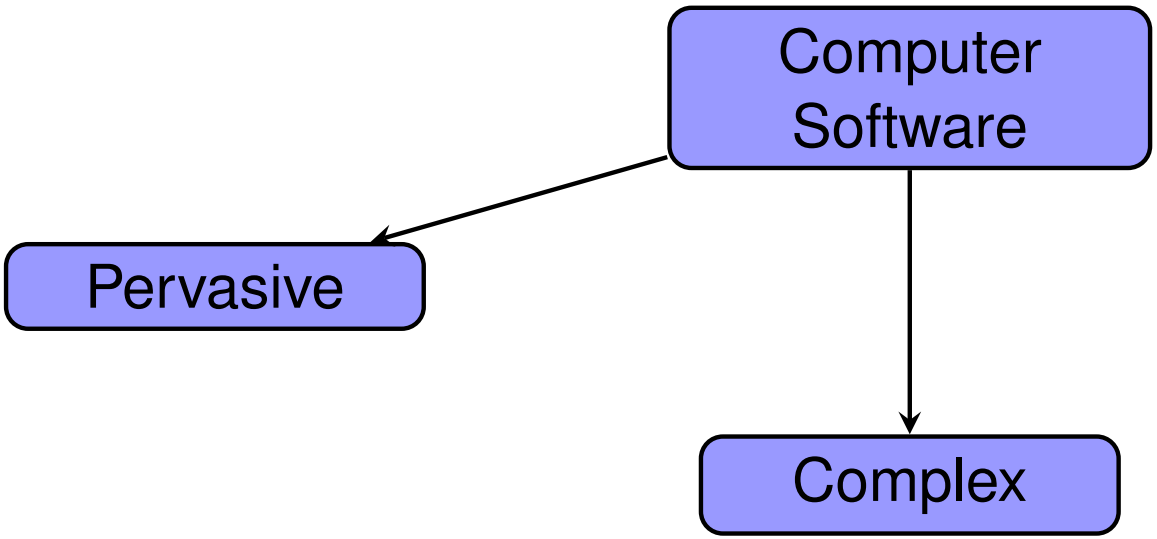
Regression Testing to the Rescue

Computer Software

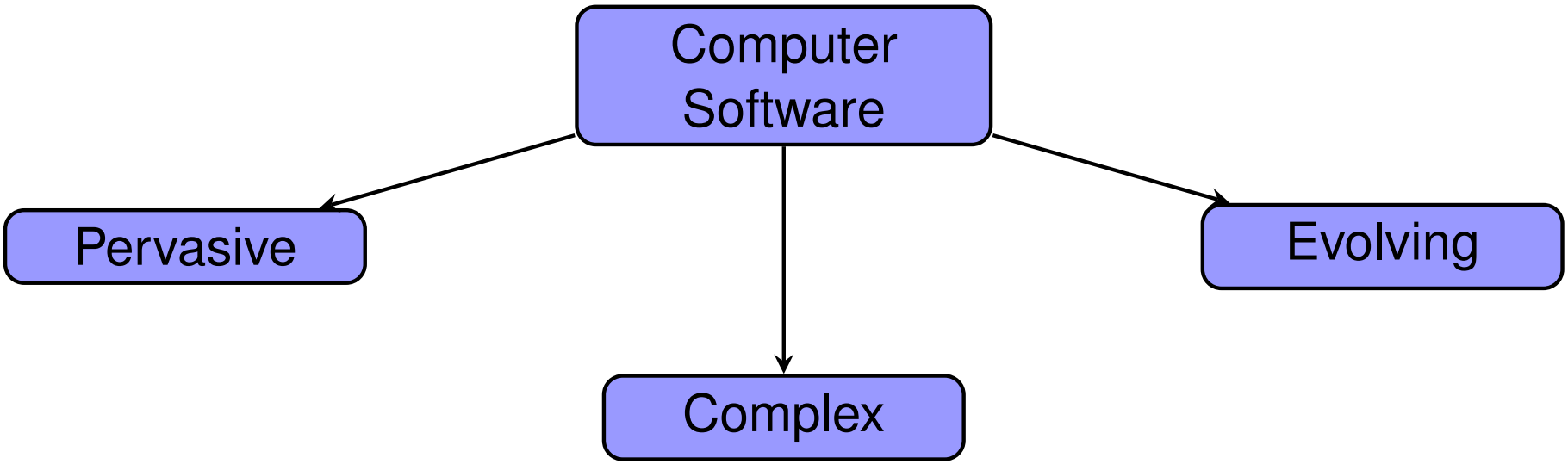
Regression Testing to the Rescue



Regression Testing to the Rescue

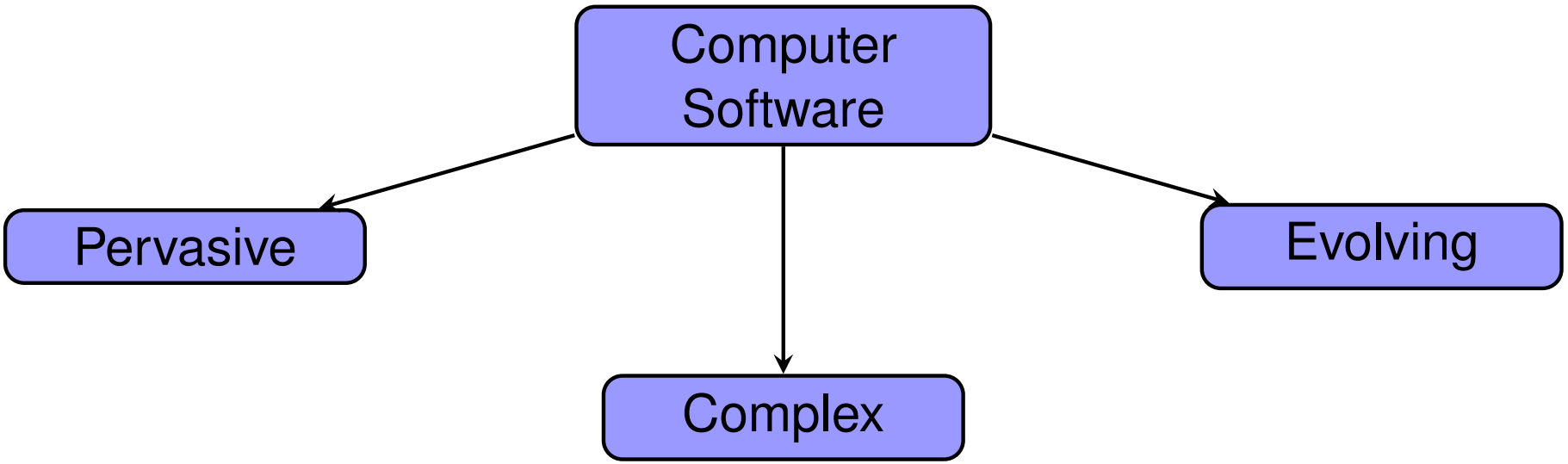


Regression Testing to the Rescue

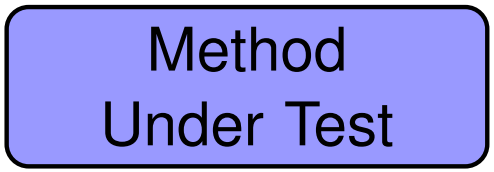


Regression Testing to the Rescue

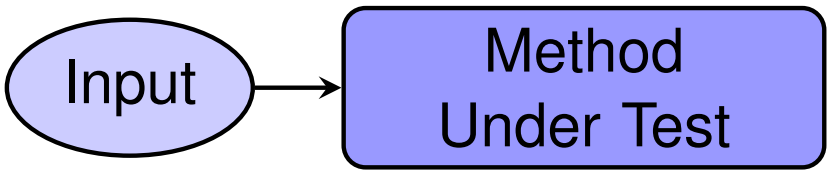
Regression Testing supports the efficient construction of pervasive software that is complex and rapidly evolving



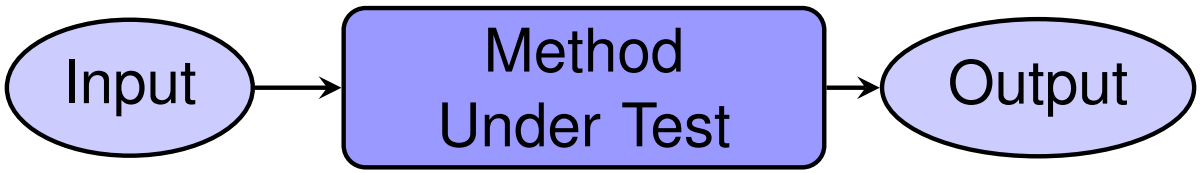
What is a Test Case?



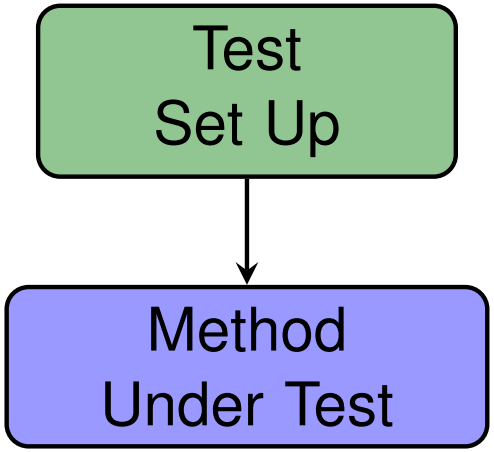
What is a Test Case?



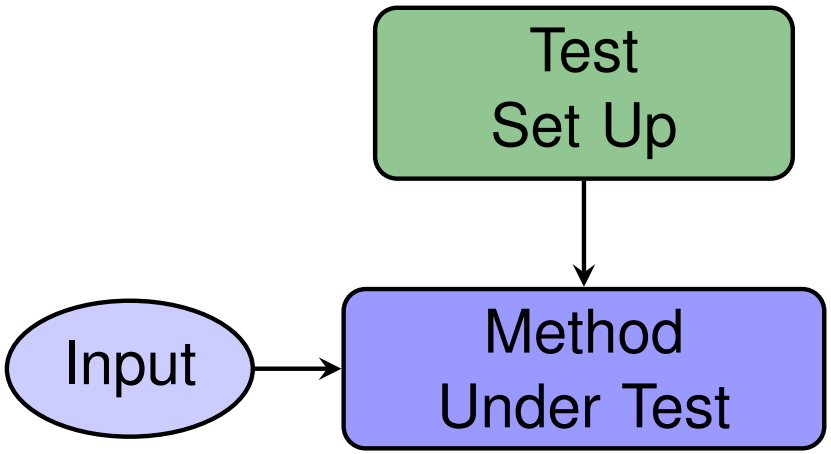
What is a Test Case?



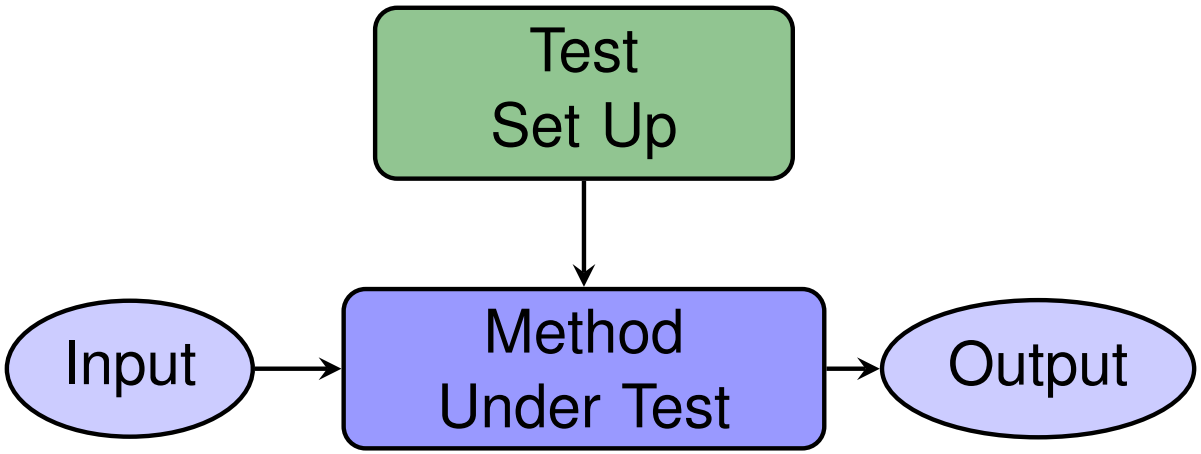
What is a Test Case?



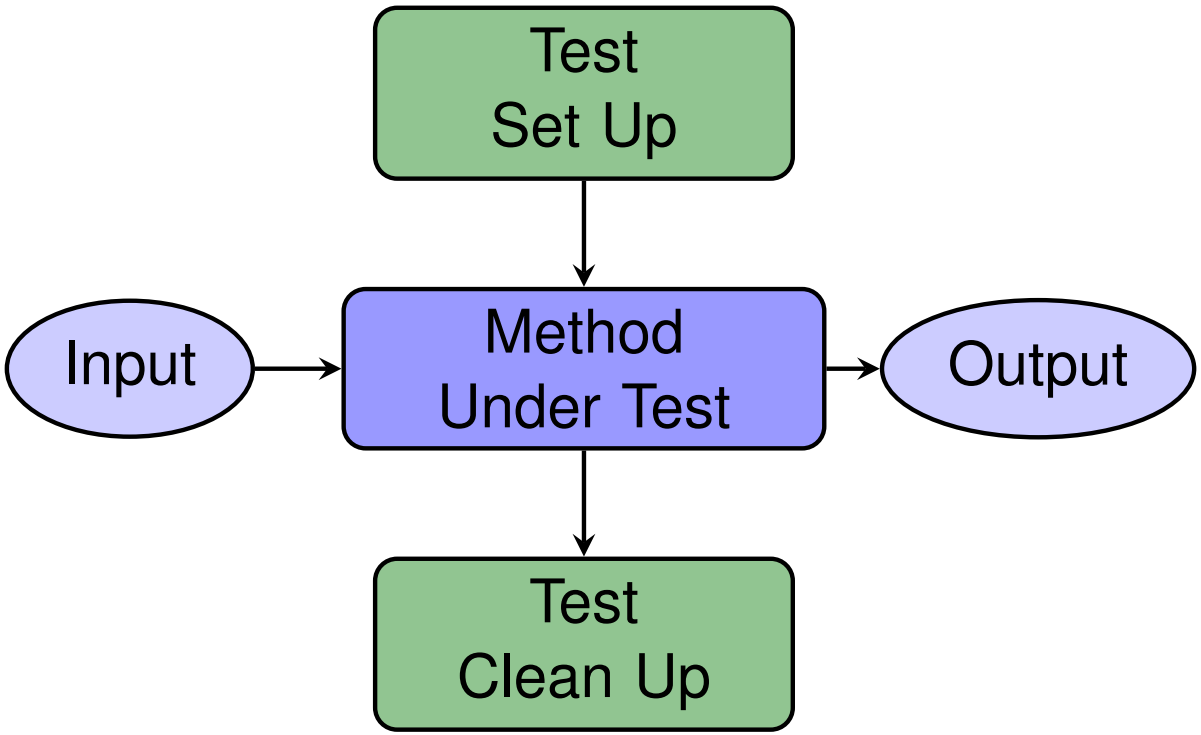
What is a Test Case?



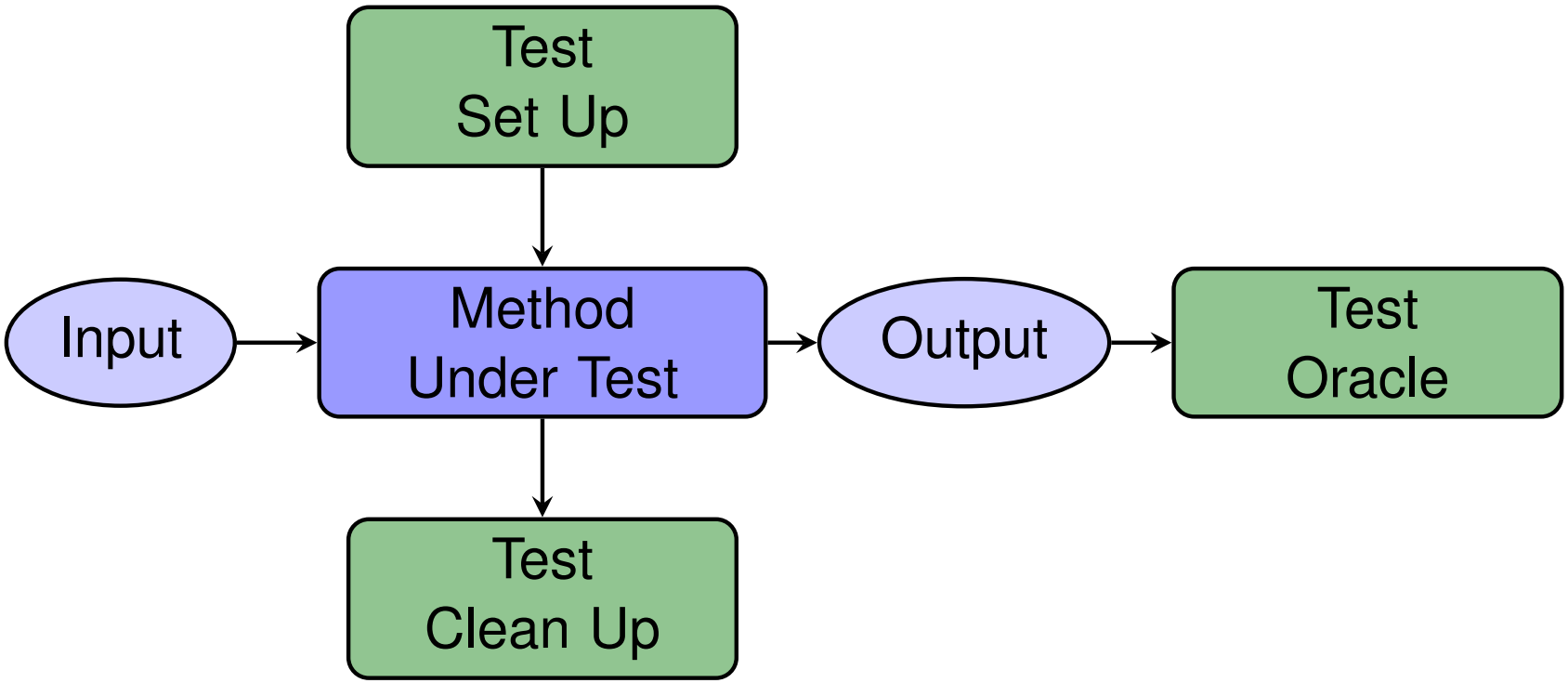
What is a Test Case?



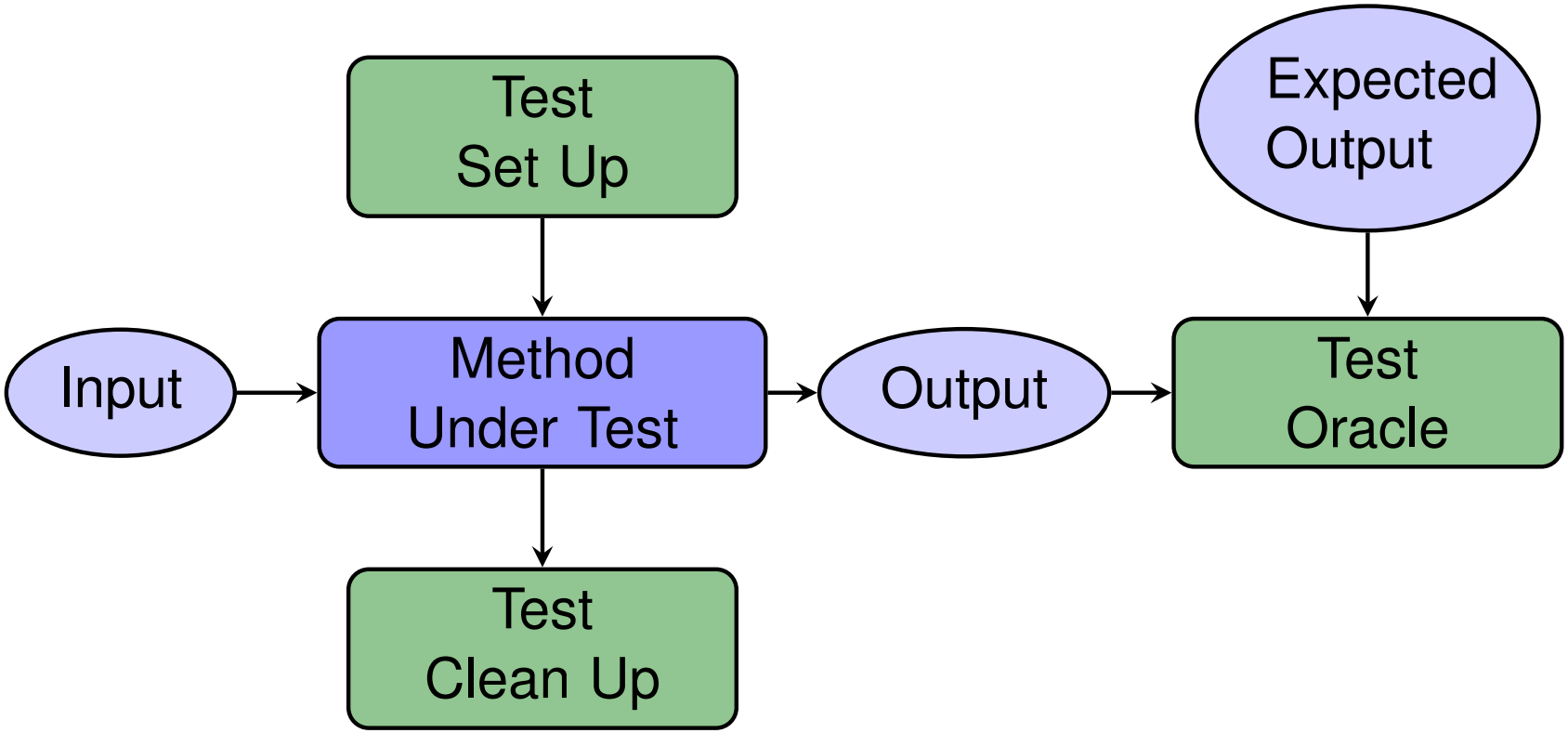
What is a Test Case?



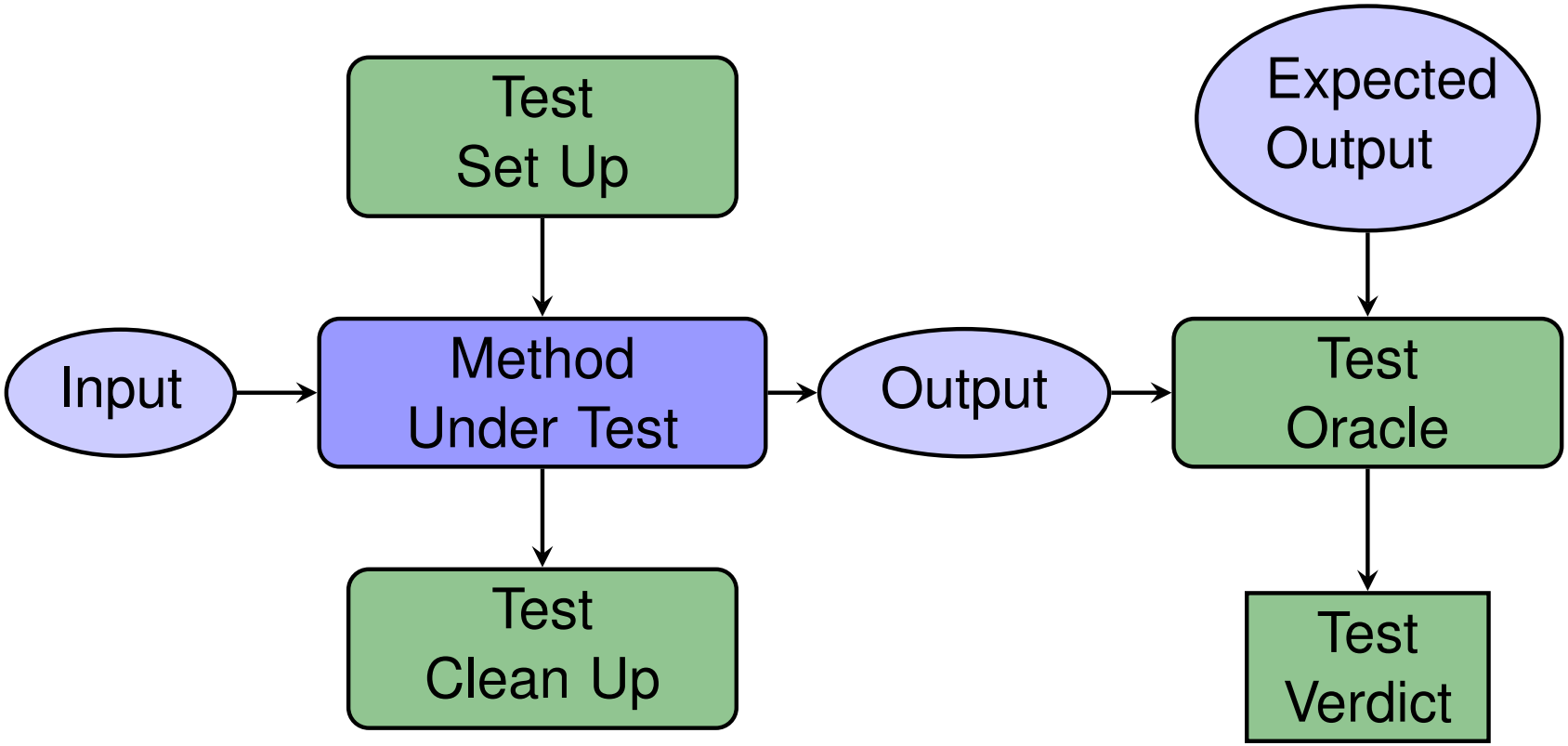
What is a Test Case?



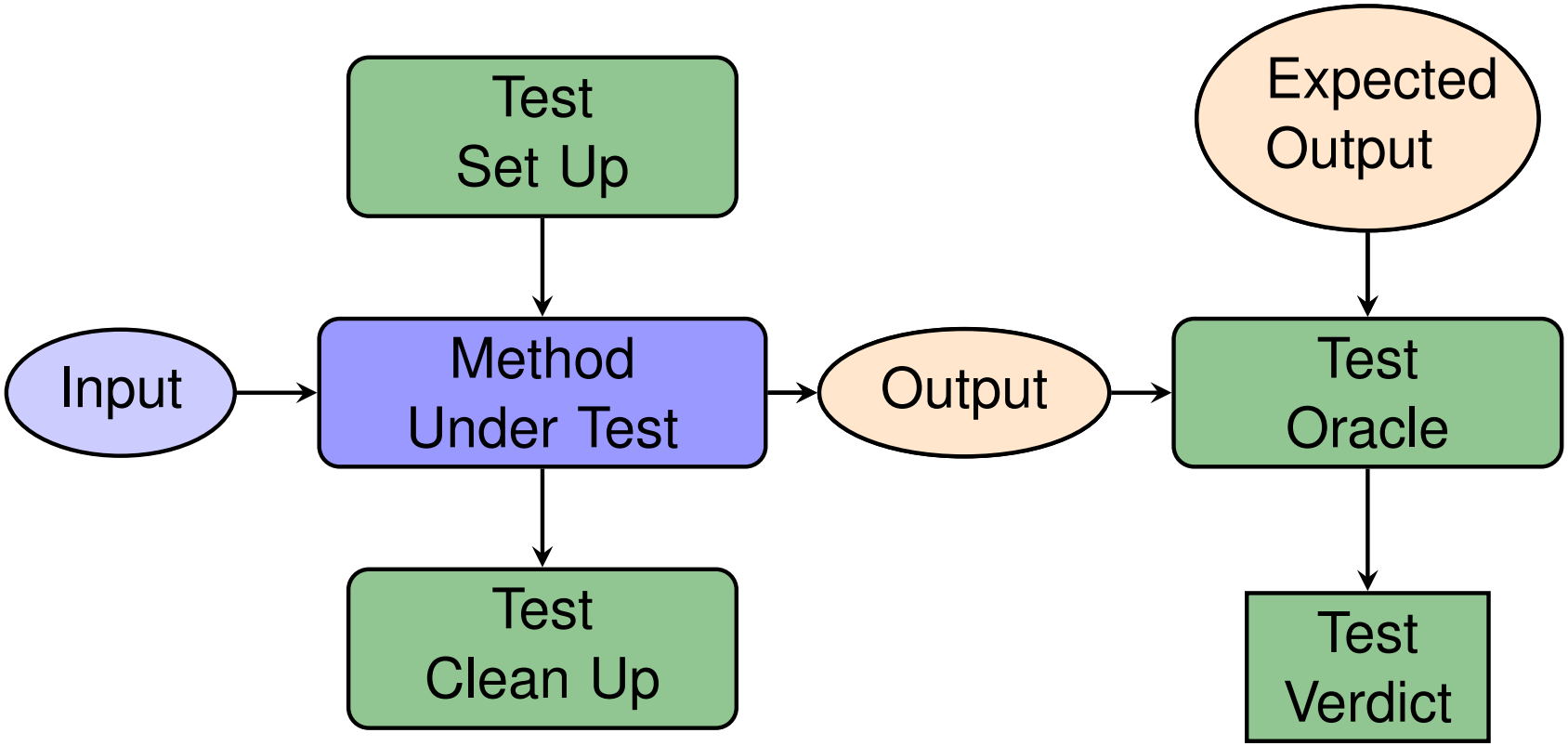
What is a Test Case?



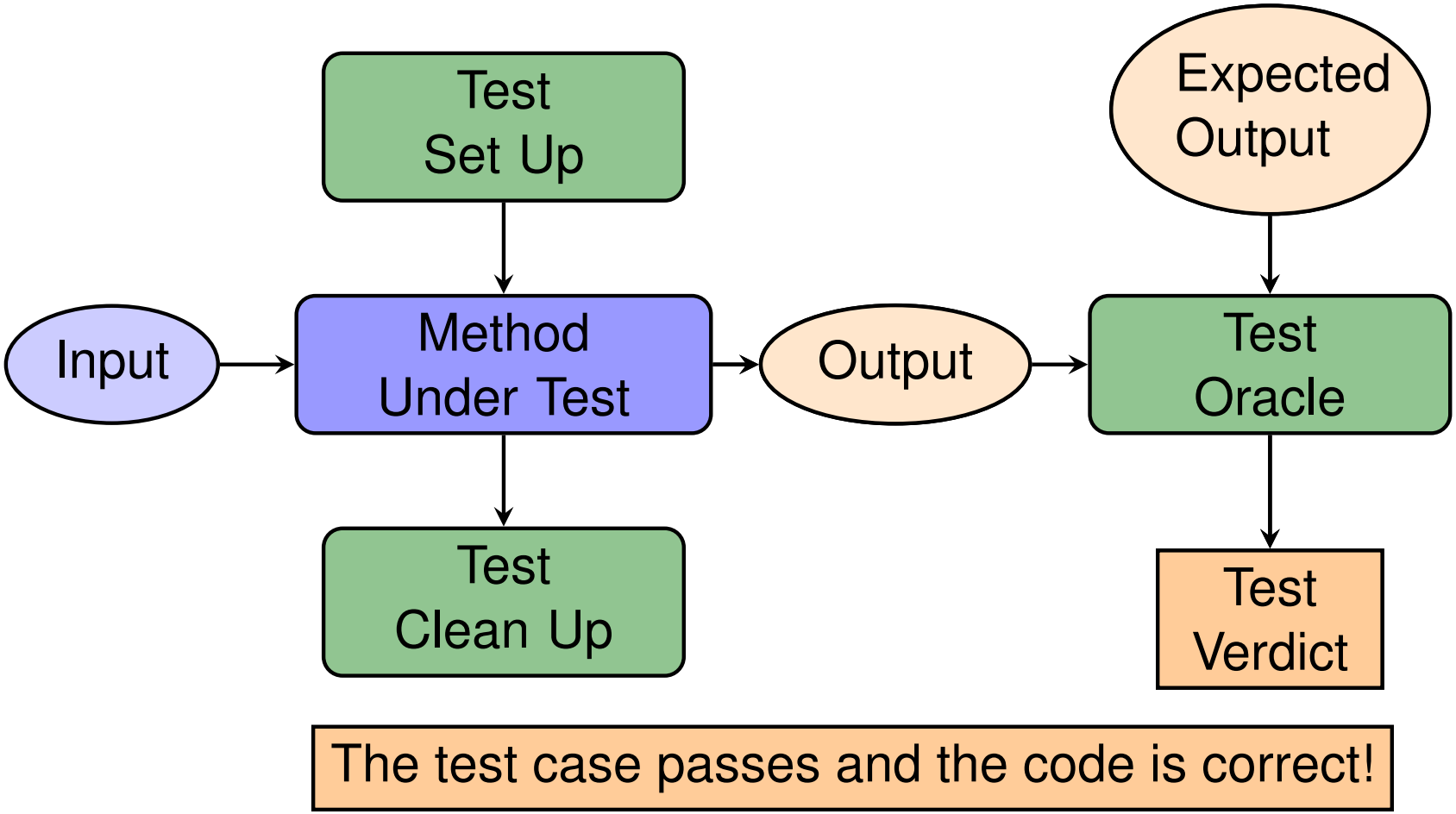
What is a Test Case?



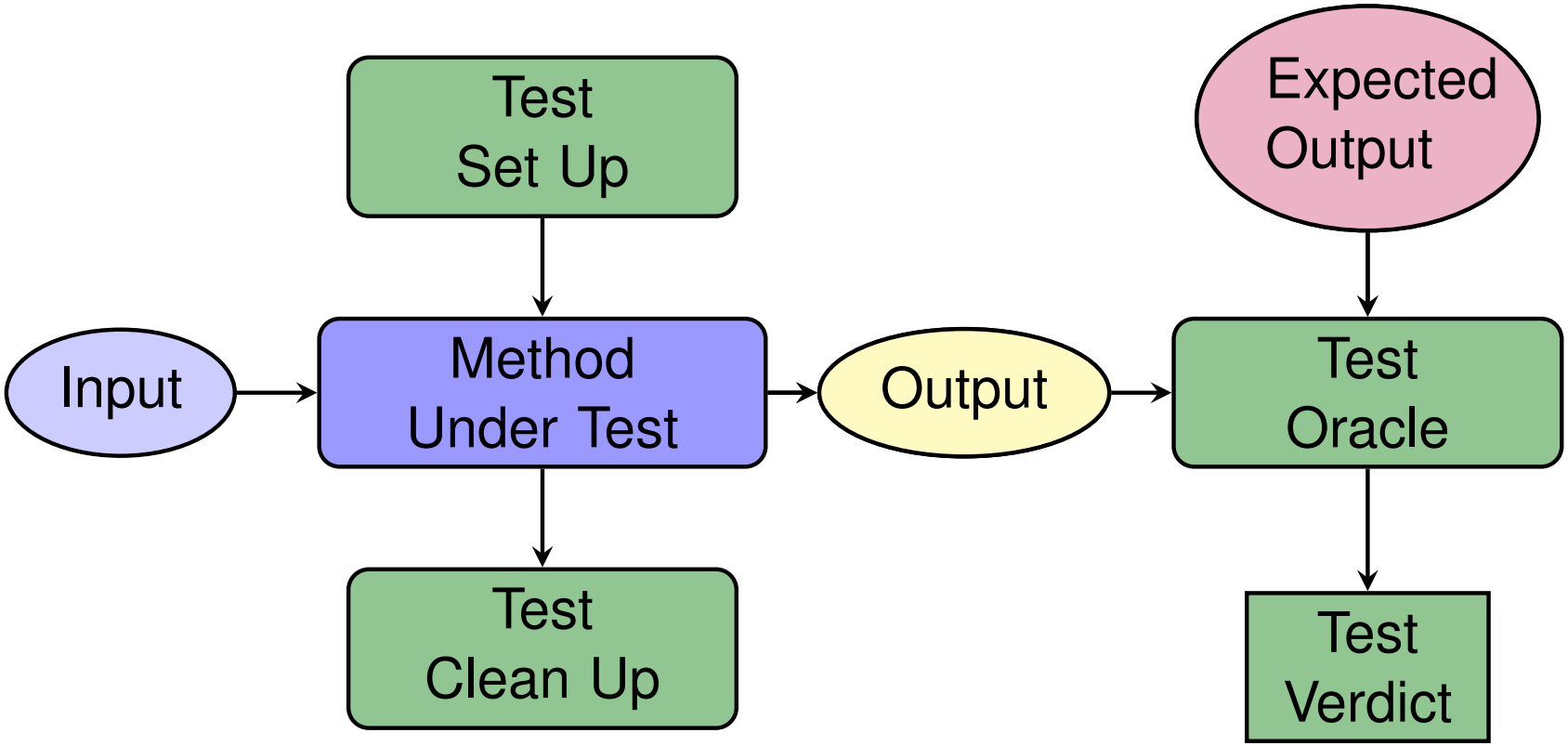
What is a Test Case?



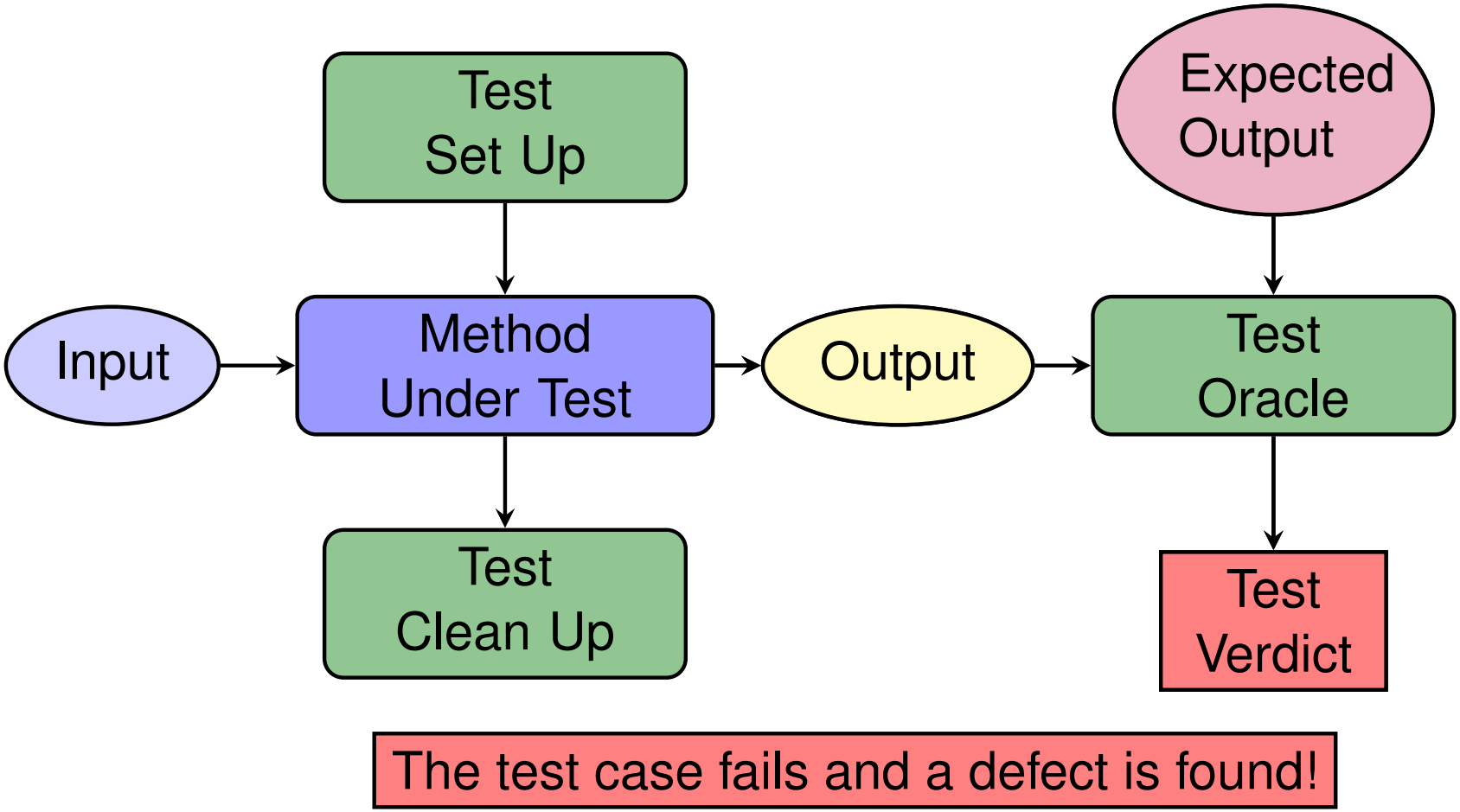
What is a Test Case?



What is a Test Case?



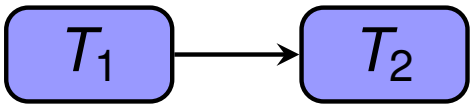
What is a Test Case?



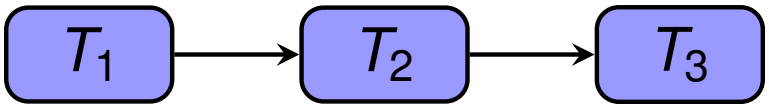
What is a Test Suite?

T_1

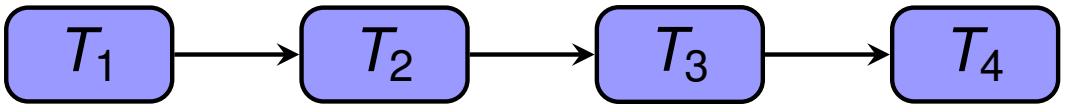
What is a Test Suite?



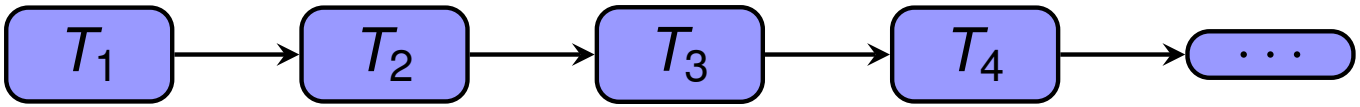
What is a Test Suite?



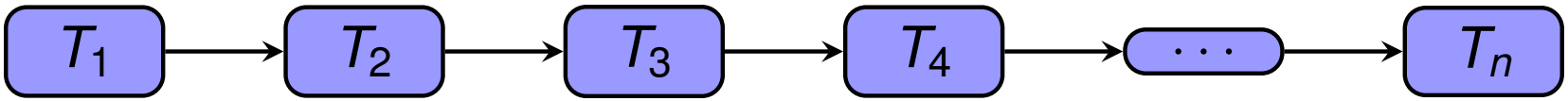
What is a Test Suite?



What is a Test Suite?

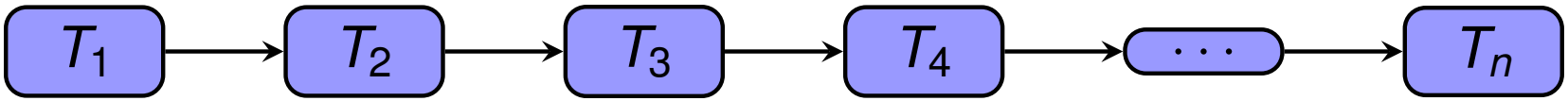


What is a Test Suite?



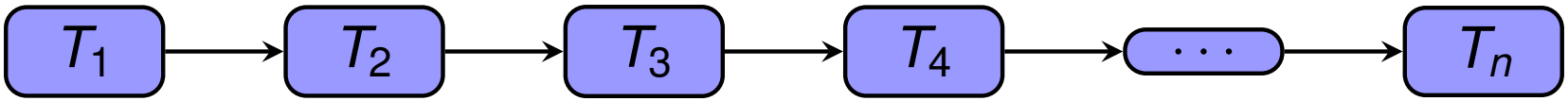
What is a Test Suite?

Organize the Test Cases into a Test Suite



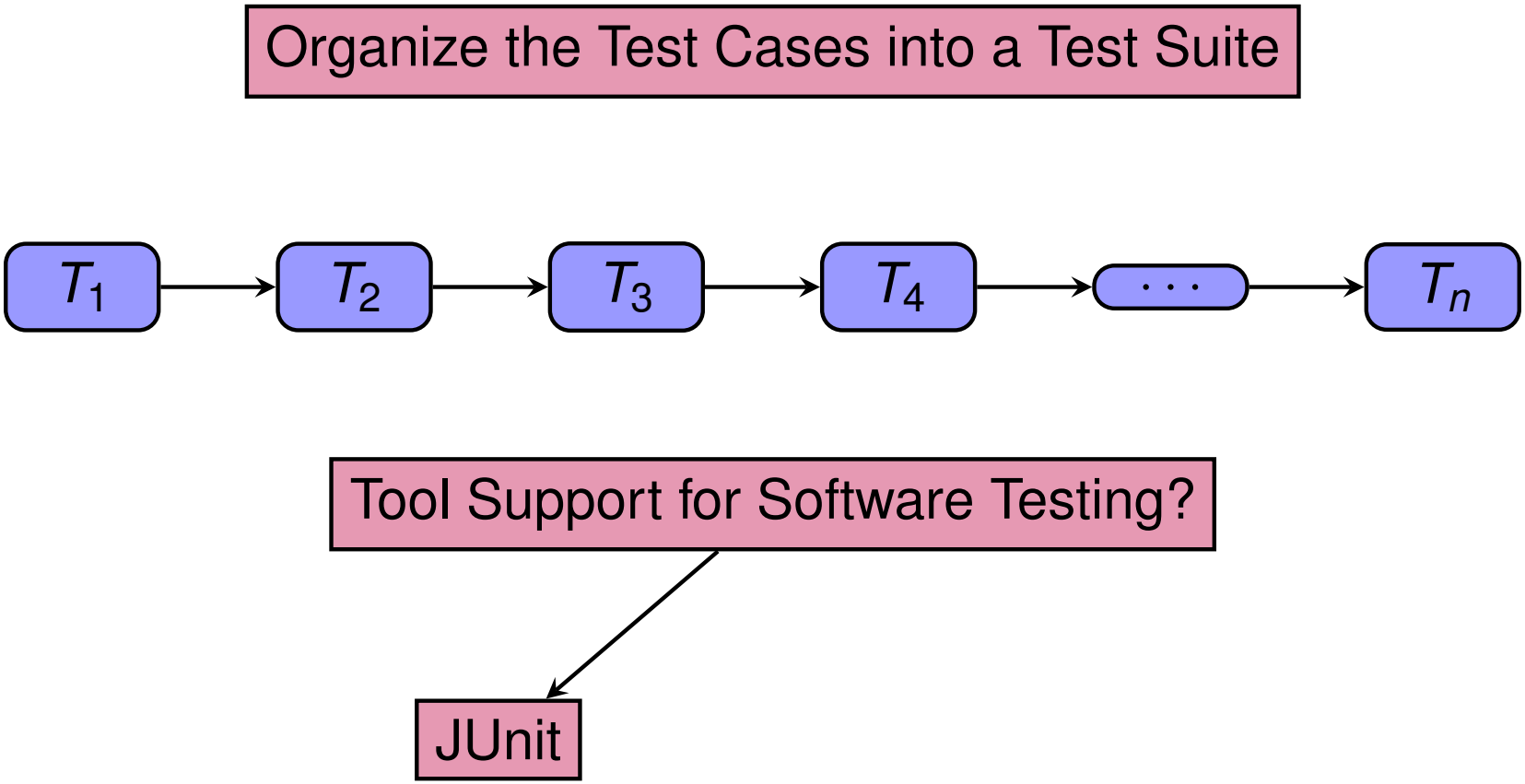
What is a Test Suite?

Organize the Test Cases into a Test Suite

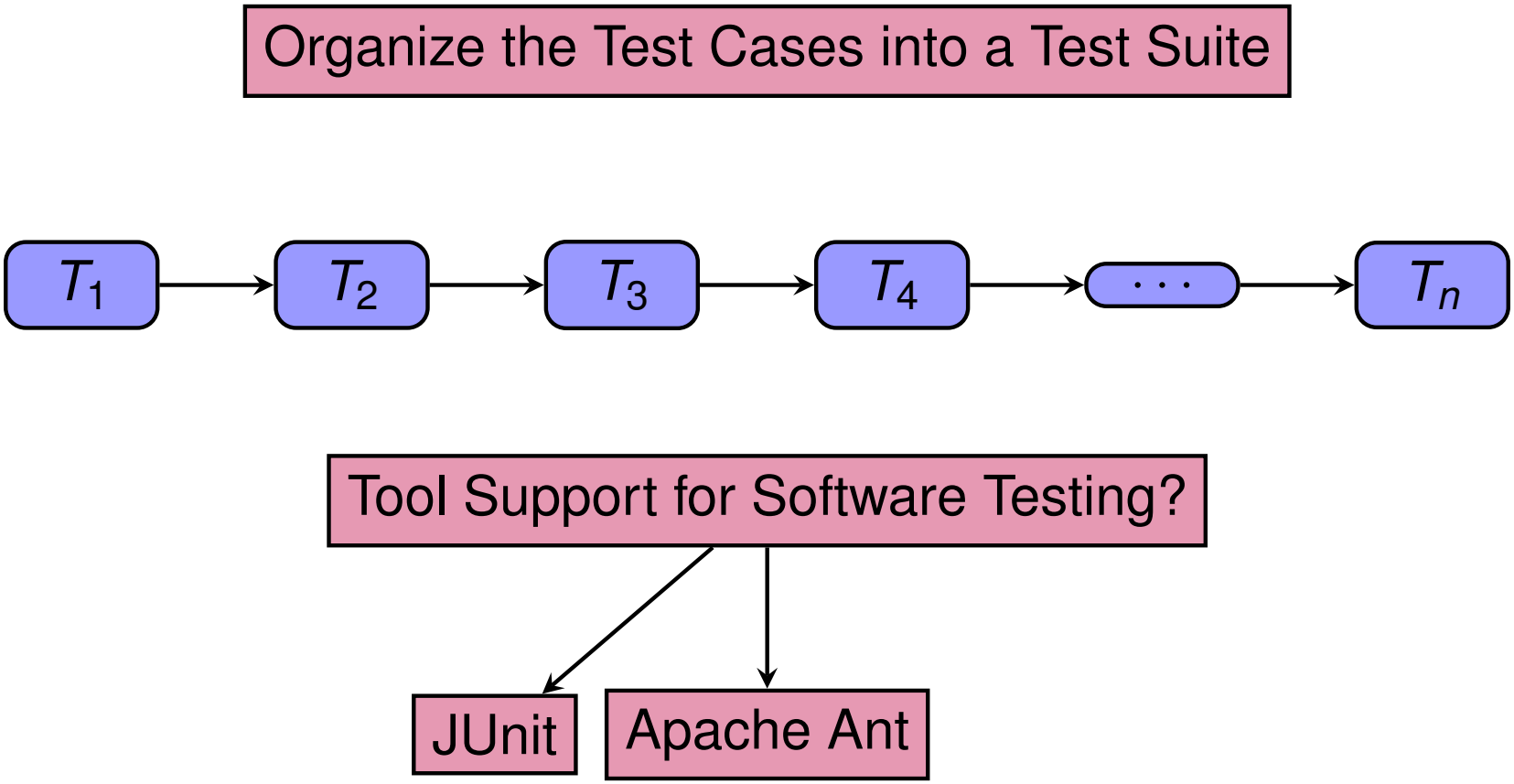


Tool Support for Software Testing?

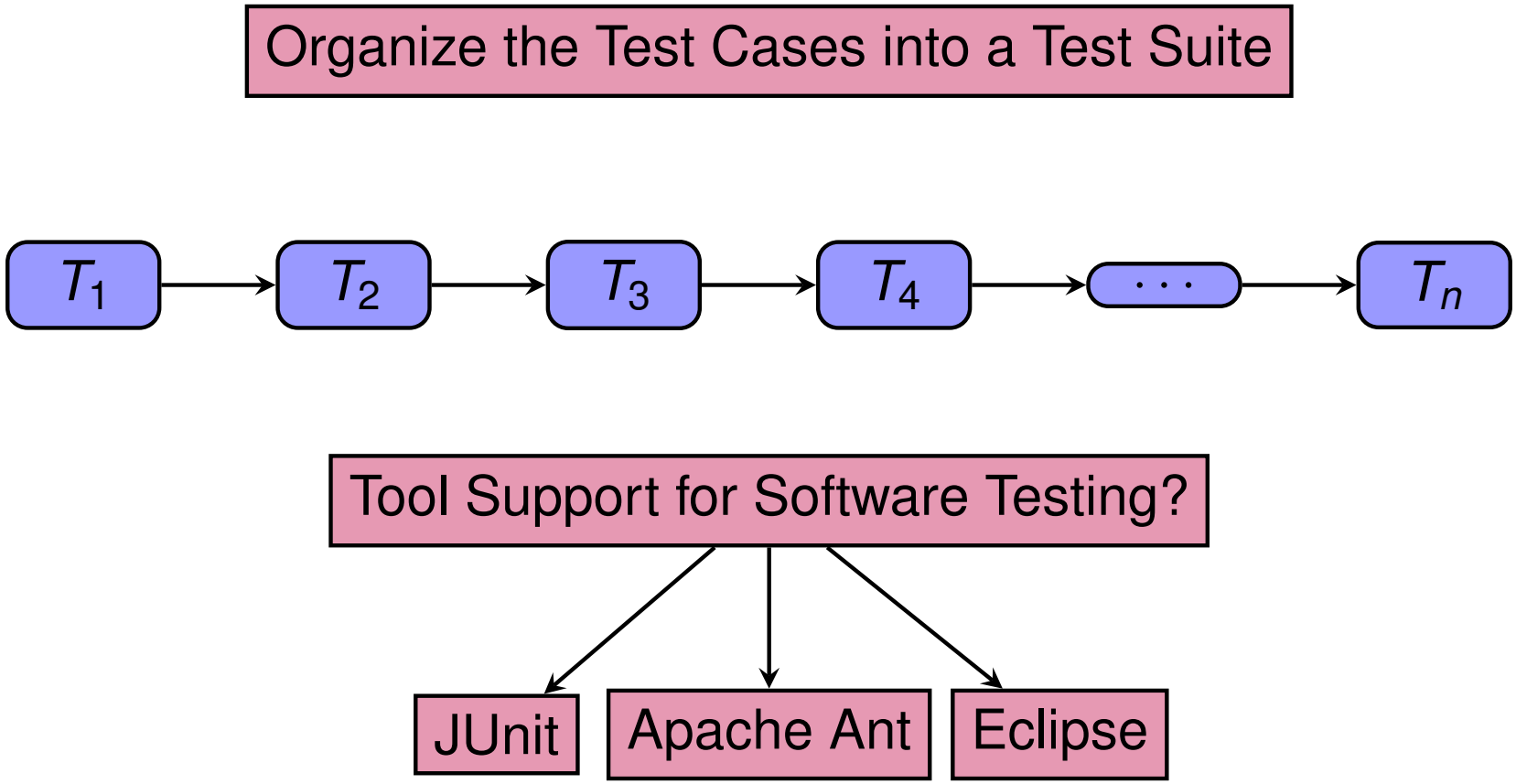
What is a Test Suite?



What is a Test Suite?

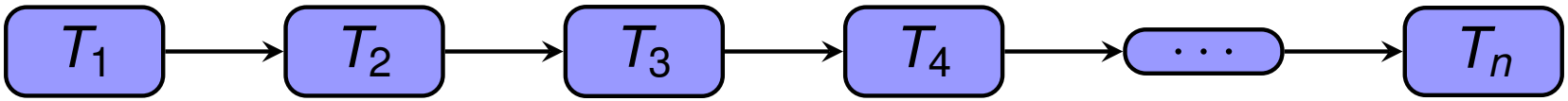


What is a Test Suite?

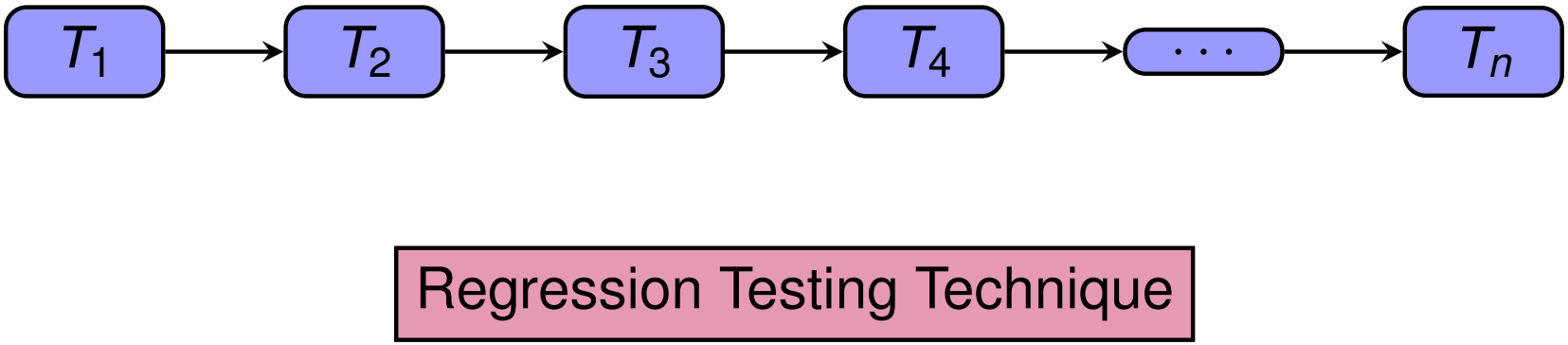


Test Suite Management

Organize the Test Cases into a Test Suite

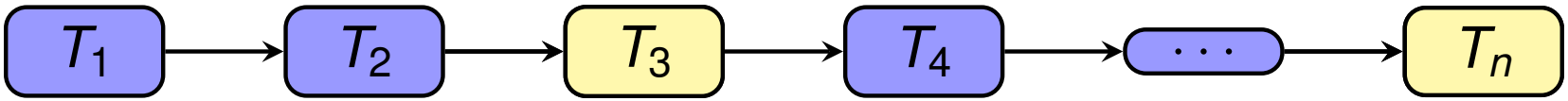


Test Suite Management



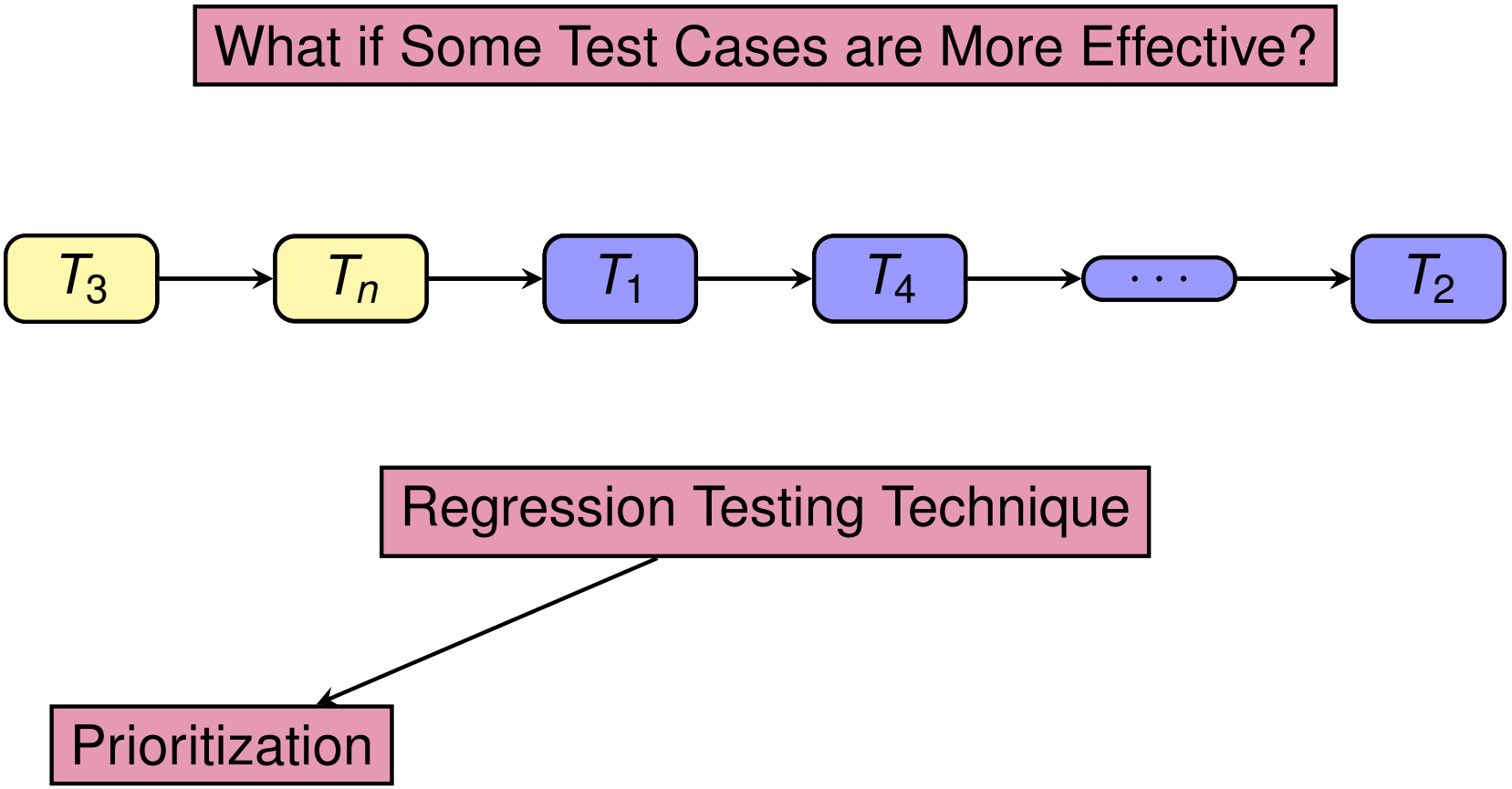
Test Suite Management

What if Some Test Cases are More Effective?

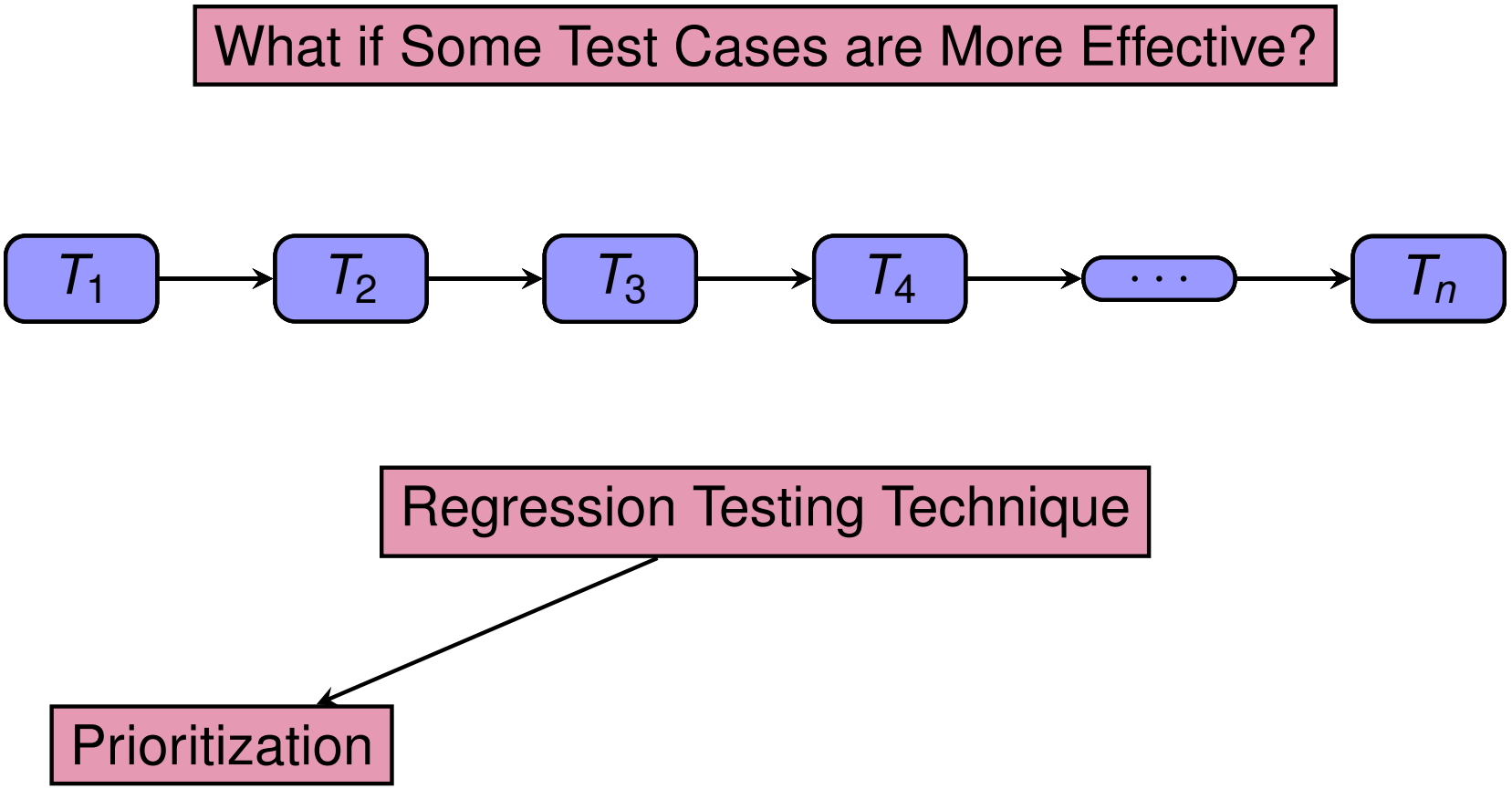


Regression Testing Technique

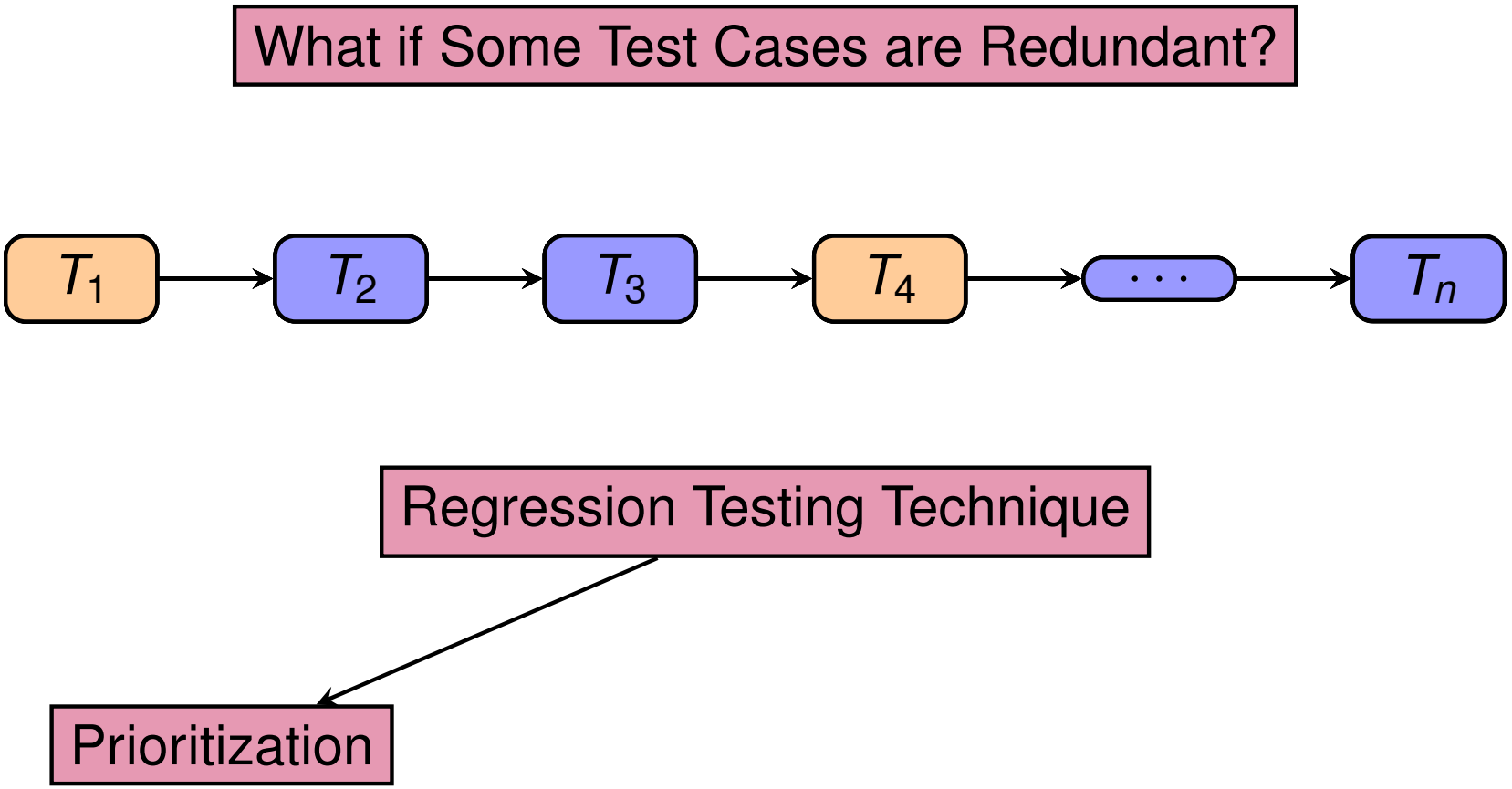
Test Suite Management



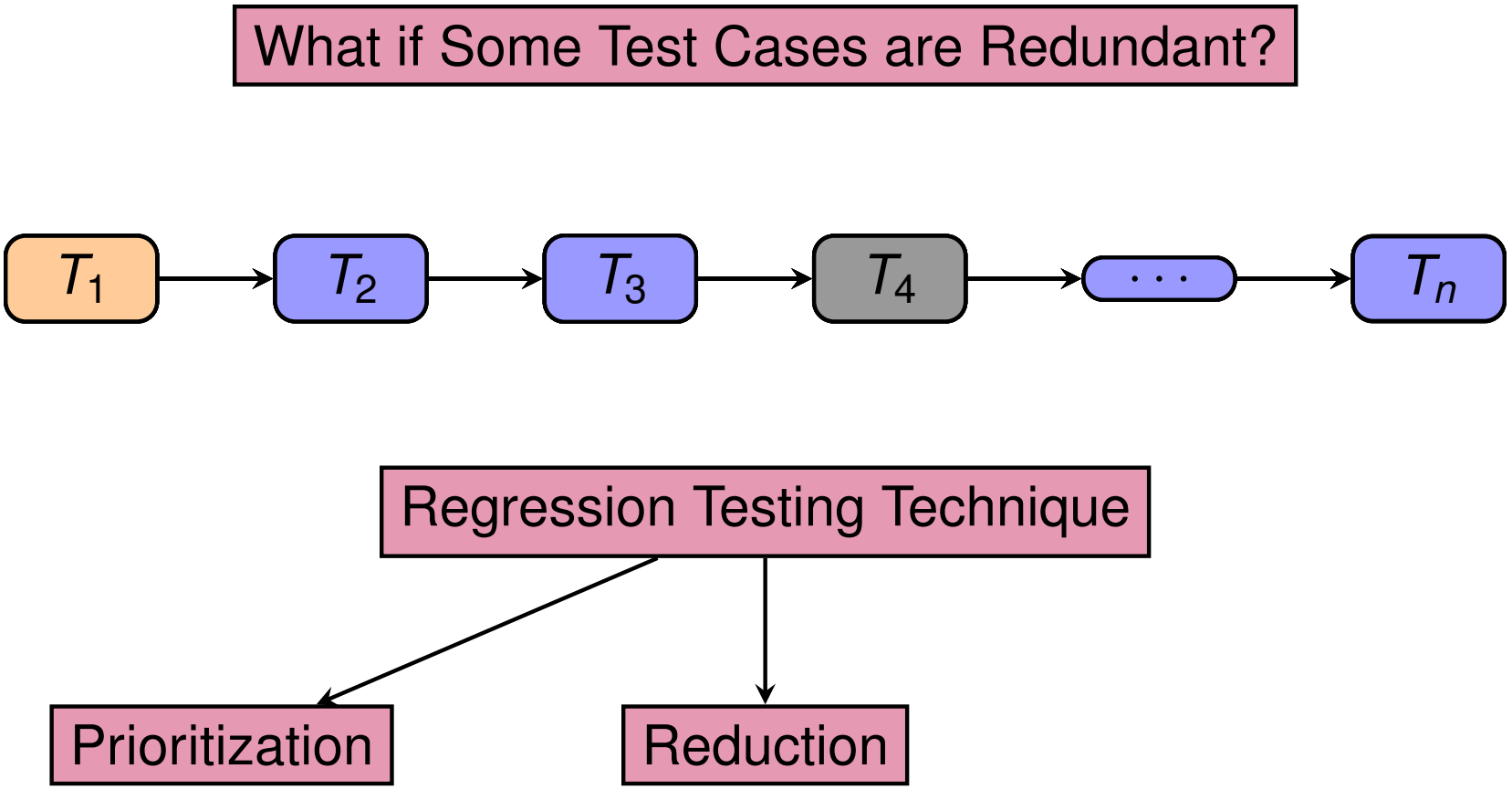
Test Suite Management



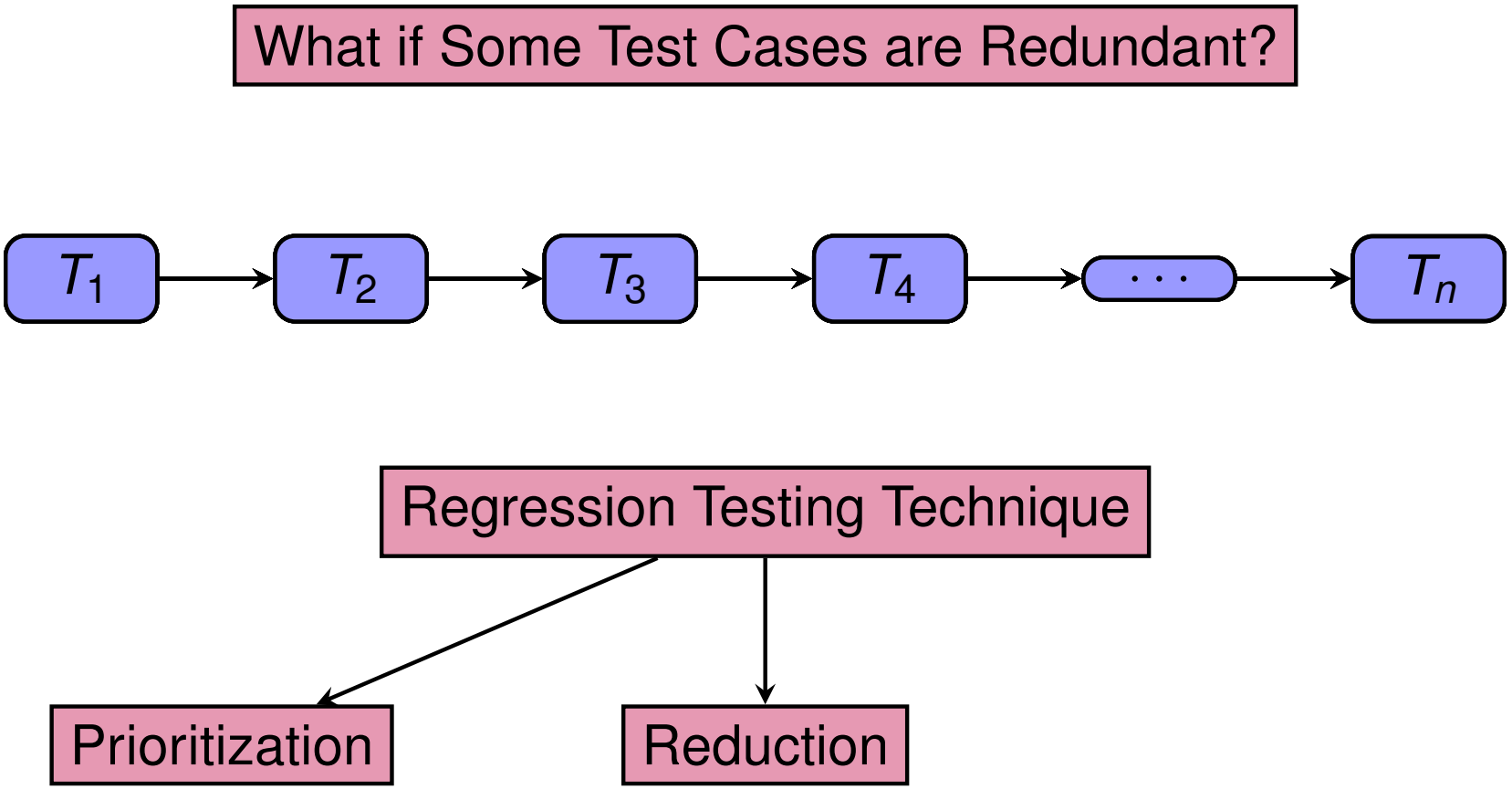
Test Suite Management



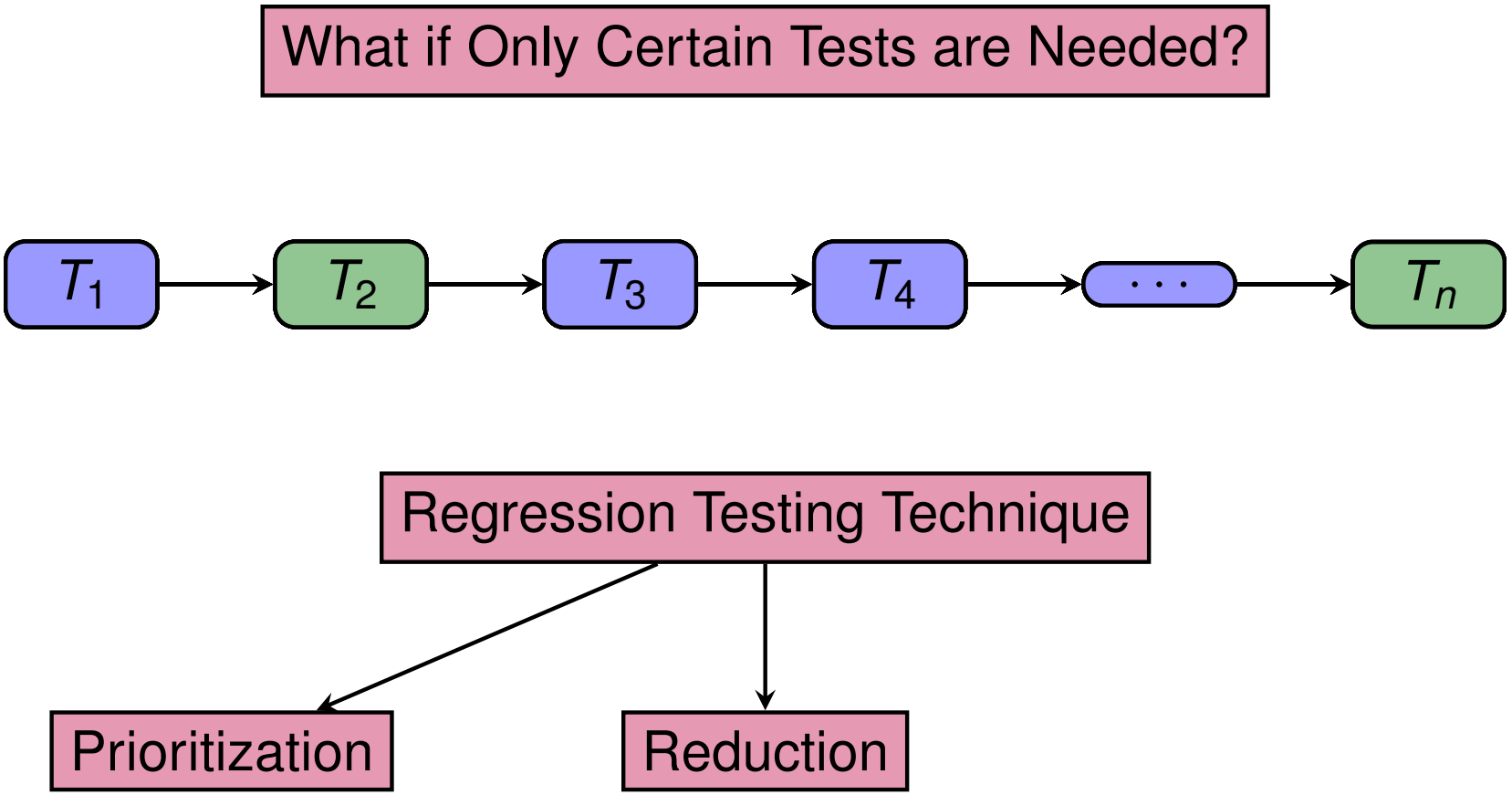
Test Suite Management



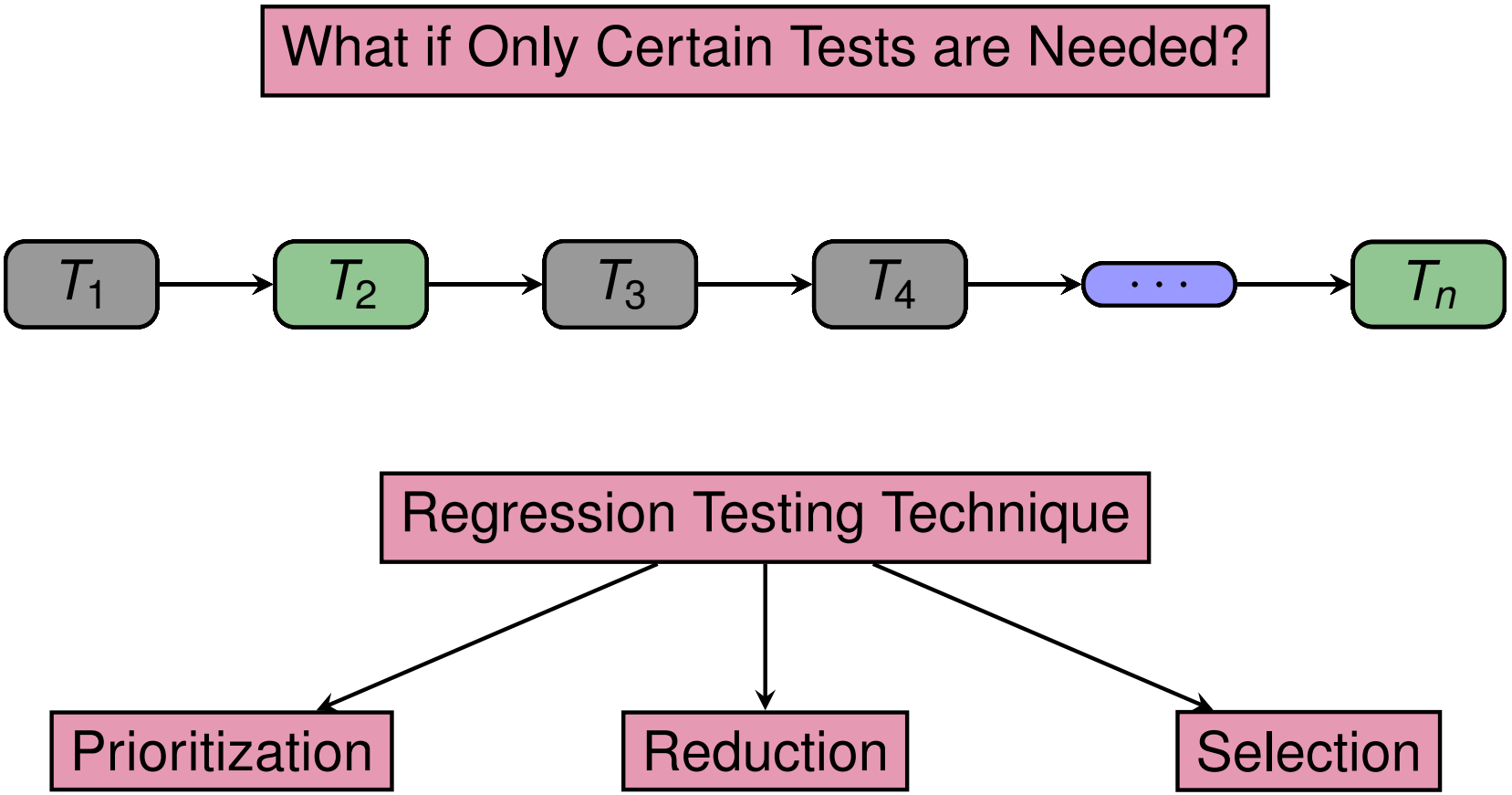
Test Suite Management



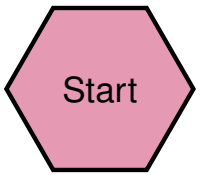
Test Suite Management



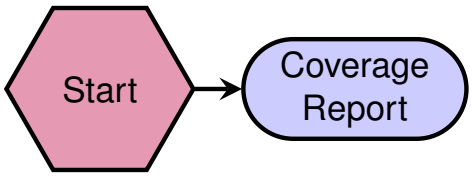
Test Suite Management



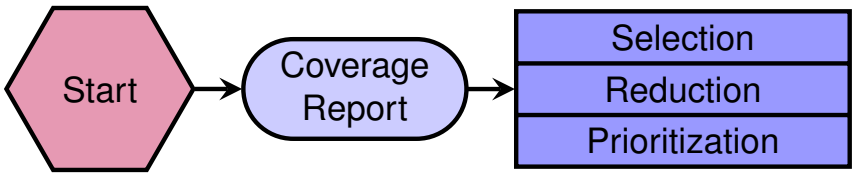
Model of Regression Testing



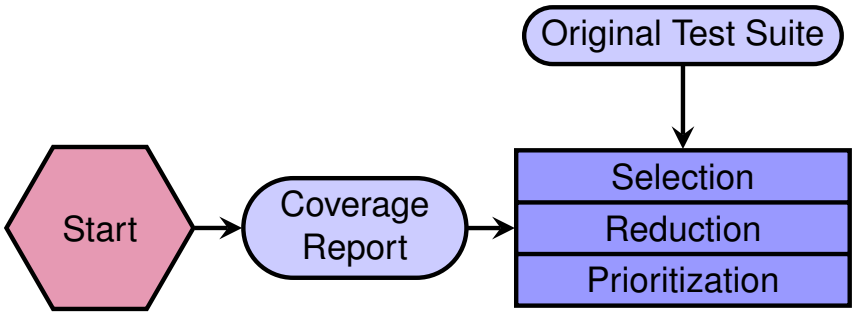
Model of Regression Testing



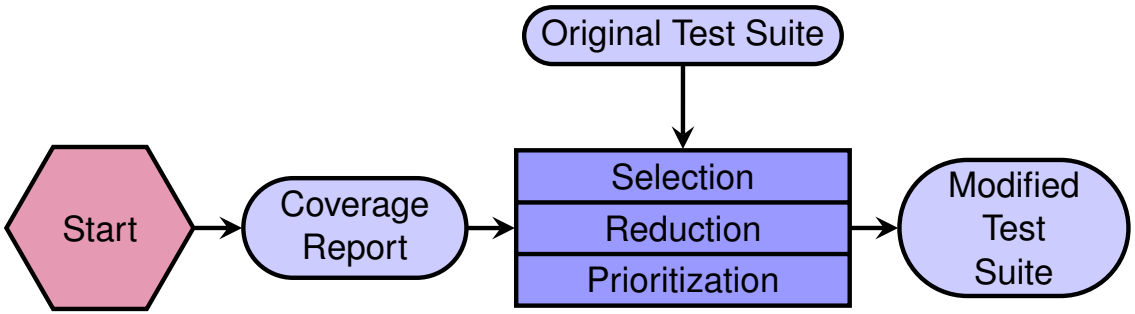
Model of Regression Testing



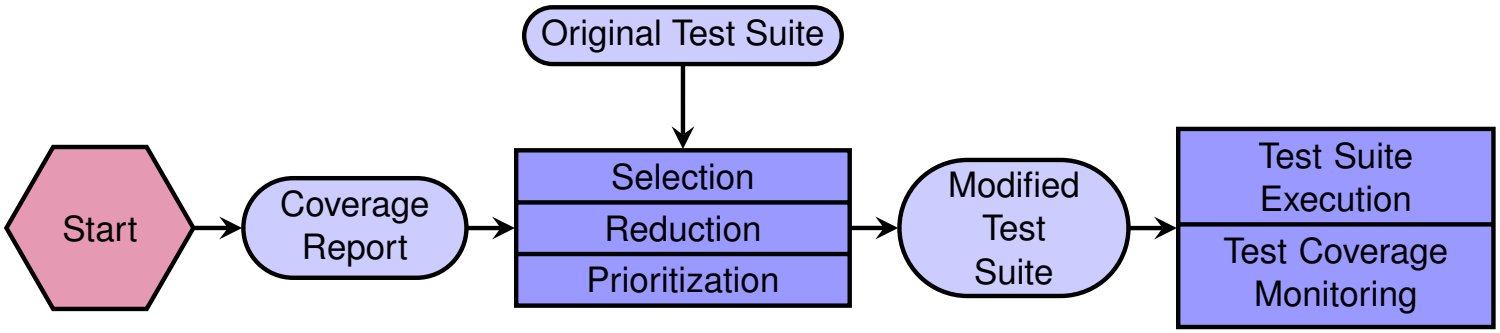
Model of Regression Testing



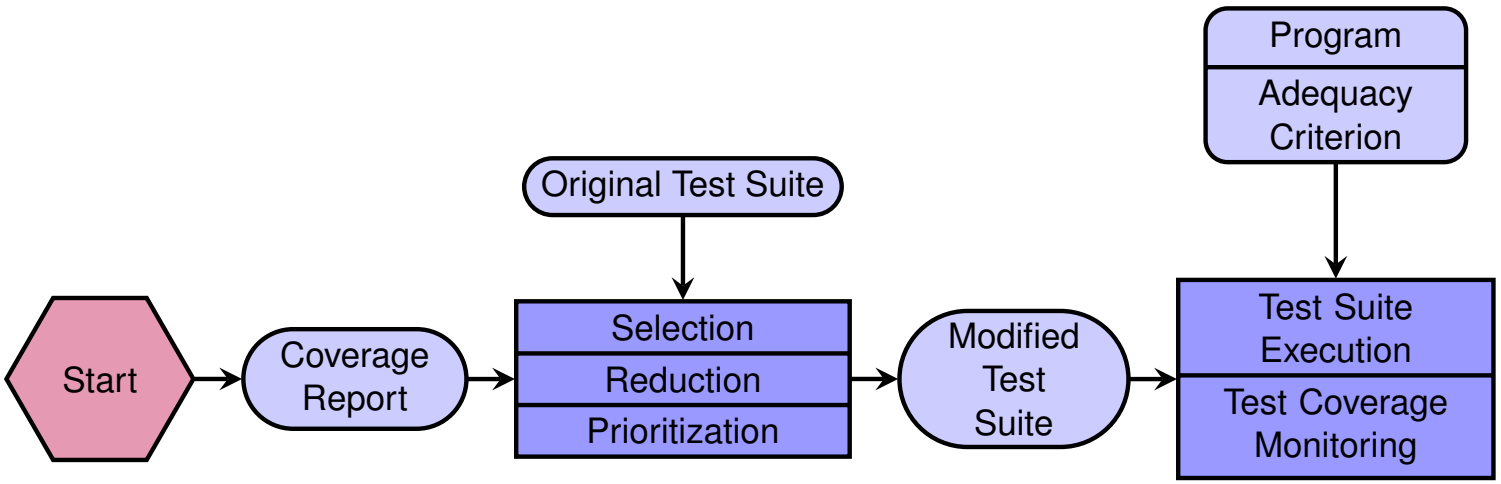
Model of Regression Testing



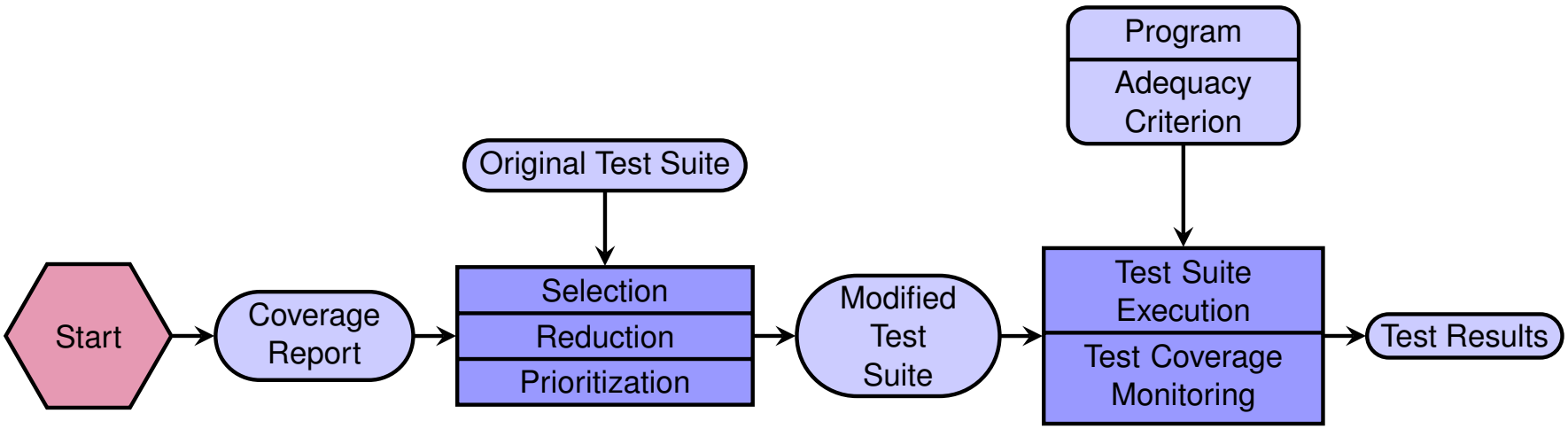
Model of Regression Testing



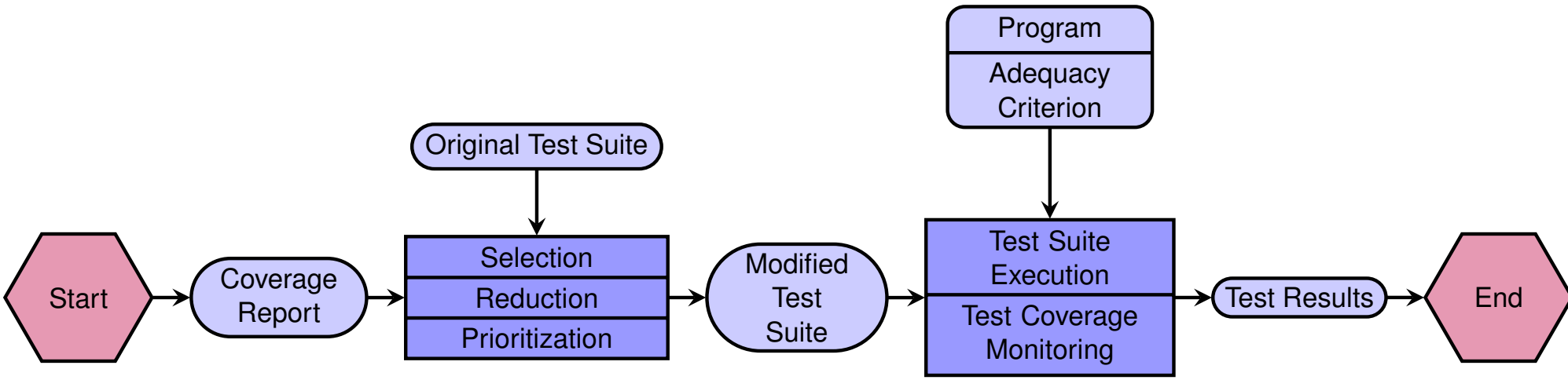
Model of Regression Testing



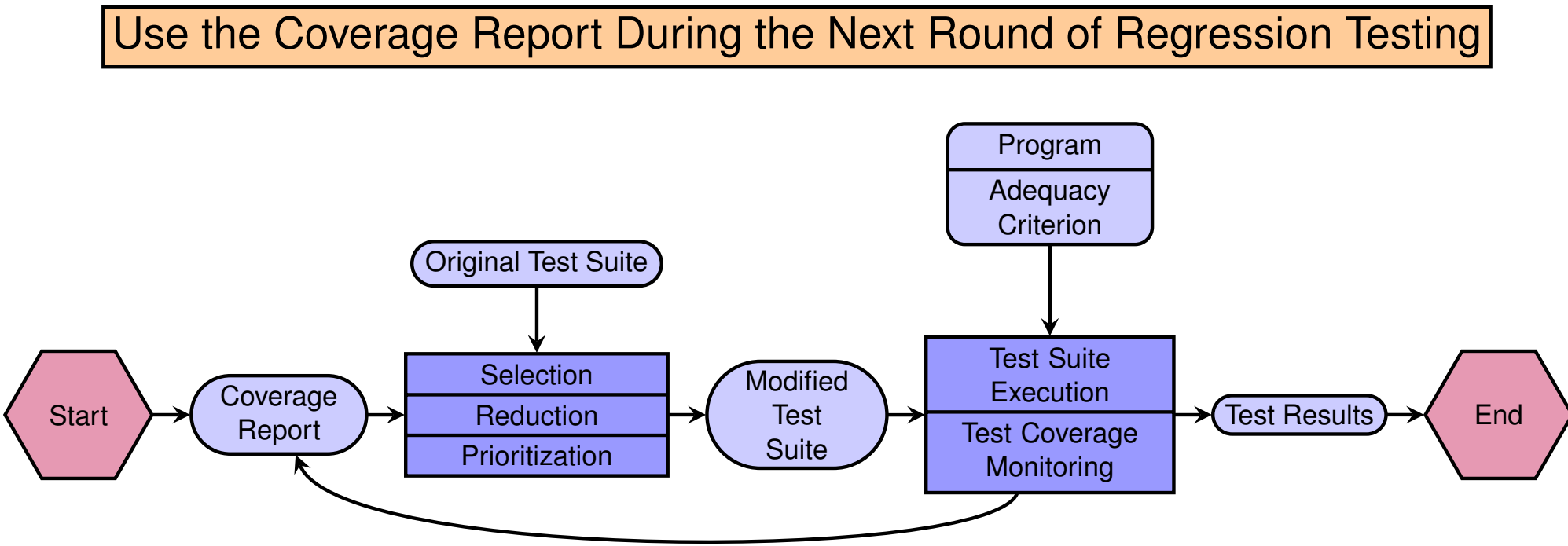
Model of Regression Testing



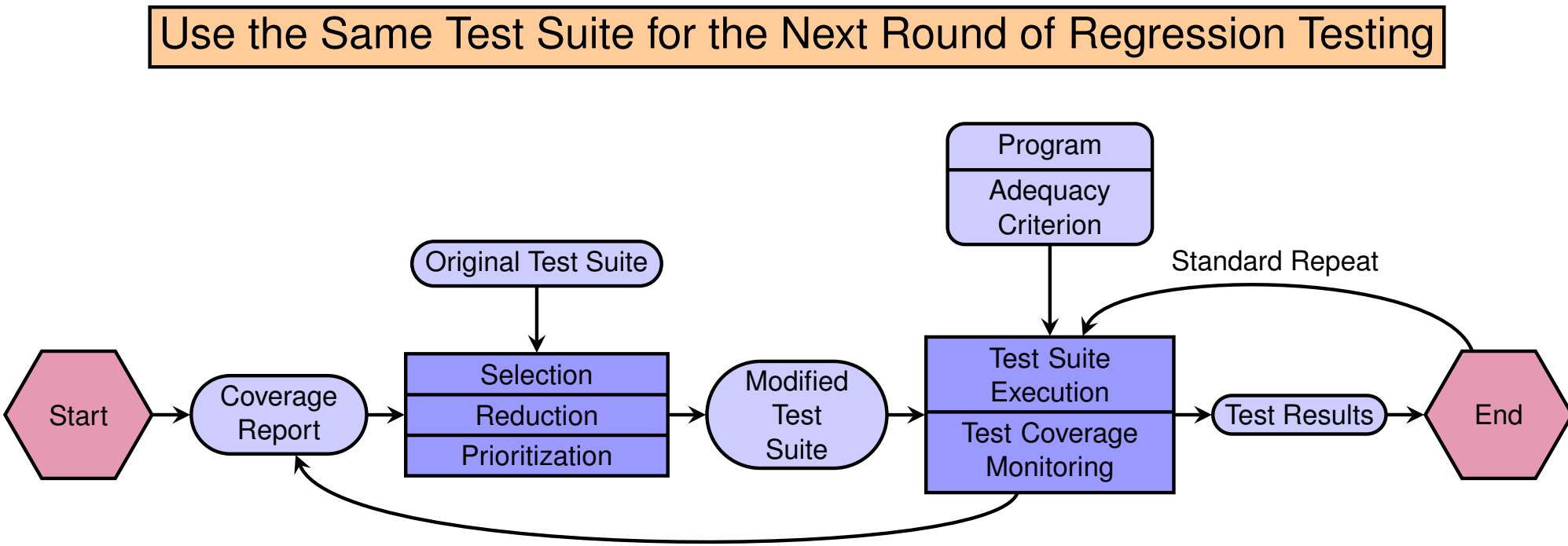
Model of Regression Testing



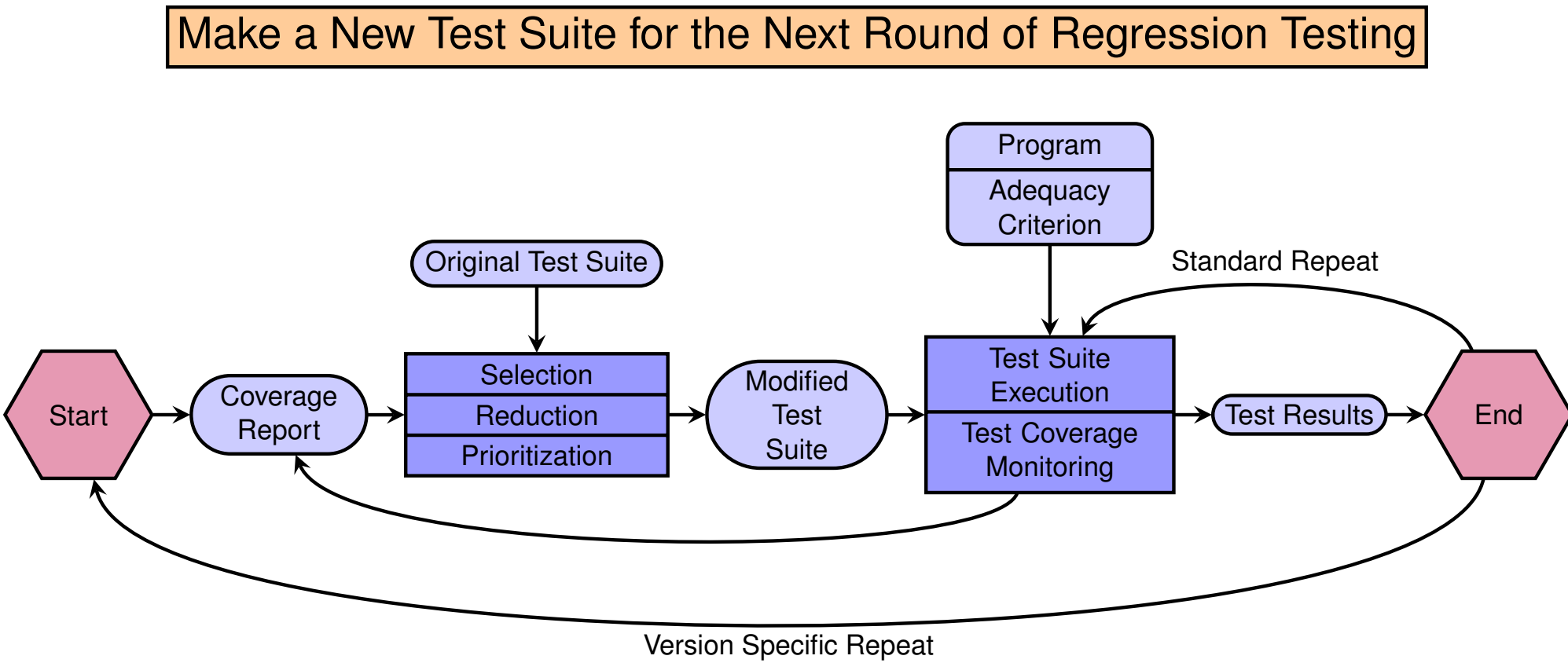
Model of Regression Testing



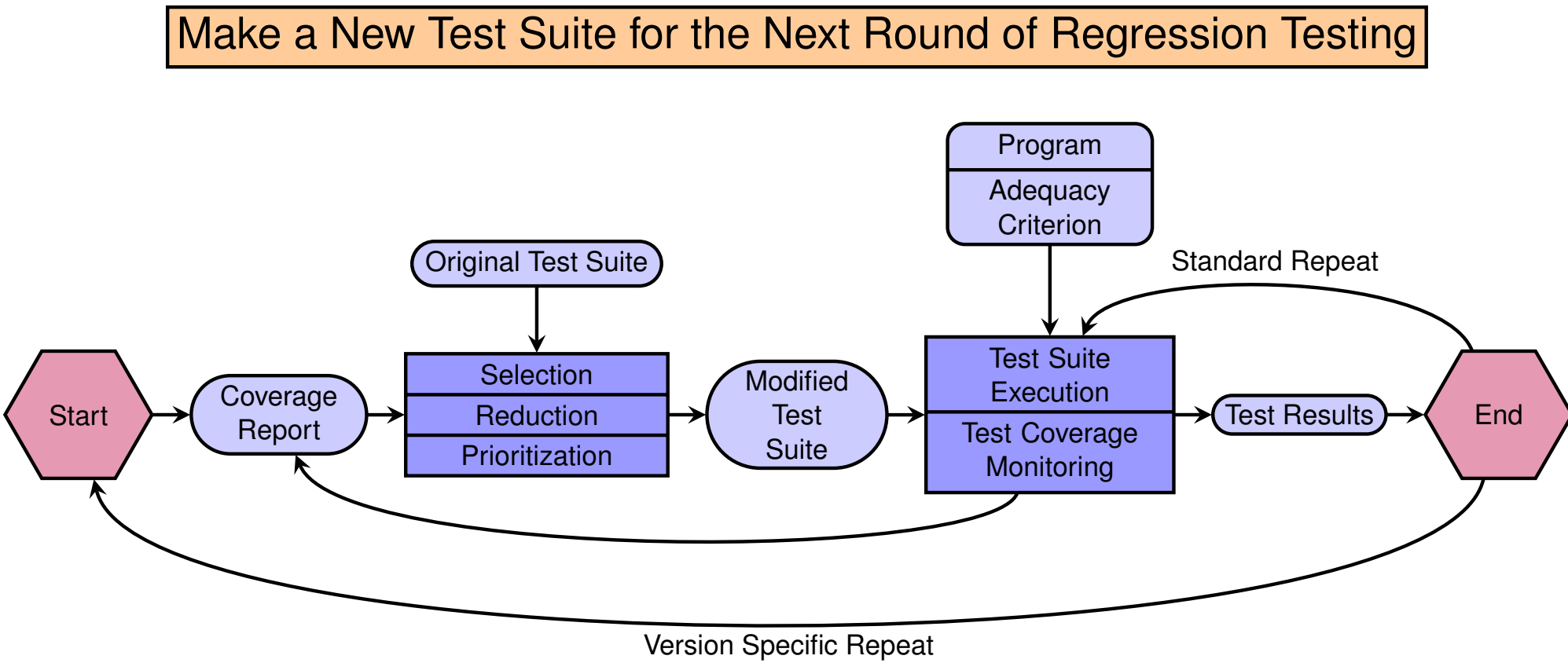
Model of Regression Testing



Model of Regression Testing



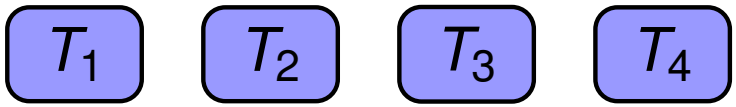
Model of Regression Testing



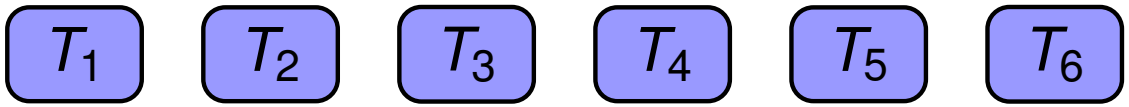
Test Suite Adequacy

T_1 T_2

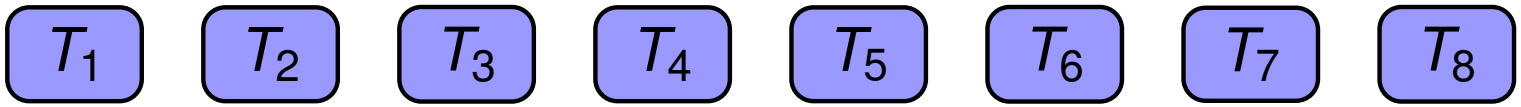
Test Suite Adequacy



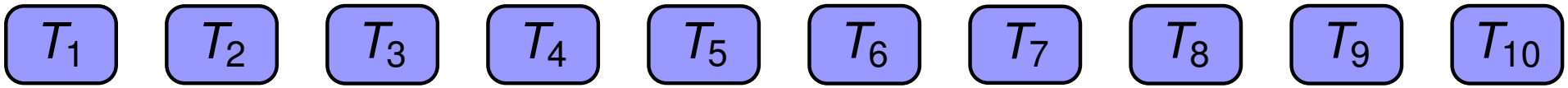
Test Suite Adequacy



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

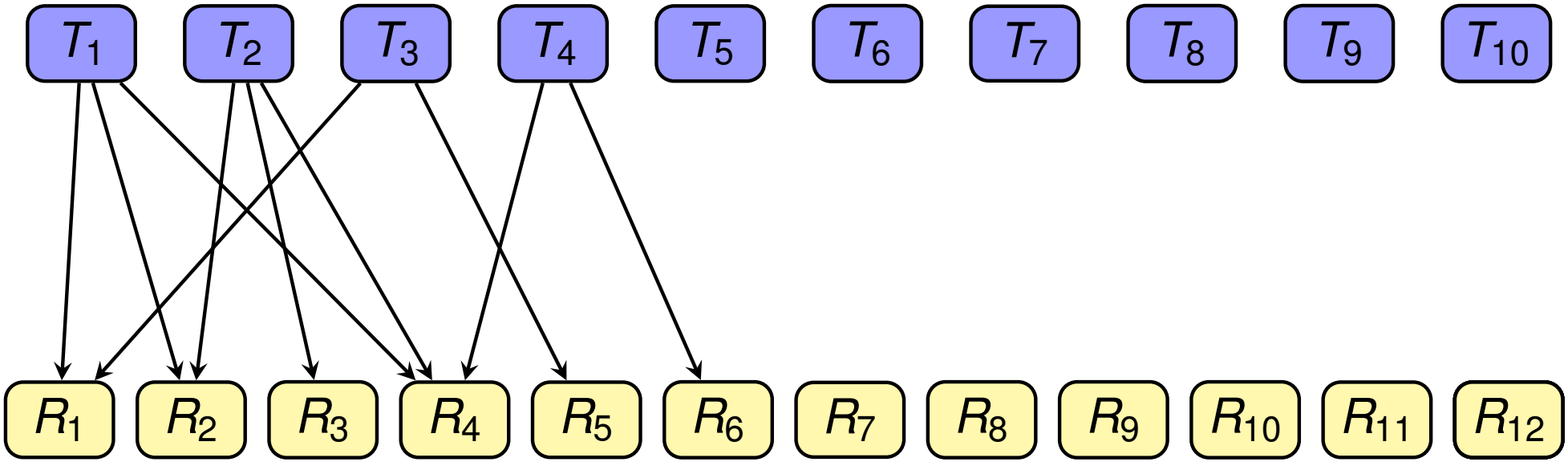
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10}

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

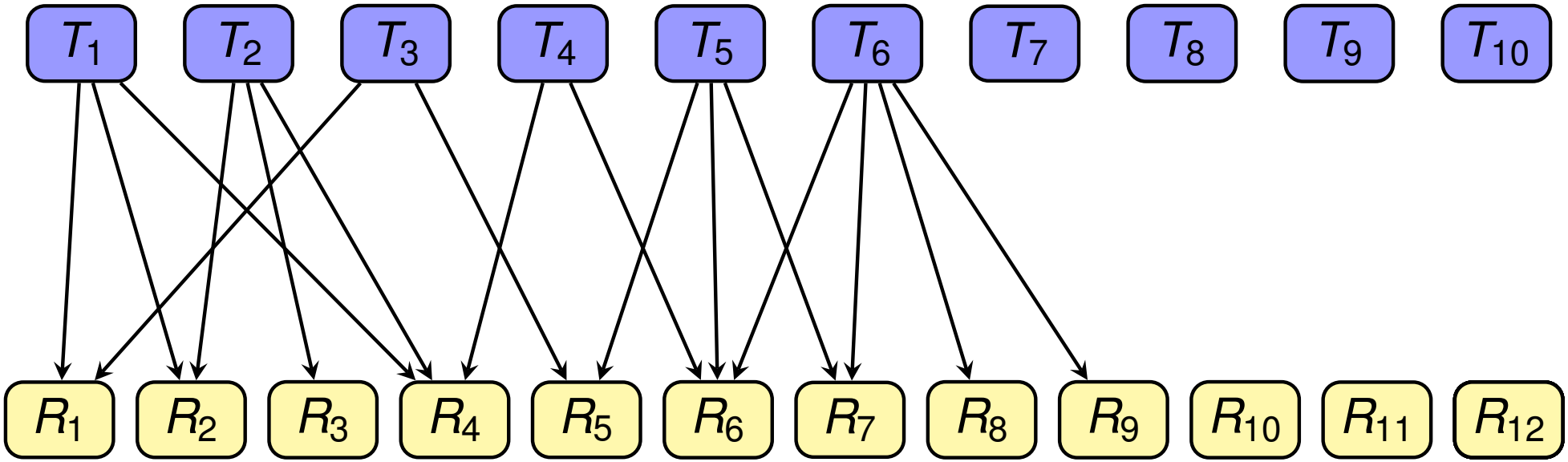
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$



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

Test Suite Adequacy

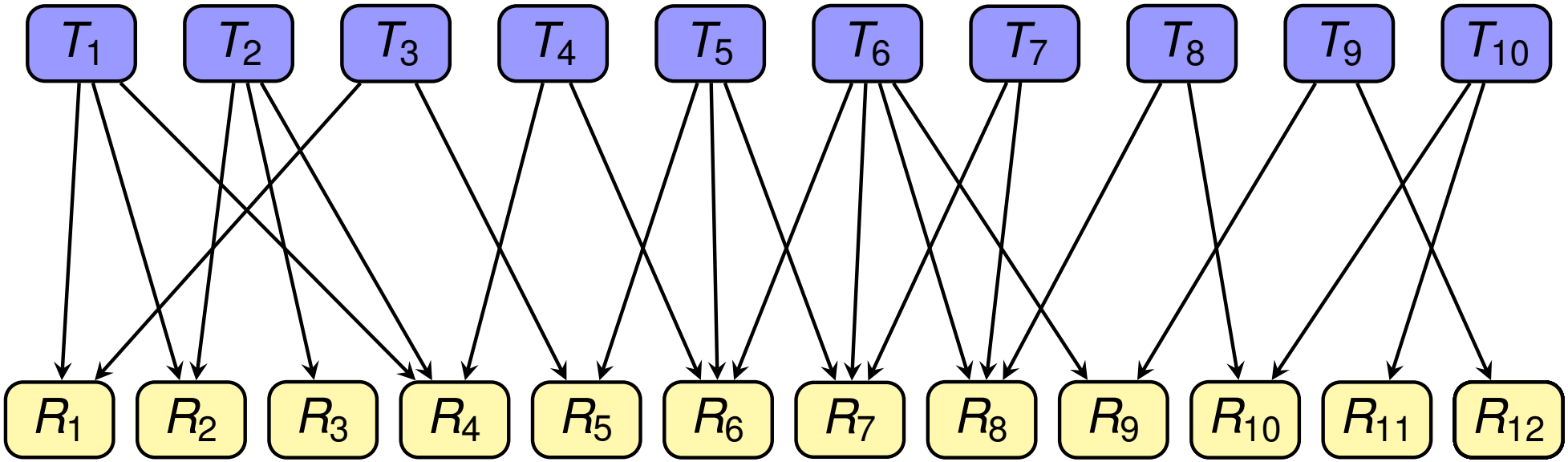
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$



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

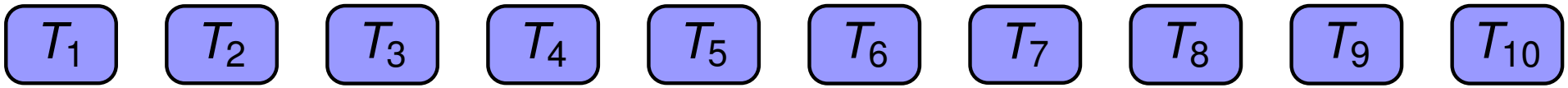
Test Suite Adequacy

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

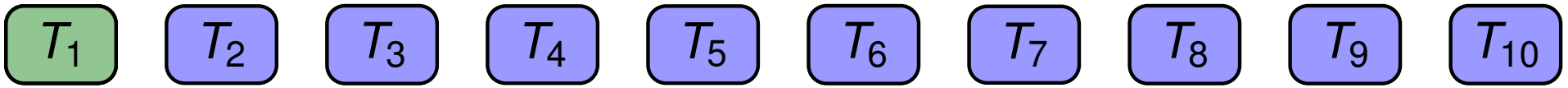
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10}

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

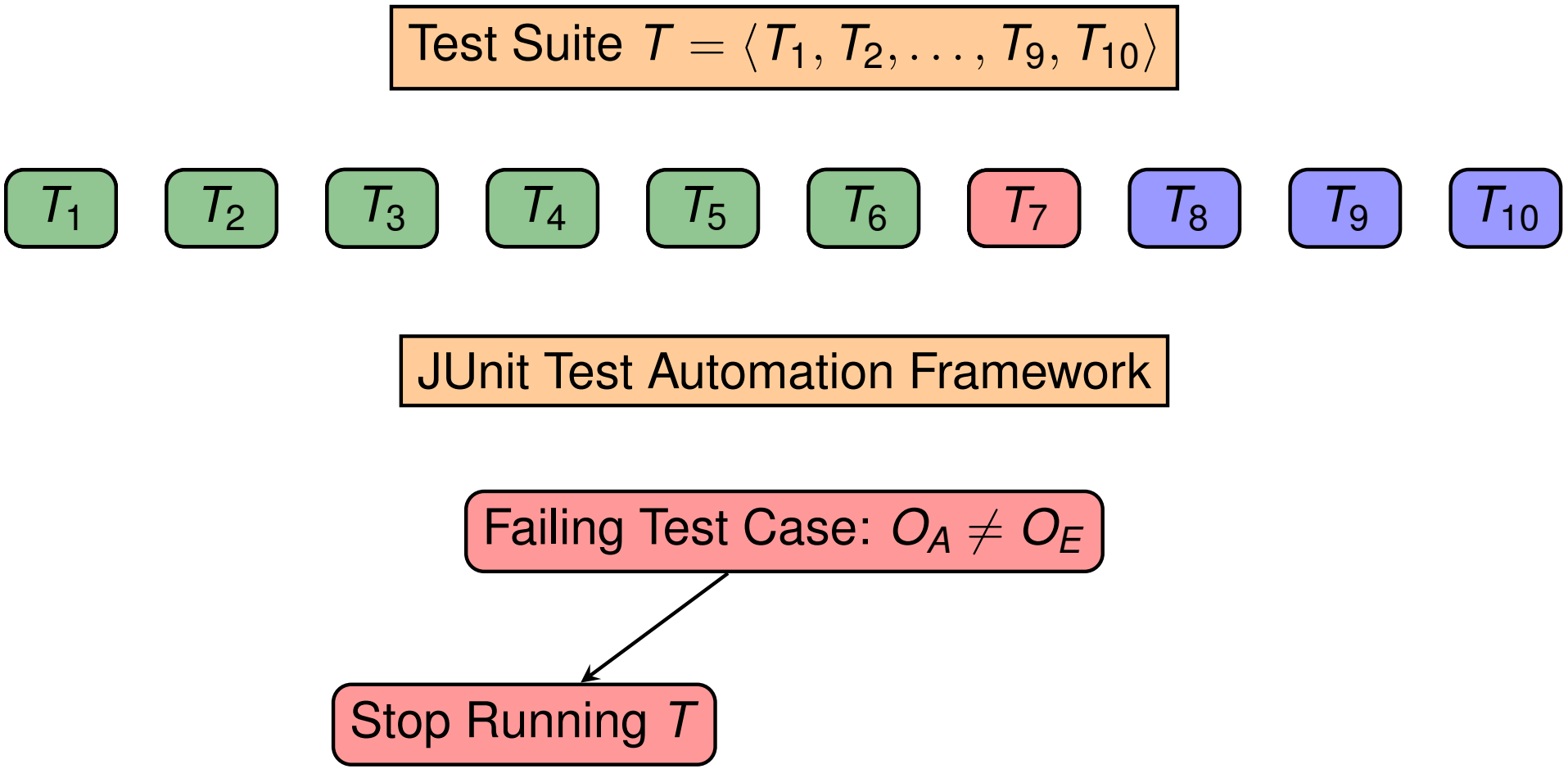
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$

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

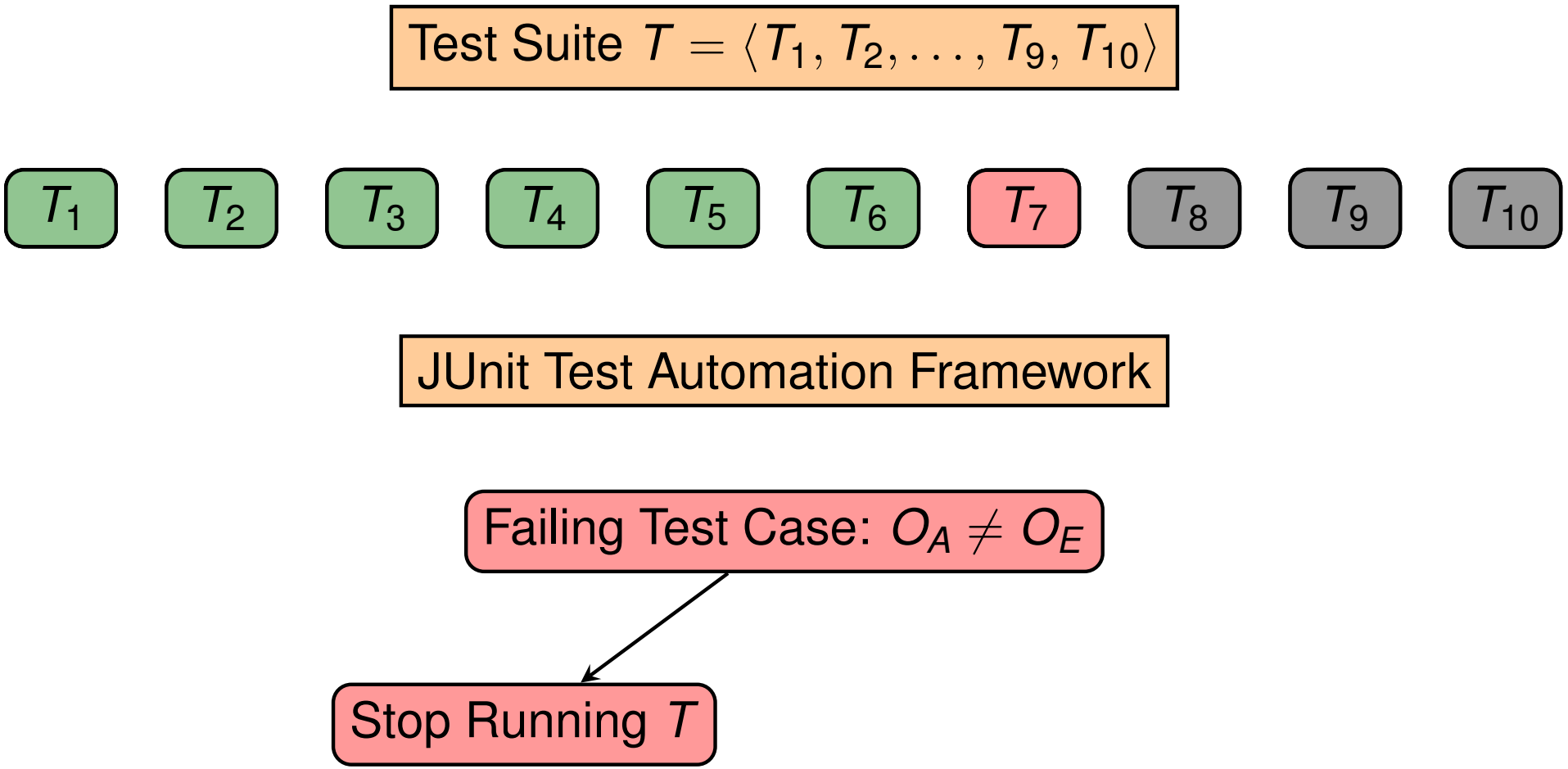
JUnit Test Automation Framework

Failing Test Case: $O_A \neq O_E$

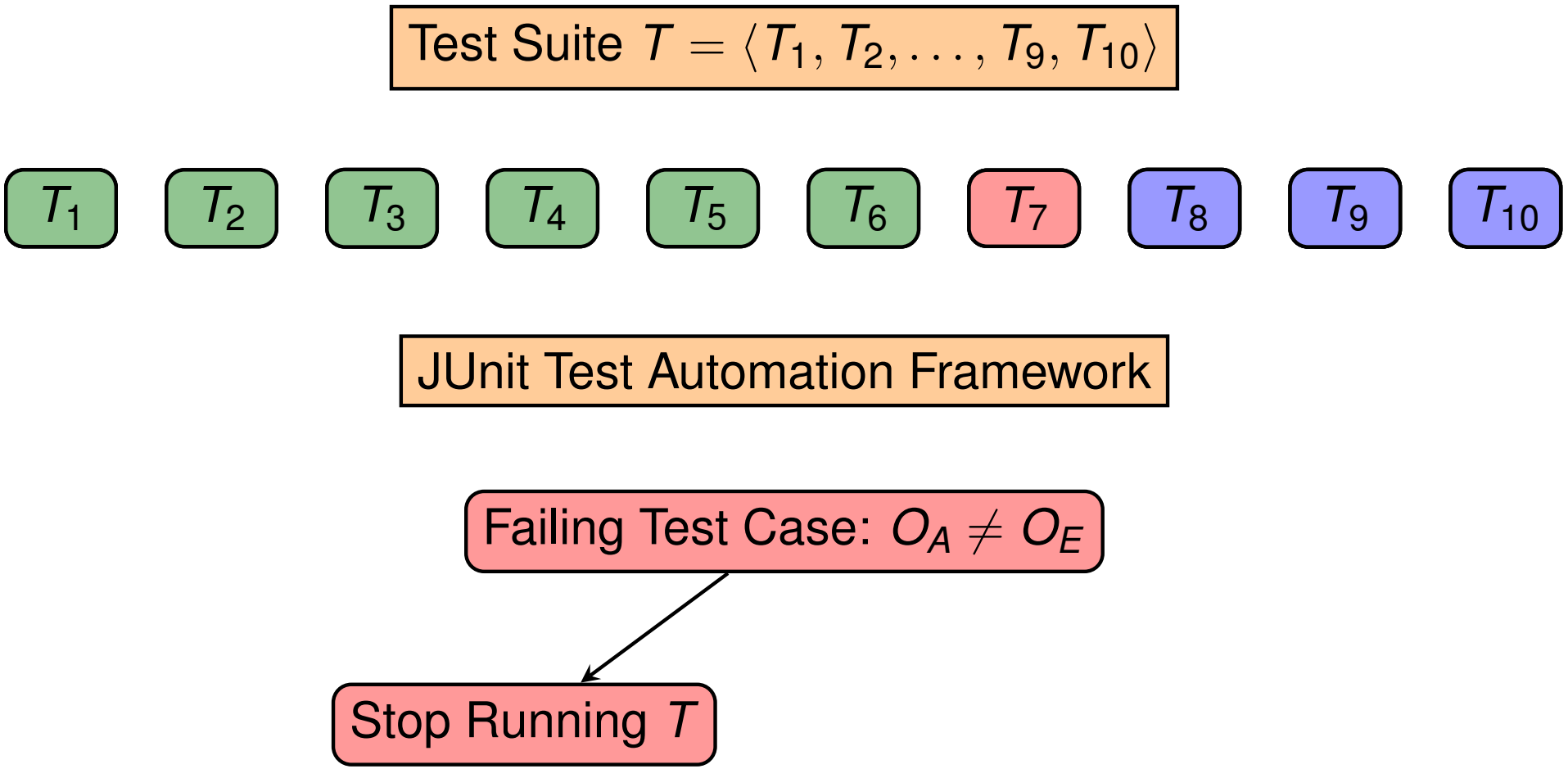
Test Suite Execution



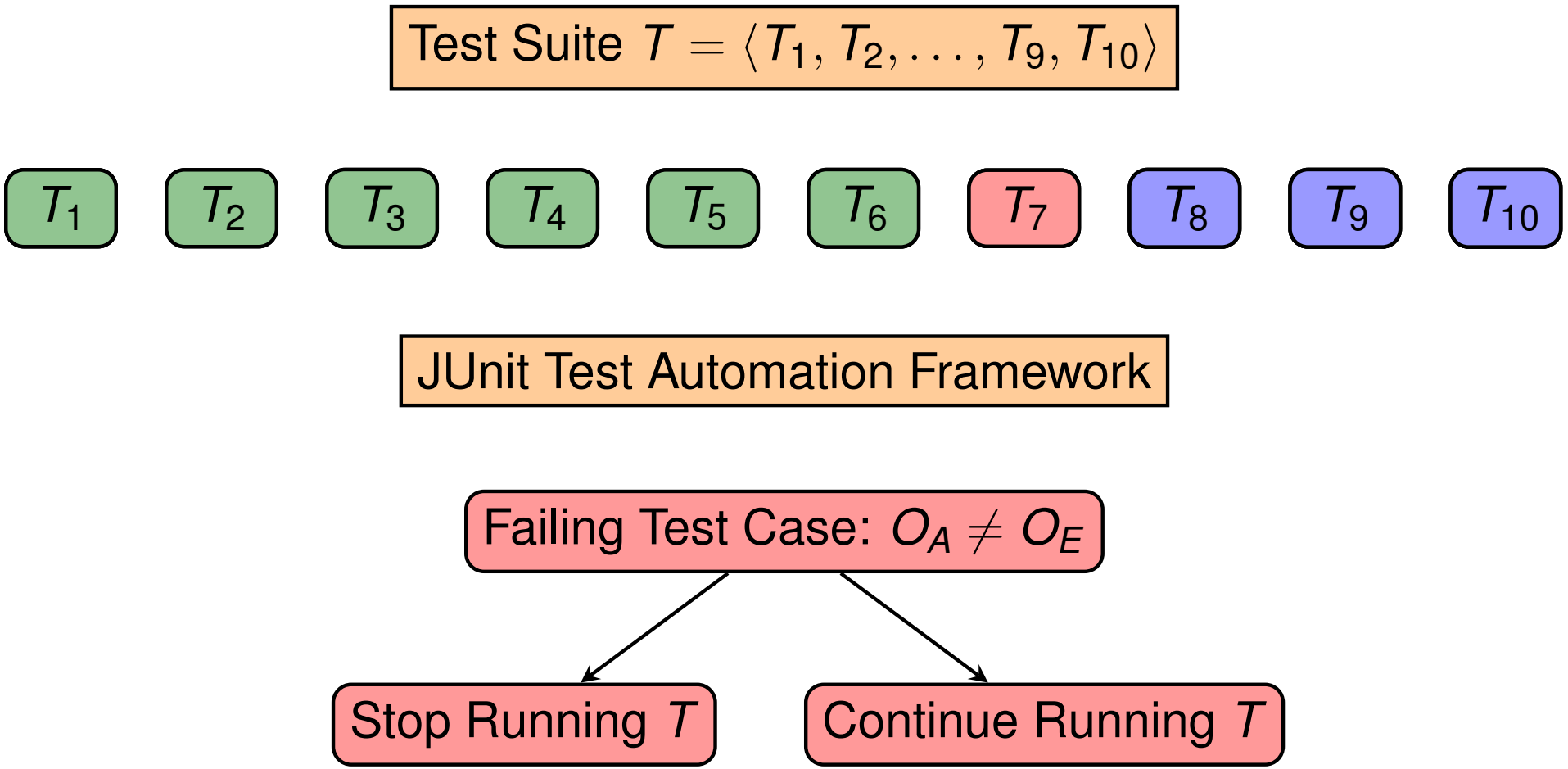
Test Suite Execution



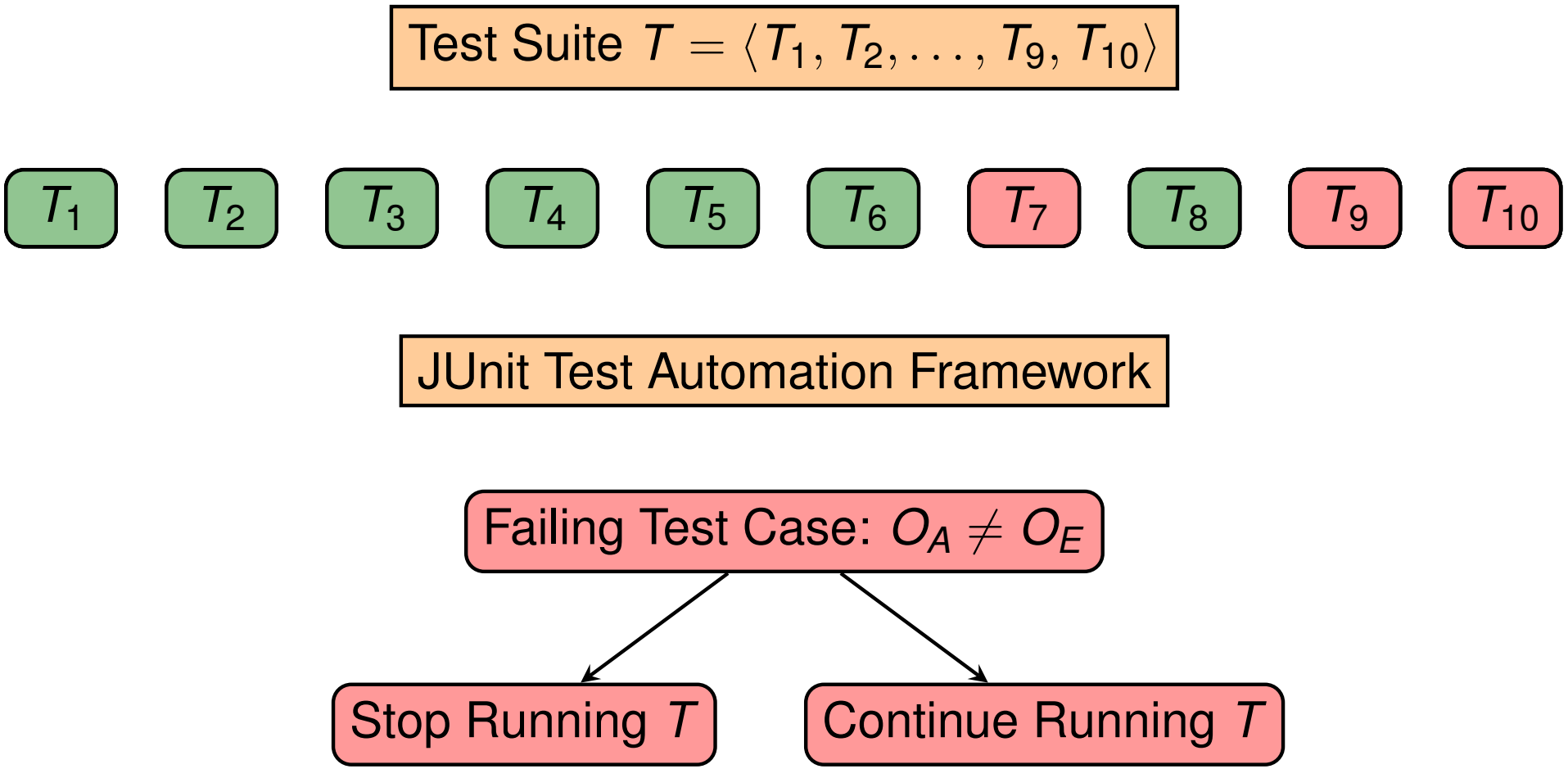
Test Suite Execution



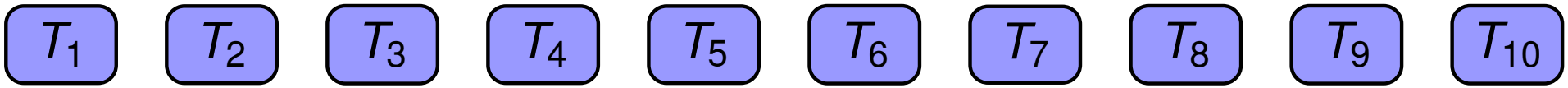
Test Suite Execution



Test Suite Execution



Test Coverage Monitoring



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

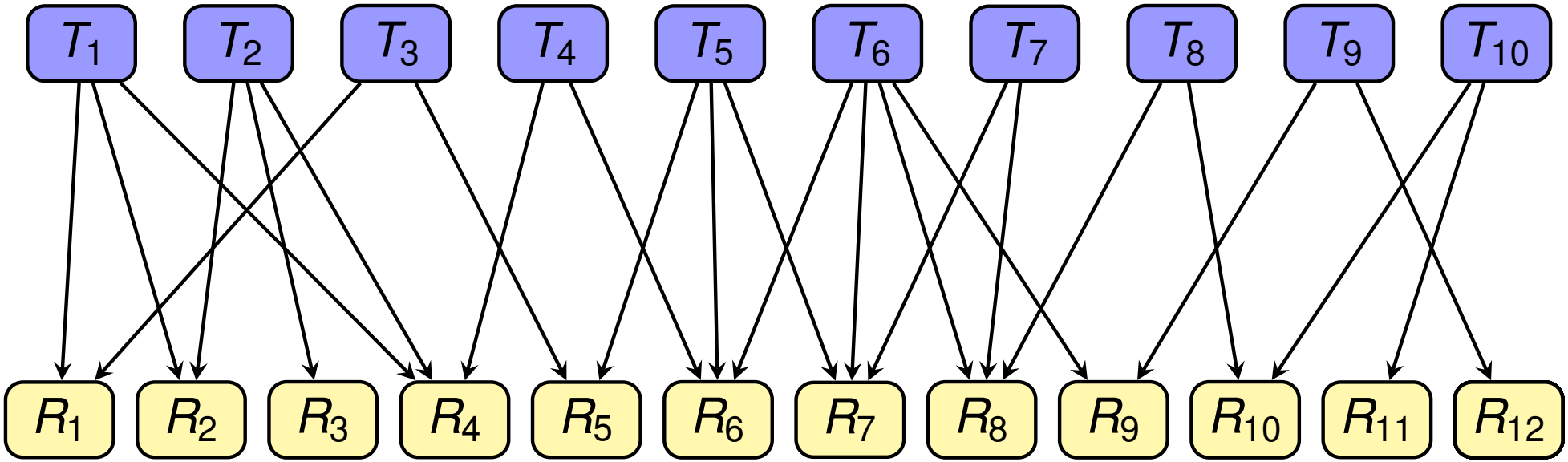
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10}

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

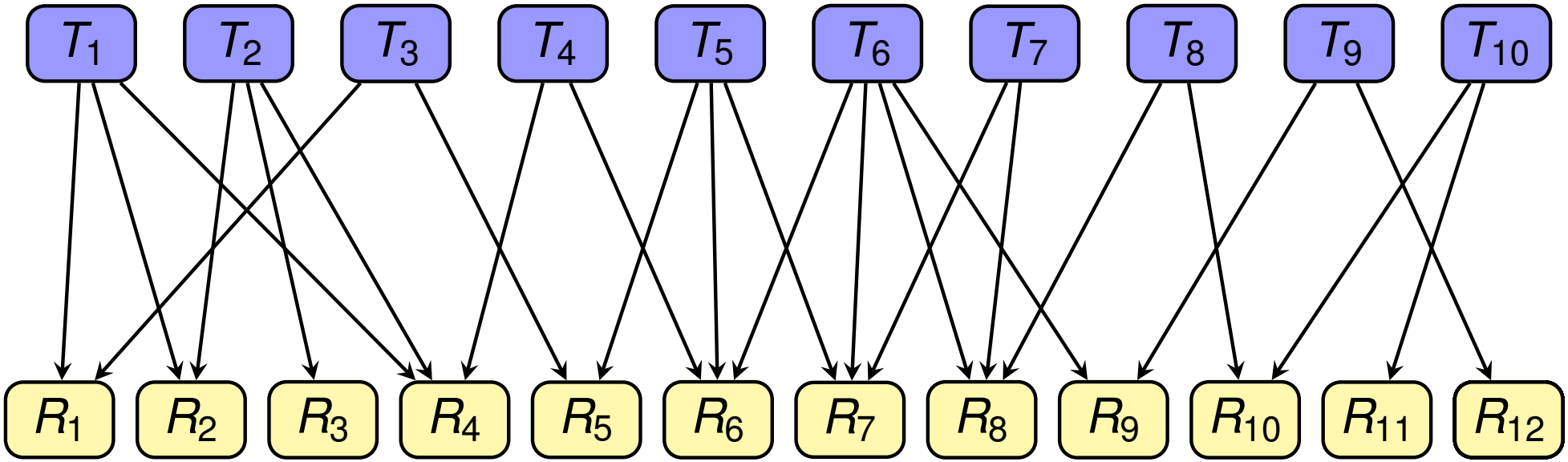
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$



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

Test Coverage Monitoring

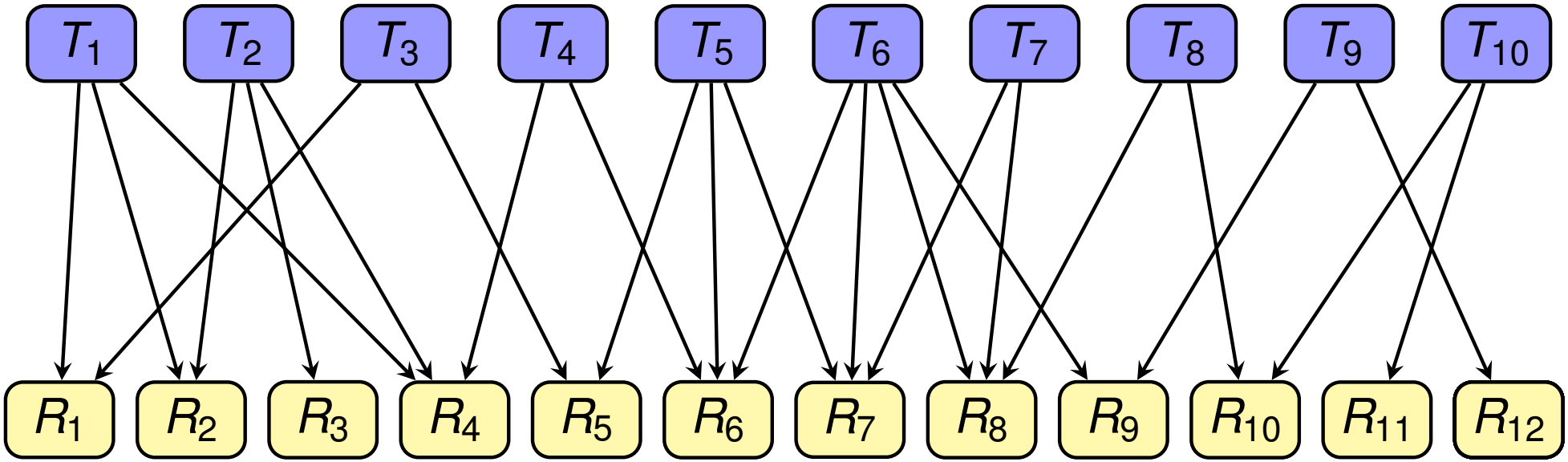
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$



Requirements Set R for ... Statement Coverage

Test Coverage Monitoring

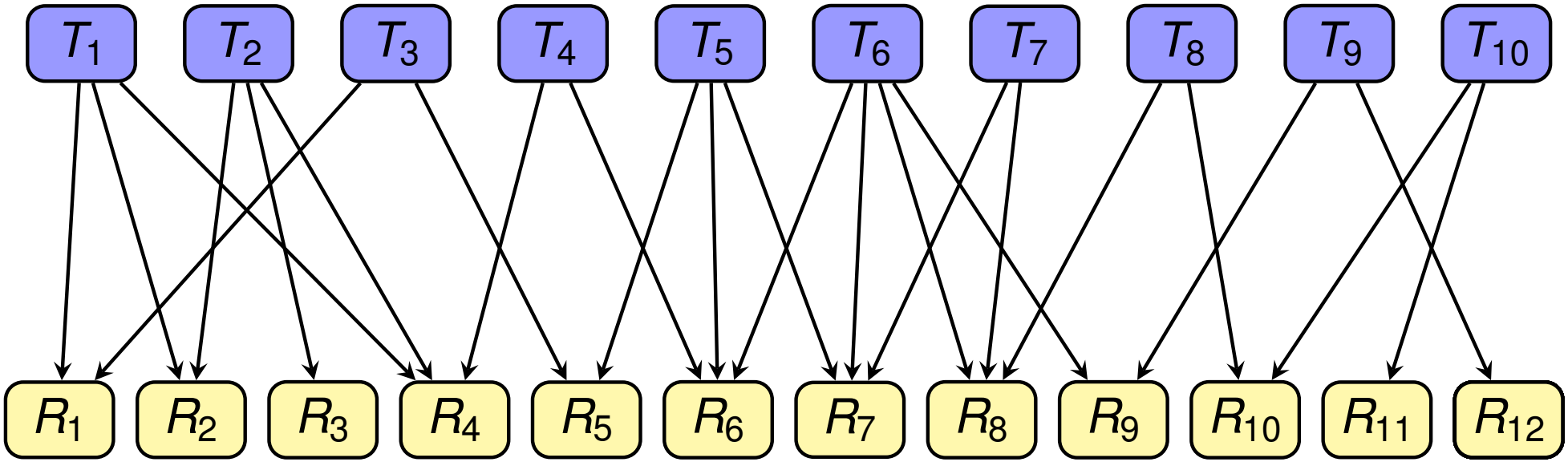
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$



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

Hands-on: Preparing the Applications and Tools

1

Compile the Sudoku Case Study Application

```
cd proteja-installation-directory
cd apps/SK
ant init
ant compile
cd ../../
```

Hands-on: Preparing the Applications and Tools

1

Compile the Sudoku Case Study Application

```
cd proteja-installation-directory
cd apps/SK
ant init
ant compile
cd ../../
```

2

Compile the Proteja Test Coverage Monitor

```
cd ../../
ant init
ant build
```

Hands-on: Executing Tests and Monitoring Coverage

1

Run Sudoku's Test Suite Without Coverage Monitoring

```
cp data/settingsFiles/SK/* .
ant addAnnotations
ant proteja
```

Hands-on: Executing Tests and Monitoring Coverage

1

Run Sudoku's Test Suite Without Coverage Monitoring

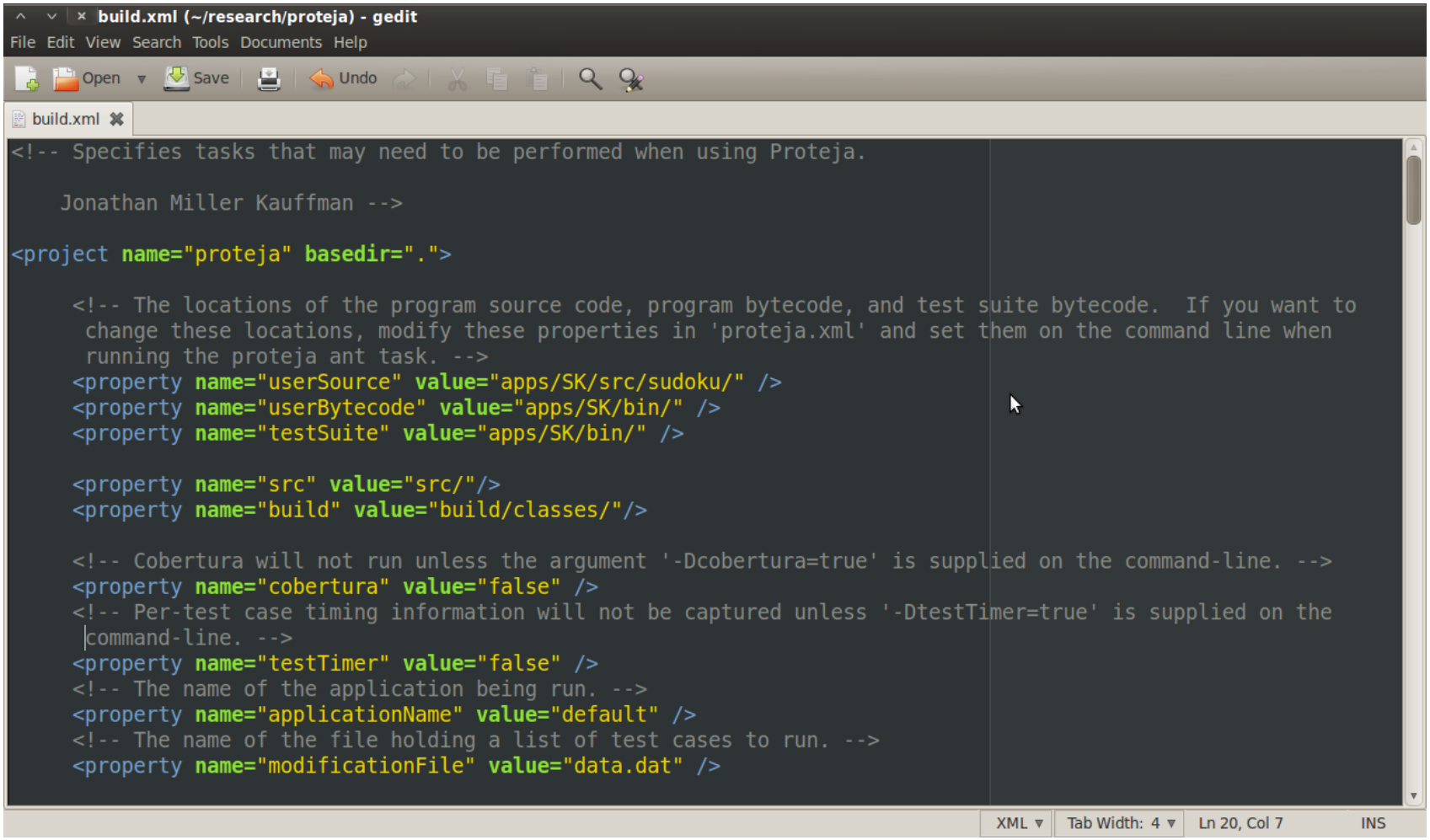
```
cp data/settingsFiles/SK/* .
ant addAnnotations
ant proteja
```

2

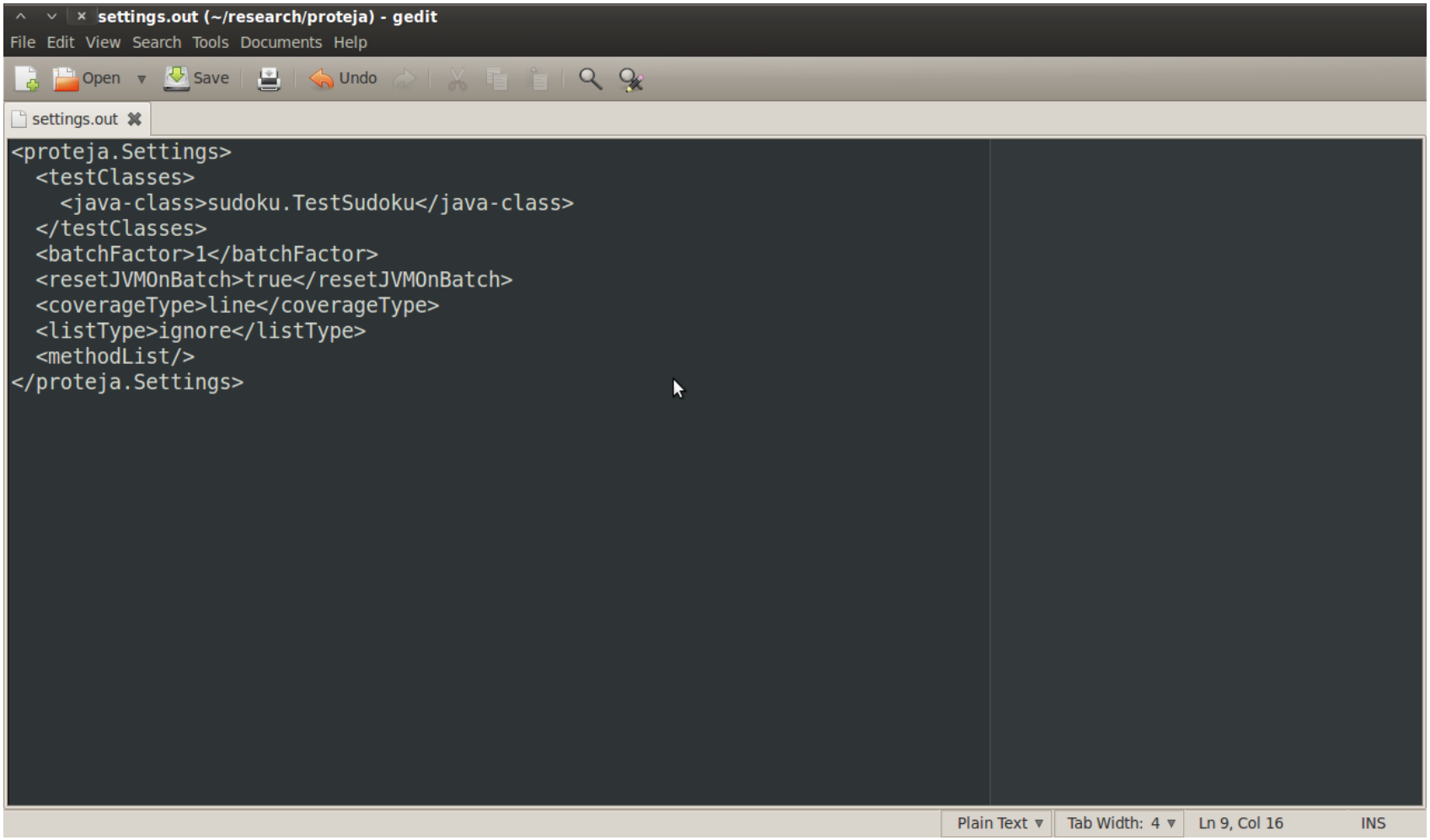
Run Sudoku's Test Suite With Coverage Monitoring

```
ant proteja -Dcobertura=true
-DtestTimer=true -DapplicationName=Sudoku
```

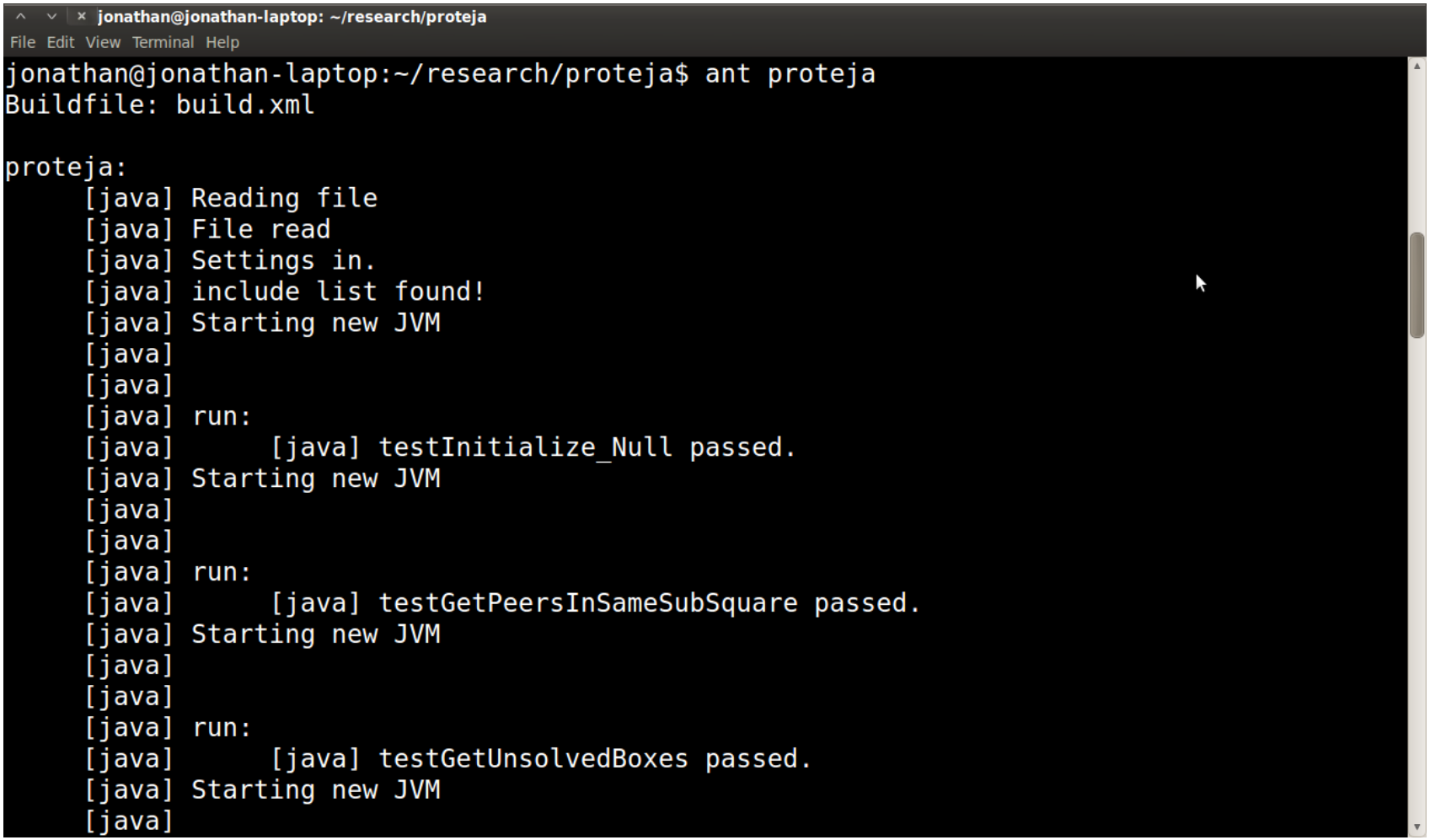
Screenshot: Configuring Test Suite Execution



Screenshot: Configuring the Test Coverage Monitor



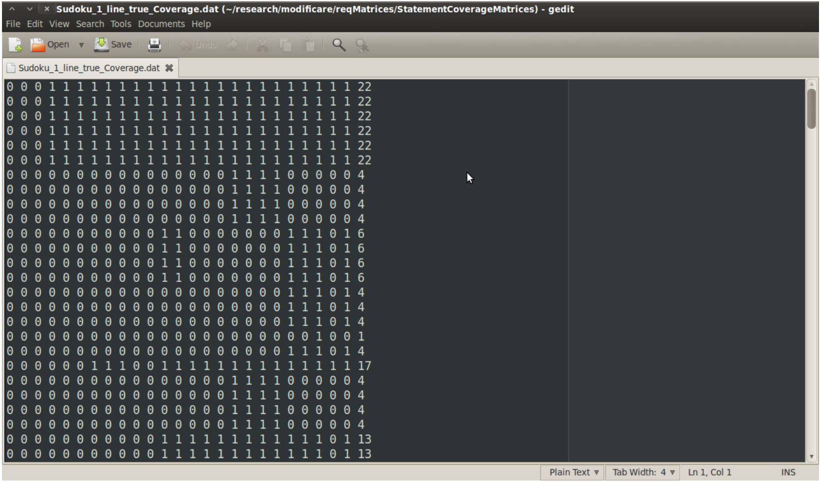
Screenshot: Running the Test Suite



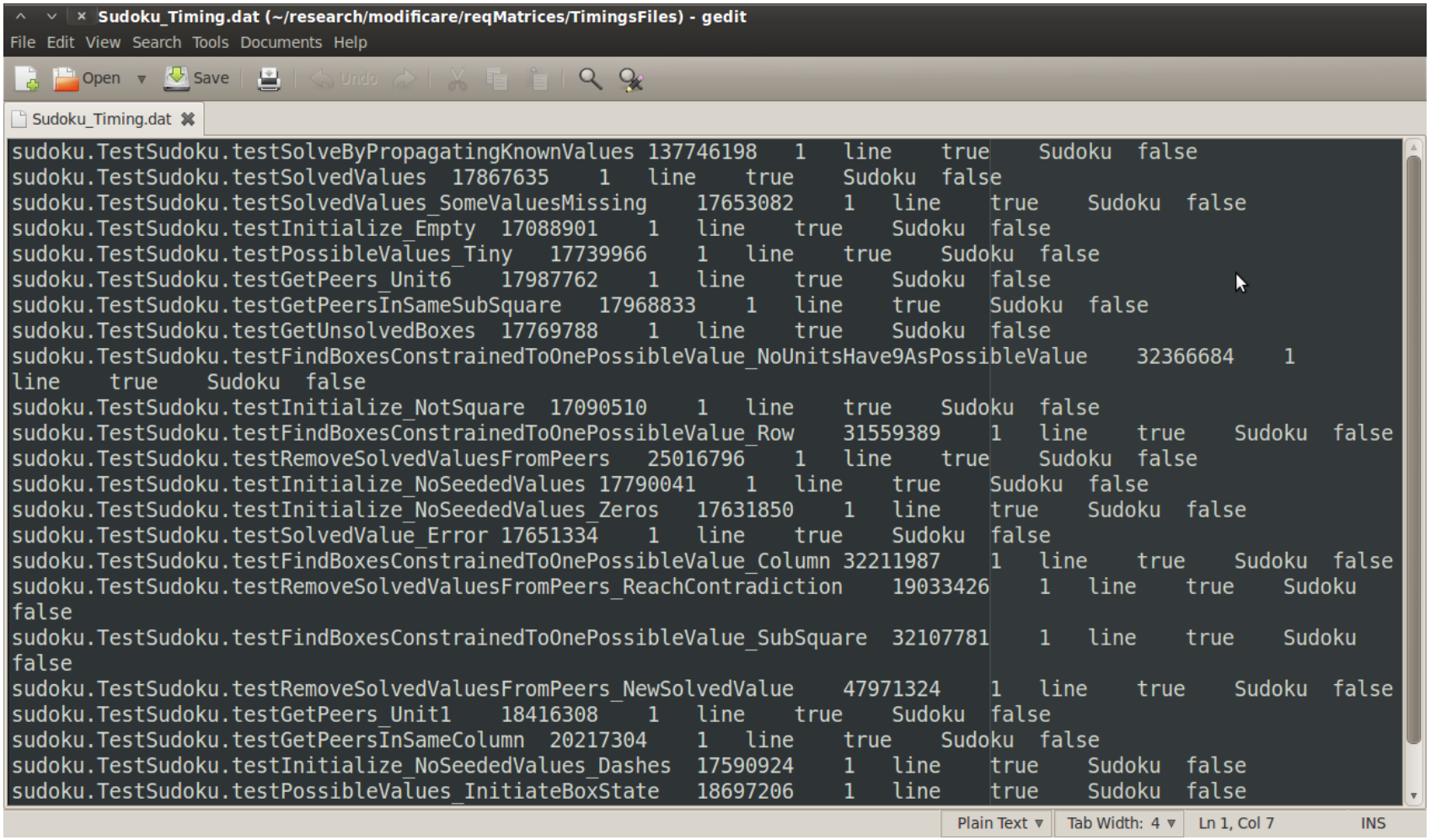
Screenshot: Performing Test Coverage Monitoring

```
jonathan@jonathan-laptop: ~/research/proteja
File Edit View Terminal Help
jonathan@jonathan-laptop:~/research/proteja$ ant proteja -Dcobertura=true -DtestTimer=true
 -DapplicationName=Sudoku
Buildfile: build.xml

proteja:
     [java] Reading file
     [java] File read
     [java] Settings in.
     [java] include list found!
     [java]
     [java] inst:
     [java]     [mkdir] Created dir: /home/jonathan/research/proteja/instrumentedClasses
     [java] [cobertura-instrument] Cobertura 1.9.1 - GNU GPL License (NO WARRANTY) - See C
OPYRIGHT file
     [java] [cobertura-instrument] Instrumenting 4 files to /home/jonathan/research/protej
a/instrumentedClasses
     [java] [cobertura-instrument] Cobertura: Saved information on 4 classes.
     [java] [cobertura-instrument] Instrument time: 192ms
     [java]
     [java] createCopy:
     [java]     [mkdir] Created dir: /home/jonathan/research/proteja/temp
     [java]      [copy] Copying 1 file to /home/jonathan/research/proteja/temp
     [java] Starting new JVM
     [java]
     [java]
```

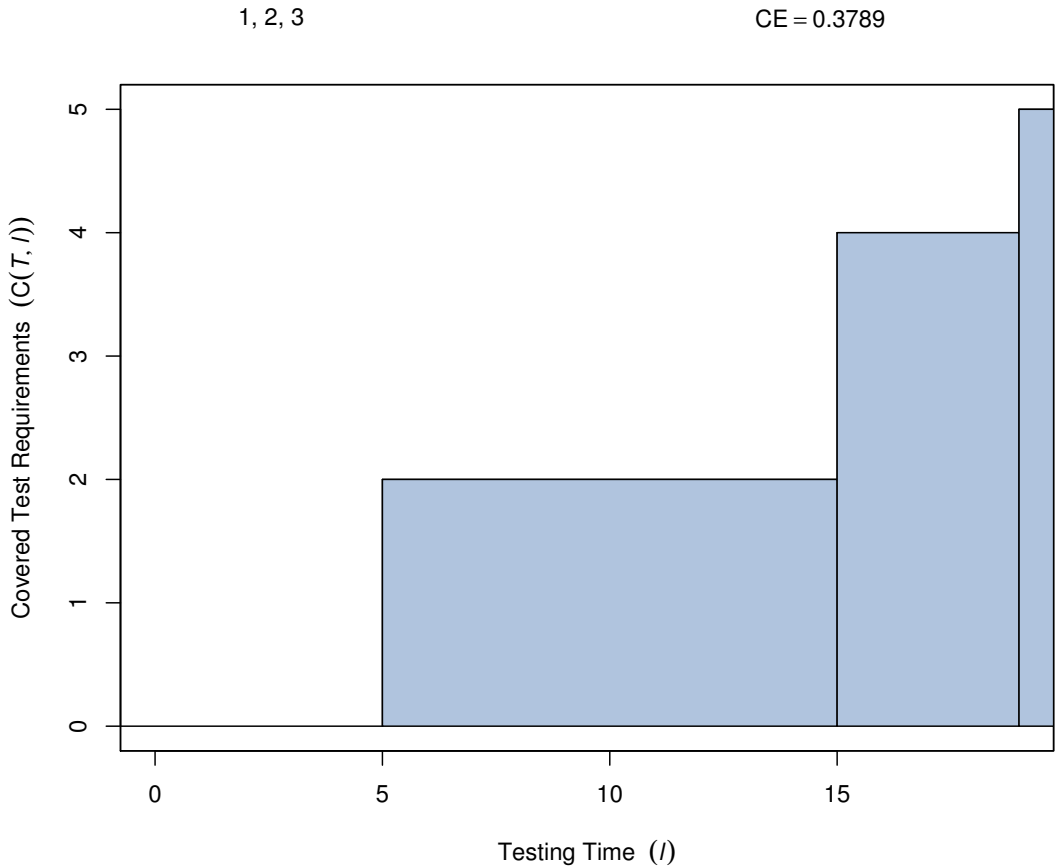
Screenshot: Viewing a Test Coverage Report



Screenshot: Viewing a Test Case Timings Report

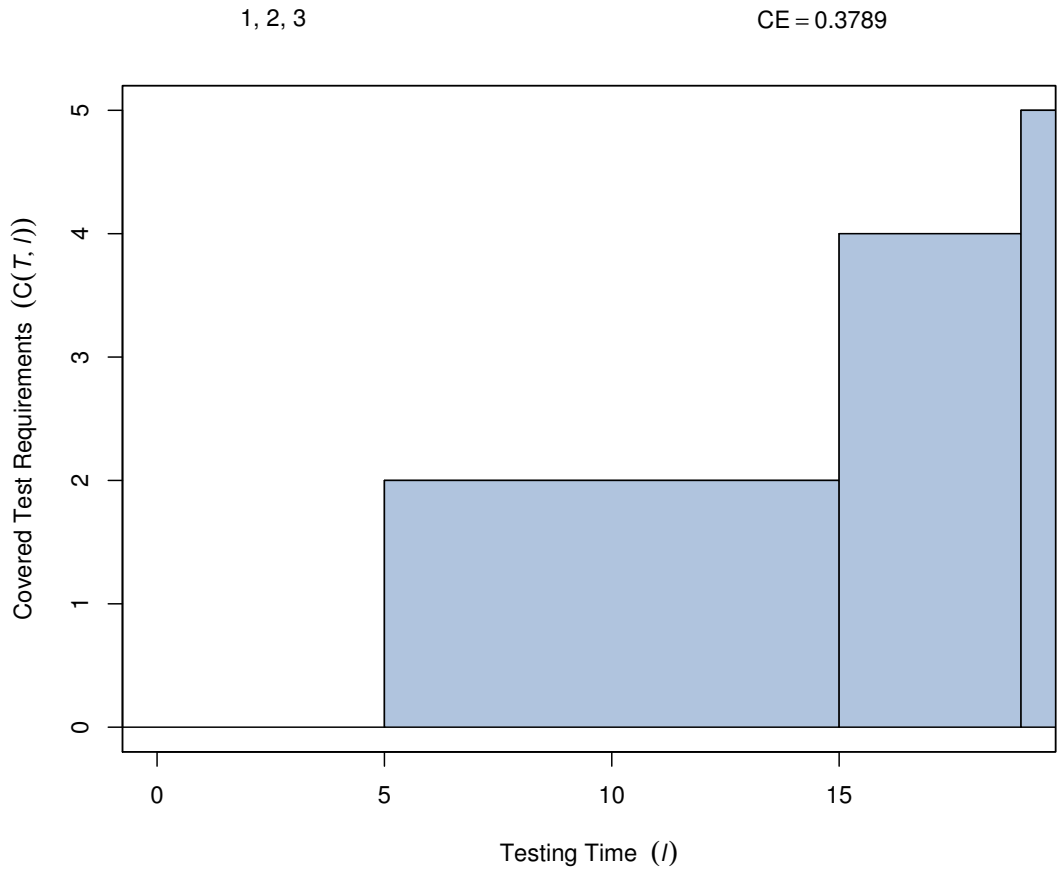


Importance of Test Suite Prioritization



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

Importance of Test Suite Prioritization

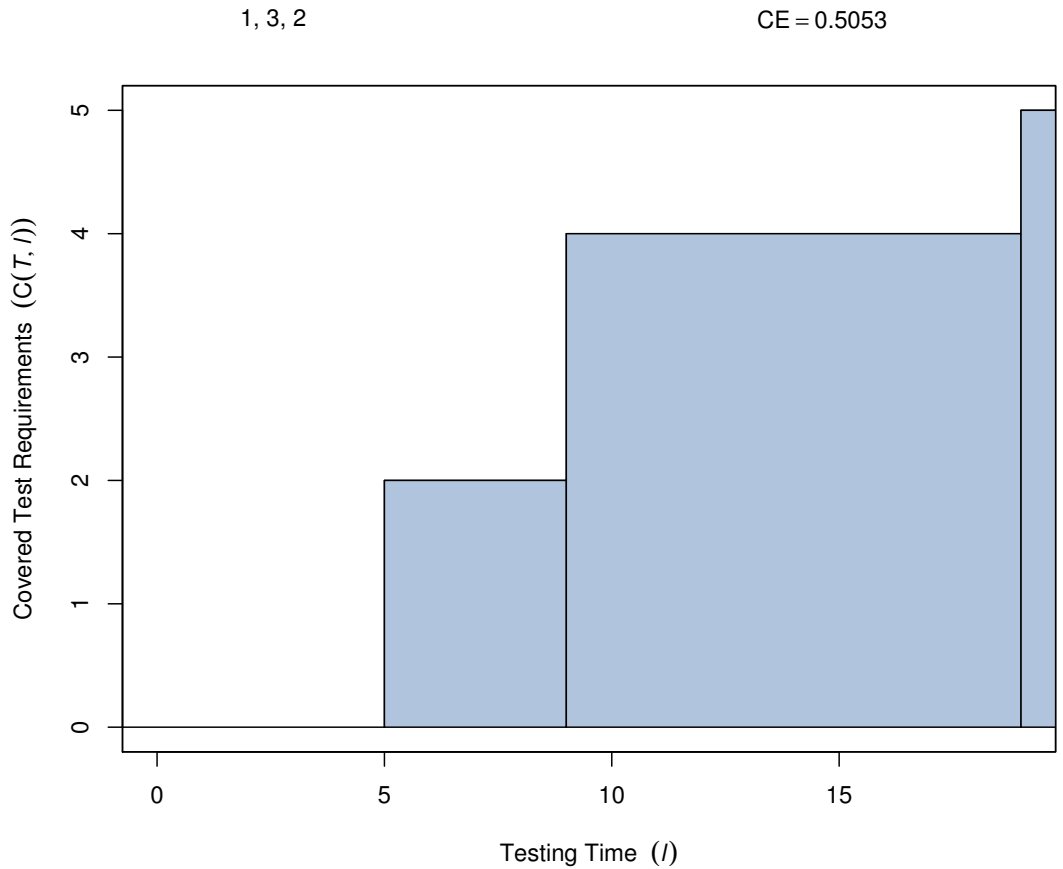


Test Orderings

1,2,3

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

Importance of Test Suite Prioritization



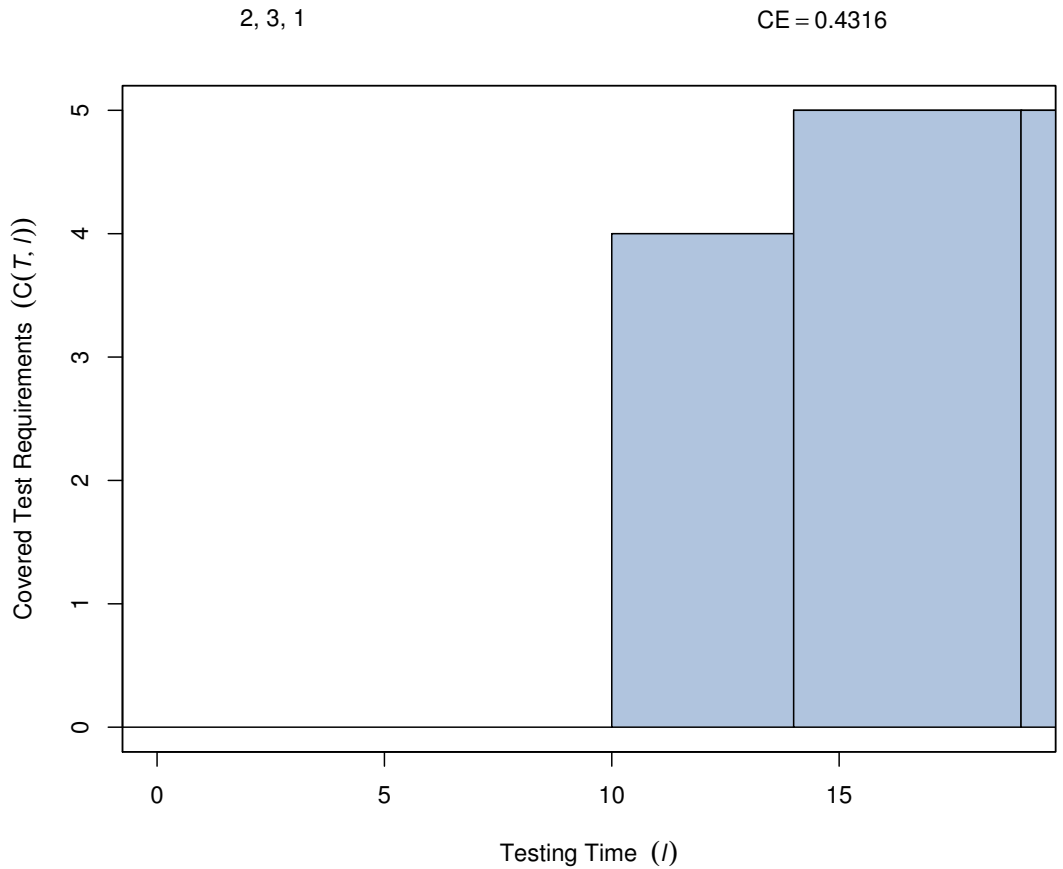
Test Orderings

1,2,3

1,3,2

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

Importance of Test Suite Prioritization



Test Orderings

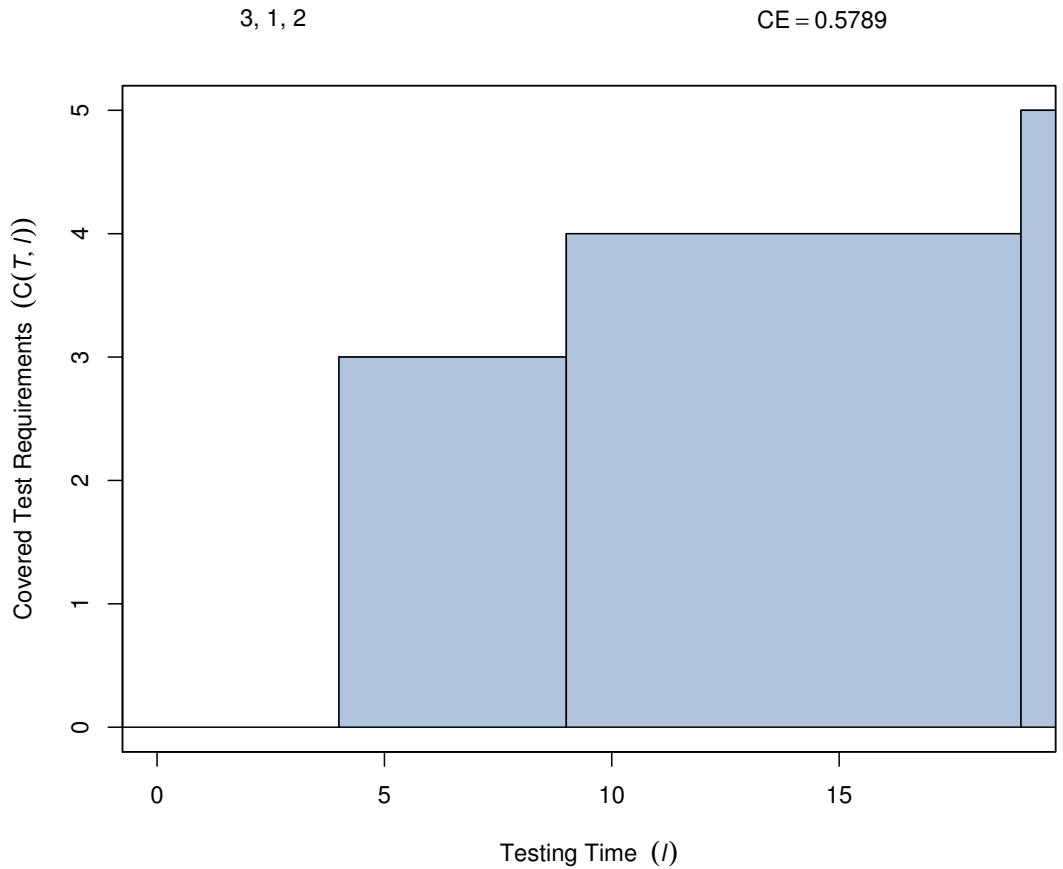
1,2,3

1,3,2

2,3,1

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

Importance of Test Suite Prioritization

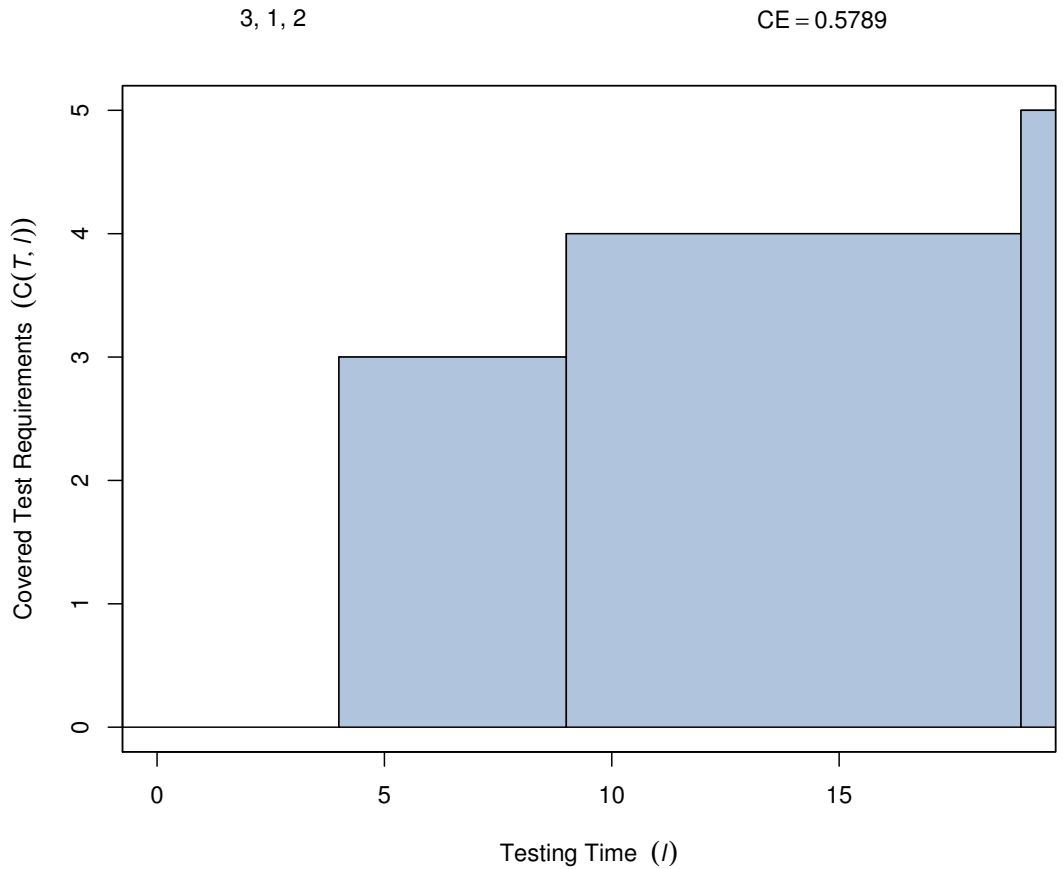


Test Orderings

- 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

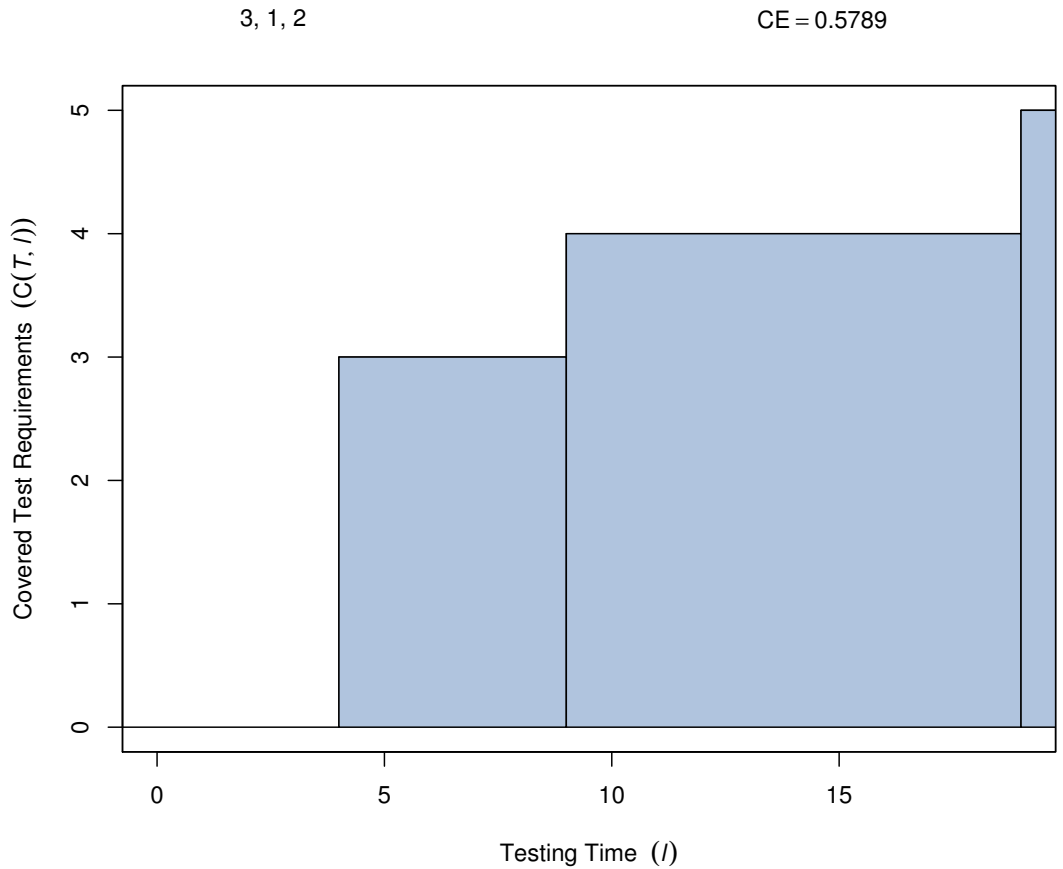
1,3,2

2,3,1

3,1,2

Greedy methods often produce high-effectiveness orderings

Importance of Test Suite Prioritization



Test Orderings

1,2,3

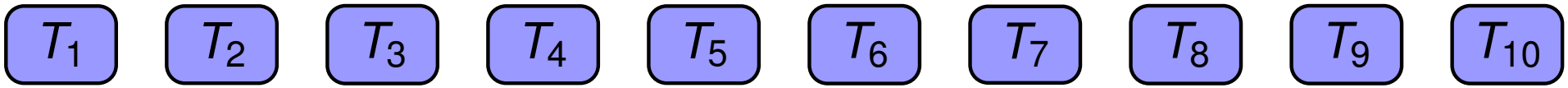
1,3,2

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

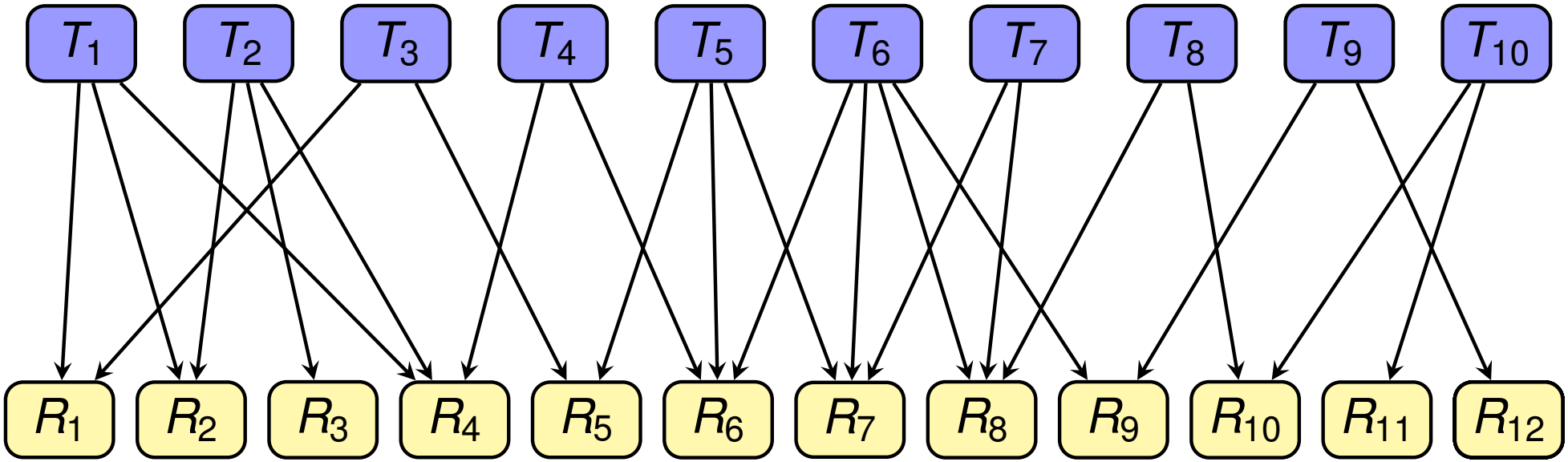
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10}

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

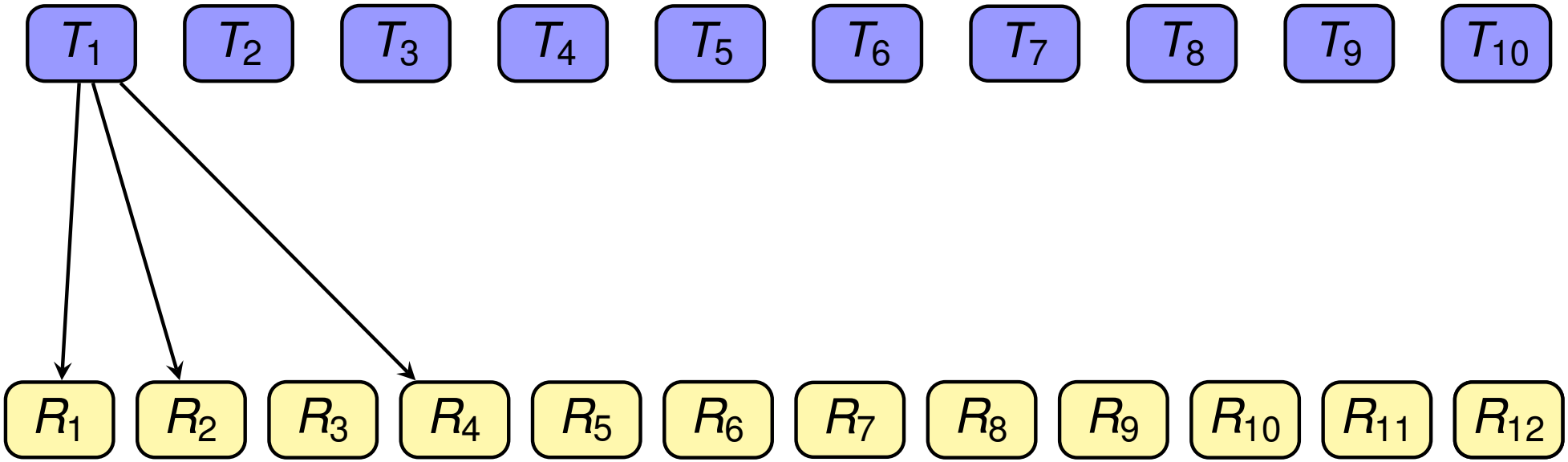
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$



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

Greedy Algorithms

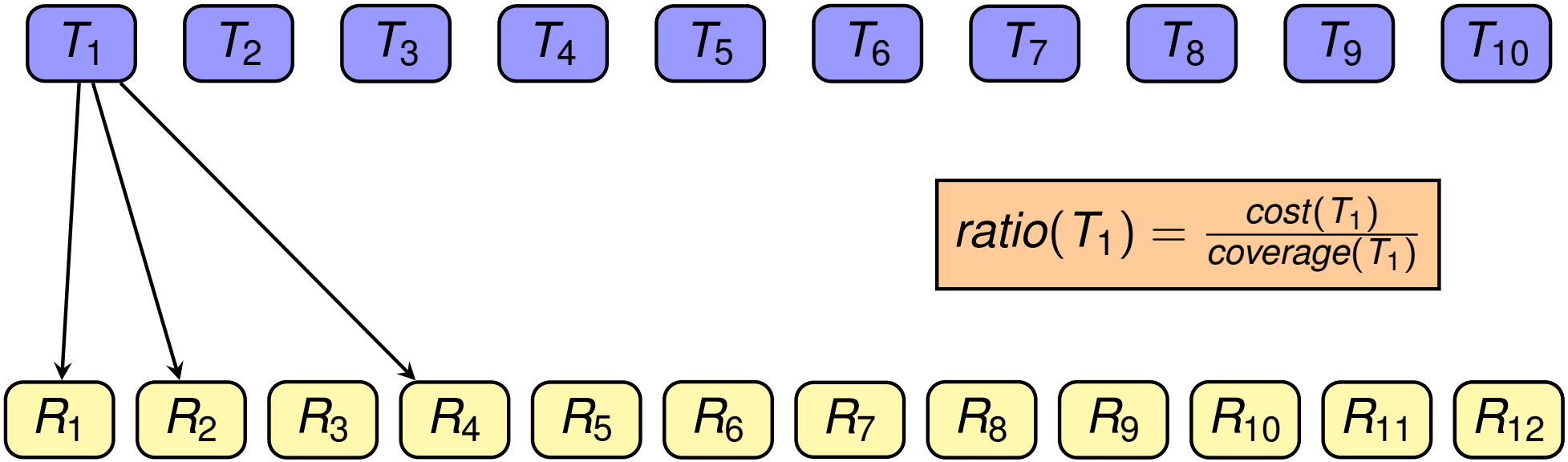
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$



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

Greedy Algorithms

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

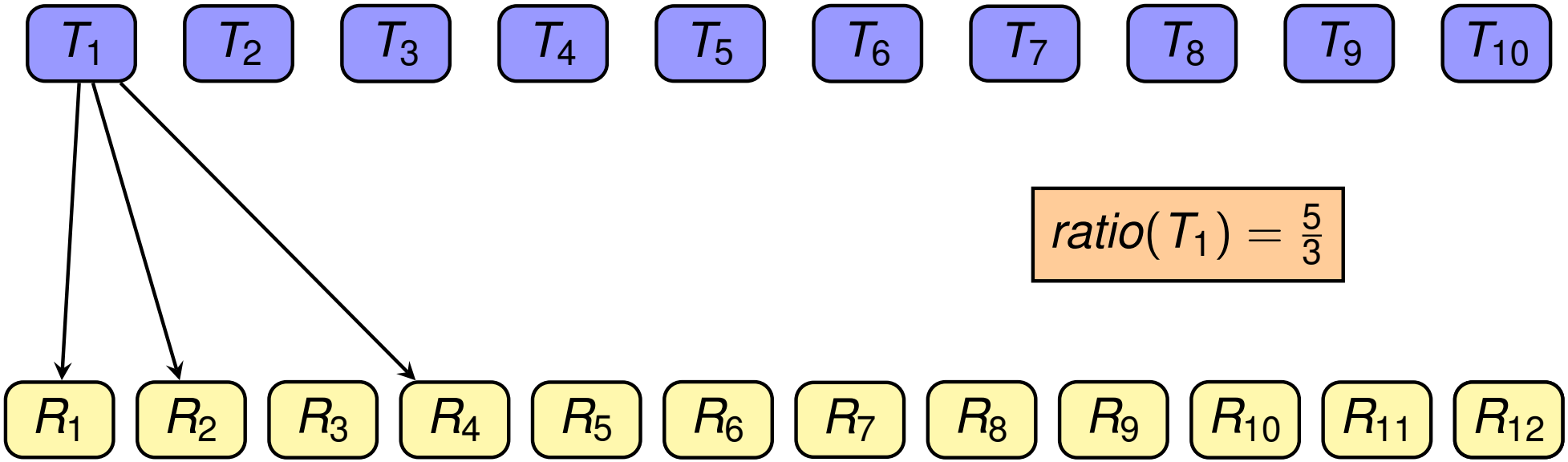


$$ratio(T_1) = \frac{cost(T_1)}{coverage(T_1)}$$

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

Greedy Algorithms

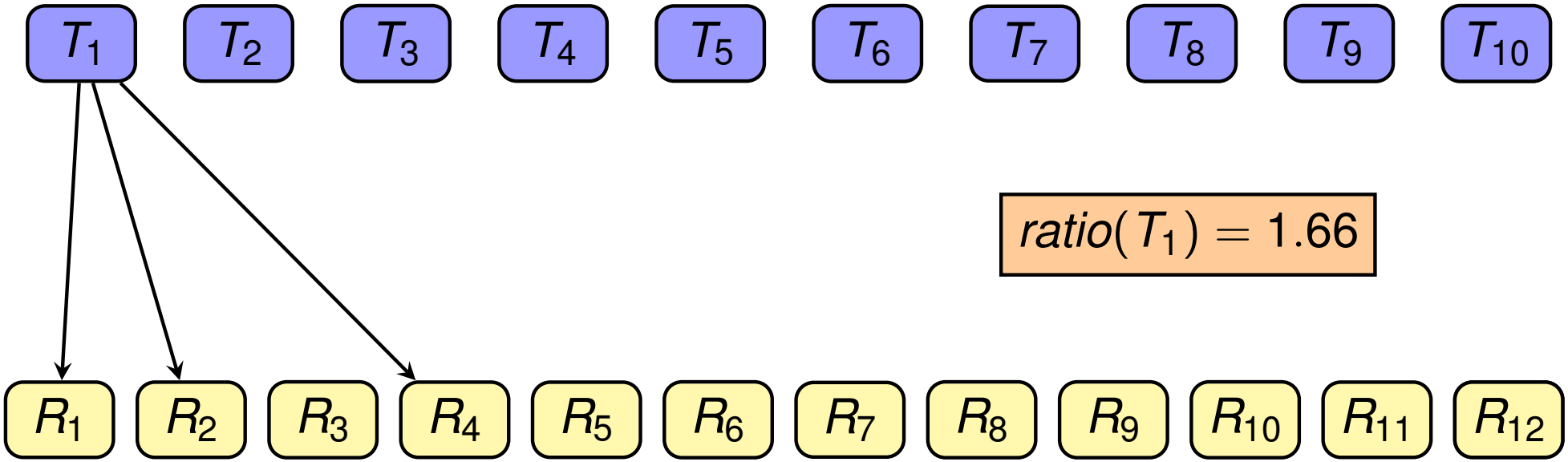
Test Suite $T = \langle T_1, T_2, \ldots, T_9, T_{10} \rangle$



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

Greedy Algorithms

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

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

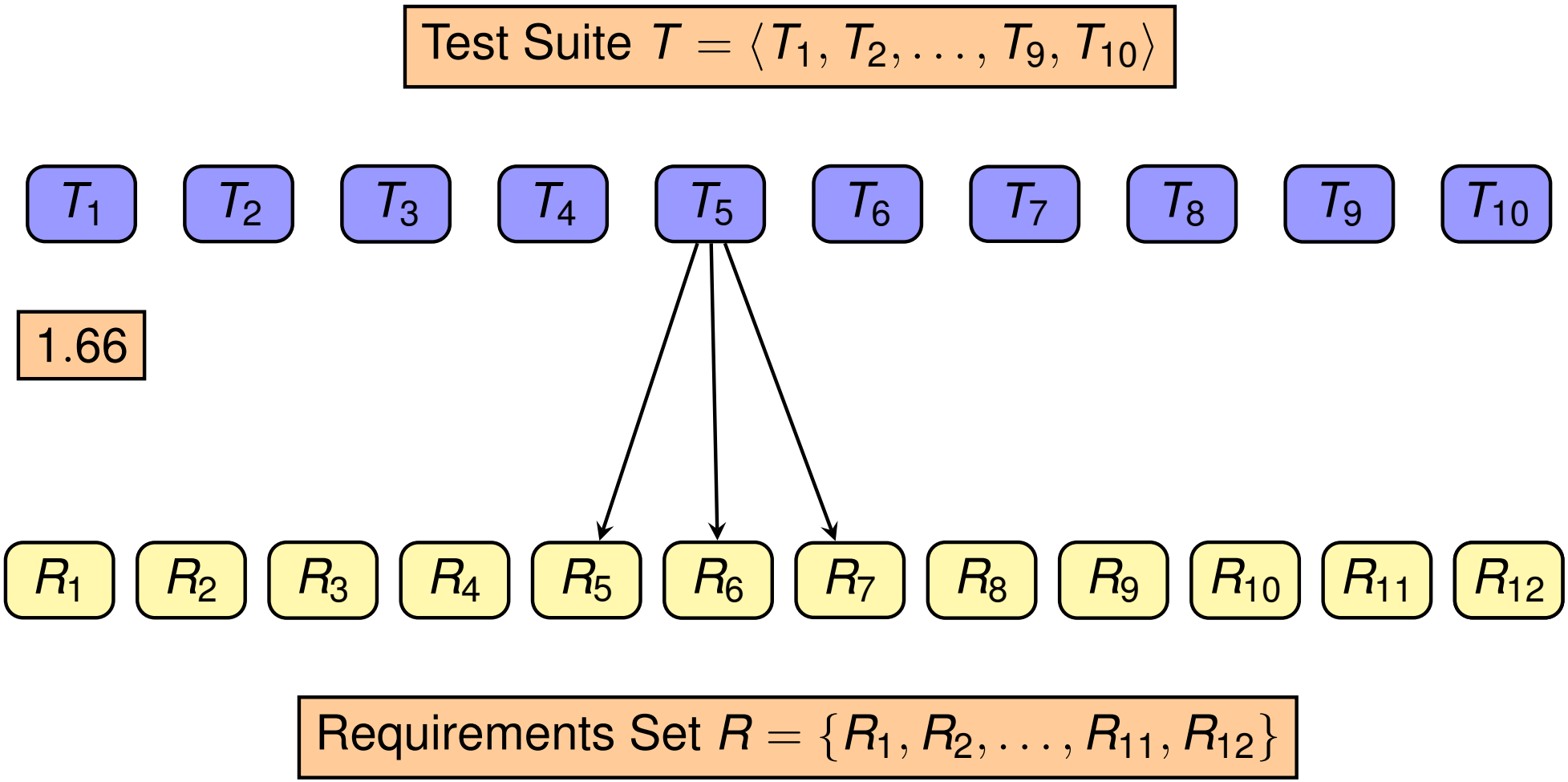
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10}

1.66

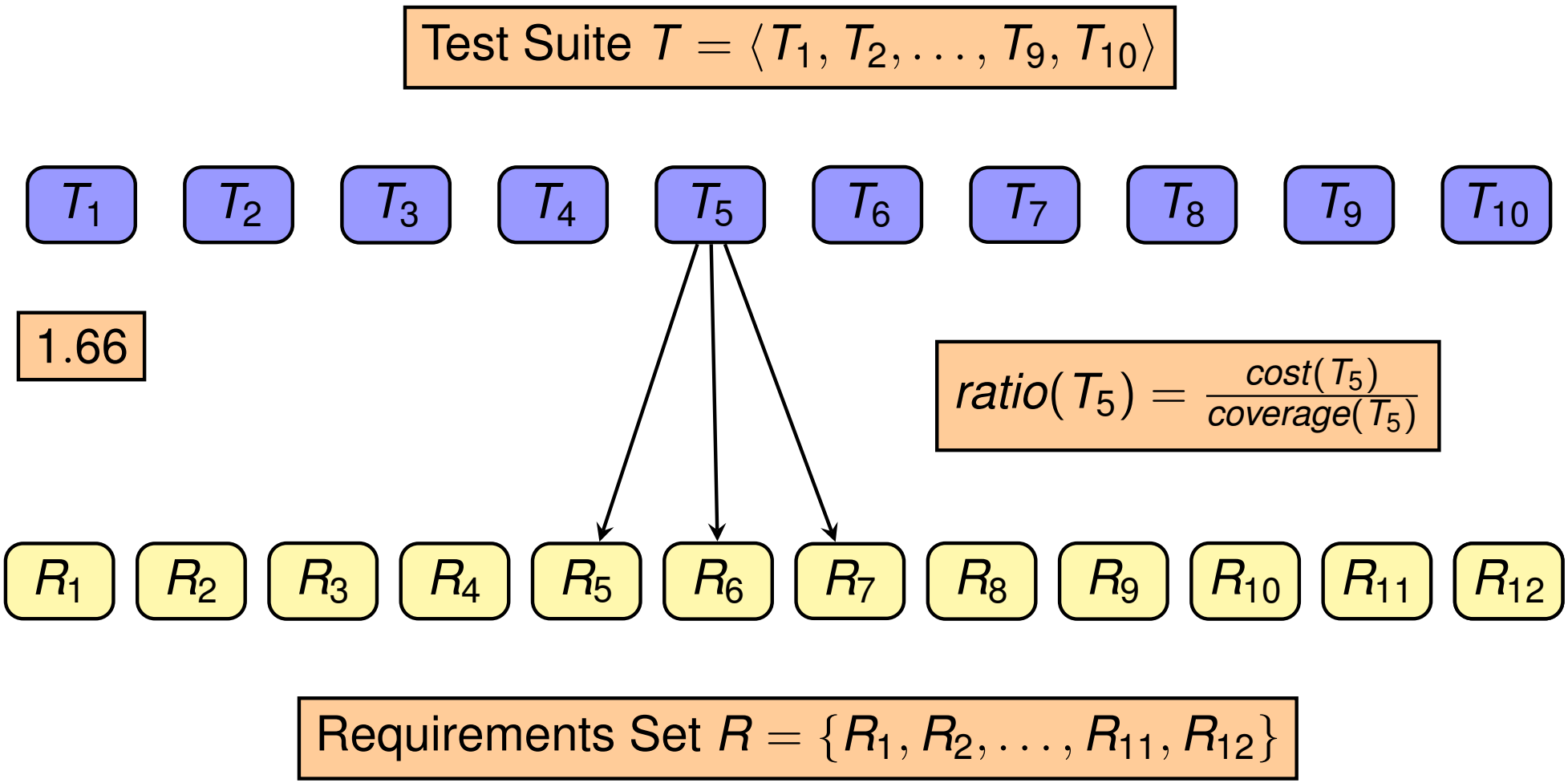
R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9 R_{10} R_{11} R_{12}

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

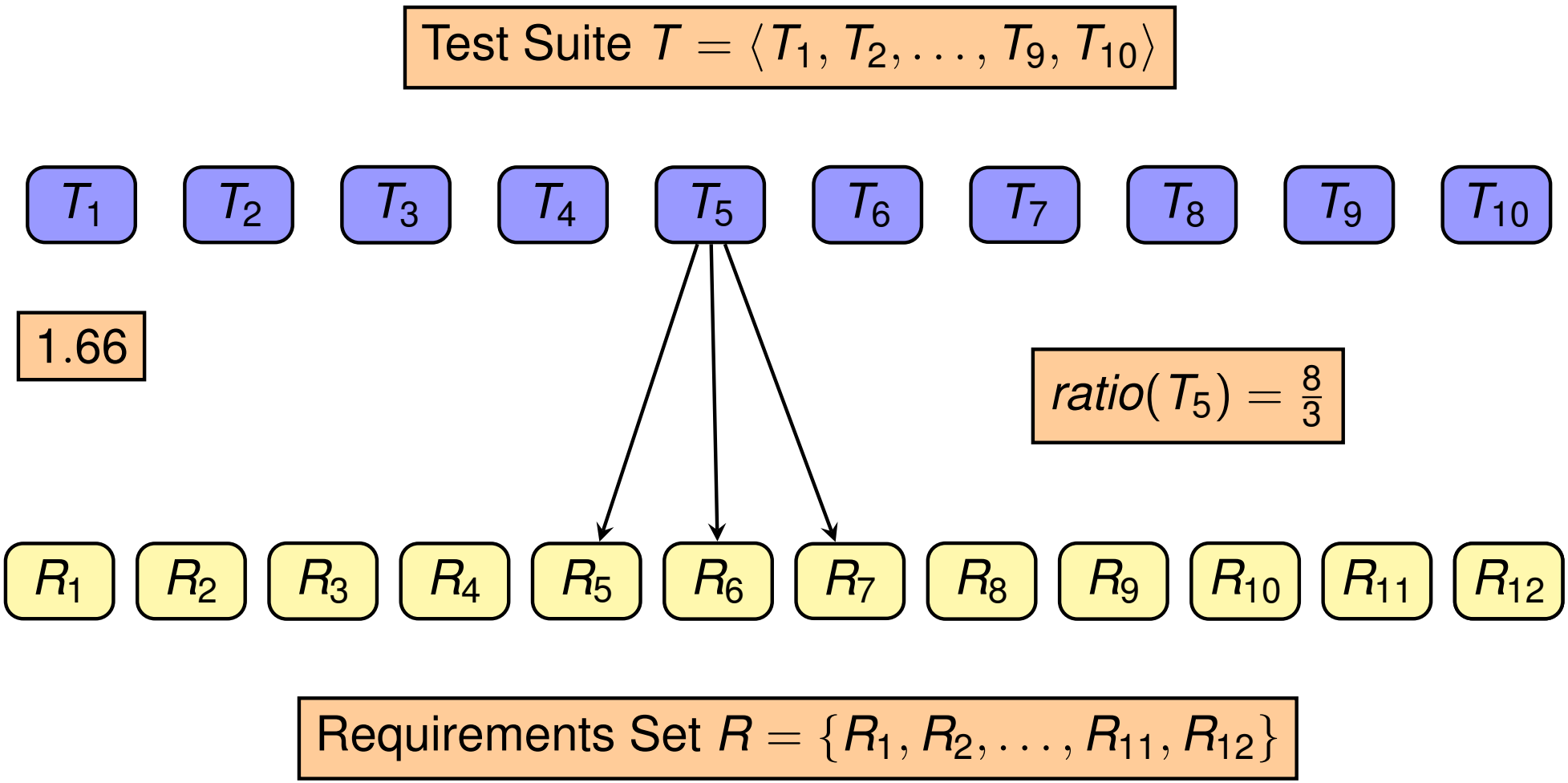
Greedy Algorithms



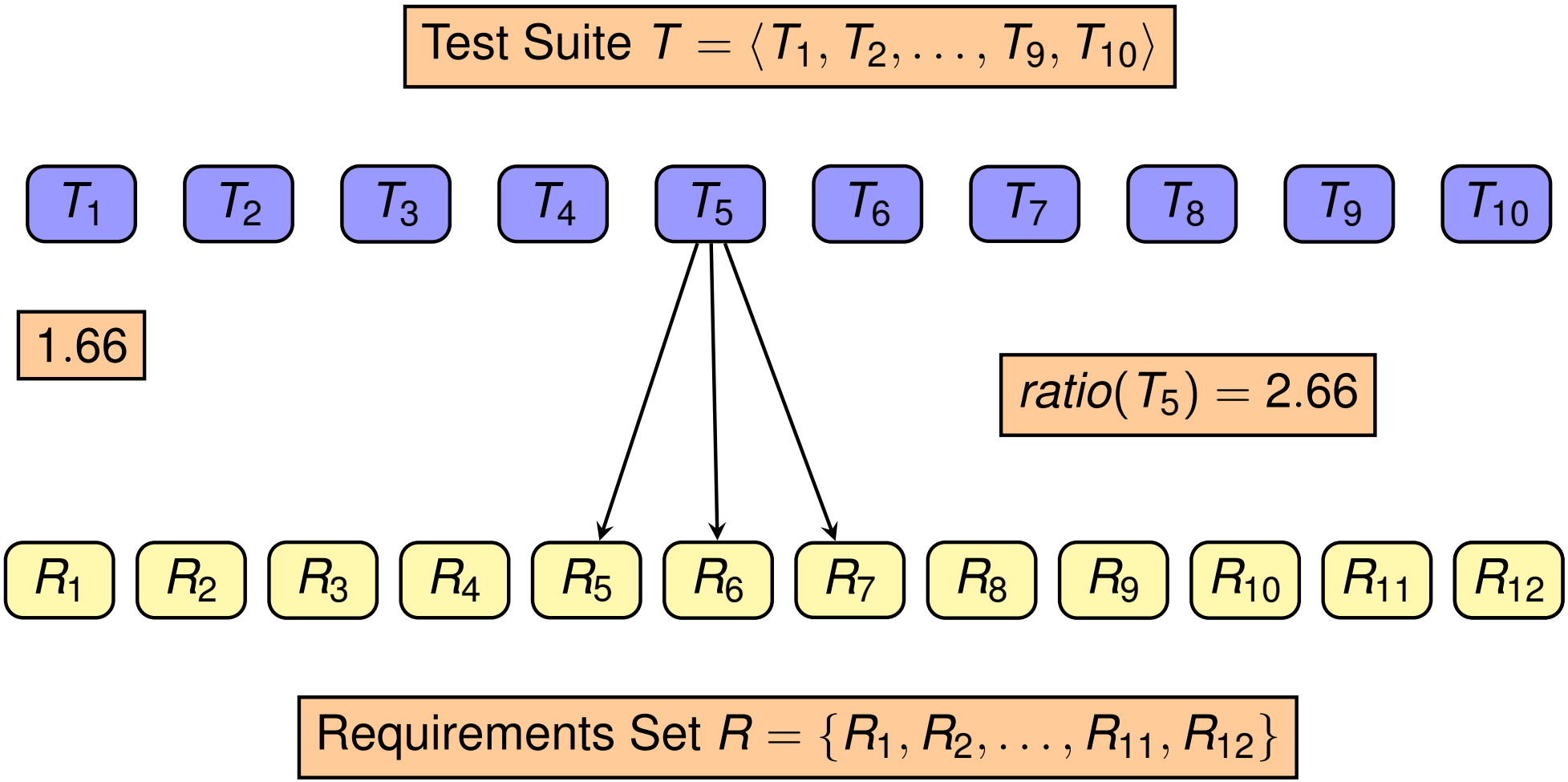
Greedy Algorithms



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

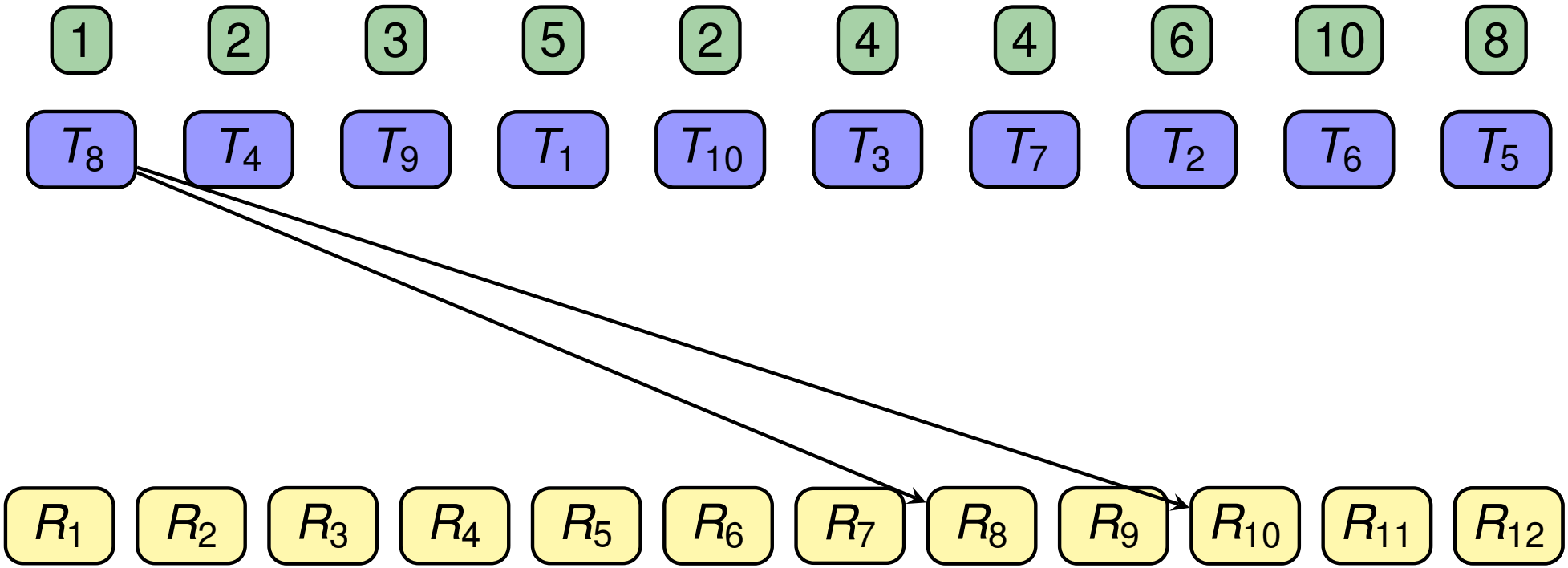
Test Suite $T = \langle T_8, T_4, T_9, T_1, T_{10}, T_3, T_7, T_2, T_6, T_5 \rangle$

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}

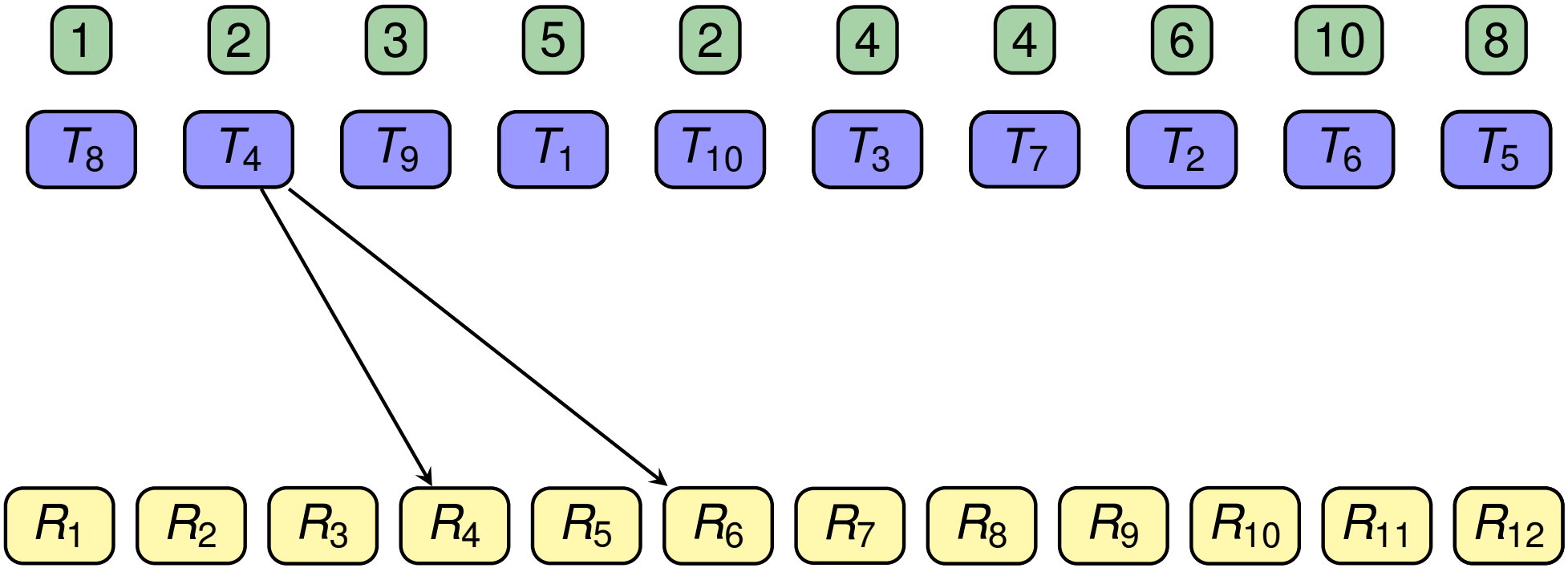
Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

Greedy Algorithms



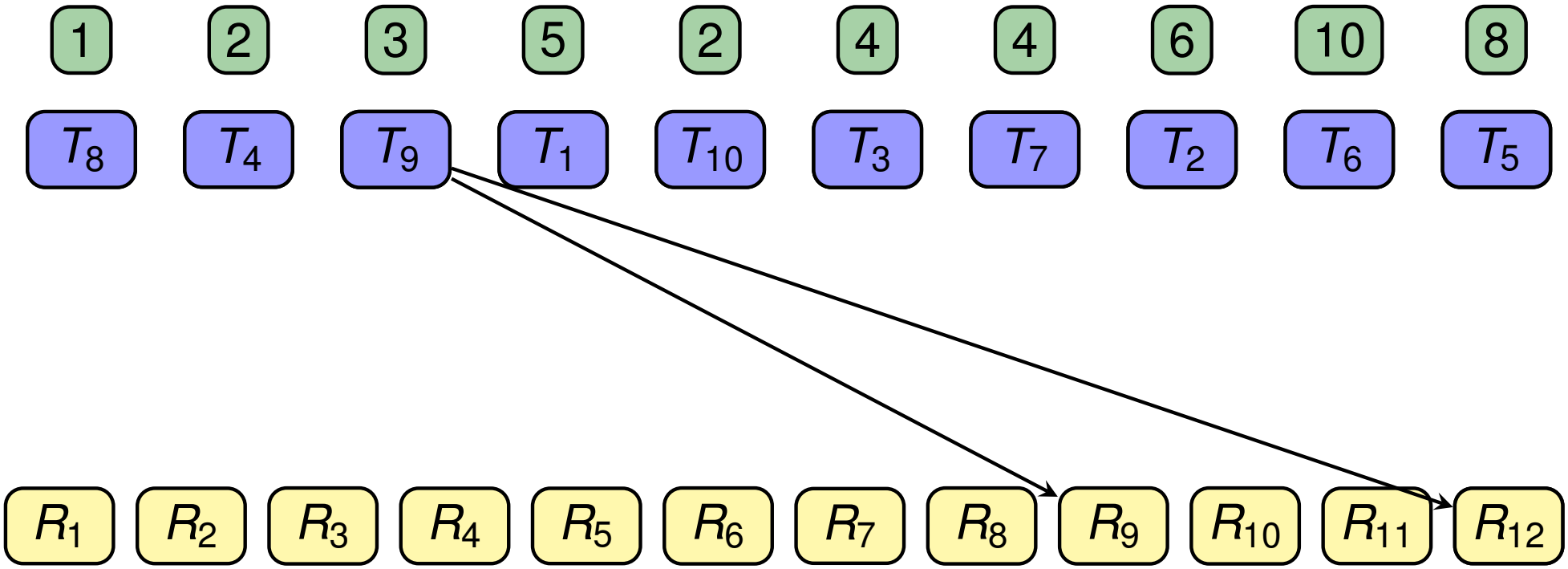
Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

Greedy Algorithms



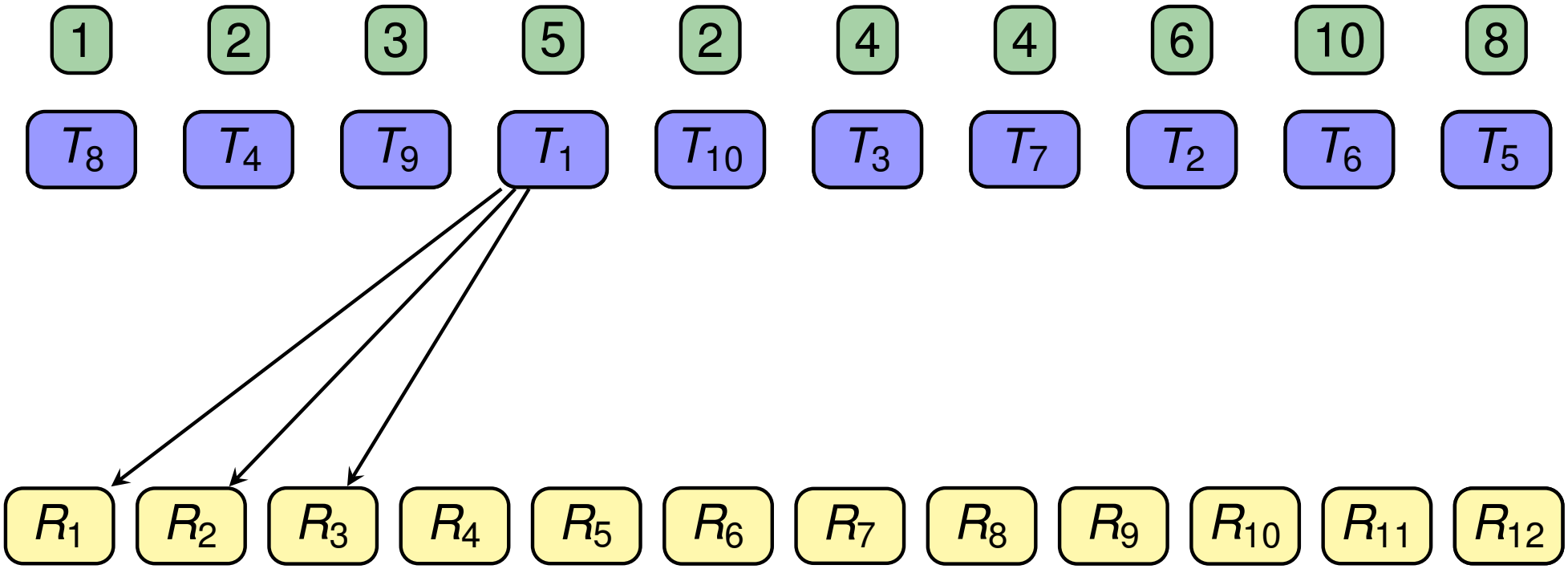
Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

Greedy Algorithms



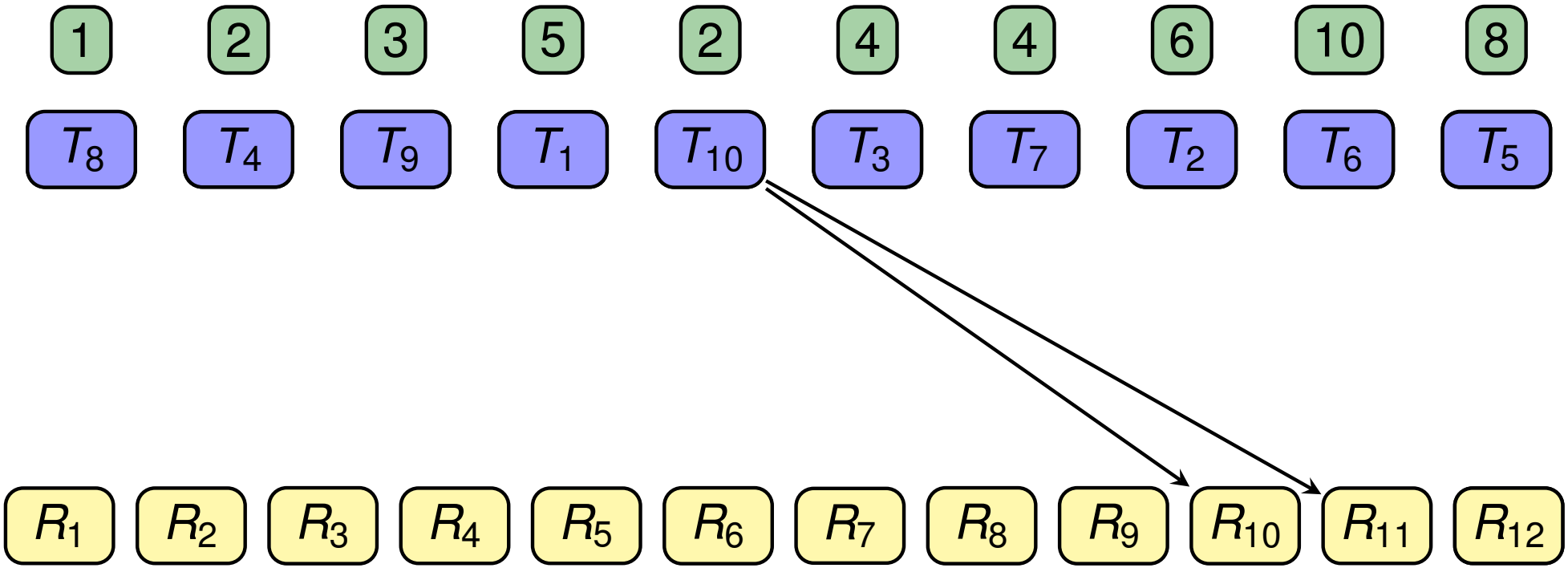
Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

Greedy Algorithms



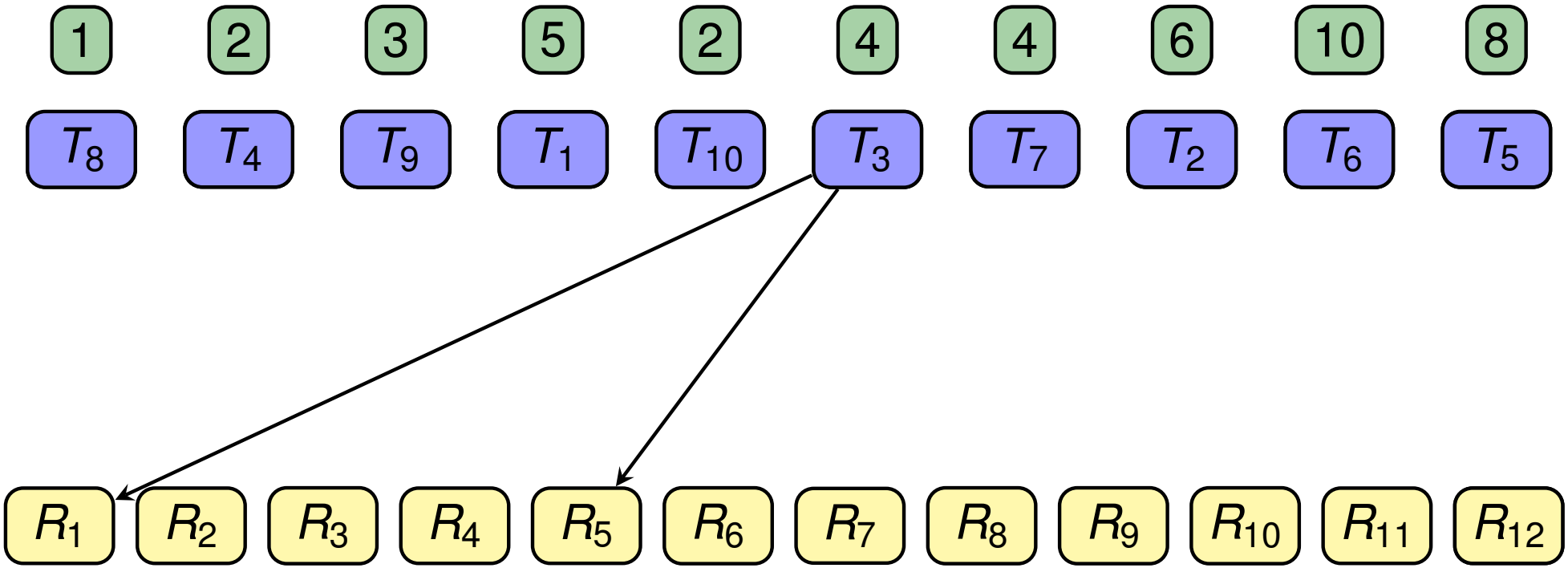
Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

Greedy Algorithms



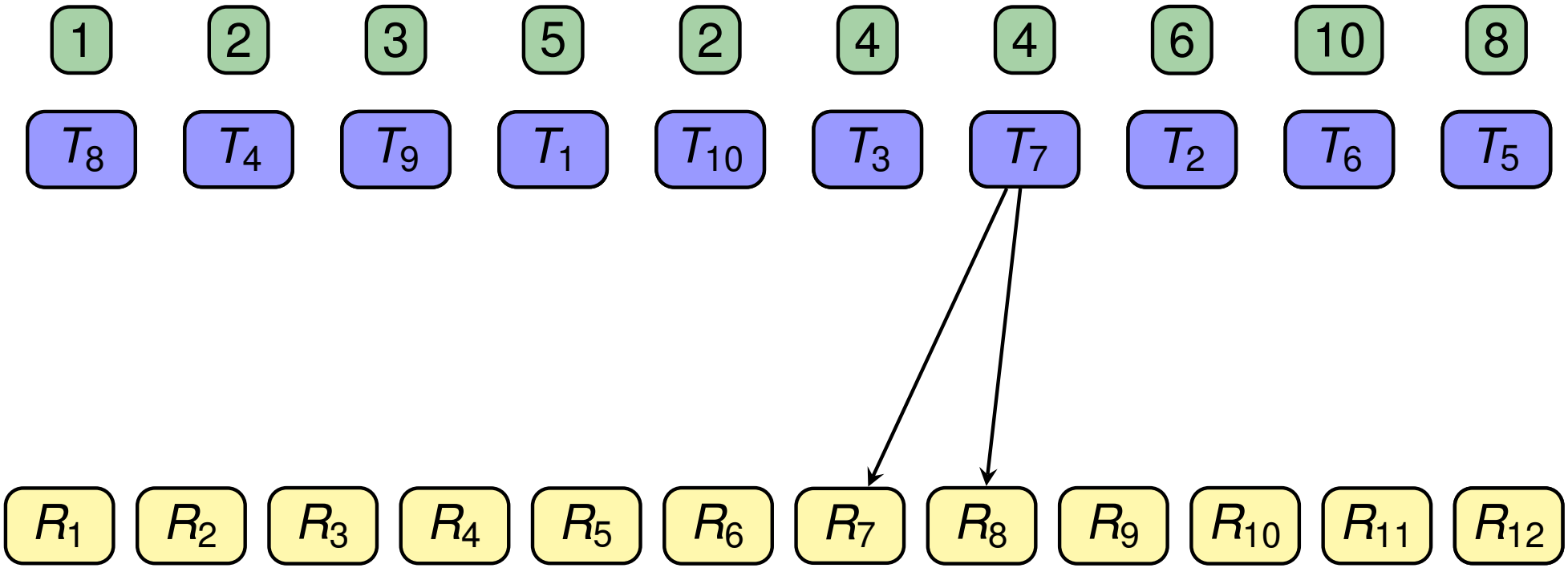
Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

Greedy Algorithms



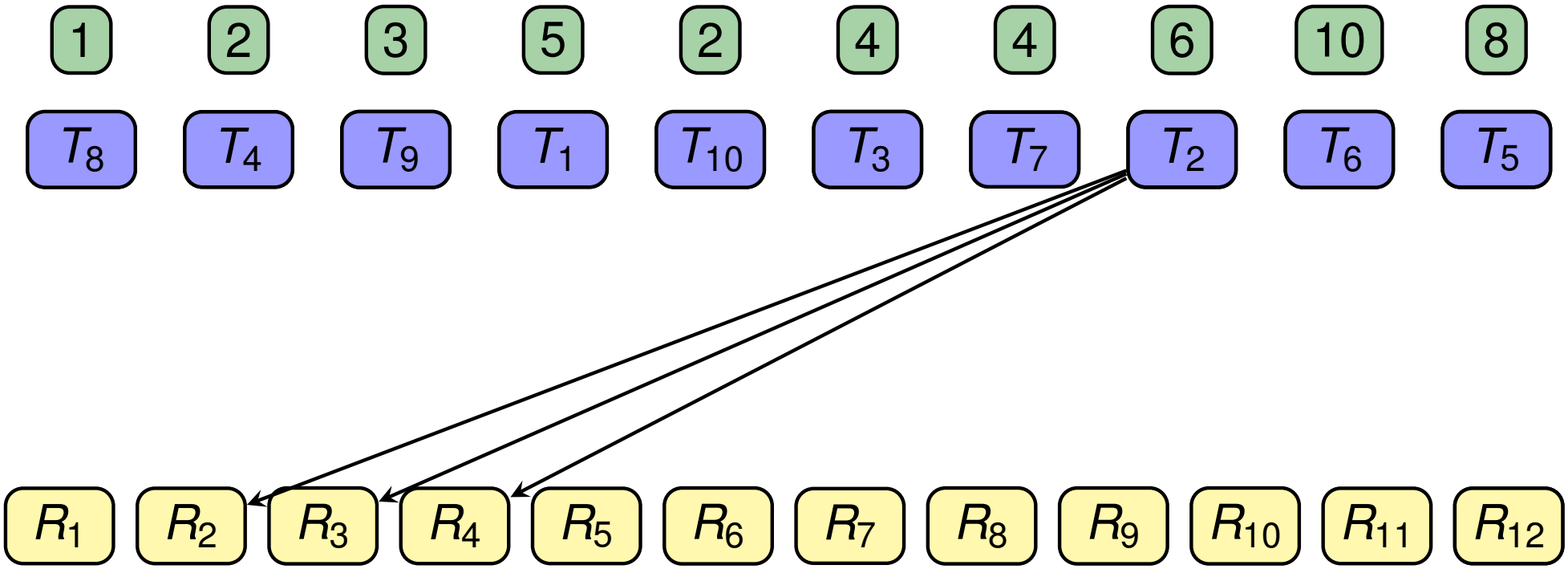
Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

Greedy Algorithms



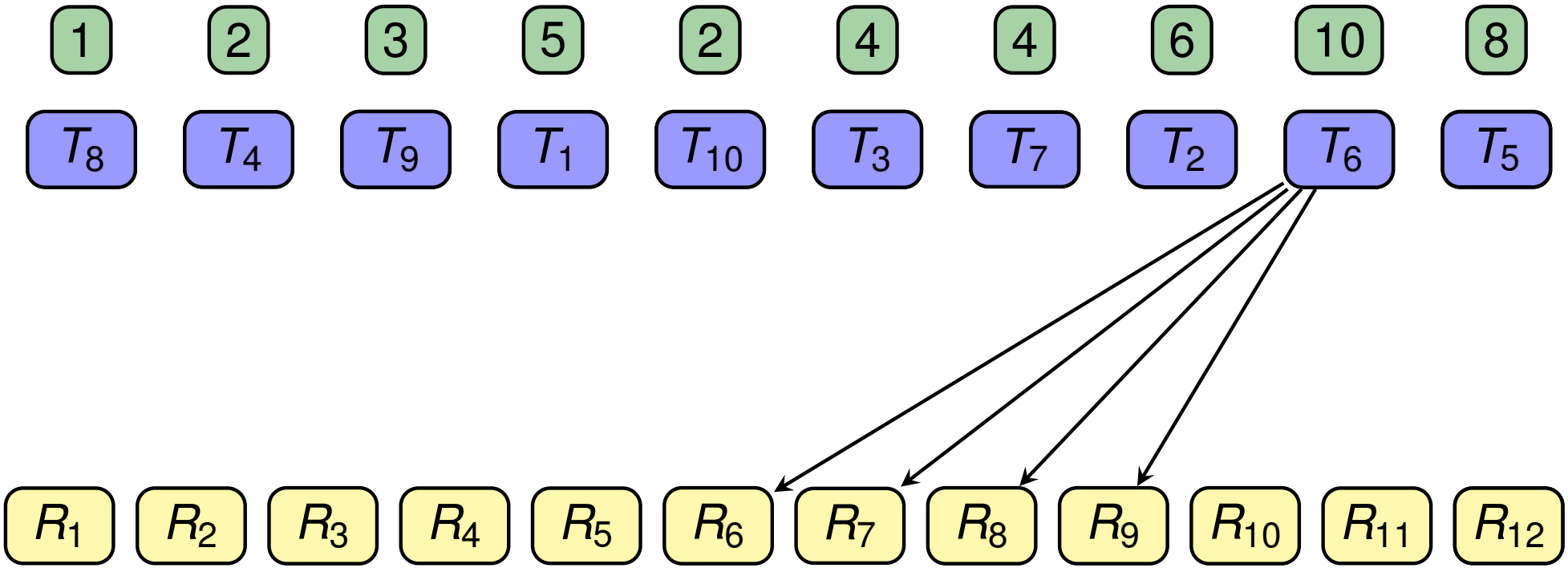
Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

Greedy Algorithms



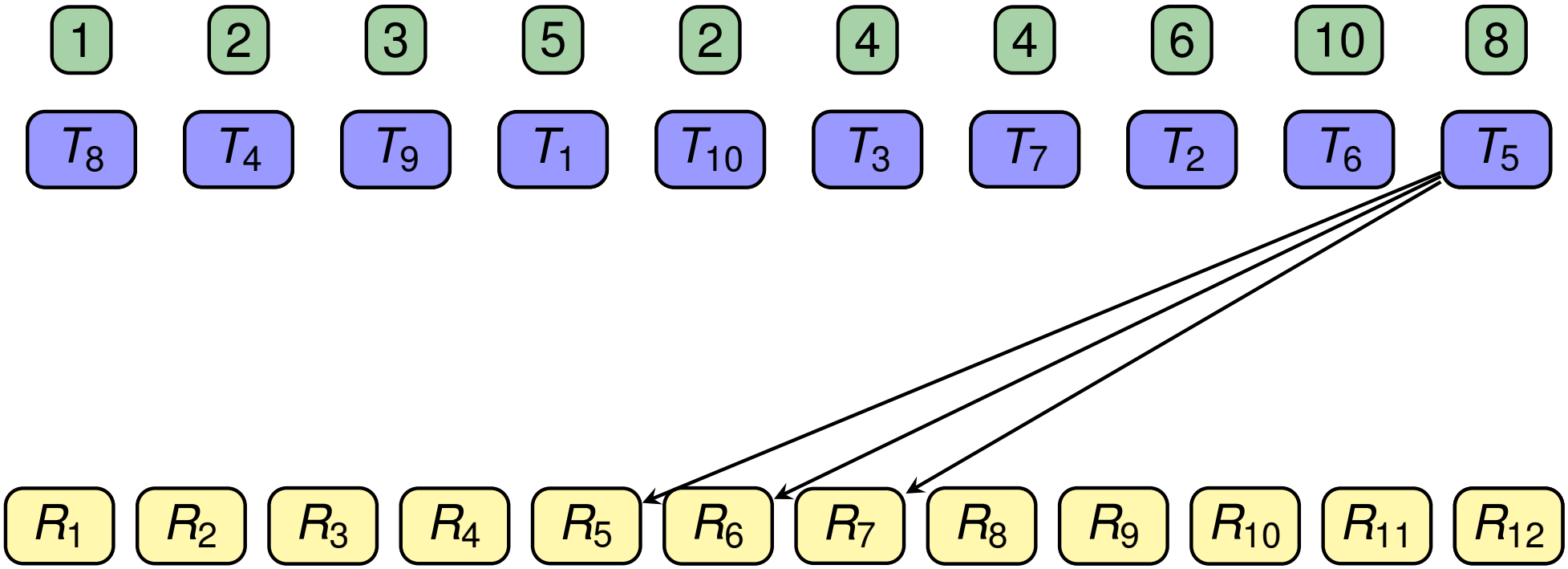
Requirements Set $R = \{R_1, R_2, \ldots, R_{11}, R_{12}\}$

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

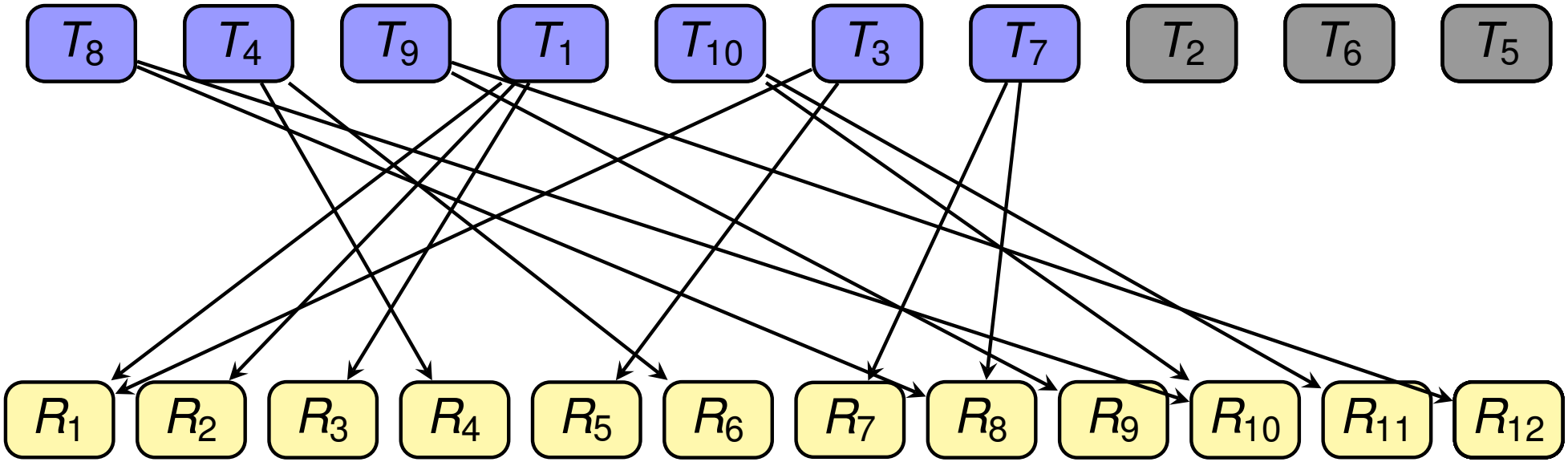
T_8 T_4 T_9 T_1 T_{10} T_3 T_7 T_2 T_6 T_5

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

Hands-on: Running the Greedy Algorithms

1

Load the Regression Testing Techniques

```
cd modificare-installation-directory
R
source(``FDP_Start.R'')
zLoadFit()
zLoadGRD()
zLoadHC()
zLoadGA()
zLoadProduceOrderingReductionDataFile()
```

Hands-on: Running the Greedy Algorithms

2

Run the Greedy Reduction Algorithm

```
x1<-GRD_reduction(coverageMatrixName=
    "reqMatrices/StatementCoverageMatrices/
    Sudoku_1_line_true_Coverage.dat",
    timingsFileName="reqMatrices/
    TimingsFiles/Sudoku_Timing.dat")
```

Hands-on: Running the Greedy Algorithms

3

Run the Greedy Prioritization Algorithm

```
x2<-GRD(coverageMatrixName=
    "reqMatrices/StatementCoverageMatrices/
    Sudoku_1_line_true_Coverage.dat",
    timingsFileName="reqMatrices/
    TimingsFiles/Sudoku_Timing.dat")
```

Hands-on: Running the Greedy Algorithms

4

Map the Test Indices to Test Names

```
y1<-produceOrderingReductionDataFile
    (testSuite=x1$Ord,
    timingsFile="reqMatrices/
    TimingsFiles/Sudoku_Timing.dat")

y2<-produceOrderingReductionDataFile
    (testSuite=x2$Ord,
    timingsFile="reqMatrices/
    TimingsFiles/Sudoku_Timing.dat")
```

Hands-on: Running the Greedy Algorithms

5

Store the Test Case Timings

```
write.table(y1,file="path-to-proteja/
    SK_ordering1.dat", row.names=FALSE,
    col.names=FALSE,quote=FALSE)

write.table(y2,file="path-to-proteja/
    SK_ordering2.dat", row.names=FALSE,
    col.names=FALSE,quote=FALSE)
```

Hands-on: Running the Greedy Algorithms

6

Execute the Reduced Test Suite

```
cd proteja-installation-directory
ant modifyTestSuite -DmodificationFile=
    "SK_ordering1.dat"
ant proteja
```

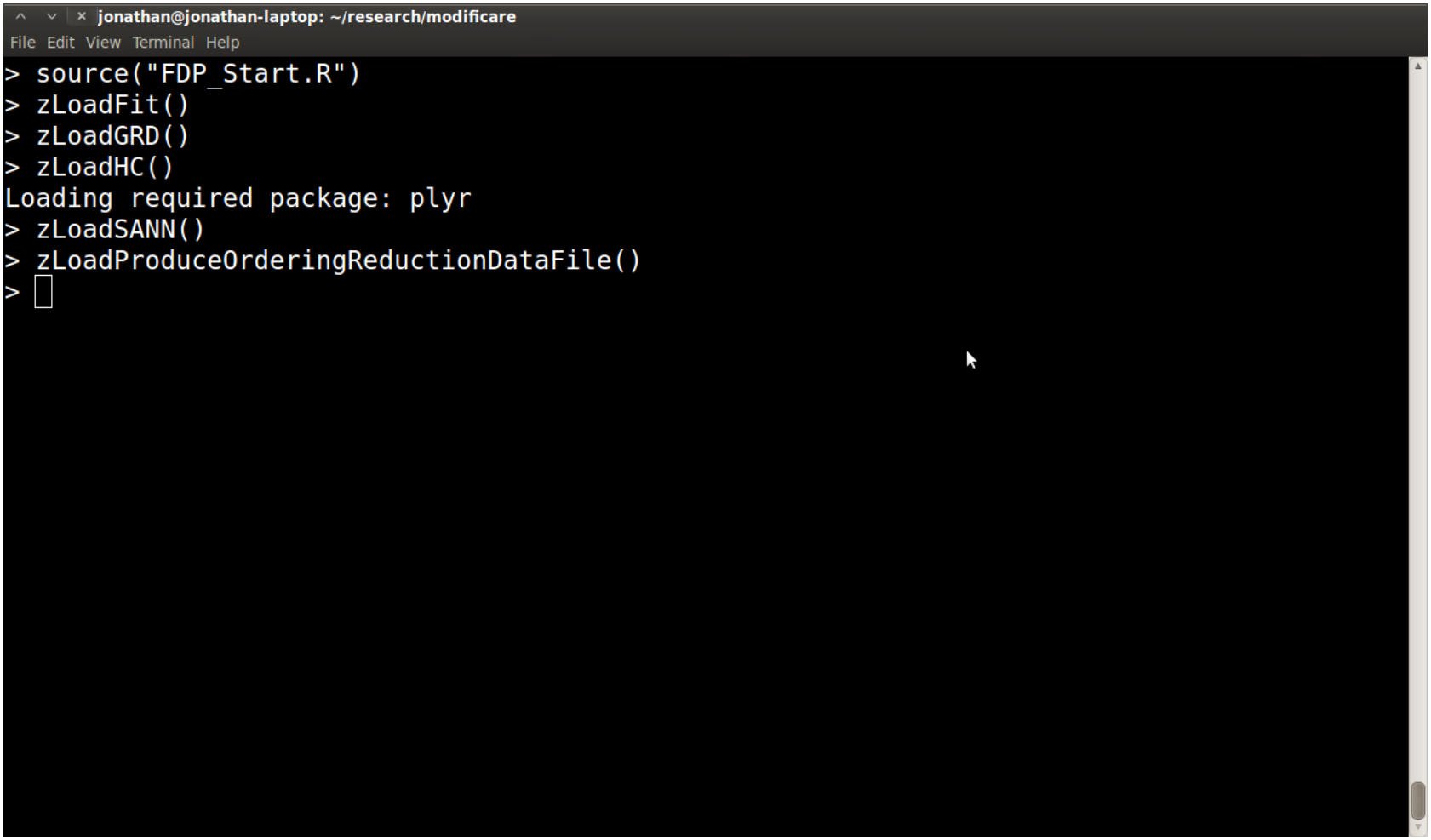
Hands-on: Running the Greedy Algorithms

7

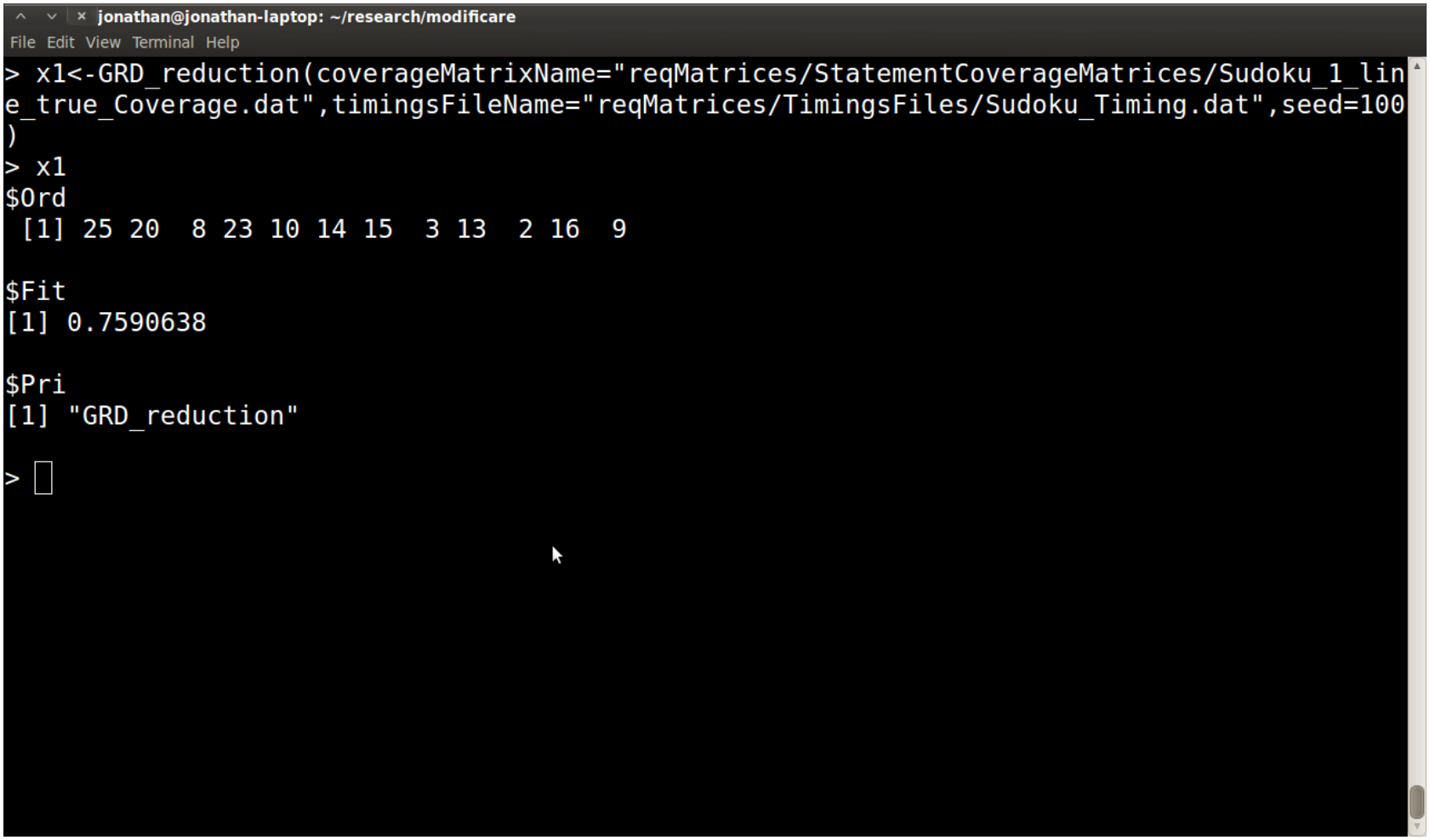
Execute the Prioritized Test Suite

```
cd proteja-installation-directory
ant modifyTestSuite -DmodificationFile=
    "SK_ordering2.dat"
ant proteja
```

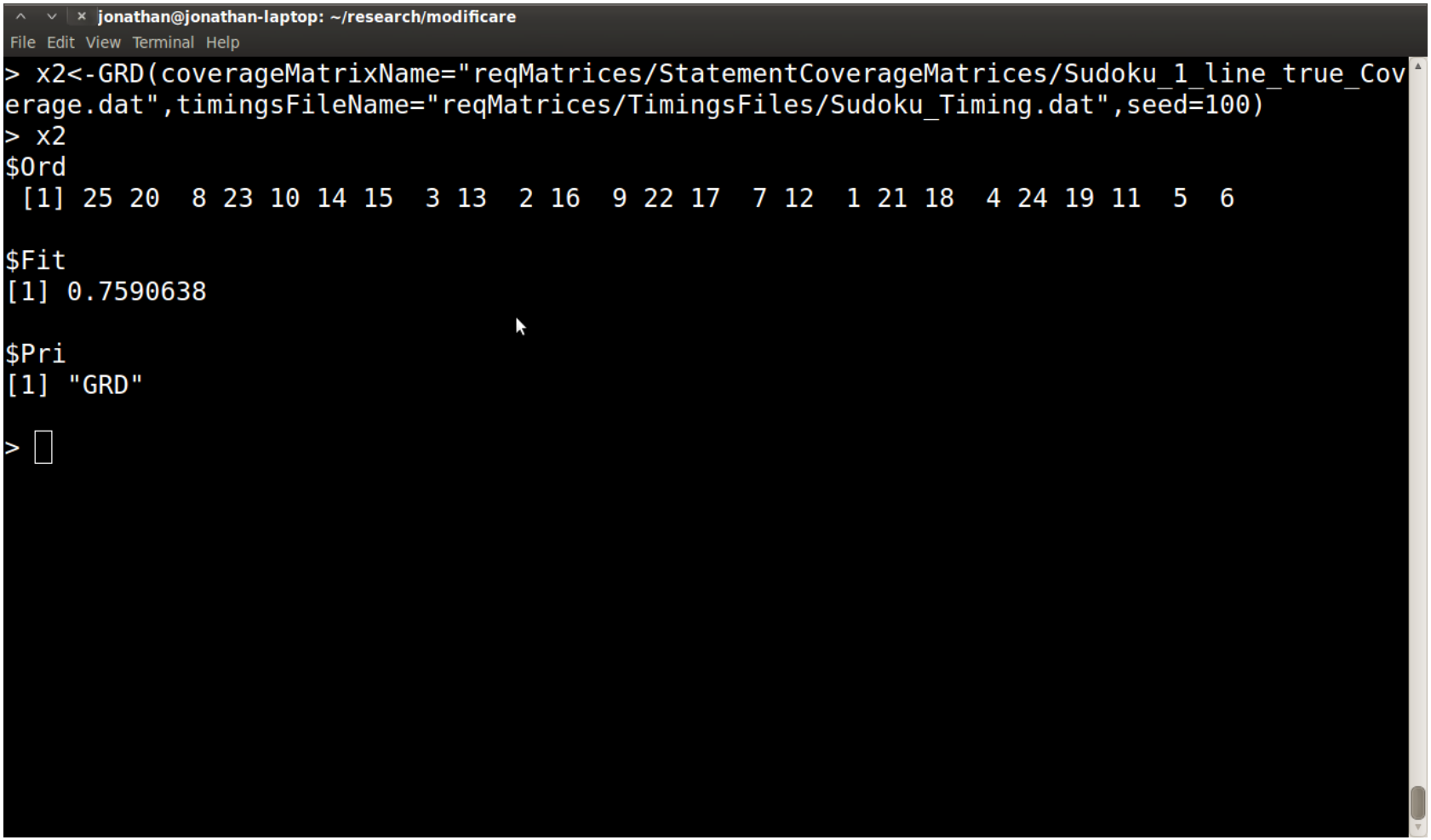
Screenshot: Loading Regression Testing



Screenshot: Running the Greedy Reducer



Screenshot: Running the Greedy Prioritizer

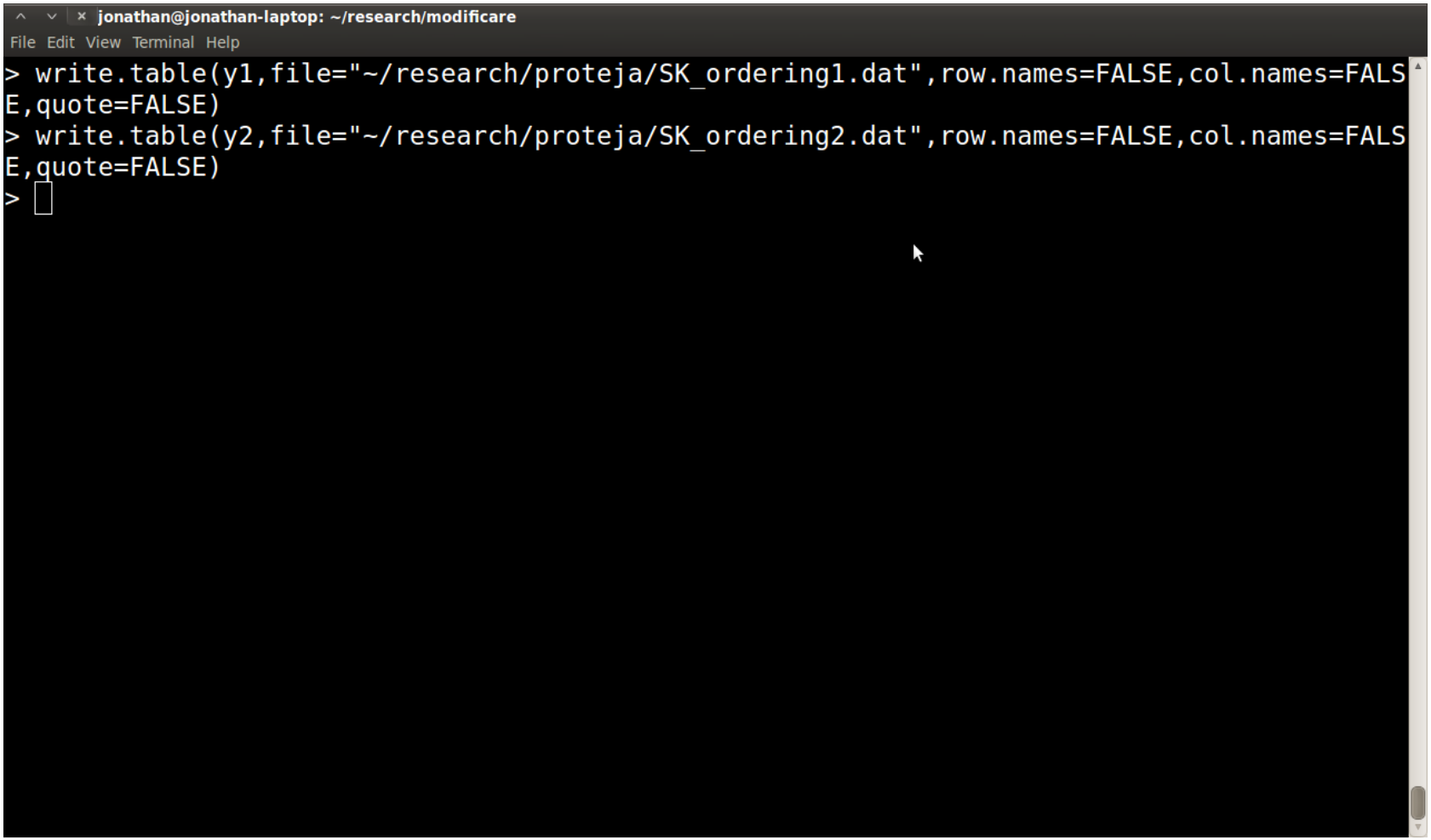


Screenshot: Test Index to Test Name Mapping

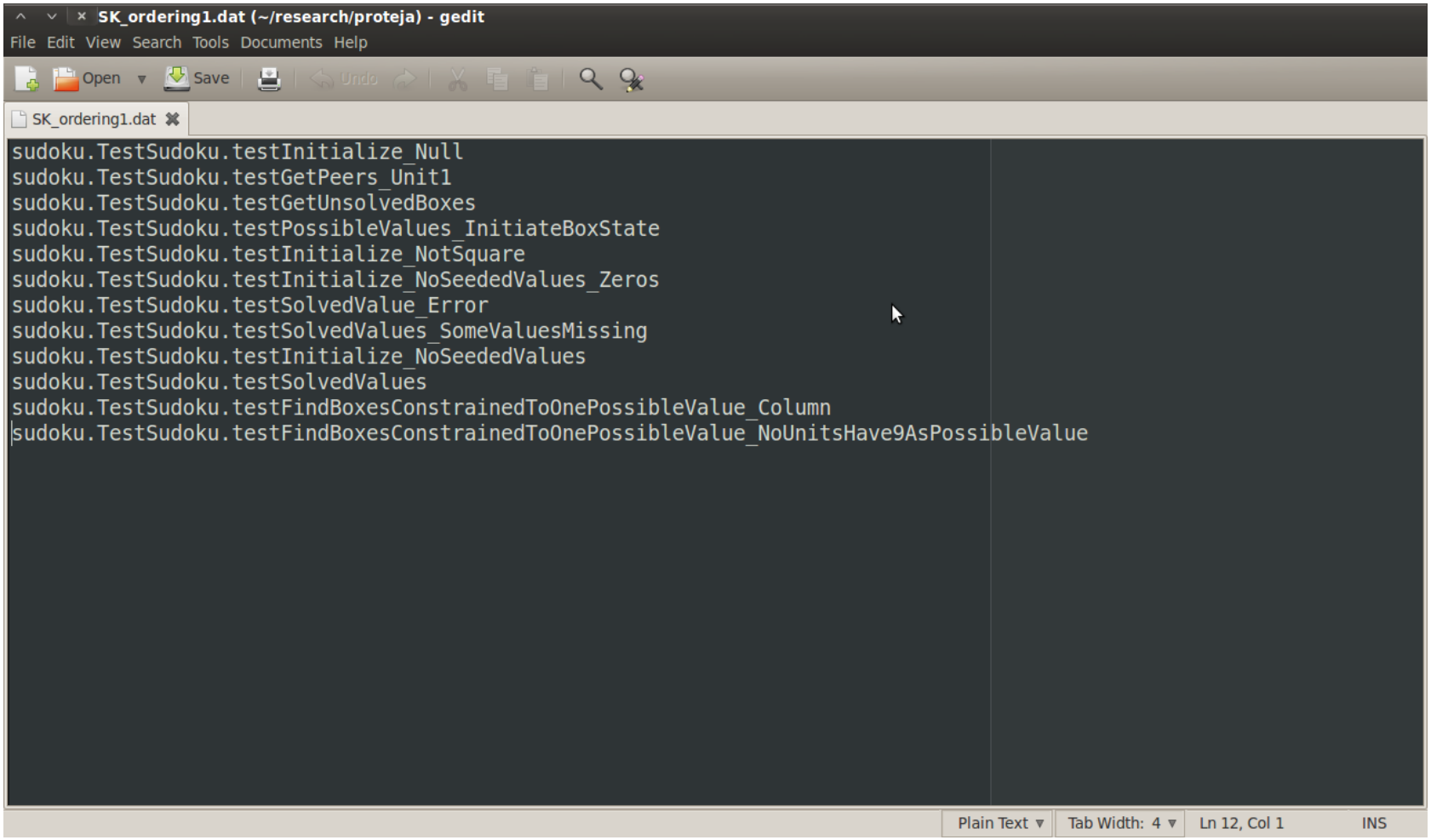
jonathan@jonathan-laptop: ~/research/modificare
File Edit View Terminal Help

```
> y1<-produceOrderingReductionDataFile(testSuite=x1$Ord,timingsFile="reqMatrices/TimingsFi
les/Sudoku_Timing.dat")
> y1
 [1] "sudoku.TestSudoku.testInitialize_Null"

 [2] "sudoku.TestSudoku.testGetPeers_Unit1"

 [3] "sudoku.TestSudoku.testGetUnsolvedBoxes"

 [4] "sudoku.TestSudoku.testPossibleValues_InitiateBoxState"

 [5] "sudoku.TestSudoku.testInitialize_NotSquare"

 [6] "sudoku.TestSudoku.testInitialize_NoSeededValues_Zeros"

 [7] "sudoku.TestSudoku.testSolvedValue_Error"

 [8] "sudoku.TestSudoku.testSolvedValues_SomeValuesMissing"

 [9] "sudoku.TestSudoku.testInitialize_NoSeededValues"

[10] "sudoku.TestSudoku.testSolvedValues"

[11] "sudoku.TestSudoku.testFindBoxesConstrainedToOnePossibleValue_Column"
```

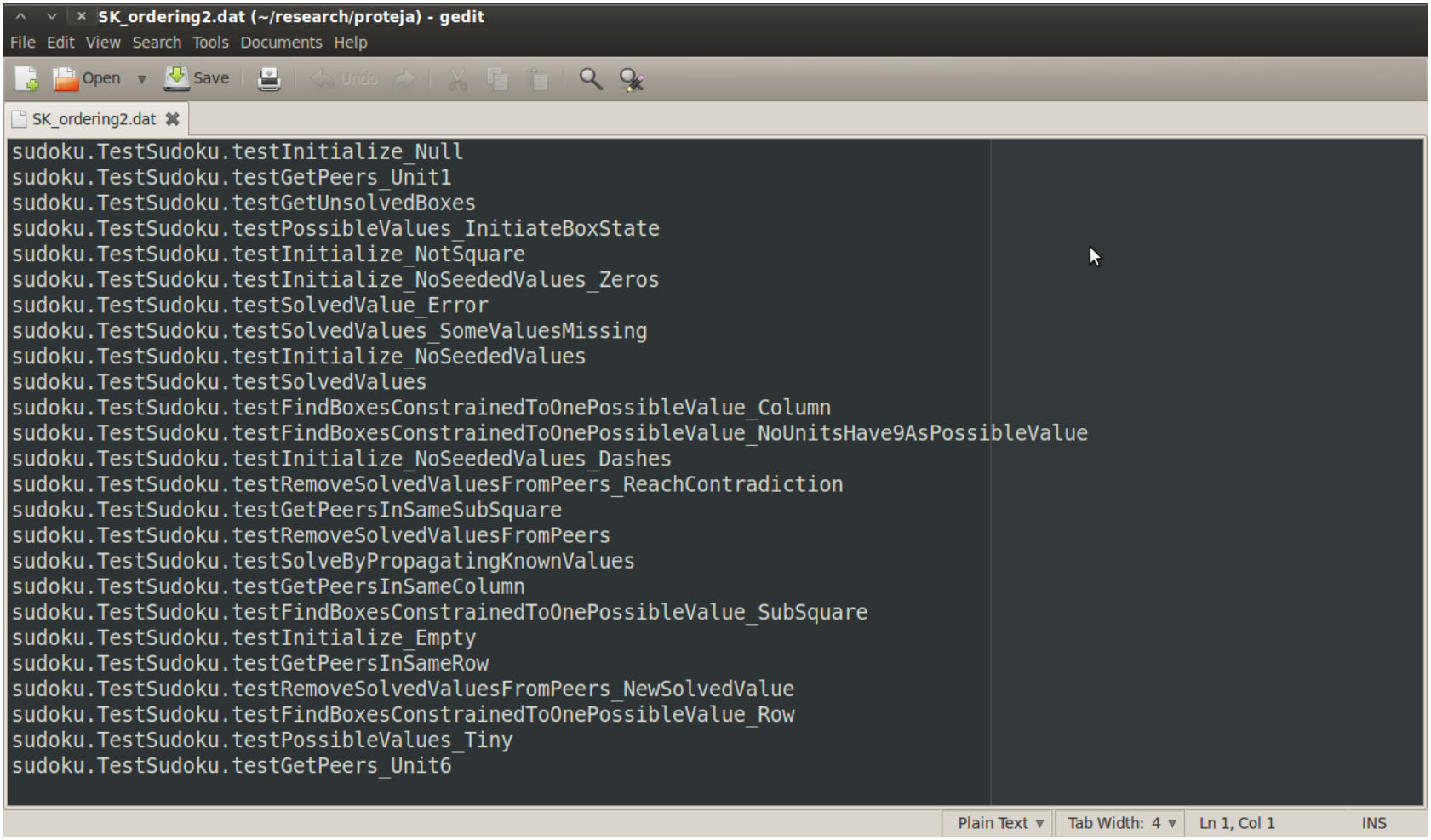
Screenshot: Storing the Test Case Timings



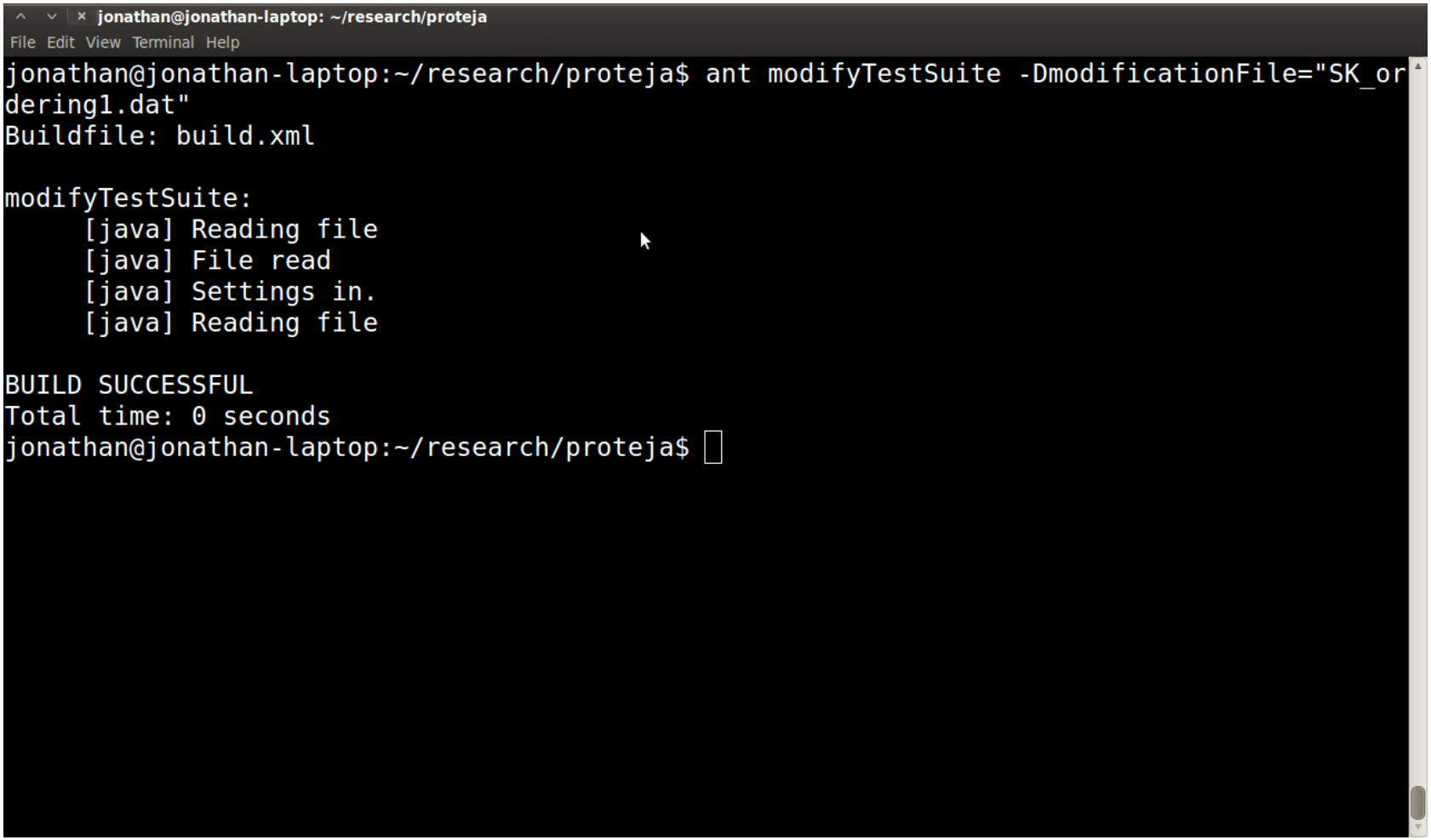
Screenshot: The Reduced Test Suite



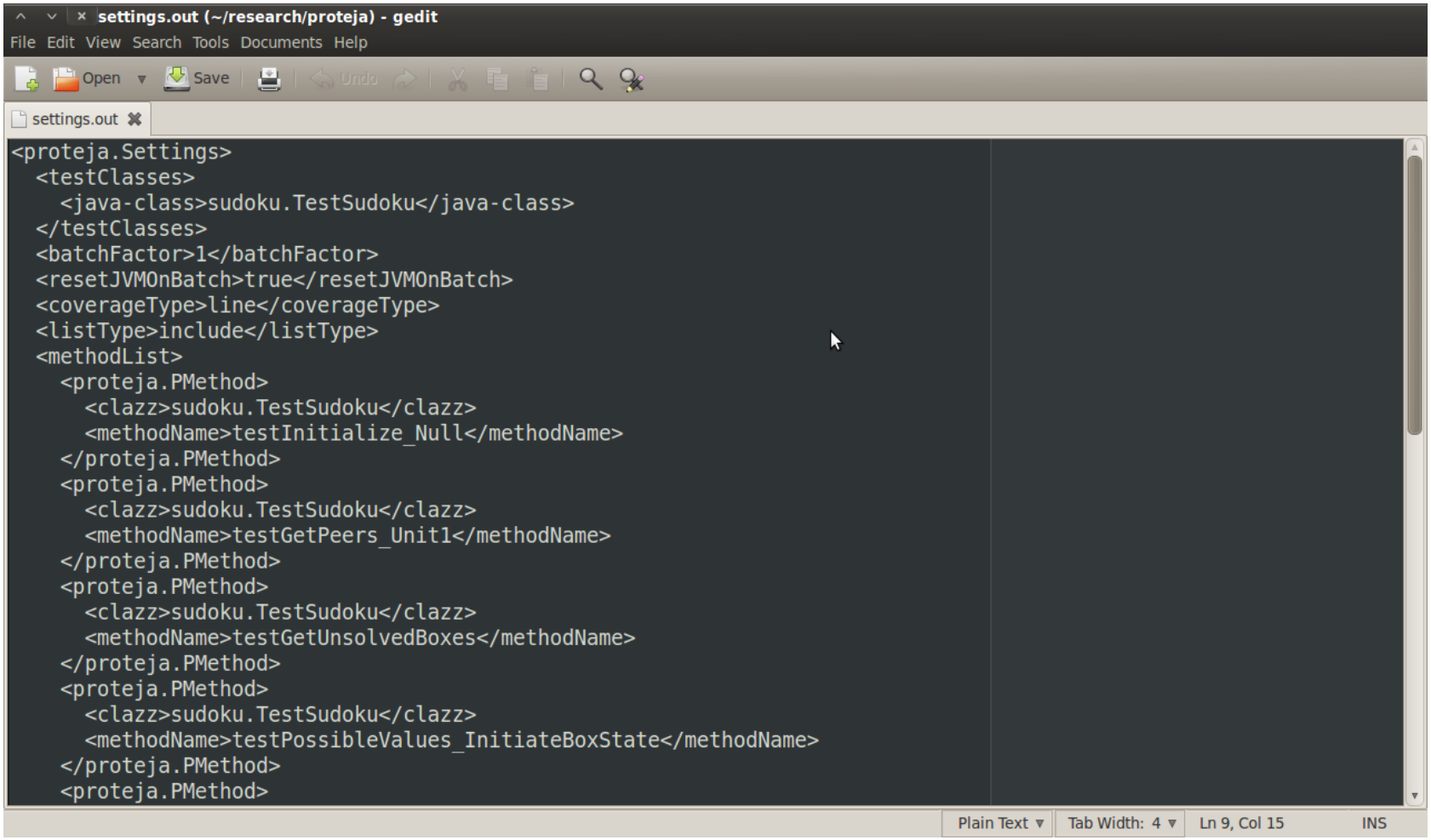
Screenshot: The Prioritized Test Suite



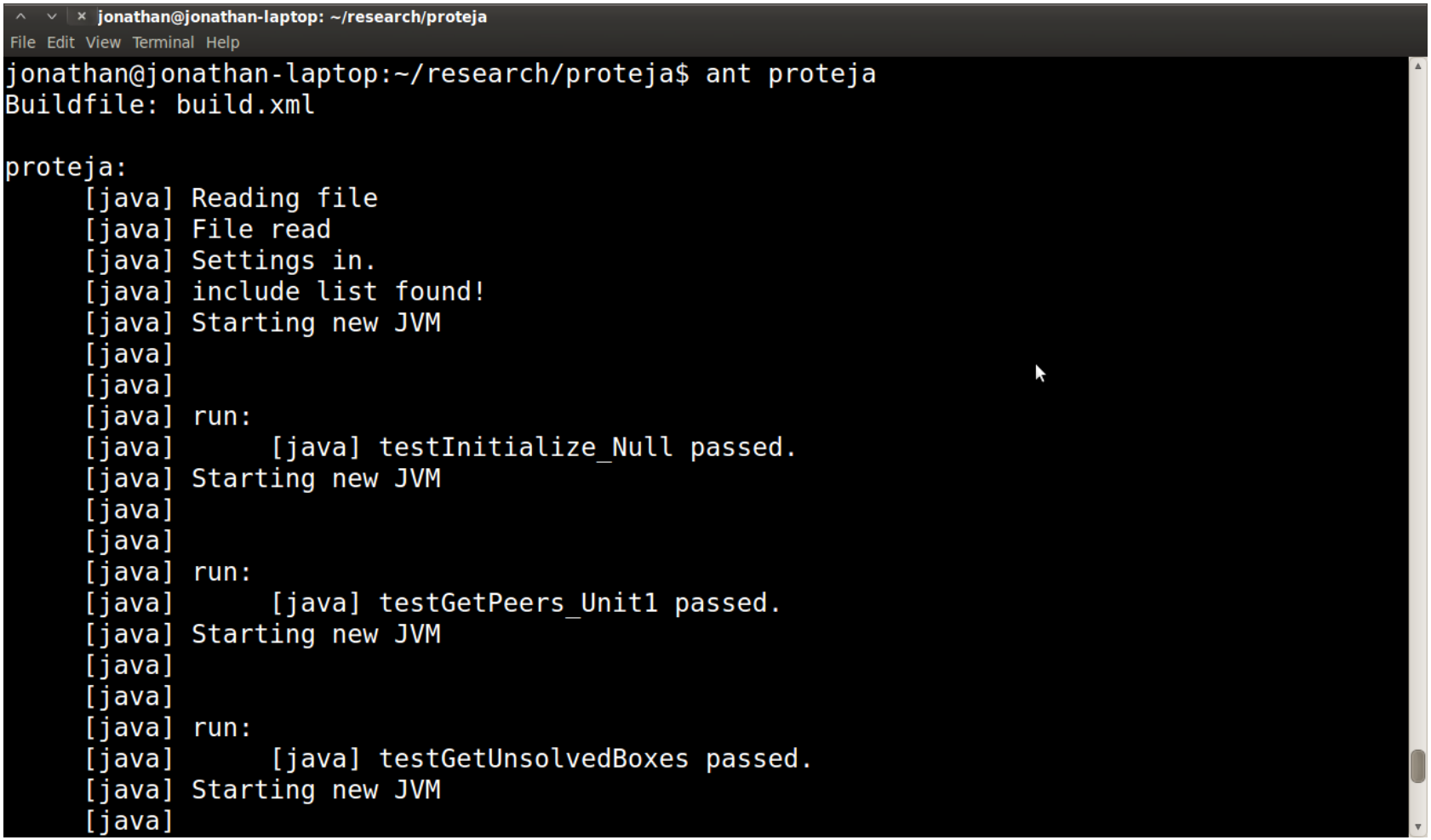
Screenshot: Preparing the Reduced Test Suite



Screenshot: Viewing the Reduced Test Suite



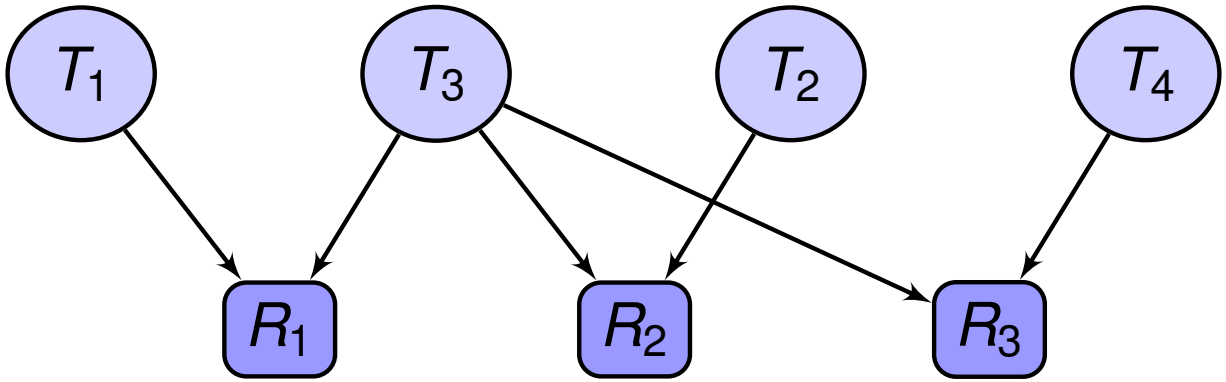
Screenshot: Running the Reduced Test Suite



Limitations of Greedy Algorithms

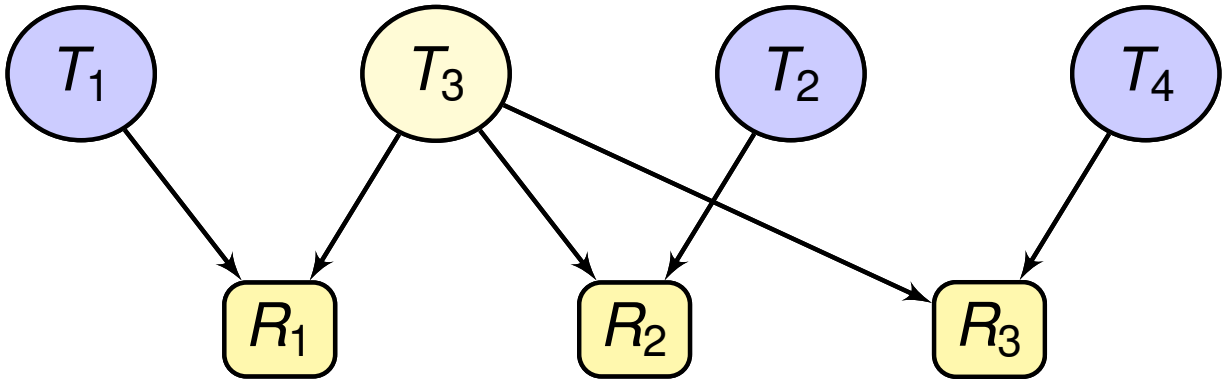
Possible configuration of the **coverage report**

Limitations of Greedy Algorithms



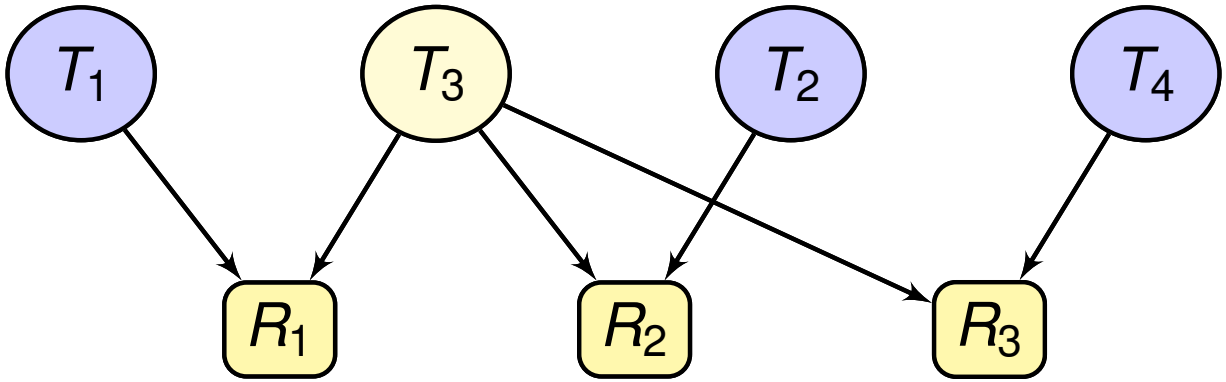
Possible configuration of the **coverage report**

Limitations of Greedy Algorithms



Possible configuration of the **coverage report**

Limitations of Greedy Algorithms



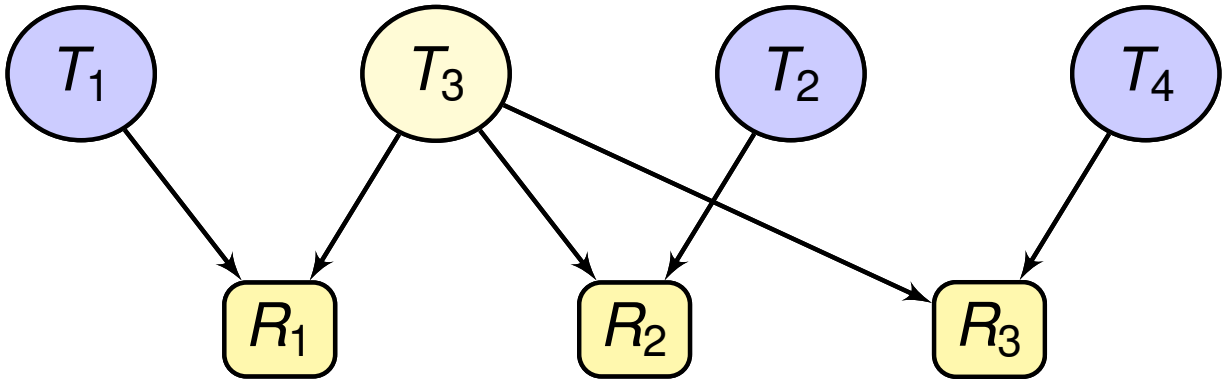
$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

Execution time of the test cases may mislead greedy

Limitations of Greedy Algorithms



$time(T_1) = 1$

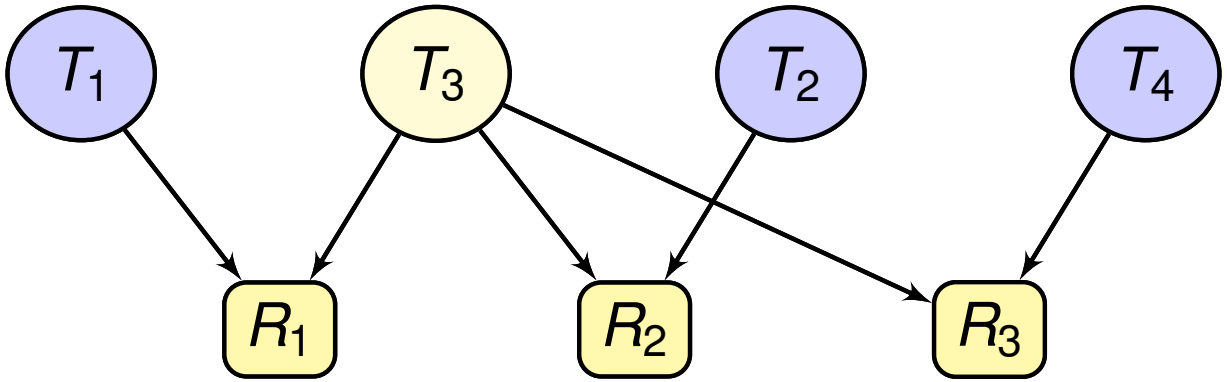
$time(T_2) = 1$

$time(T_4) = 1$

$time(T_3) = 2.45$

Execution time of the test cases may mislead greedy

Limitations of Greedy Algorithms

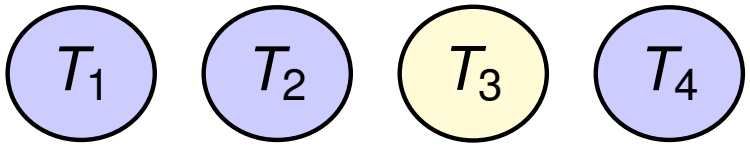


$time(T_1) = 1$

$time(T_2) = 1$

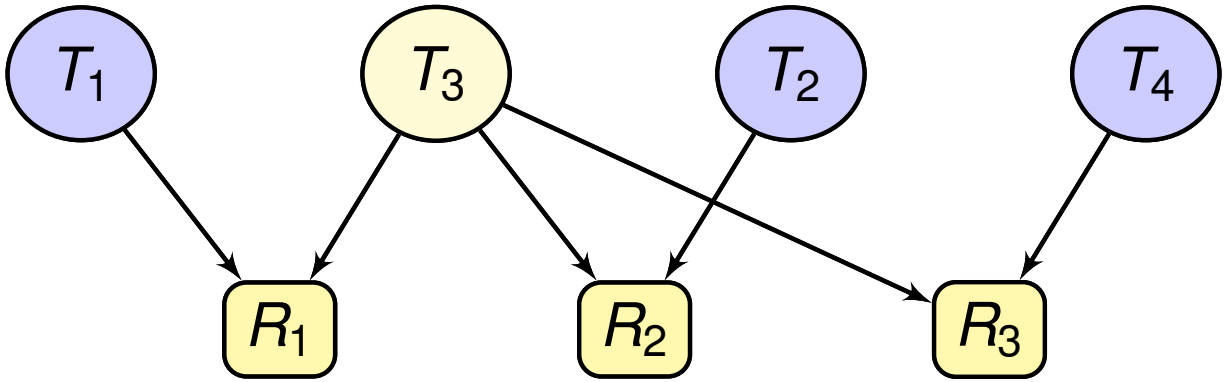
$time(T_4) = 1$

$time(T_3) = 2.45$



Original ordering has low effectiveness score

Limitations of Greedy Algorithms

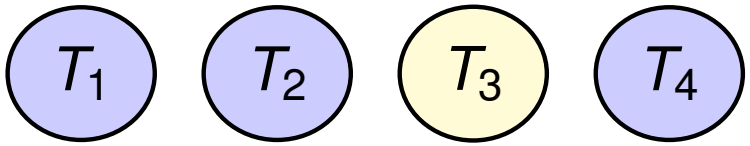


$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

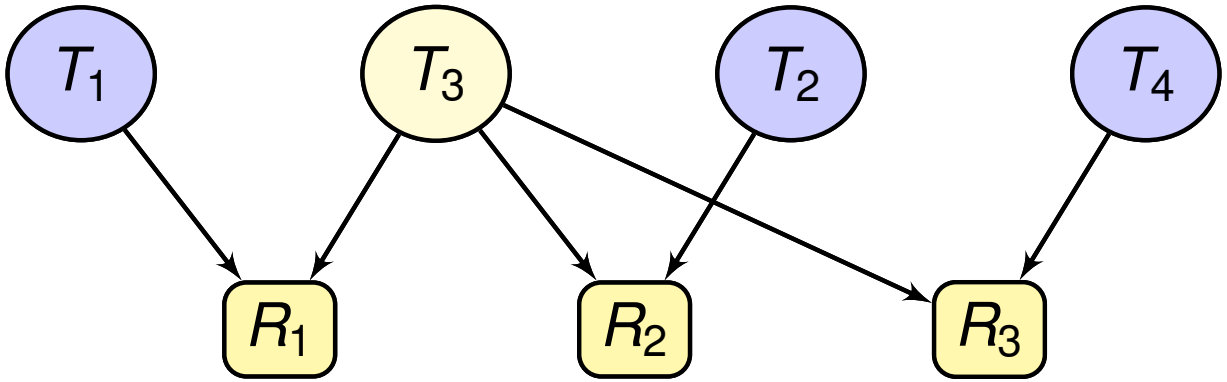
$time(T_3) = 2.45$



$CE(T) = 0.54$

Original ordering has low effectiveness score

Limitations of Greedy Algorithms

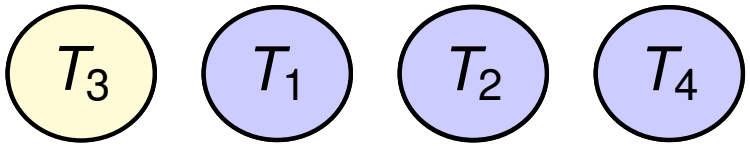


$time(T_1) = 1$

$time(T_2) = 1$

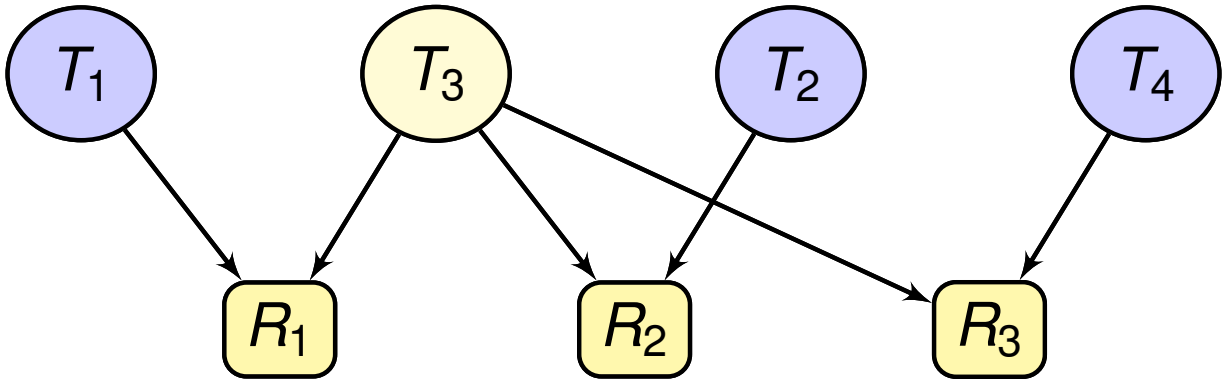
$time(T_4) = 1$

$time(T_3) = 2.45$



Greedy method constructs suite with marginal improvement

Limitations of Greedy Algorithms

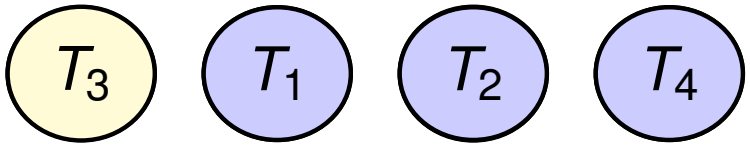


$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

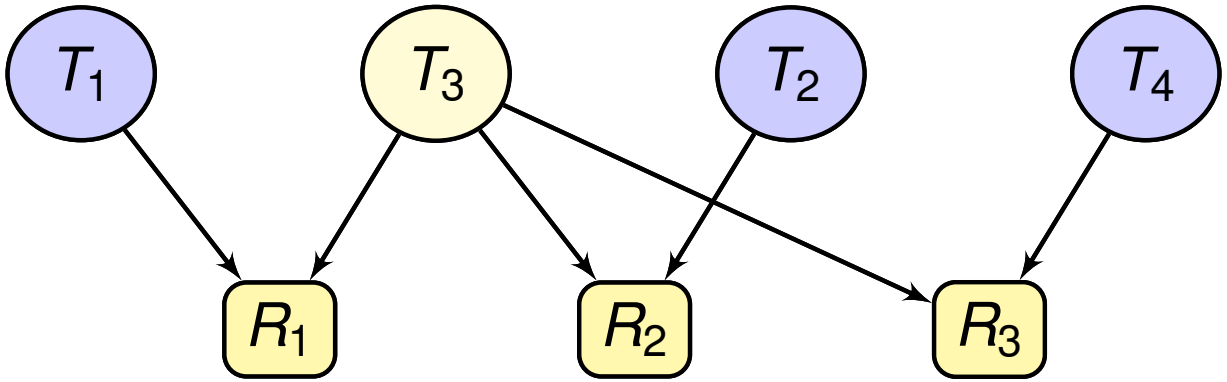
$time(T_3) = 2.45$



$CE(T') = 0.55$

Greedy method constructs suite with marginal improvement

Limitations of Greedy Algorithms

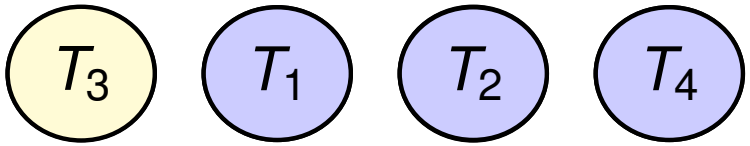


$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

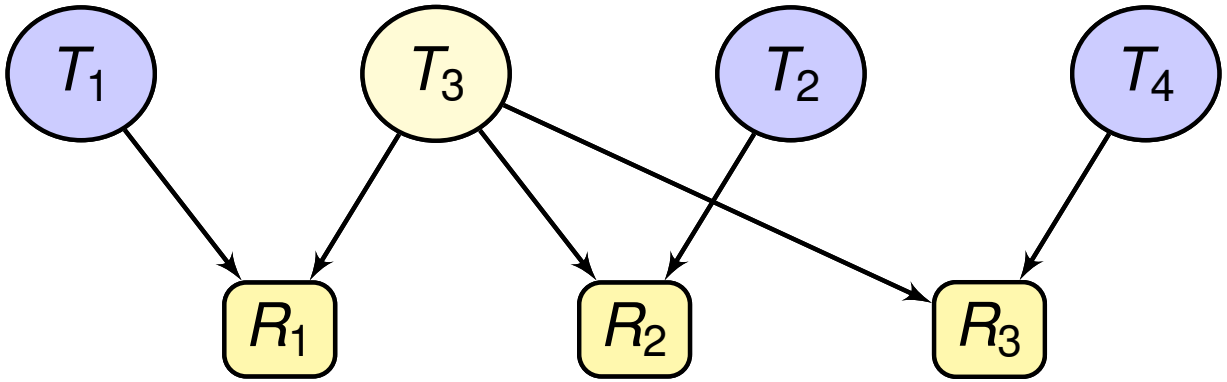
$time(T_3) = 2.45$



$CE(T') = 0.55$

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

Limitations of Greedy Algorithms

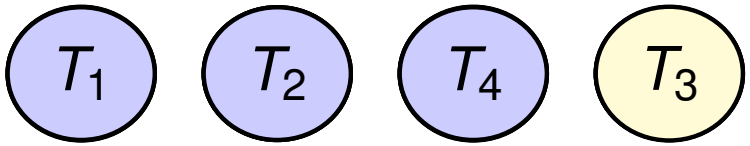


$time(T_1) = 1$

$time(T_2) = 1$

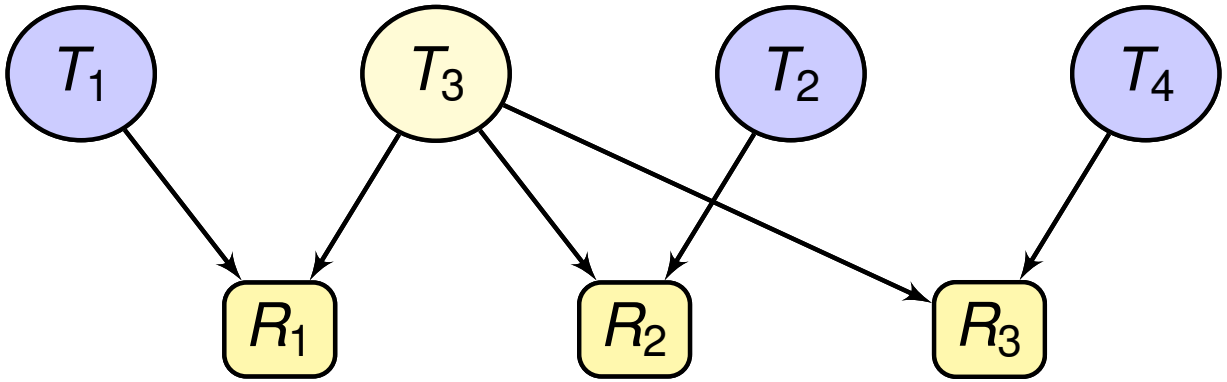
$time(T_4) = 1$

$time(T_3) = 2.45$



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

Limitations of Greedy Algorithms

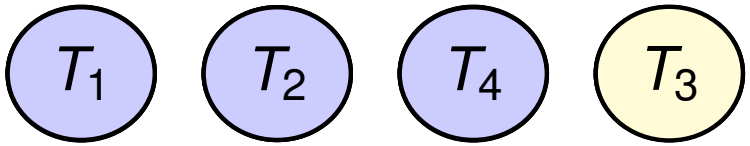


$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

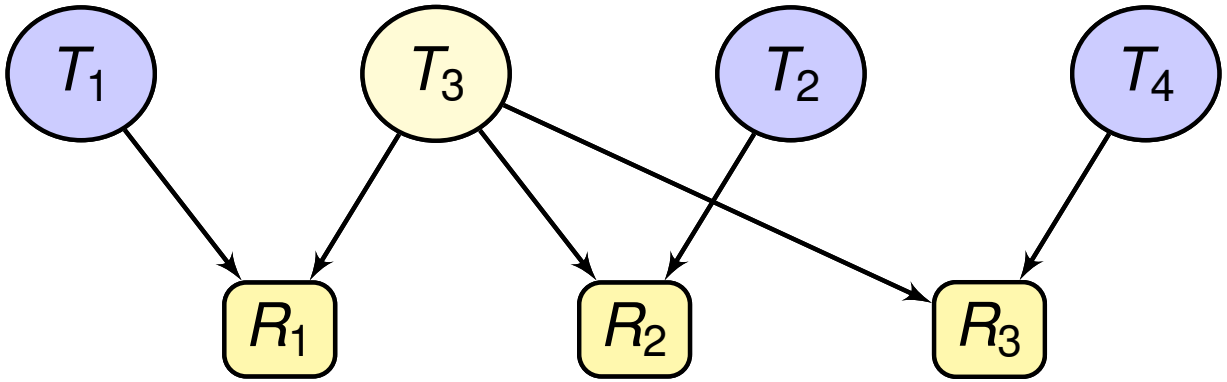
$time(T_3) = 2.45$



$CE(T') = 0.63$

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

Limitations of Greedy Algorithms

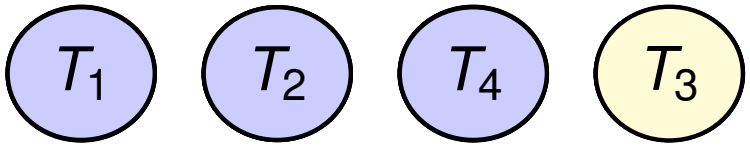


$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

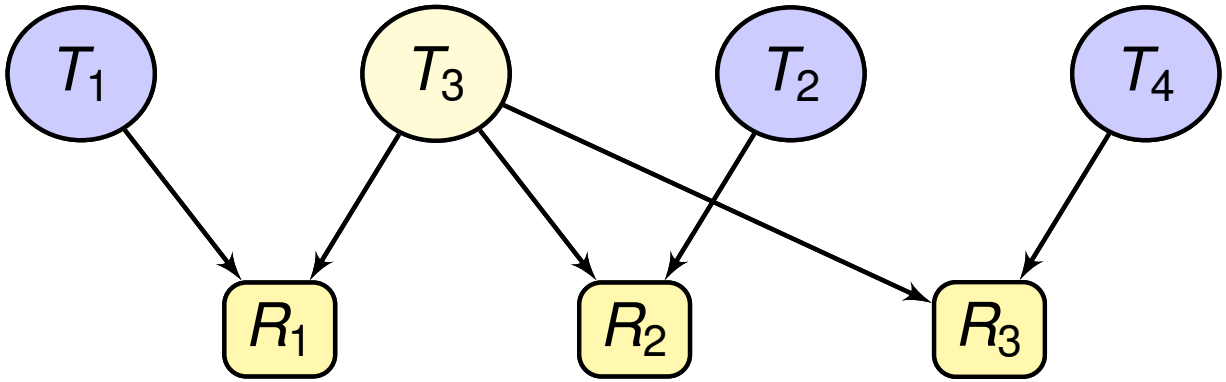
$time(T_3) = 2.45$



$CE(T') = 0.63$

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

Limitations of Greedy Algorithms

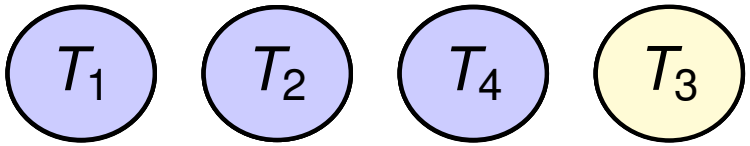


$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

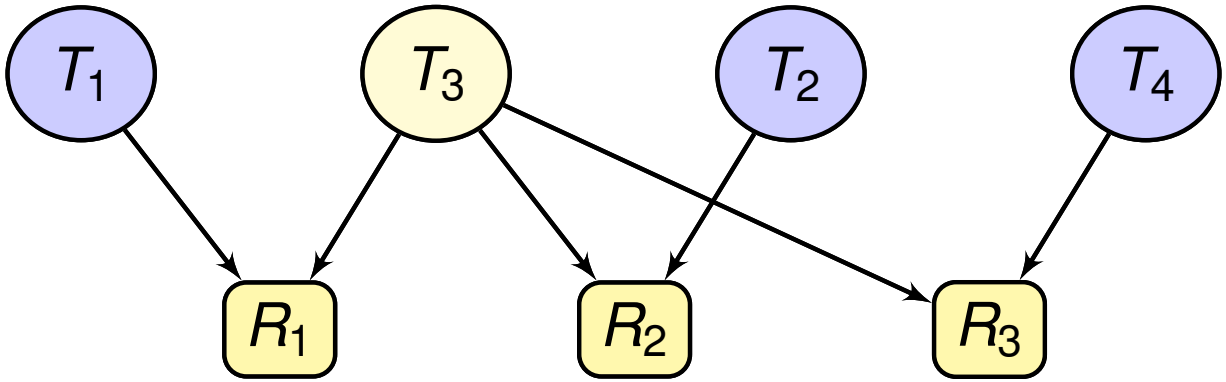
$time(T_3) = 2.45$



$CE(T') = 0.63$

Search-based algorithms support **"human in the loop"**

Limitations of Greedy Algorithms

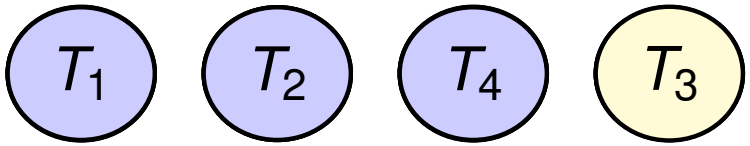


$time(T_1) = 1$

$time(T_2) = 1$

$time(T_4) = 1$

$time(T_3) = 2.45$



$CE(T') = 0.63$

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

Hill Climbing Algorithm

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

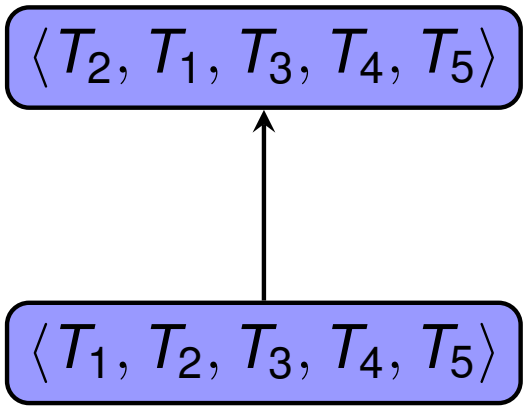
Hill Climbing Algorithm

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

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

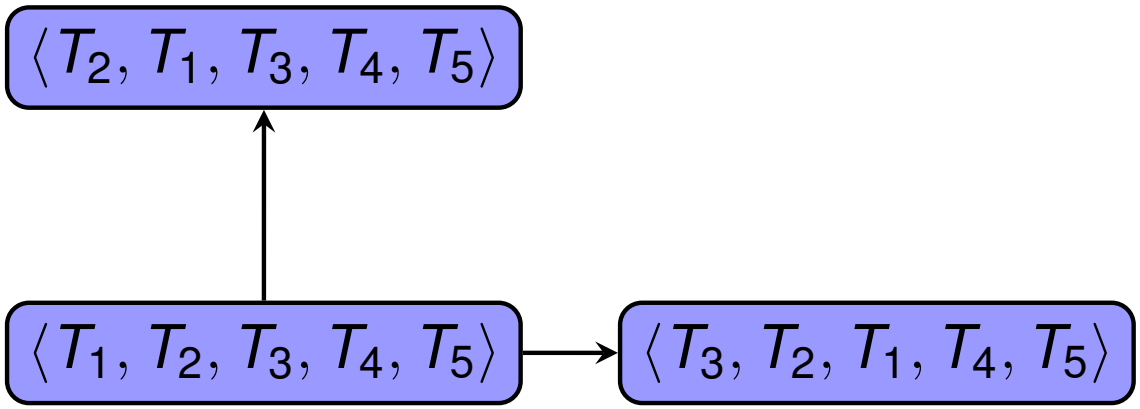
Hill Climbing Algorithm

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



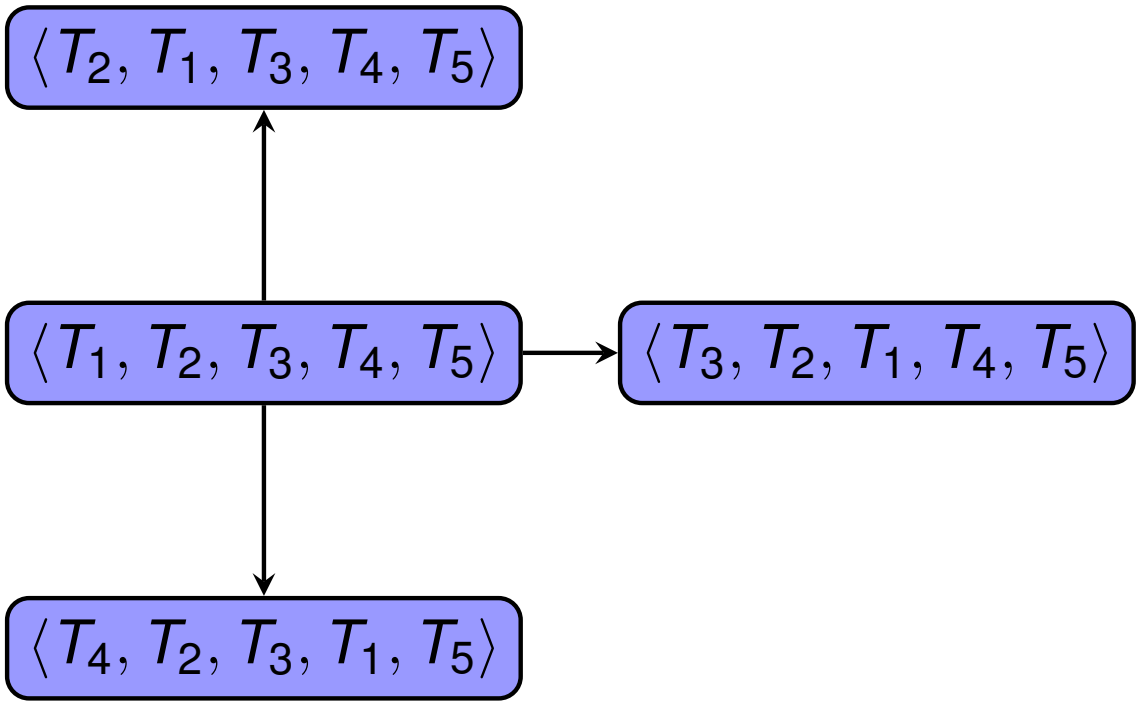
Hill Climbing Algorithm

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



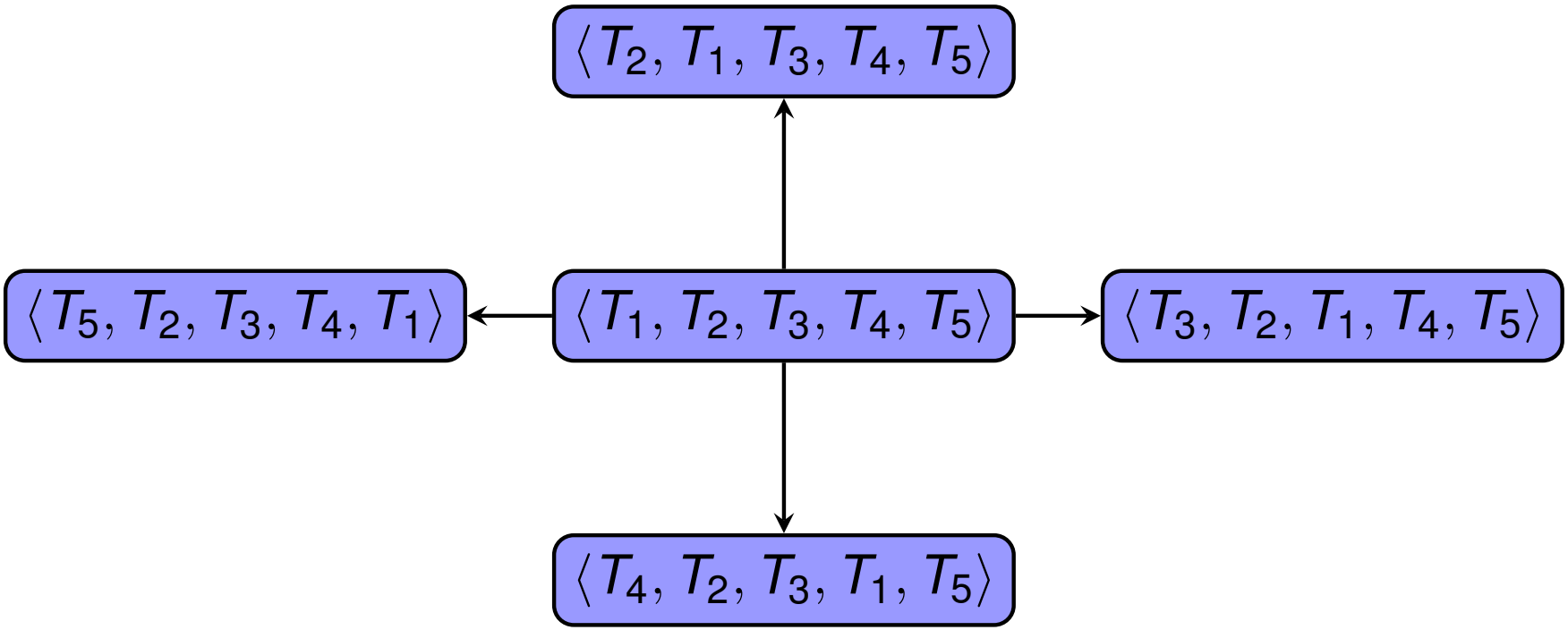
Hill Climbing Algorithm

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



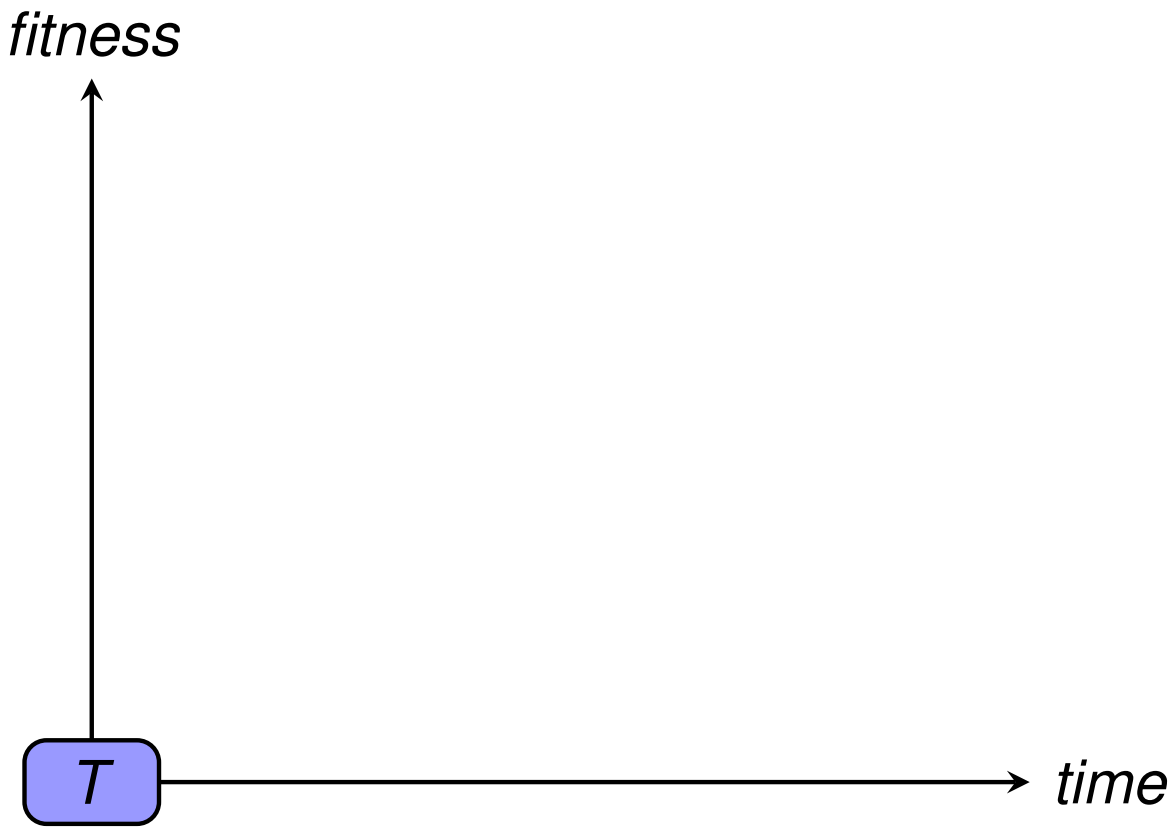
Hill Climbing Algorithm

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

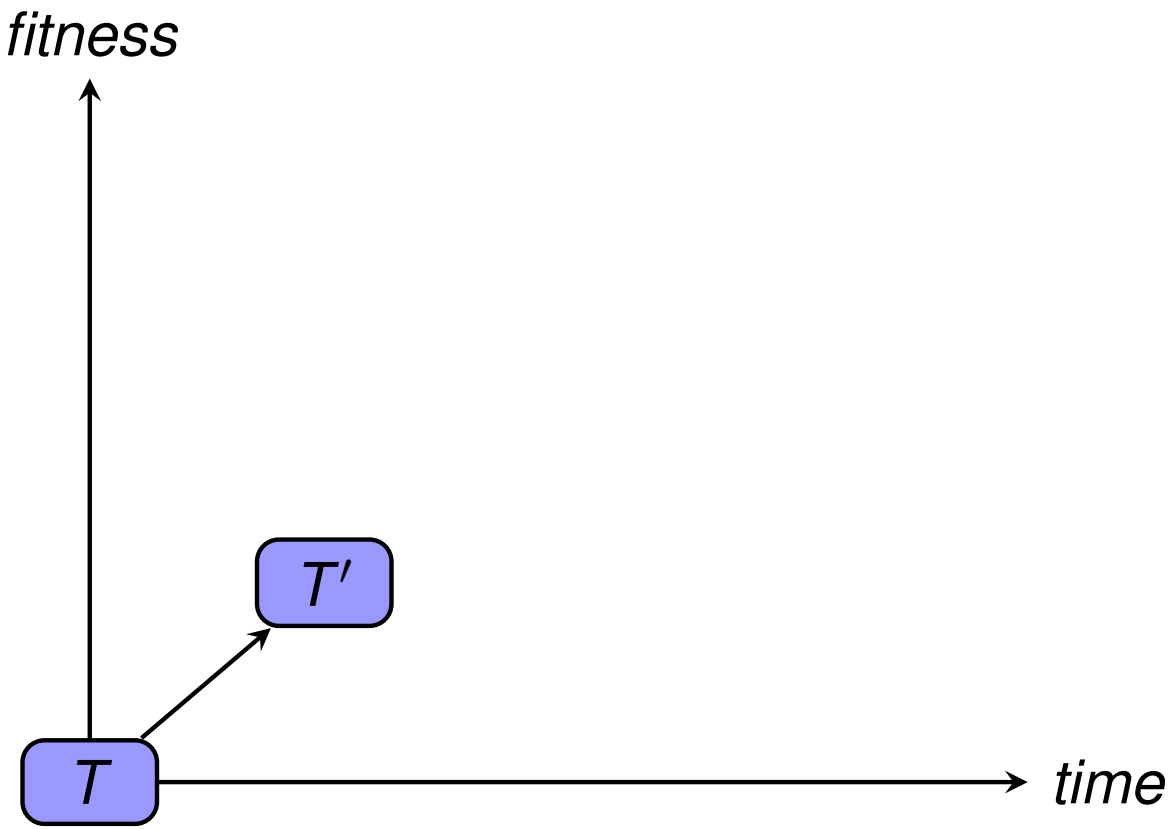


Hill Climbing Algorithm

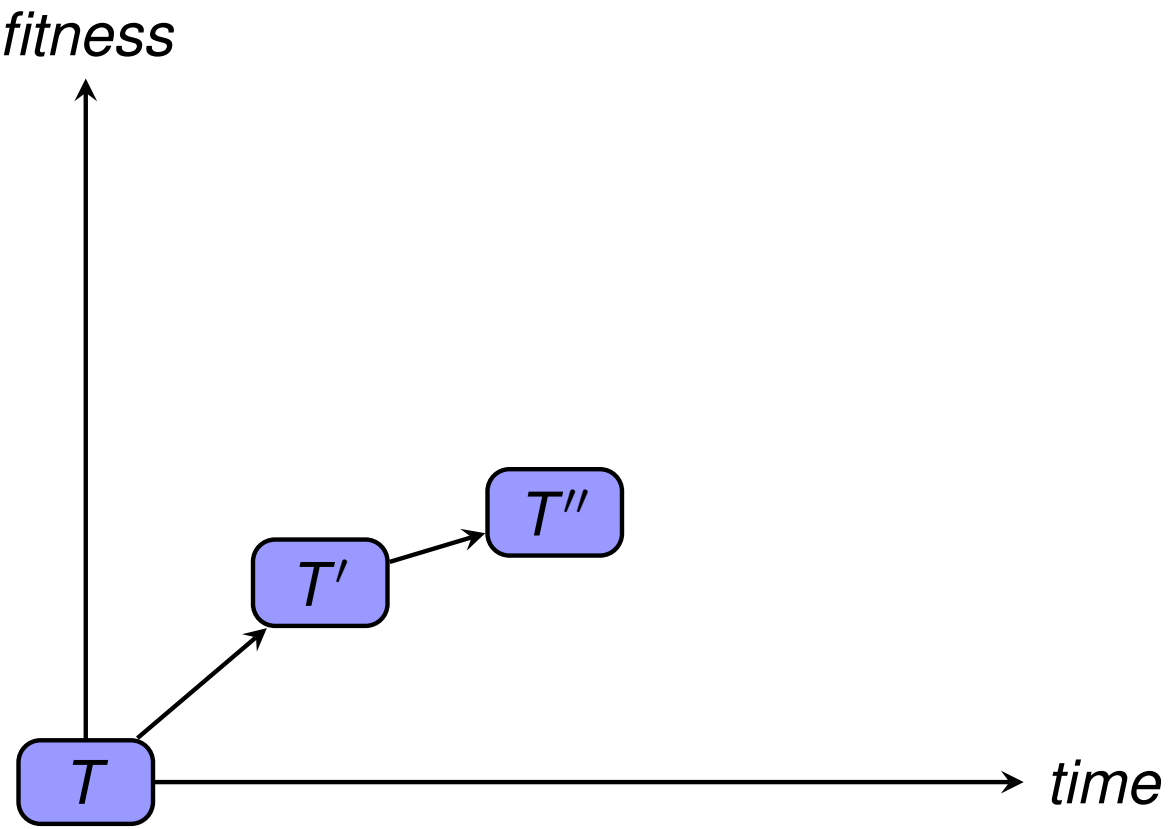
Hill Climbing Algorithm



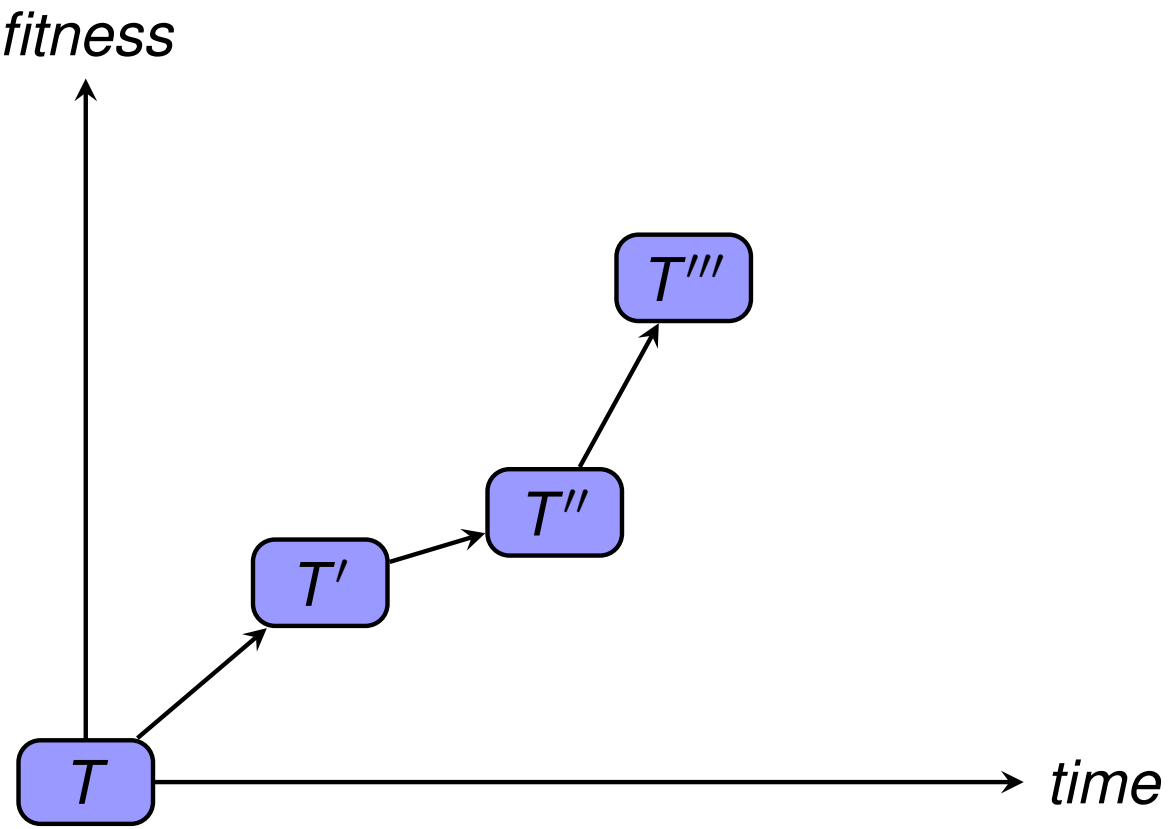
Hill Climbing Algorithm



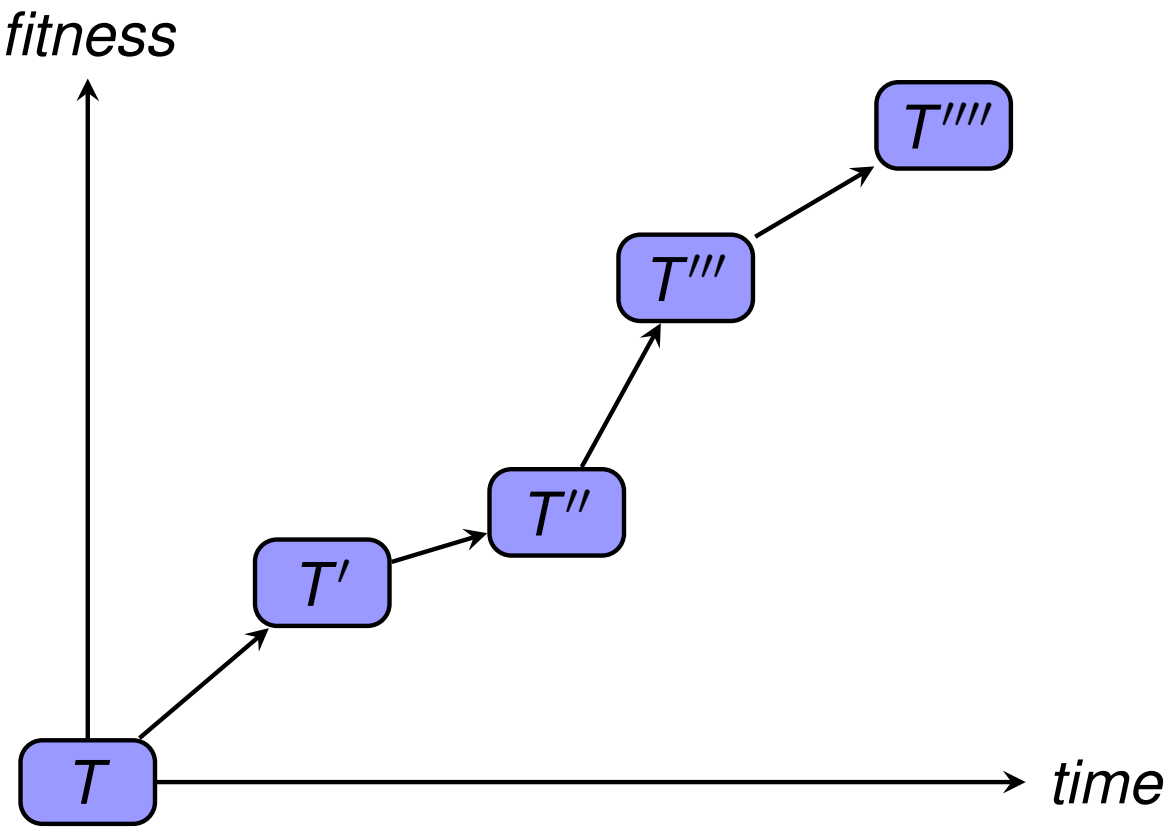
Hill Climbing Algorithm



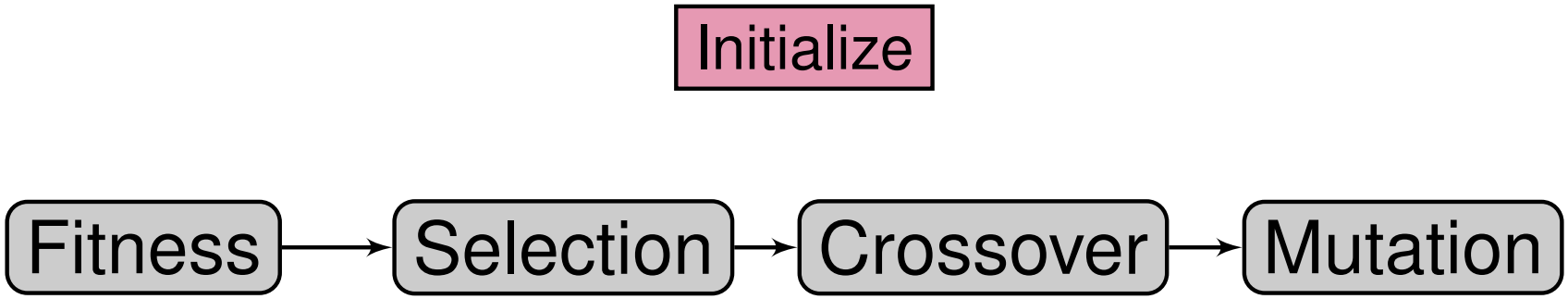
Hill Climbing Algorithm



Hill Climbing Algorithm

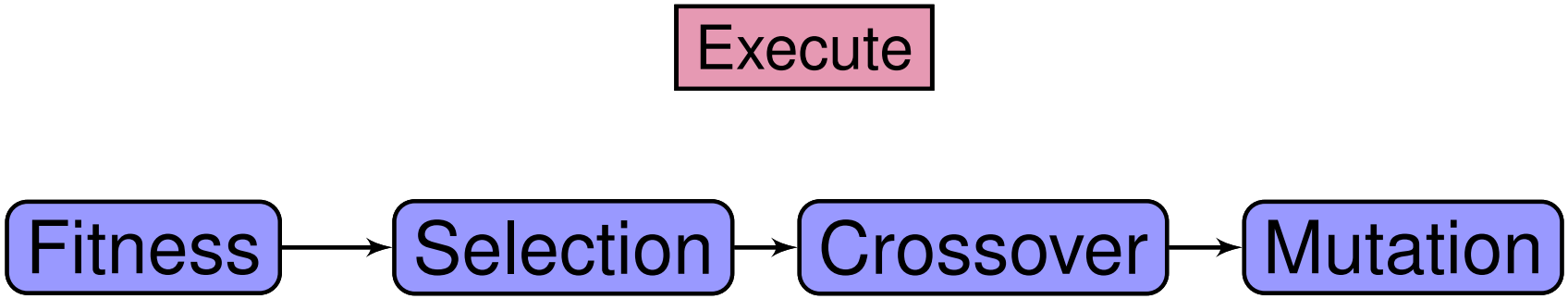


Genetic Algorithm



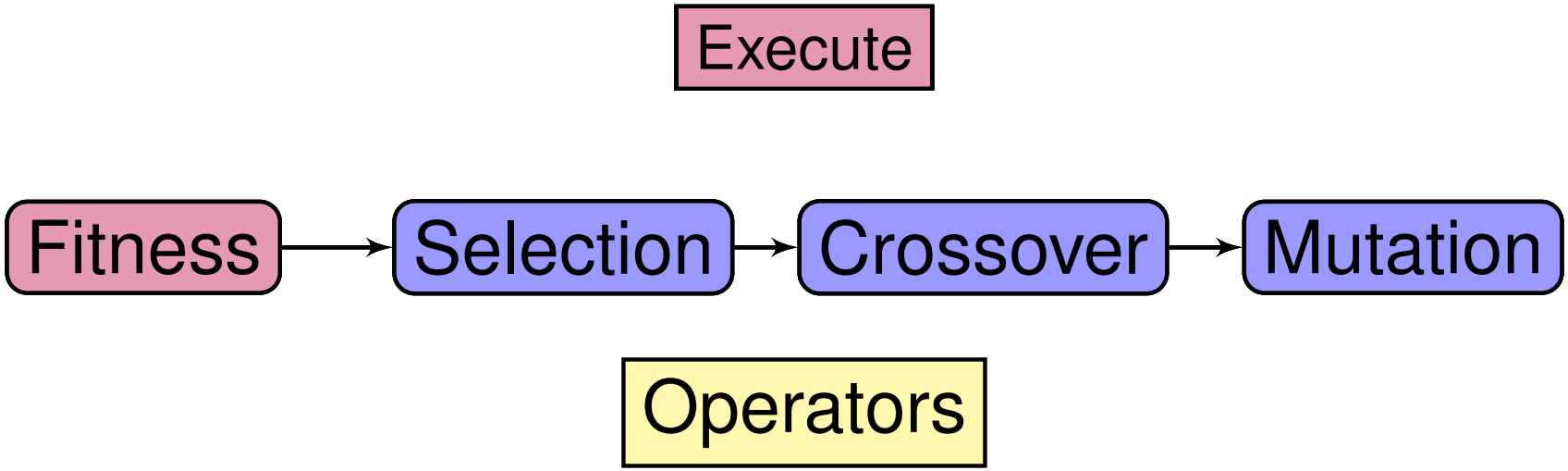
Randomly create suites by repeatedly **shuffling** $\langle T_1, \ldots, T_n \rangle$

Genetic Algorithm



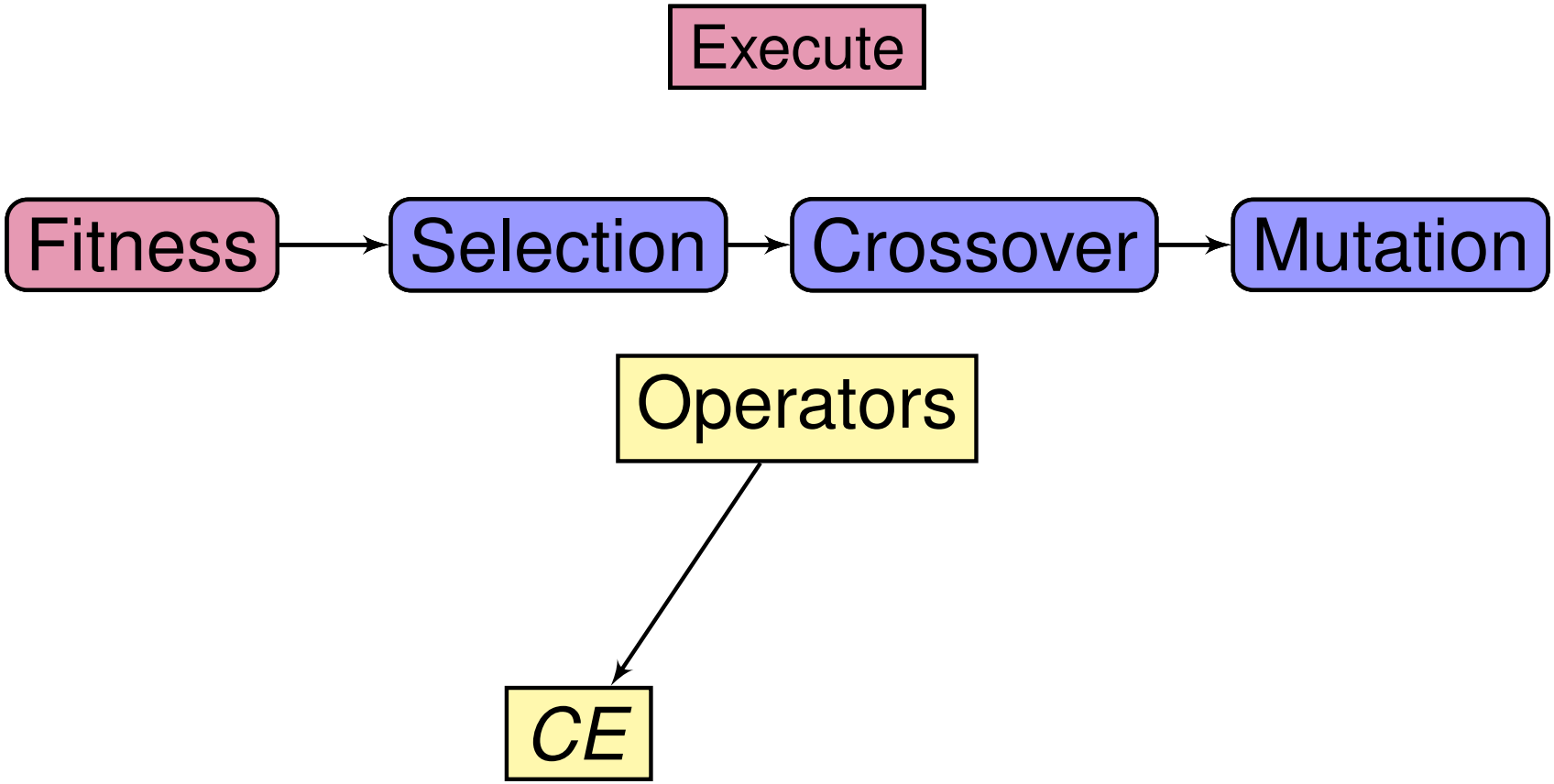
Execute the phases until the genetic algorithm **stagnates**

Genetic Algorithm



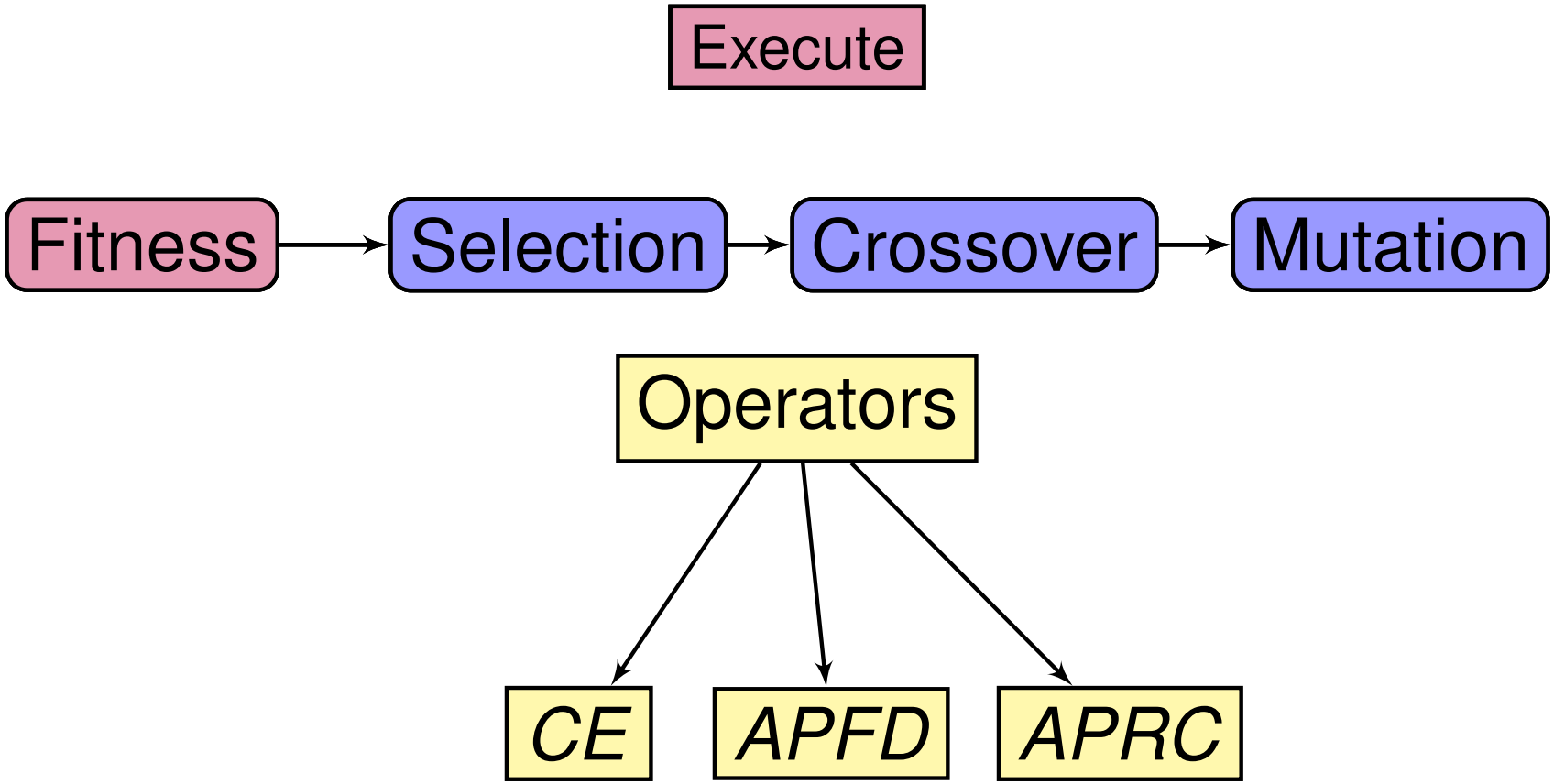
Use a “higher is better” effectiveness metric

Genetic Algorithm



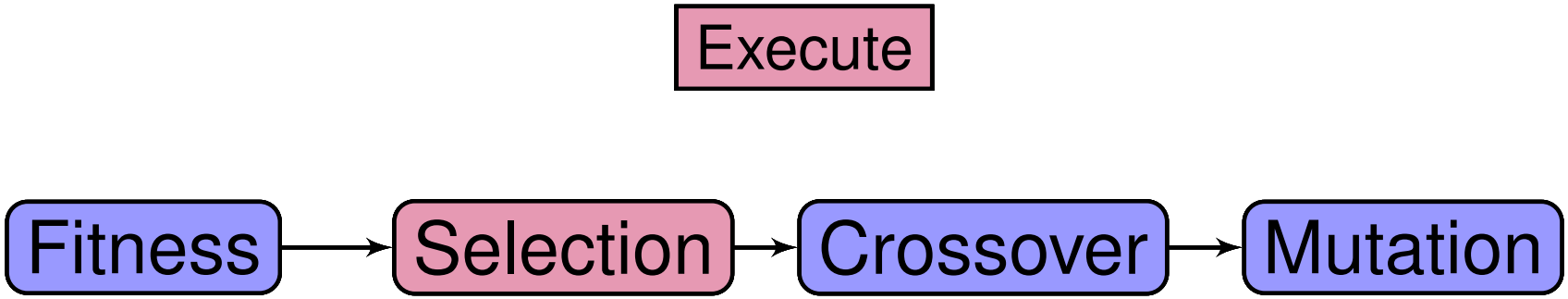
Use a “higher is better” effectiveness metric

Genetic Algorithm



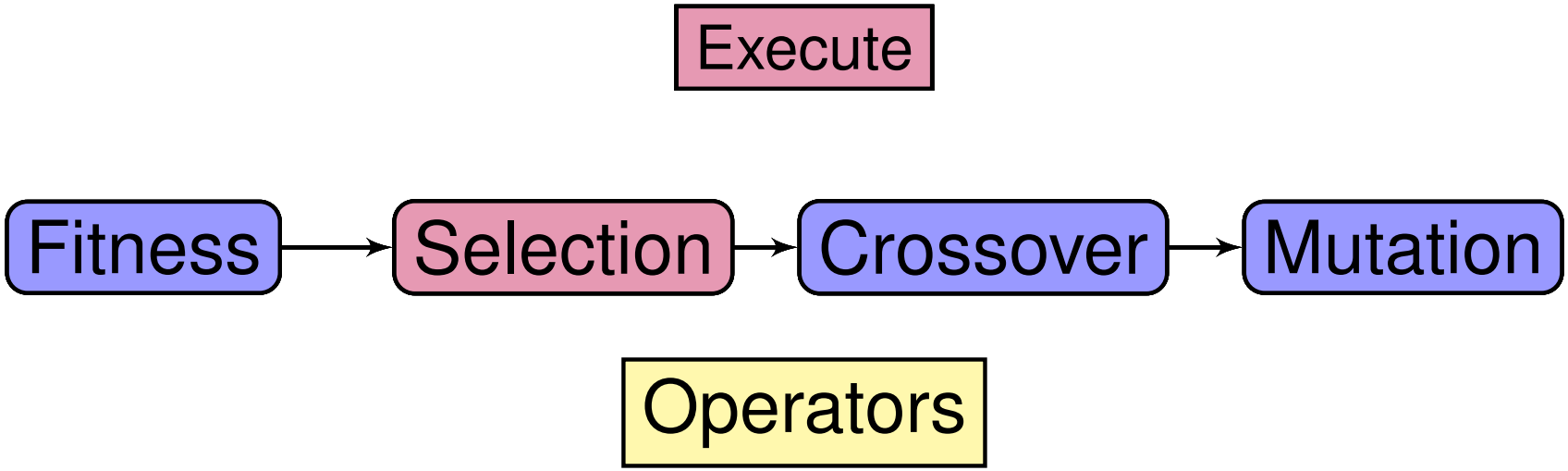
Use a “higher is better” effectiveness metric

Genetic Algorithm



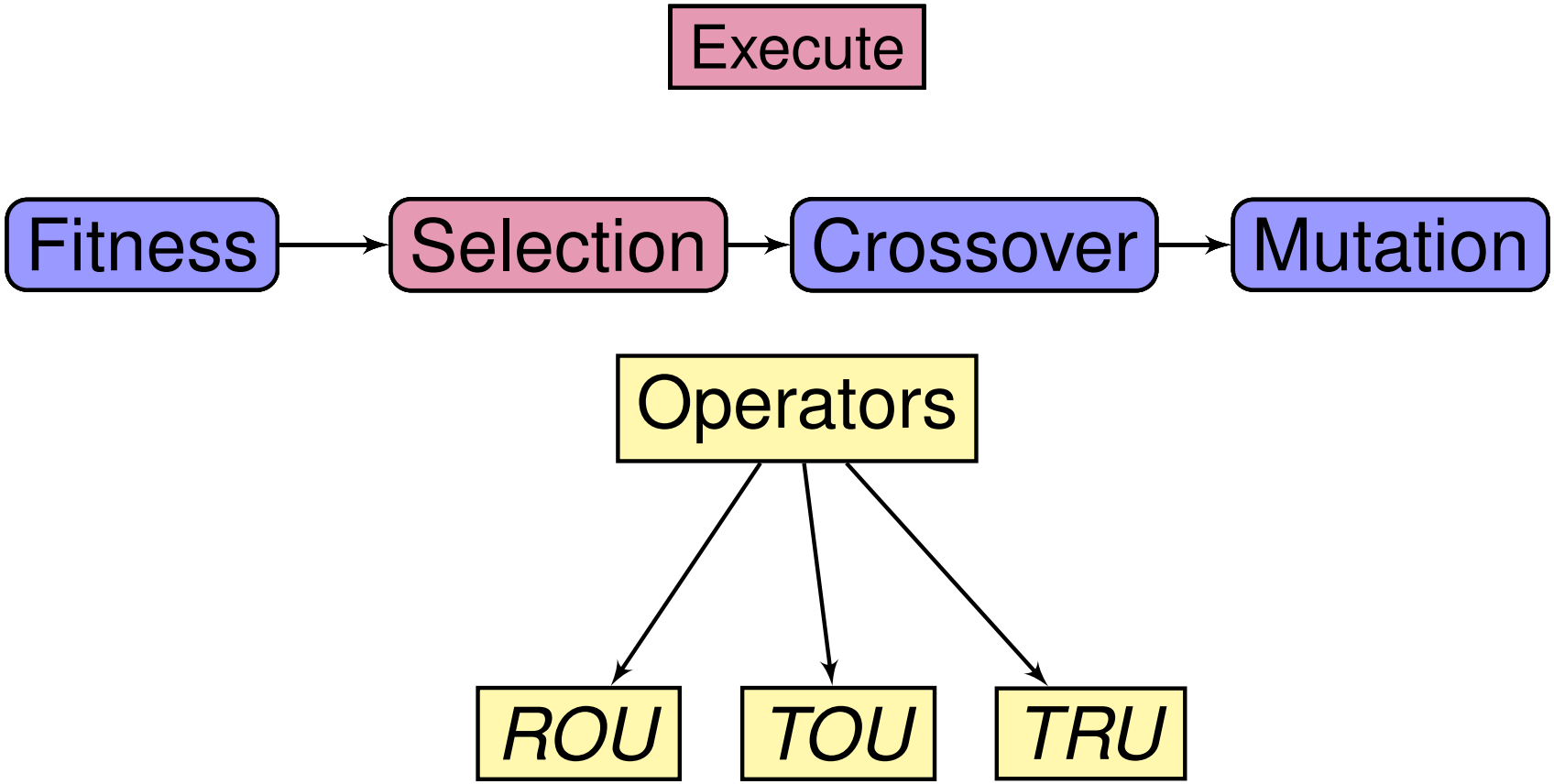
Choose orderings to become **parents** of next generation

Genetic Algorithm



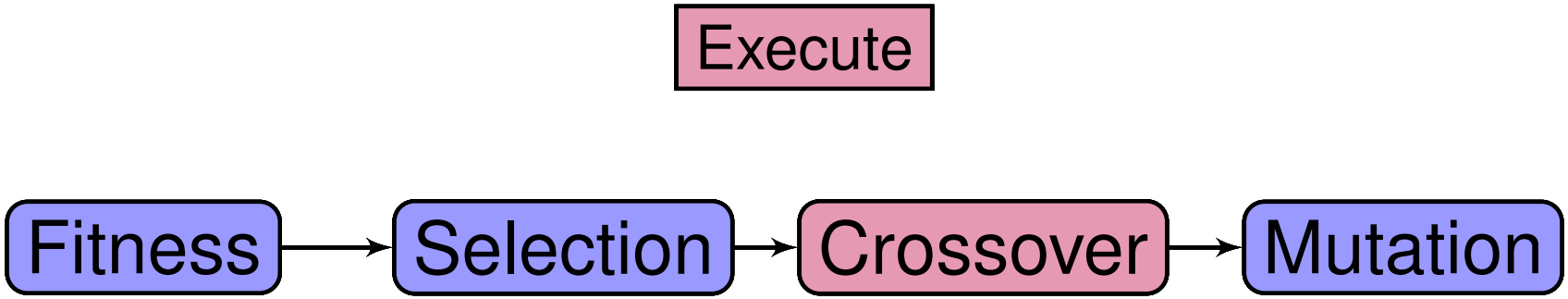
Choose orderings to become **parents** of next generation

Genetic Algorithm



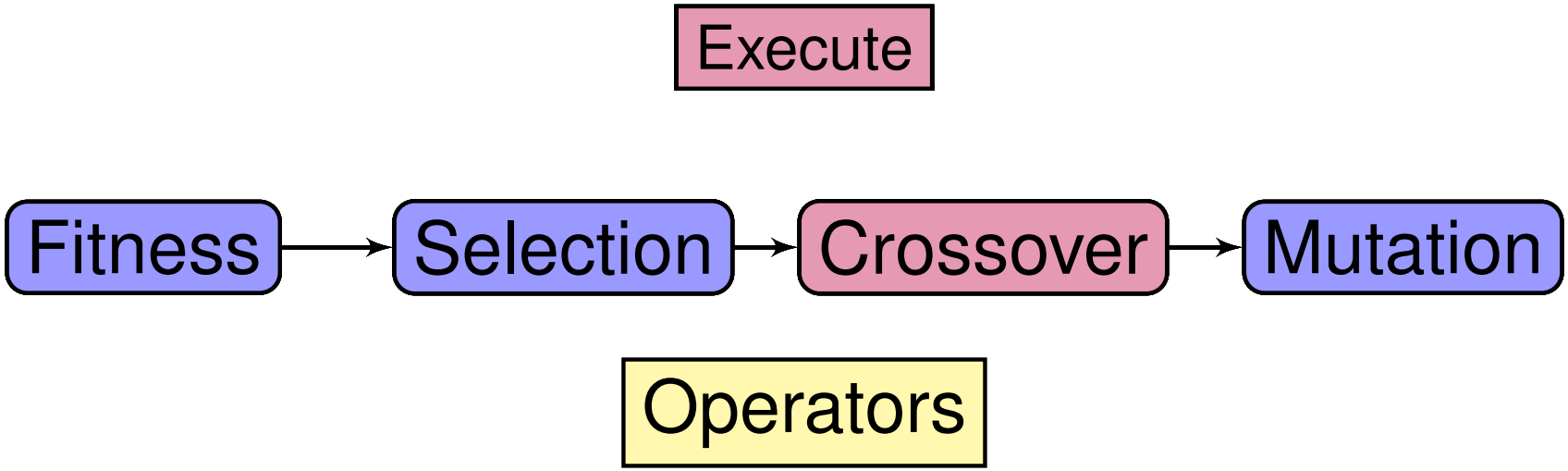
Choose orderings to become **parents** of next generation

Genetic Algorithm



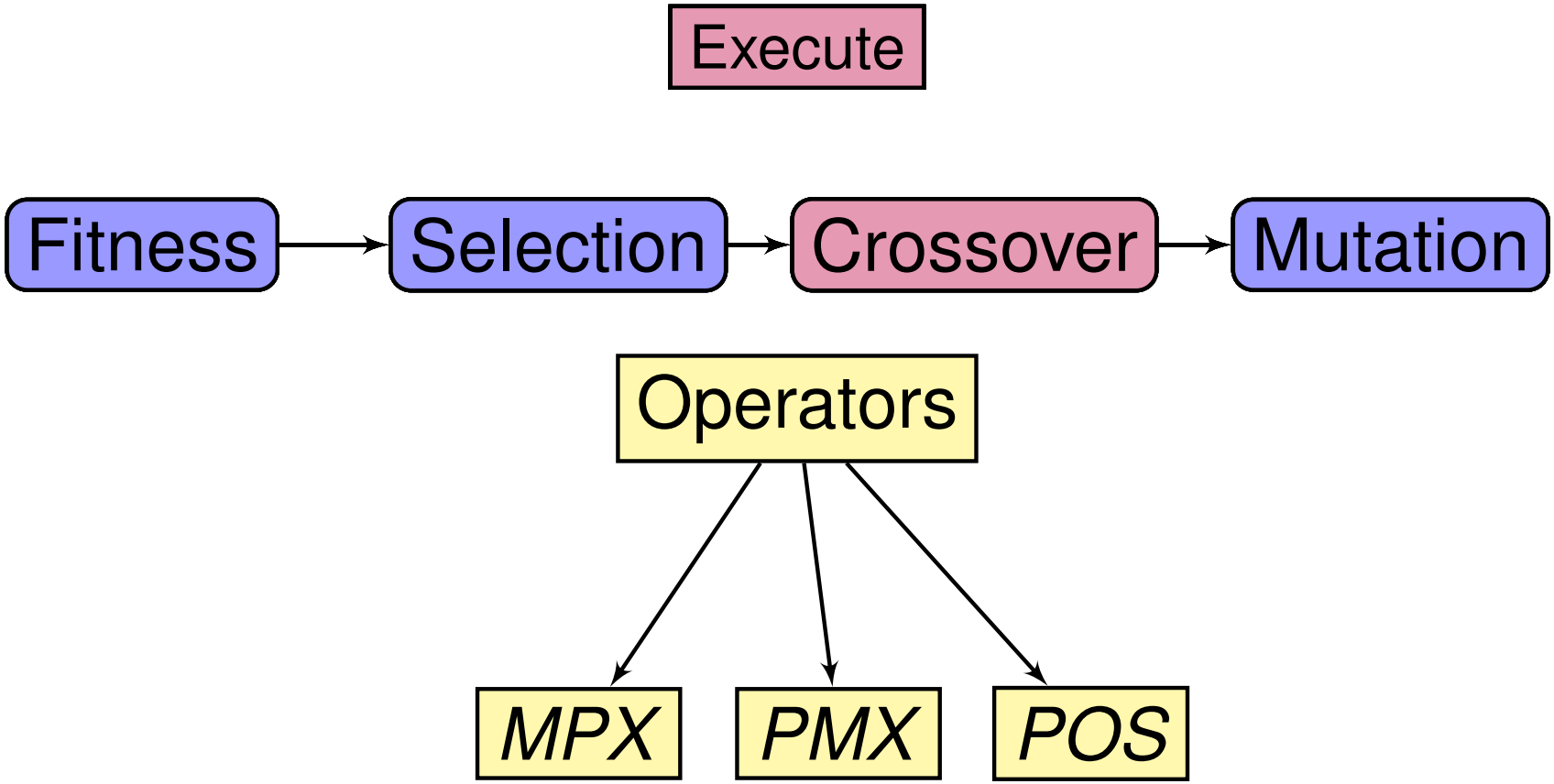
Five possible operators **combine** parents to produce children

Genetic Algorithm



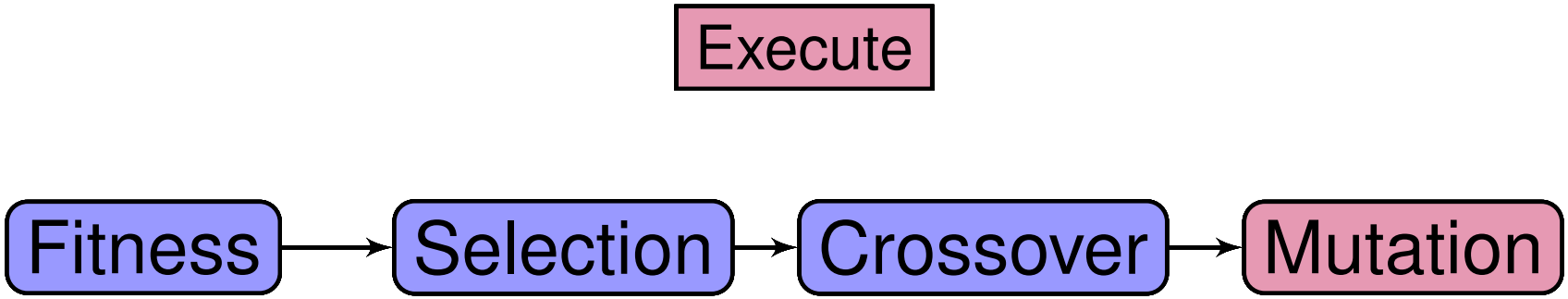
Five possible operators **combine** parents to produce children

Genetic Algorithm



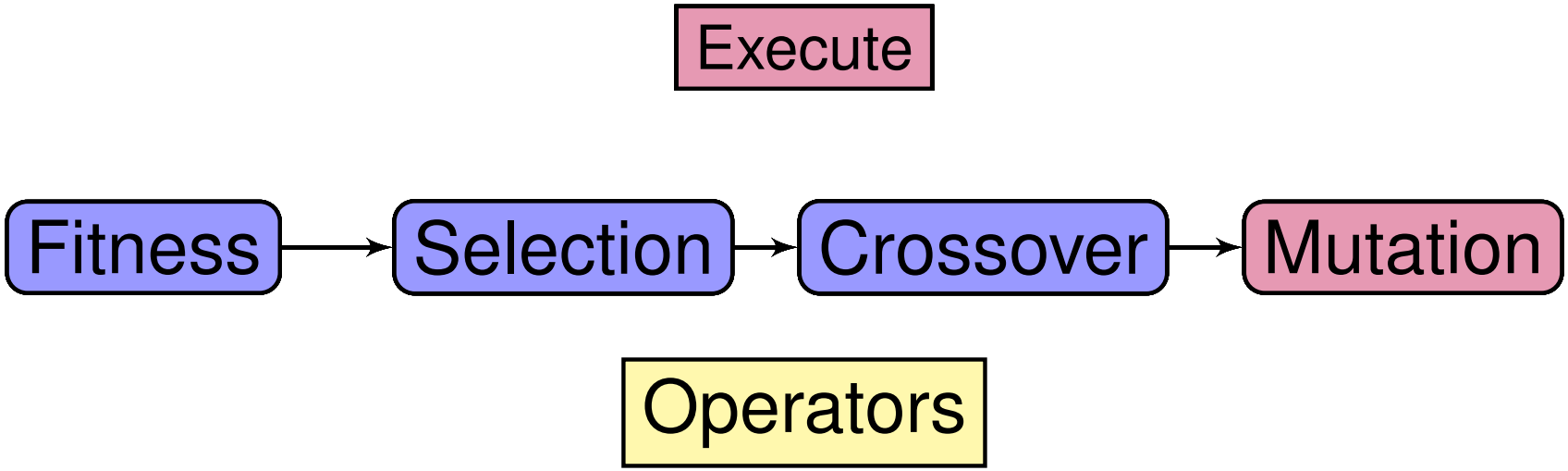
Five possible operators **combine** parents to produce children

Genetic Algorithm



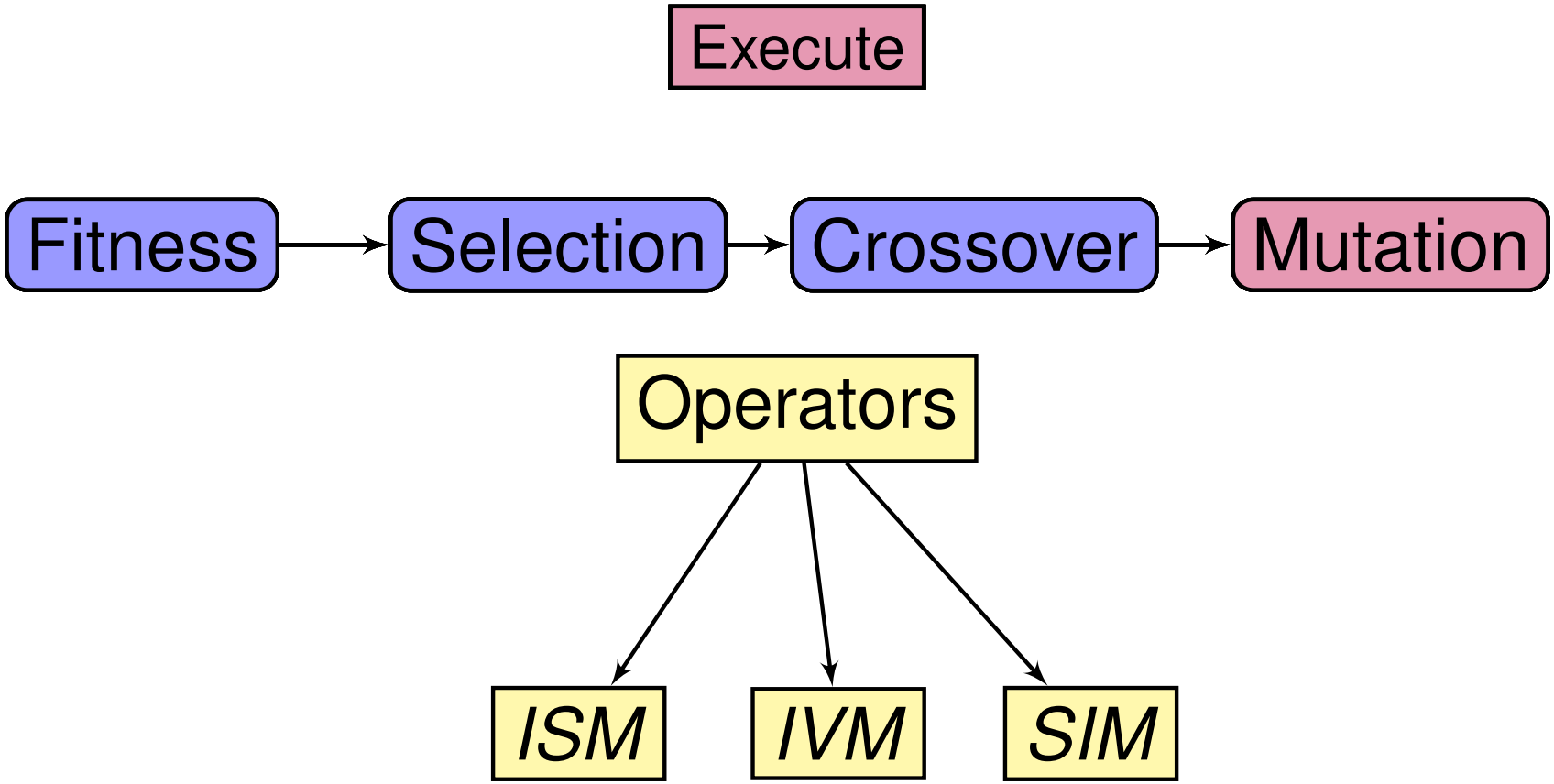
Six possible operators make random **changes** to orderings

Genetic Algorithm



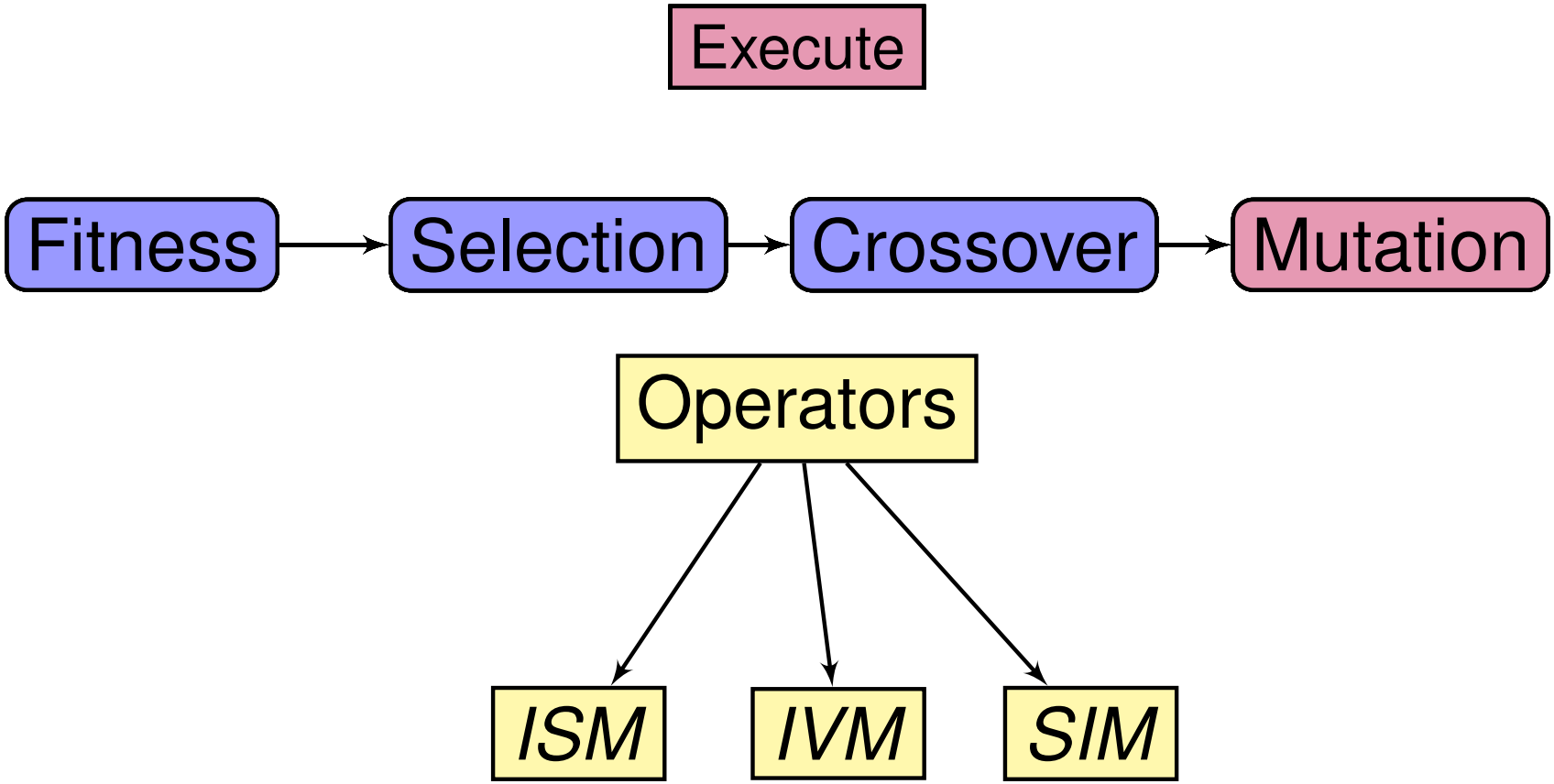
Six possible operators make random **changes** to orderings

Genetic Algorithm



Six possible operators make random **changes** to orderings

Genetic Algorithm



See (Conrad et. al, GECCO 2010) for additional details

Hands-on: Running the Search-Based Algorithms

1

Run the Hill Climbing Prioritization Algorithm

```
HC_FA(lFM=makeLogFM(read.table(
    "reqMatrices/StatementCoverageMatrices/
    Sudoku_1_line_true_Coverage.dat")),
    NG="NG_FS",Seed=100)
```

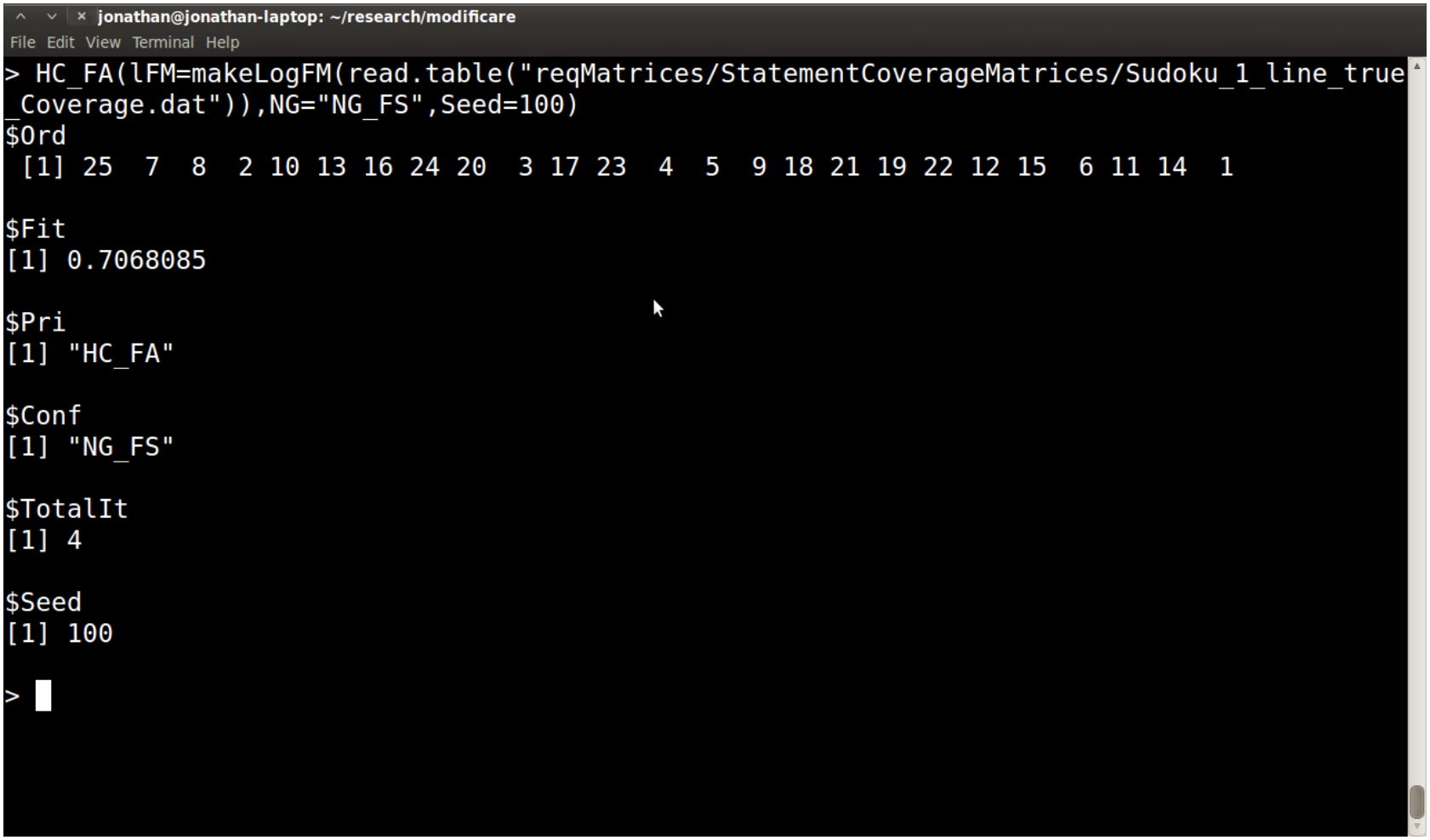
Hands-on: Running the Search-Based Algorithms

2

Run the Genetic Prioritization Algorithm

```
GA(lFM=makeLogFM(read.table("reqMatrices/
    StatementCoverageMatrices/
    Sudoku_1_line_true_Coverage.dat")),
    cChildDens=.6,cMutRate=.2,cPopSize=20,
    CrossOp="CO_OX1",MutOp="MO_DM",
    SelOp="SO_ROU", TransOp="TO_EXP",
    TermCond="TermTotalIt",
    EndCond=10,Seed=100)
```

Screenshot: Running the Hill Climbing Algorithm



Screenshot: Running the Genetic Algorithm

jonathan@jonathan-laptop: ~/research/modificare
File Edit View Terminal Help

```
> GA(lFM=makeLogFM(read.table("reqMatrices/StatementCoverageMatrices/Sudoku_1_line_true_Co
verage.dat")),cChildDens=.6,cMutRate=.2,cPopSize=20,CrossOp="CO_OX1",MutOp="MO_DM",SelOp="
SO_ROU",TransOp="TO_EXP",TermCond="TermTotalIt",EndCond=10,Seed=100)
$Ord
 [1]  2  5 24  9 10  1 14 18  6 11 13 23  7 21  8  3 22 16 19  4 17 15 25 20 12

$Fit
[1] 0.752766

$cPopSize
[1] 20

$cChildDens
[1] 0.6

$cMutRate
[1] 0.2

$CrossOp
[1] "CO_OX1"

$MutOp
[1] "MO_DM"

$SelOp
```

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
1: read(”Enter 3 numbers:”,x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓
4: if(x< y)	✓	✓			✓		✓	✓
5: m=y;		✓						
6: else if(x<z)	✓				✓		✓	✓
7: m=y; // *** bug ***	✓						✓	✓
8: else			✓	✓		✓		
9: if(x>y)			✓	✓		✓		
10: m=y;			✓			✓		
11: else if (x>z)				✓				
12: m=z;								
13: print(”Middle number is:”,m);	✓	✓	✓	✓	✓	✓	✓	✓
} Pass/Fail Status	P	P	P	P	P	P	F	F

Fault Localization

$$suspiciousness(s) = \frac{failed(s)}{\sqrt{totalfailed * (failed(s) + passed(s))}}$$

Fault Localization

$$suspiciousness(s) = \frac{failed(s)}{\sqrt{totalfailed * (failed(s) + passed(s))}}$$

Fault Localization

$$suspiciousness(s) = \frac{failed(s)}{\sqrt{totalfailed * (failed(s) + passed(s))}}$$

Fault Localization

$$suspiciousness(s) = \frac{failed(s)}{\sqrt{totalfailed * (failed(s) + passed(s))}}$$

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Susp	Rank
1: read(”Enter 3 numbers:”,x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
4: if(x< y)	✓	✓			✓		✓	✓	0.63	3
5: m=y;		✓							0.0	13
6: else if(x<z)	✓				✓		✓	✓	0.71	2
7: m=y; // *** bug ***	✓						✓	✓	0.82	1
8: else			✓	✓		✓			0.0	13
9: if(x>y)			✓	✓		✓			0.0	13
10: m=y;			✓			✓			0.0	13
11: else if (x>z)				✓					0.0	13
12: m=z;									0.0	13
13: print(”Middle number is:”,m);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
} Pass/Fail Status	P	P	P	P	P	P	F	F		

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Susp	Rank
1: read(”Enter 3 numbers:”,x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
4: if(x< y)	✓	✓			✓		✓	✓	0.63	3
5: m=y;		✓							0.0	13
6: else if(x<z)	✓				✓		✓	✓	0.71	2
7: m=y; // *** bug ***	✓						✓	✓	0.82	1
8: else			✓	✓		✓			0.0	13
9: if(x>y)			✓	✓		✓			0.0	13
10: m=y;			✓			✓			0.0	13
11: else if (x>z)				✓					0.0	13
12: m=z;									0.0	13
13: print(”Middle number is:”,m);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
} Pass/Fail Status	P	P	P	P	P	P	F	F		

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Susp	Rank
1: read(”Enter 3 numbers:”,x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
4: if(x< y)	✓	✓			✓		✓	✓	0.63	3
5: m=y;		✓							0.0	13
6: else if(x<z)	✓				✓		✓	✓	0.71	2
7: m=y; // *** bug ***	✓						✓	✓	0.82	1
8: else			✓	✓		✓			0.0	13
9: if(x>y)			✓	✓		✓			0.0	13
10: m=y;			✓			✓			0.0	13
11: else if (x>z)				✓					0.0	13
12: m=z;									0.0	13
13: print(”Middle number is:”,m);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
} Pass/Fail Status	P	P	P	P	P	P	F	F		

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Susp	Rank
1: read("Enter 3 numbers:",x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
4: if(x< y)	✓	✓			✓		✓	✓	0.63	3
5: m=y;		✓							0.0	13
6: else if(x<z)	✓				✓		✓	✓	0.71	2
7: m=y; // *** bug ***	✓						✓	✓	0.82	1
8: else			✓	✓		✓			0.0	13
9: if(x>y)			✓	✓		✓			0.0	13
10: m=y;			✓			✓			0.0	13
11: else if (x>z)				✓					0.0	13
12: m=z;									0.0	13
13: print("Middle number is:",m);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
} Pass/Fail Status	P	P	P	P	P	P	F	F		

Examine the most suspicious statement first

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Susp	Rank
1: read(”Enter 3 numbers:”,x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
4: if(x< y)	✓	✓			✓		✓	✓	0.63	3
5: m=y;		✓							0.0	13
6: else if(x<z)	✓				✓		✓	✓	0.71	2
7: m=y; // *** bug ***	✓						✓	✓	0.82	1
8: else			✓	✓		✓			0.0	13
9: if(x>y)			✓	✓		✓			0.0	13
10: m=y;			✓			✓			0.0	13
11: else if (x>z)				✓					0.0	13
12: m=z;									0.0	13
13: print(”Middle number is:”,m);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
} Pass/Fail Status	P	P	P	P	P	P	F	F		

Assumption: Developer examines source code statements in order

Fault Localization

mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Susp	Rank
1: read(”Enter 3 numbers:”,x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
4: if(x< y)	✓	✓			✓		✓	✓	0.63	3
5: m=y;		✓							0.0	13
6: else if(x<z)	✓				✓		✓	✓	0.71	2
7: m=y; // *** bug ***	✓						✓	✓	0.82	1
8: else			✓	✓		✓			0.0	13
9: if(x>y)			✓	✓		✓			0.0	13
10: m=y;			✓			✓			0.0	13
11: else if (x>z)				✓					0.0	13
12: m=z;									0.0	13
13: print(”Middle number is:”,m);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
} Pass/Fail Status	P	P	P	P	P	P	F	F		

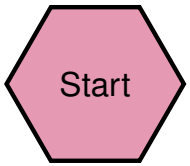
Assumption: Developer can identify faulty statement

Fault Localization

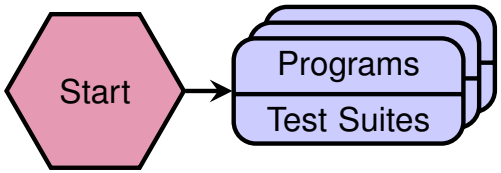
mid() { int x,y,z,m;	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Susp	Rank
1: read(”Enter 3 numbers:”,x,y,z);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
2: m=z;	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
3: if(y<z)	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
4: if(x< y)	✓	✓			✓		✓	✓	0.63	3
5: m=y;		✓							0.0	13
6: else if(x<z)	✓				✓		✓	✓	0.71	2
7: m=y; // *** bug ***	✓						✓	✓	0.82	1
8: else			✓	✓		✓			0.0	13
9: if(x>y)			✓	✓		✓			0.0	13
10: m=y;			✓			✓			0.0	13
11: else if (x>z)				✓					0.0	13
12: m=z;									0.0	13
13: print(”Middle number is:”,m);	✓	✓	✓	✓	✓	✓	✓	✓	0.5	7
} Pass/Fail Status	P	P	P	P	P	P	F	F		

Our framework supports fault localization with a wide variety of metrics

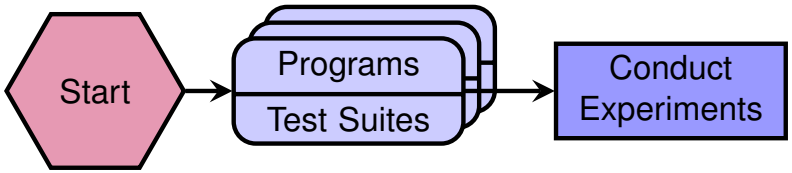
How Do I Evaluate Regression Testing Methods?



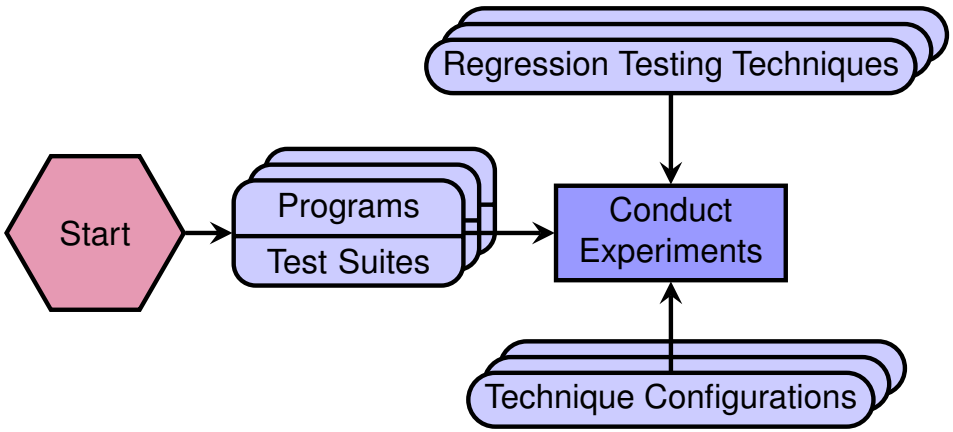
How Do I Evaluate Regression Testing Methods?



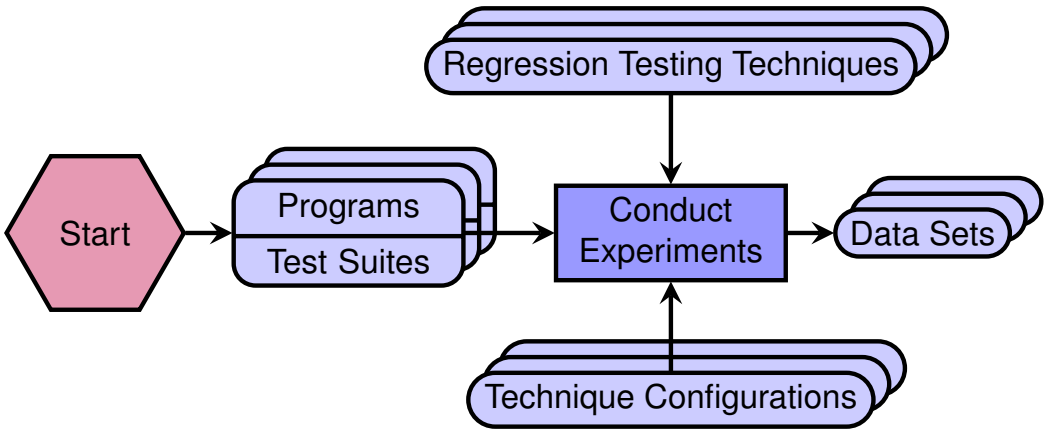
How Do I Evaluate Regression Testing Methods?



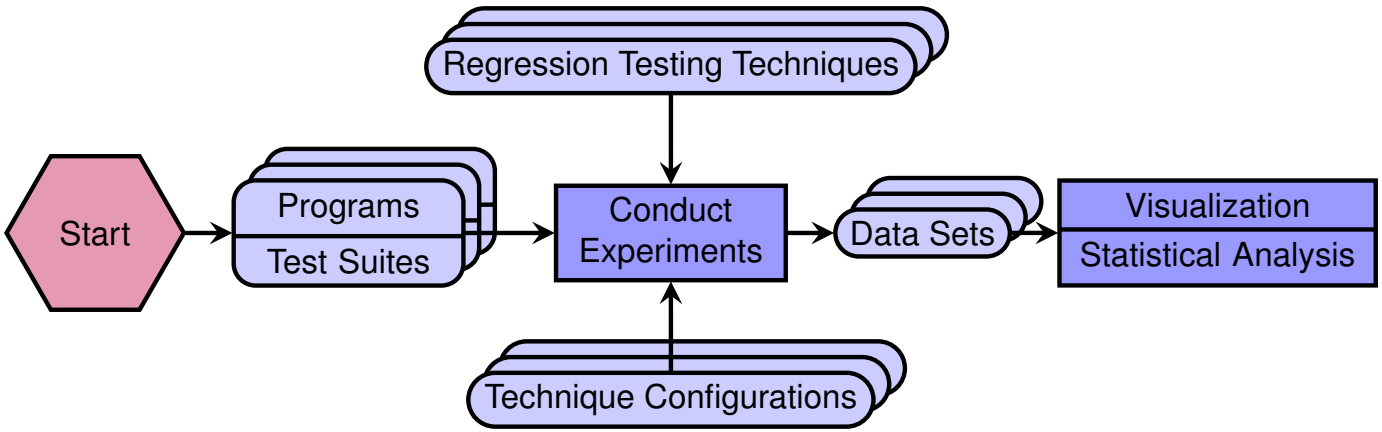
How Do I Evaluate Regression Testing Methods?



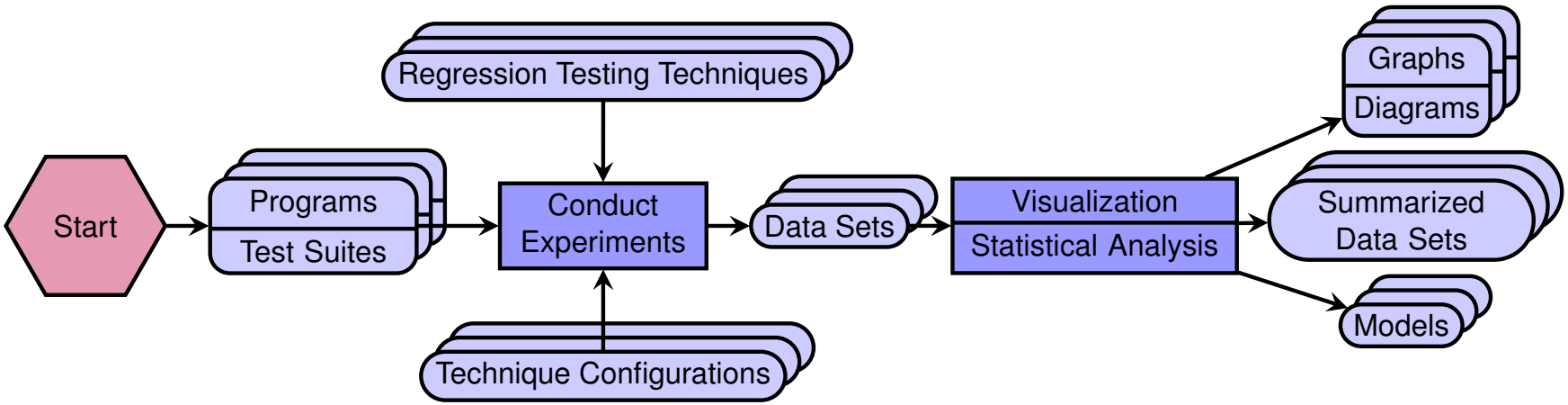
How Do I Evaluate Regression Testing Methods?



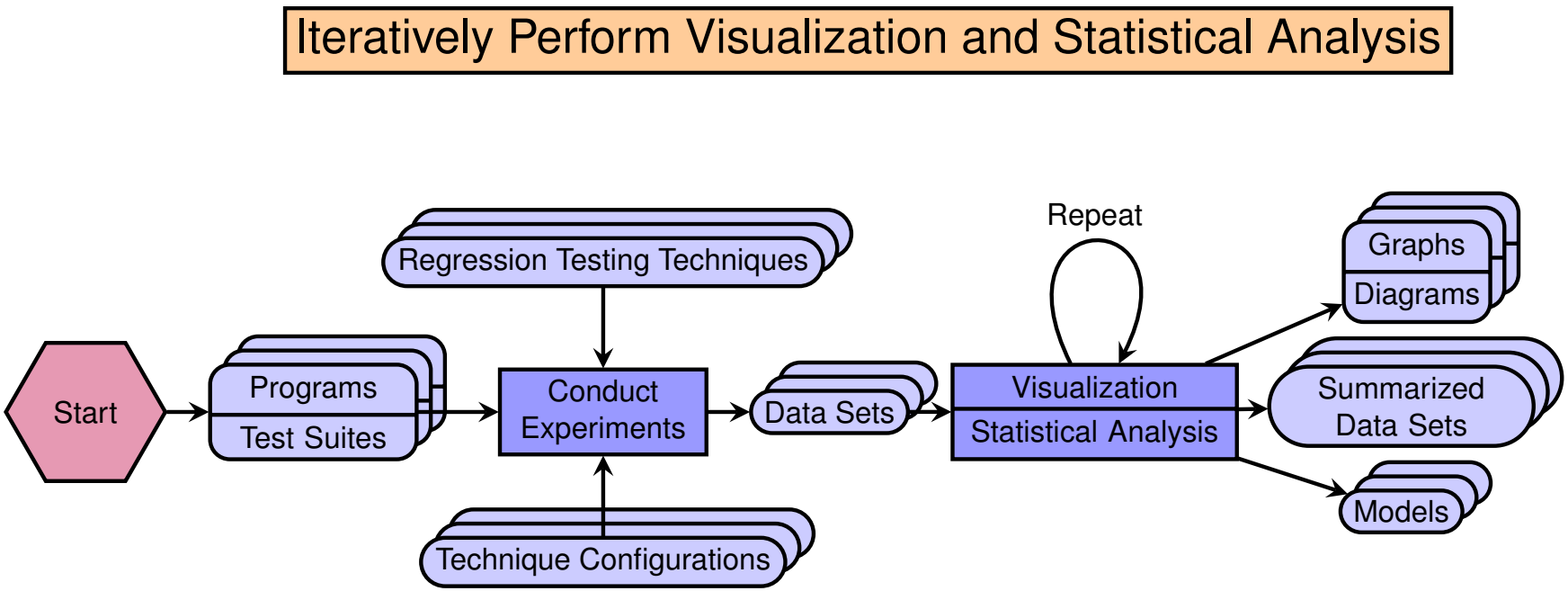
How Do I Evaluate Regression Testing Methods?



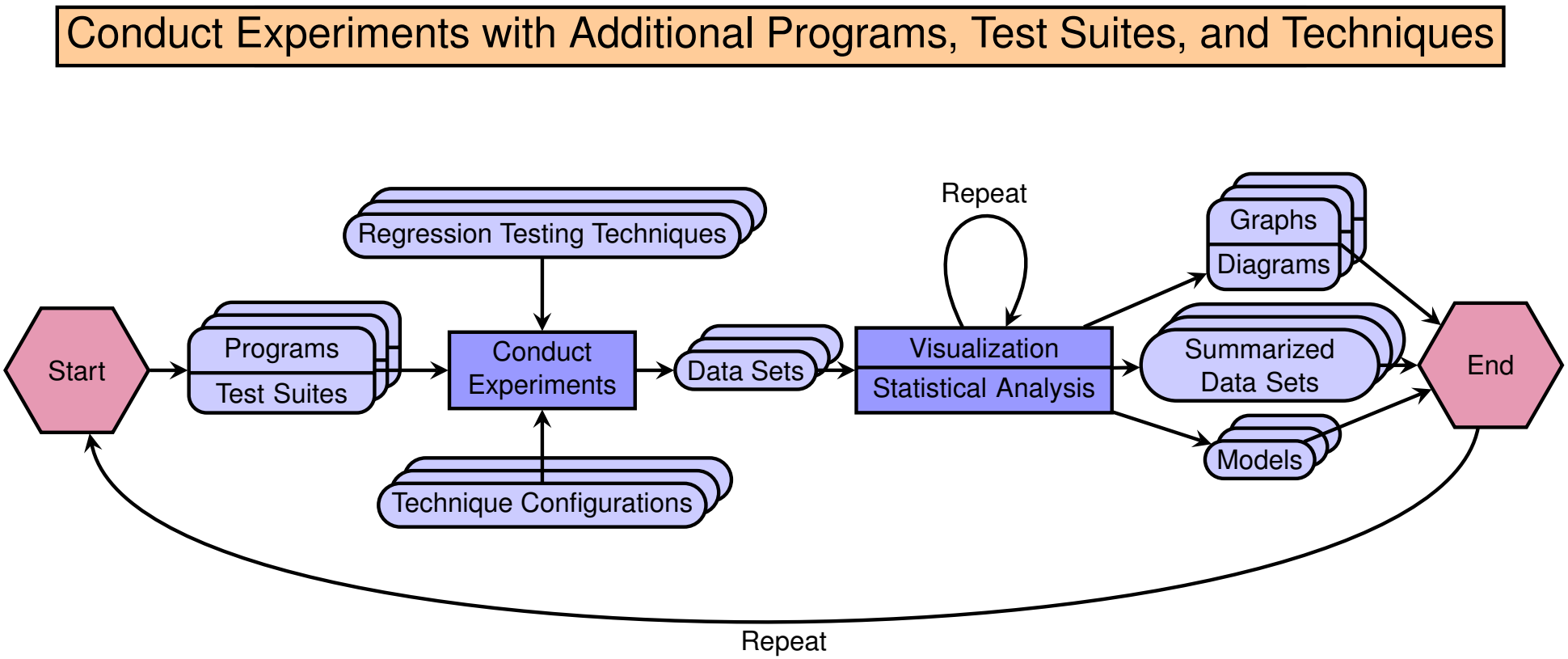
How Do I Evaluate Regression Testing Methods?



How Do I Evaluate Regression Testing Methods?

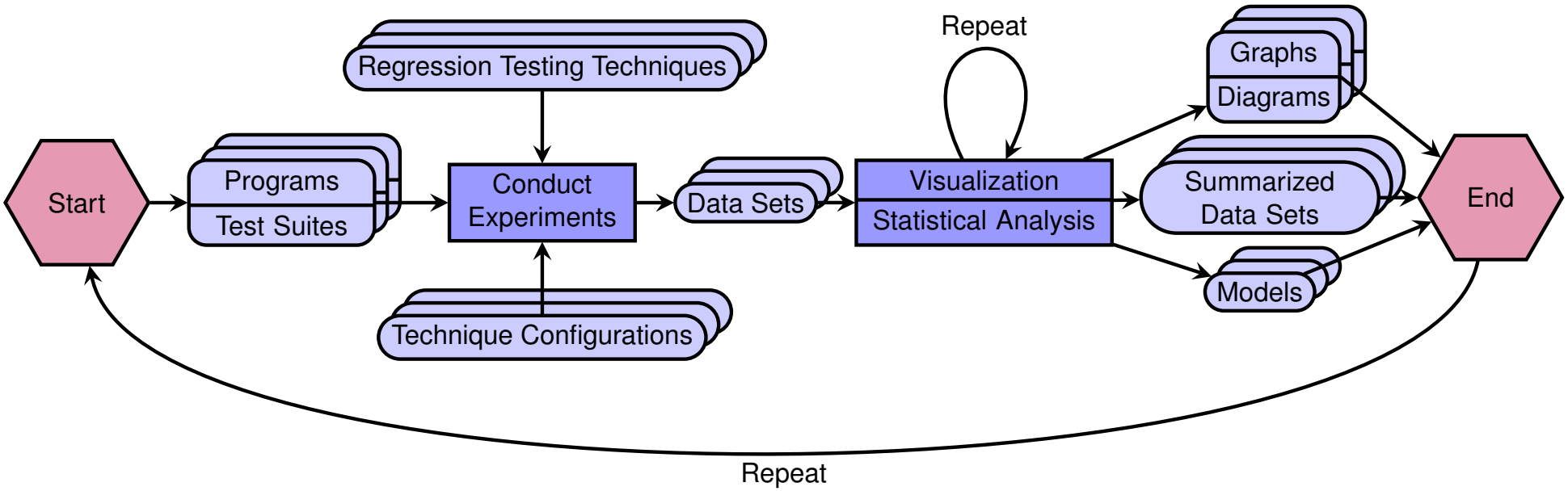


How Do I Evaluate Regression Testing Methods?



How Do I Evaluate Regression Testing Methods?

Conduct Experiments with Additional Programs, Test Suites, and Techniques



Our tools support all of these tasks!

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

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

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

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 Insights

Prioritizer	Fitness	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 Insights

Prioritizer	Fitness	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 Insights

Prioritizer	Fitness	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 Insights

Prioritizer	Fitness	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 Insights

Prioritizer	Fitness	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 Insights

Prioritizer	Fitness	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 Insights

Prioritizer	Fitness	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 Insights

Prioritizer	Fitness	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 Insights

Prioritizer	Fitness	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 Insights

Prioritizer	Fitness	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 Insights

Prioritizer	Fitness	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 fitness increase may result in a large runtime increase

Empirical Insights

Prioritizer	Fitness	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

Our tools support efficient experimentation through parallelism

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
- 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
- Develop statistically meaningful empirical results

Regression Testing: Theoretical Underpinnings, Practical Techniques, and Empirical Insights

Gregory M. Kapfhammer* and Jonathan Miller Kauffman†

*http://www.cs.allegheny.edu/~gkapfham/
†https://sites.google.com/a/allegheny.edu/jonathan-kauffman/

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

ALLEGHENY COLLEGE