Class 17

- Questions/comments
- Graders for Problem Set 6 (4); Graders for Problem set 7 (2-3) (solutions for all); will be posted on T-square
- Regression testing, Instrumentation
- Final project presentations: Dec 1, 3; 4:35-6:45
- Assign (see Schedule for links)
  - Problem Set 7 discuss
  - Readings

Discussion

- Mutation analysis and testing
  - Mutant operator
  - Mutate
  - Mutant
  - Killing mutants
  - Mutation adequacy
  - Equivalent mutants

Discussion

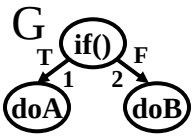
- Regression testing
  - What is it?
  - What are scenarios in which it is used?
  - Test selection
    - DejaVu algorithm?
    - Safety?
    - Precision?
    - Fault-revealing test cases?
    - Modification-revealing test cases?

DejaVu Algorithm

- Algorithm can be used at various levels
  - Branches in CFG
  - Methods, procedures in program
  - Classes
  - UML diagrams
  - Other representations of program

General Test Selection with DejaVu

1. Construct representation G for P

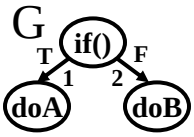


P can be
- procedural
- object-oriented
- database
- Web-based
- other

G can be
- low level—e.g., code
- higher level—e.g., UML diagrams, state-transition diagrams, architectural diagrams
- other information—e.g., CVS repository

General Test Selection with DejaVu

1. Construct representation G for P



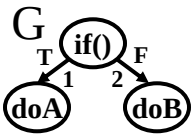
2. Associate test cases in T with entities in G

M

edge \ TC	t1	t2	t3
e1	X		
e2		X	X

General Test Selection with DejaVu

1. Construct representation G for P

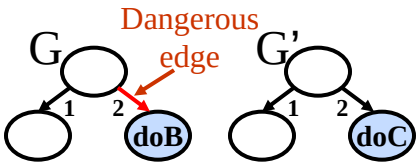


2. Associate test cases in T with entities in G

M

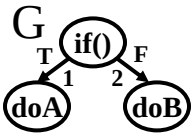
edge \ TC	t1	t2	t3
e1	X		
e2		X	X

3. Build G' and compare G and G' to find dangerous entities



General Test Selection with DejaVu

1. Construct representation G for P

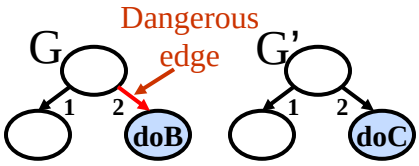


2. Associate test cases in T with entities in G

M

edge \ TC	t1	t2	t3
e1	X		
e2		X	X

3. Build G' and compare G and G' to find dangerous entities

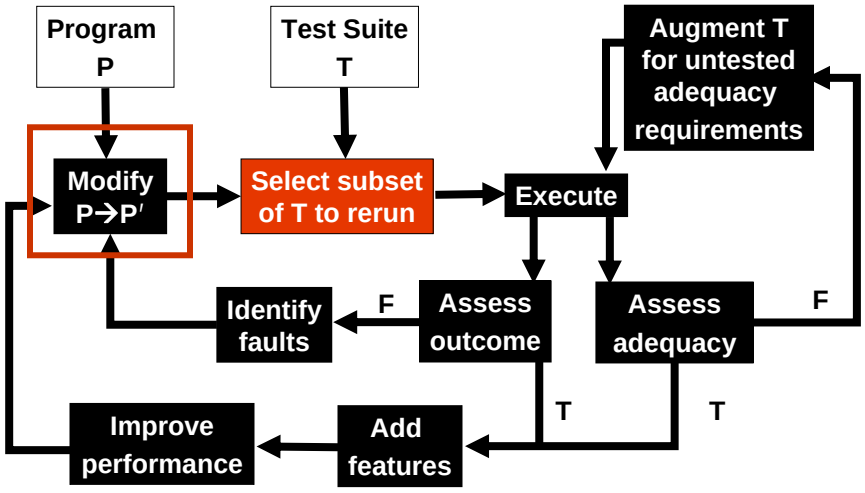


4. Select test cases based on dangerous entities

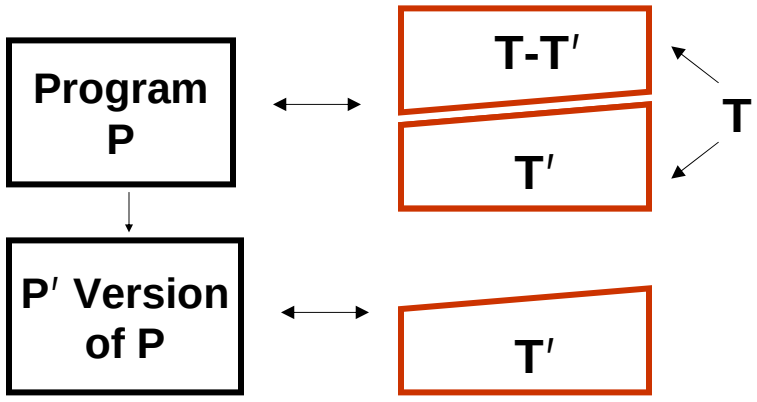
M

edge \ TC	t1	t2	t3
e1	X		
e2		X	X

Select Subset of T to Rerun

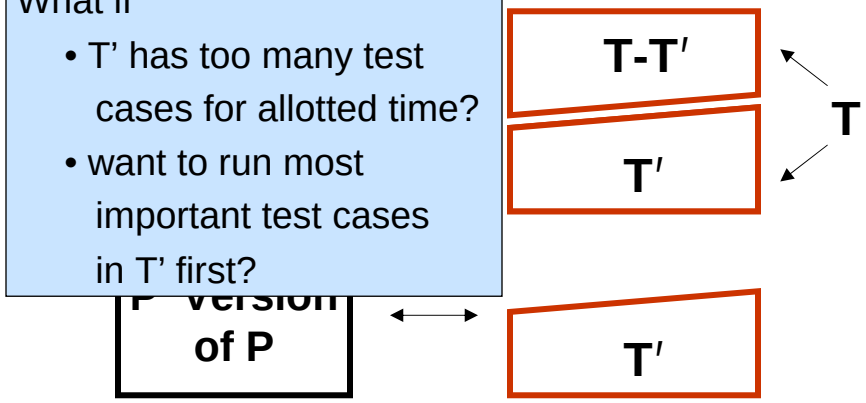


Test-Suite Prioritization Problem



Test-Suite Prioritization Problem

What if
- T' has too many test cases for allotted time?
- want to run most important test cases in T' first?

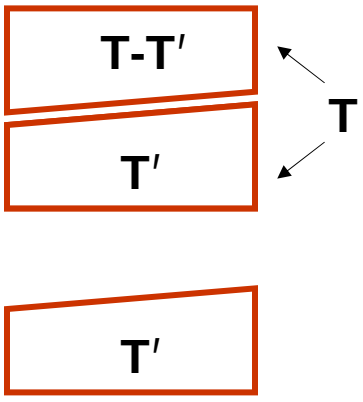


Test-Suite Prioritization Problem

What if
- T' has too many test cases for allotted time?
- want to run most important test cases in T' first?

Solution

Order (prioritize) T'
- find faults earlier
- get coverage earlier
- etc.



Test-Suite Prioritization Problem

What if
- T' has too many test cases for allotted time?
- want to run most important test cases in T' first?

Solution

Order (prioritize) T'
- find faults earlier
- get coverage earlier
- etc.

t1, t2,…,tn

Test-Suite Prioritization Problem

What if
- T' has too many test cases for allotted time?
- want to run most important test cases in T' first?

Solution

Order (prioritize) T'
- find faults earlier
- get coverage earlier
- etc.

t1, t2,…,tn

t1, t3,…,tn

. . .

t2, t1,…,tn

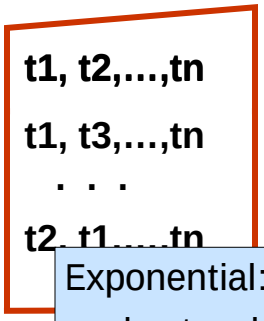
. . .

Test-Suite Prioritization Problem

What if
- T' has too many test cases for allotted time?
- want to run most important test cases in T' first?

Solution

Order (prioritize) T'
- find faults earlier
- get coverage earlier
- etc.



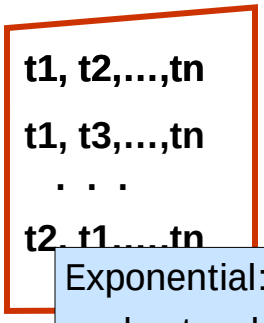
Exponential:
best ordering

Test-Suite Prioritization Problem

What if
- T' has too many test cases for allotted time?
- want to run most important test cases in T' first?

Solution

Order (prioritize) T'
- find faults earlier
- get coverage earlier
- etc.



Exponential:
best ordering
Heuristics:
good ordering

Test-Suite Prioritization Problem

Given

T, a test suite

PT, the set of permutations of T (prioritizations)

f: PT→Reals (f yields prioritization award value)

Problem

Find T' in PT such that for all T'' in PT where T'' does not equal T', f(T') > f(T'') (i.e., find the best prioritization)

Objectives for Test-suite Prioritization

- Increase likelihood of revealing faults earlier in regression testing
- Increase likelihood of revealing high-risk faults earlier in regression testing
- Increase likelihood of revealing regression errors for specific code changes
- Increase code coverage earlier in regression testing

Objectives for Test-suite Prioritization

- Increase likelihood of revealing faults earlier in regression testing
- Increase likelihood of revealing high-risk faults earlier in regression testing
- Increase likelihood of revealing regression errors for specific code changes
- Increase code coverage earlier in regression testing

Prioritizations We Used

T1	unordered
T2	random
T3	optimal
T4	branch-total
T5	branch-additional
T6	FEP-total
T7	FEP-additional
T8	statement-total
T9	statement-additional

Prioritizations We Used

T1	unordered
T2	random
T3	optimal
T4	branch-total
T5	branch-additional
T6	FEP-total
T7	FEP-additional
T8	statement-total
T9	statement-additional

Fault Exposing Potential (FEP)

The fault-exposing potential (FEP) for a statement *s* in a program *P* is the probability that, when *P* is executed with a test, a fault at *s* will cause *P* to fail

Fault Exposing Potential

Voas developed the PIE model for *P* and input distribution *D* as a measure of FEP

- E -- the probability that s will be executed by *d,* a random selection from D
- I -- the probability that the state at *s* will be changed by *d*
- P -- the probability that the modified state will propagate to output (cause a failure)

Approximating FEP

We use mutation analysis to approximate FEP

In mutation analysis, we create many versions of a program, each containing one syntactic change and try to "kill" the mutants with the test suite

The mutation score is the percentage of mutants "killed" by the test suite

Average Percentage of Fault Detection (APFD)

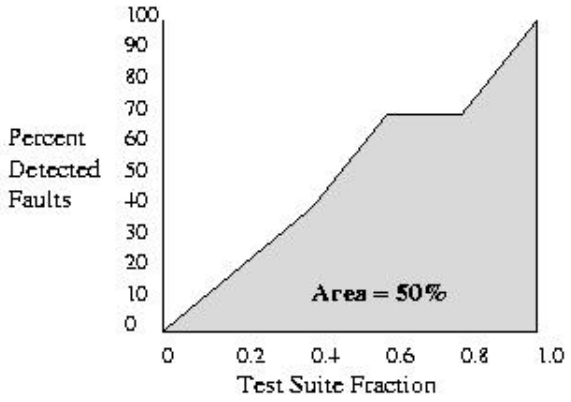
- Want to measure how rapidly a prioritized test suite detects faults
- We use a weighted average of the percentage of faults detected over the life of the test suite -- average percentage of fault detection (APFD)
- We think that higher APFD means a faster (better) fault detection rate

Example

Test	Fault									
	1	2	3	4	5	6	7	8	9	10
A	X				X					
B	X				X	X	X			
C	X	X	X	X	X	X	X			
D					X					
E							X	X		X

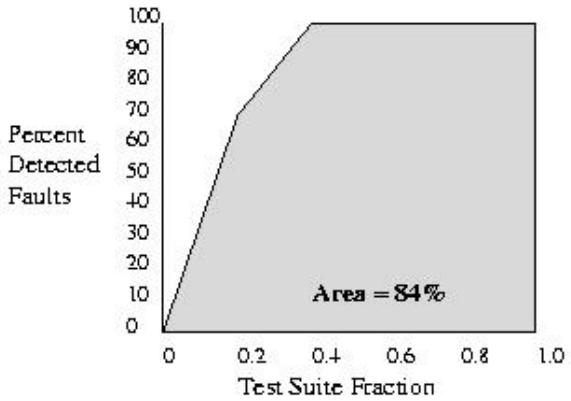
Test Order: A-B-C-D-E

A	20%
B	40%
C	70%
D	70%
E	100%



Test Order: C-E-B-A-D

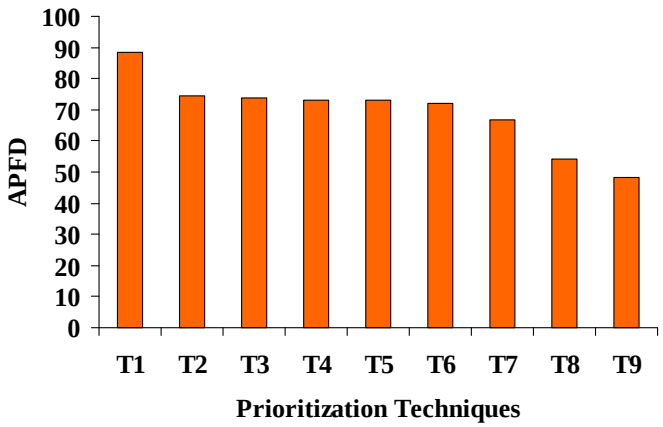
C	70%
E	100%
B	100%
A	100%
D	100%



Study: Relative Effectiveness Of Prioritization Techniques

Siemens: Seven C programs (300LOC), Siemens Labs, 7-42 versions, 1000-5000 tests (same subjects we used in Studies 1 & 2)

Results of Prioritization Study



Results of Our Experiments

- Test prioritization can substantially improve the rate of fault detection of test suites. This result is true for all heuristics we studied
- Overall, additional FEP prioritization outperformed all prioritization techniques but not significantly (this was contrary to our initial belief about FEP)

Results of Our Experiments

- Total branch (T4) outperforms additional branch (T5) and total statement (T8) outperforms additional statement (T9)
- Many of the heuristics performed the same statistically and even for those that differ, the difference is not great

Test Selection, Prioritization in Industry

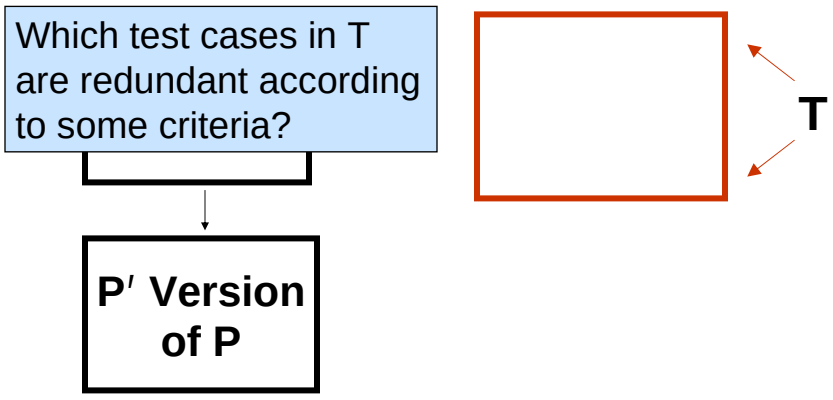
In 2002 OOPSLA talk, Bill Gates

- in-house tool—selects and prioritizes regression tests
- “the impact had been very dramatic."

In 2004 talk at Georgia Tech, Jim Gray

- prioritizing regression tests used at MS
- significant impact

Test-suite Reduction

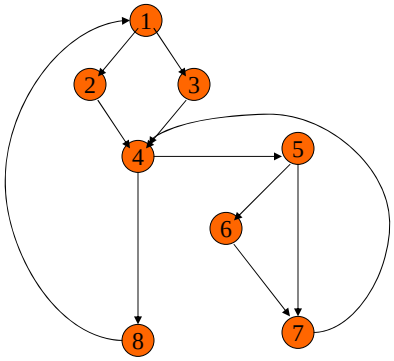


Test-suite Reduction

Which test cases in T are redundant according to some criteria?

Solution:
Identify and remove redundant test cases from T

Test-suite Reduction



Test cases:

T1: 1,2,4,5,6,7,4,5,6,7,4,8

T2: 1,2,4,5,6,7,4,5,7,4,8

T3: 1,2,4,8

T4: 1,3,4,8

T5: 1,3,4,5,6,7,4,5,6,7,4,8

Rest on board

Execution Tracing and Profiling

Execution Tracing and Profiling

- Gathering dynamic information about programs
  - Execution tracing
  - Execution profiling
  - Execution coverage
- Instrumenting for tracing, profiling, coverage
  - Postprocessing
  - Online processing
  - Preprocessing

Execution Tracing

Execution Tracing records, as the program executes with a test case (an input to the program), some *sequence* of events that occur. For example:

- the sequence of statements executed when a program is run with a test case
- the sequence of program states associated with statements executed when a program is run with a test case

Execution Profiling

Execution Profiling records, as the program executes with a test case, some *number* of times that an event occurs. For example:

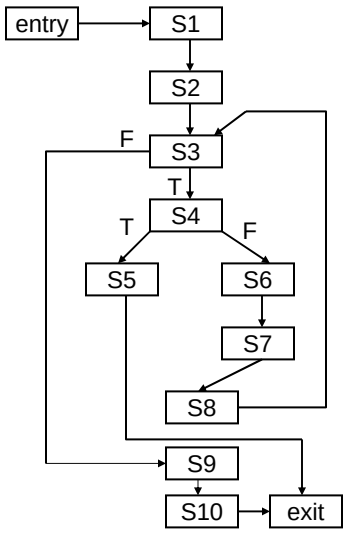
- the number of times a statement is executed when a program is run with test case
- the number of times a variable is changed when the program is run with a test case

Execution Coverage

Execution Coverage records, as the program executes with a test case, *whether* an event occurs. For example:

- whether a statement is executed when a program is run with test case
- whether a branch is taken when the program is run with a test case
- whether a variable is changed when the program is run with a test case

Execution Profiling/Tracing: Example



Suppose program executes entry, S1, S2, S3, S4, S6, S7, S8, S3, S4, S5, exit

Coverage?

Profile?

Trace?

Uses of Coverage, Profiling, Tracing

Instrumentation

Instrumentation is the process of adding code to a program (called probes) such that when the program is executed, it records information about its execution

The program with the probes is called an instrumented program

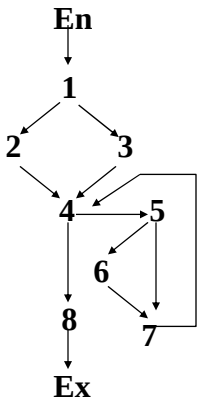
Instrumentation

Types of instrumentation

- Preprocessing
- Online processing
- Postprocessing

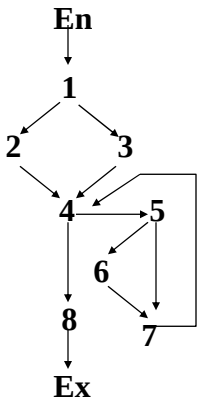
Instrumentation Process

Instrumentation - statement coverage:



Instrumentation Process

Instrumentation – edge coverage:



Instrumentation Process

Instrumentation – path coverage: