Test Suite Prioritization and Reduction by Combinational-based Criteria

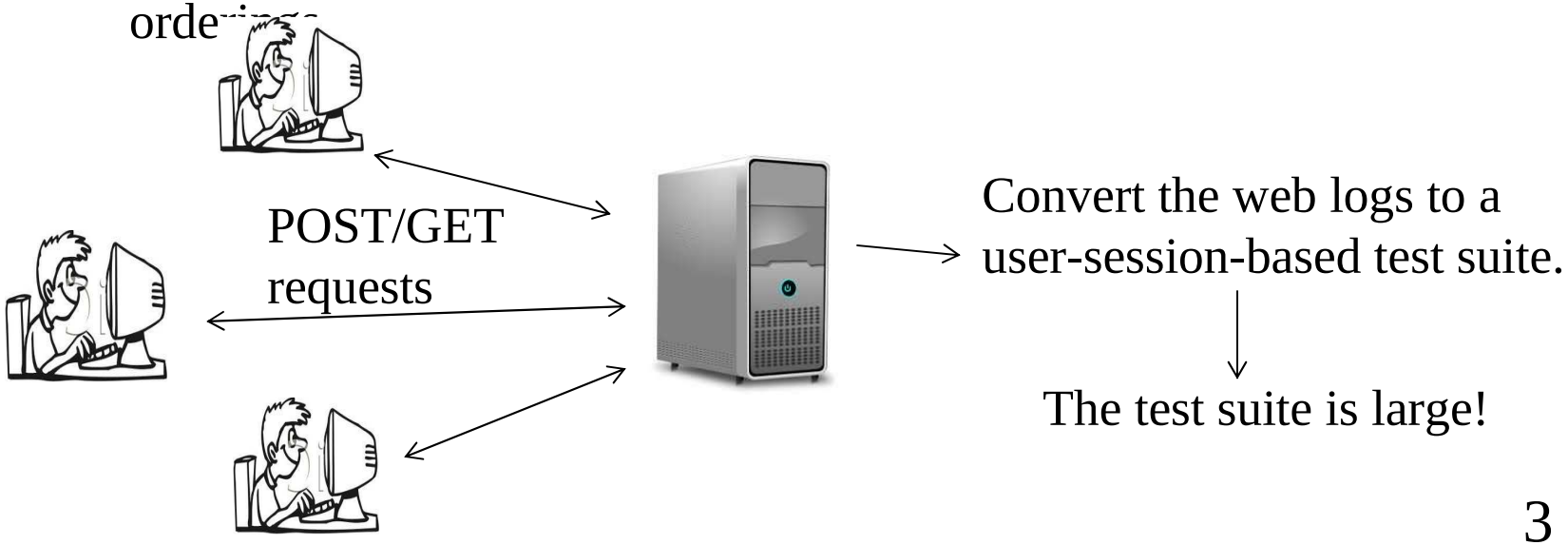
Dr. Renée Bryce
Associate Professor
University of North Texas
Renee.Bryce@unt.edu

Presentation outline

- Test Suite Prioritization
  - Exercise: Prioritize a test suite
- Test Suite Reduction
  - Exercise: Reduce a test suite using HGS
- Discussion

Test Suite Prioritization

- Test Suite Prioritization
  - Problem: Given T, a test suite, Π, the set of all test suites obtained by permuting the tests of T, and f, a function from Π to the set of real numbers, the problem is to find $\pi \in \Pi$ such that $\forall \pi' \in \Pi, f(\pi) \geq f(\pi')$. In this definition, Π refers to the possible prioritizations of T and f is a function applied to evaluate the orde[illegible]



Case Study: Prioritizing User-session-based Test Suites

- Methodology: Convert web logs to user-session-based test suites, prioritize, and write to an XML format.
- Algorithm: Efficiently prioritize by combinatorial-based coverage for large test suites
- Empirical Studies: Families of empirical studies to analyze the effectiveness in relation to characteristics of the applications and test suites.

Research Questions

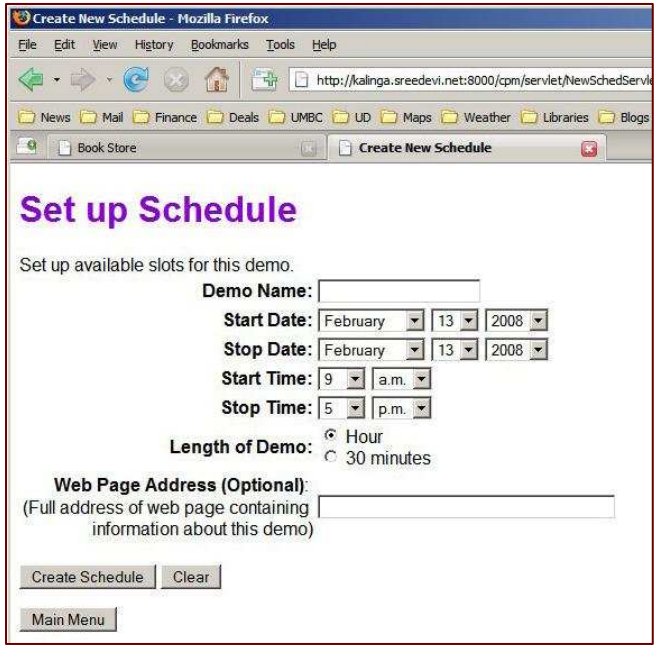
- Can we improve the rate of fault detection for user-session-based testing with new prioritization criteria?
- Which techniques are valuable in different scenarios?
  - i.e.: tests have a high/low Fault Detection Density
  - i.e.: predicted distribution of faults (deemed from prior versions of the software)
- Can we fine tune the criteria?
  - i.e.: cost-based prioritization

Prioritization Metrics

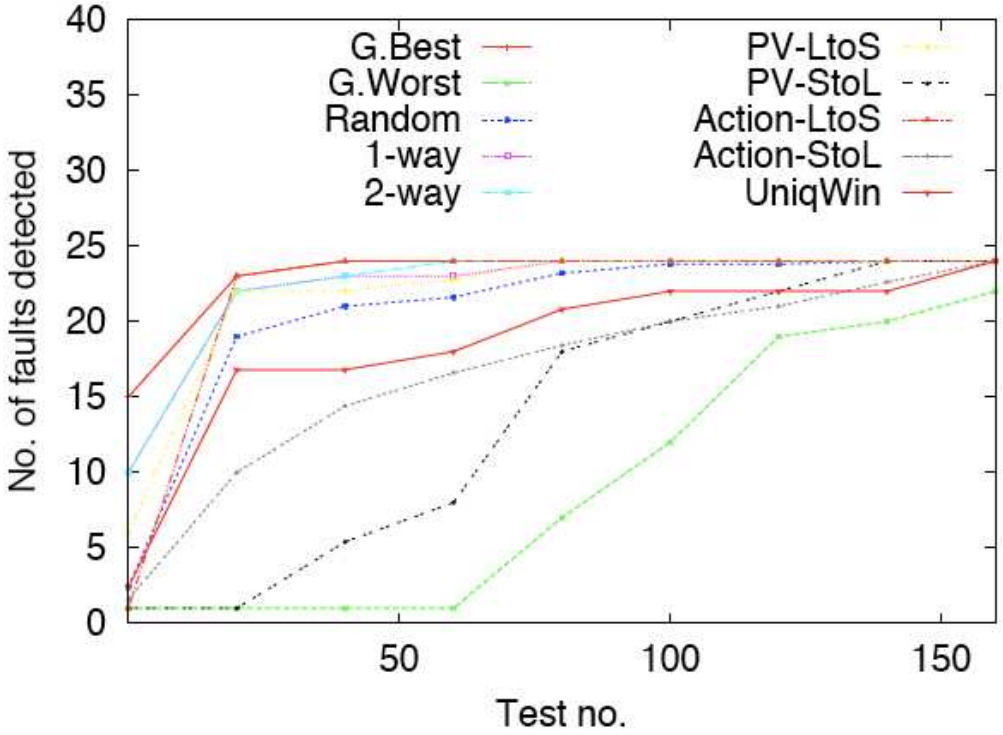
- Test length based on number of base requests:
  - order by the number of HTTP requests in a test case
- Frequency-based prioritization:
  - order such that test cases that cover most frequently accessed pages/sequence of pages are selected for execution before test cases that exercise the less frequently accessed pages/sequences of pages.
- Unique coverage of parameter-values:
  - order tests to cover all unique parameter-values as soon as possible
- 2-way parameter-value interaction coverage:
  - order tests to cover all pair-wise combinations of parameter-values between pages as soon as possible
- Test length based on number of parameter-value:
  - order by number of parameter-values used in a test case
- Random:
  - randomly permute the order of tests

Empirical Studies

- TerpCalc, TerpPaint, Terp Spreadsheet, and TerpWord
- Online Bookstore
- Online Course Project Manager (CPM)
- Online Conference Management System
- SchoolMate
- Online Music Store
- Metavist (sponsored by USDA)



Results for an on-line system for a Course Project Manager and 890 Test Cases



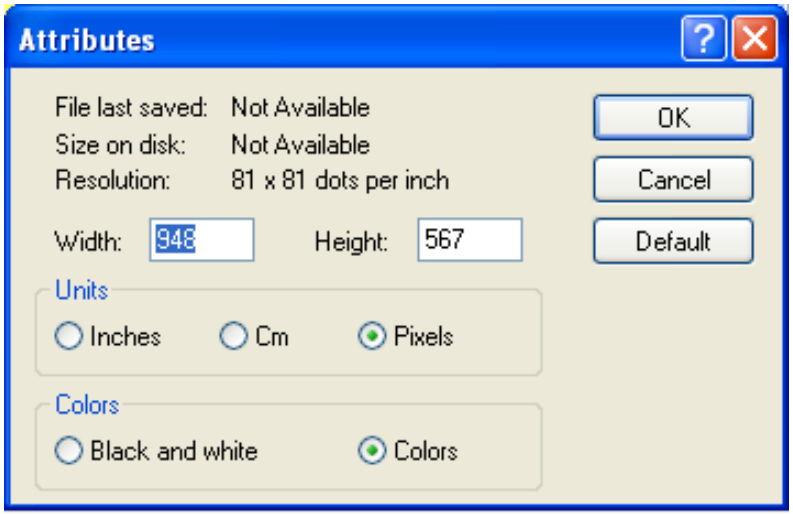
[1] R. Bryce, S. Sampath, A. Memon. Developing a Single Model and Test Prioritization Strategies for Event-Driven Software, Transactions on Software Engineering, (January 2011), 37(1):48-64.

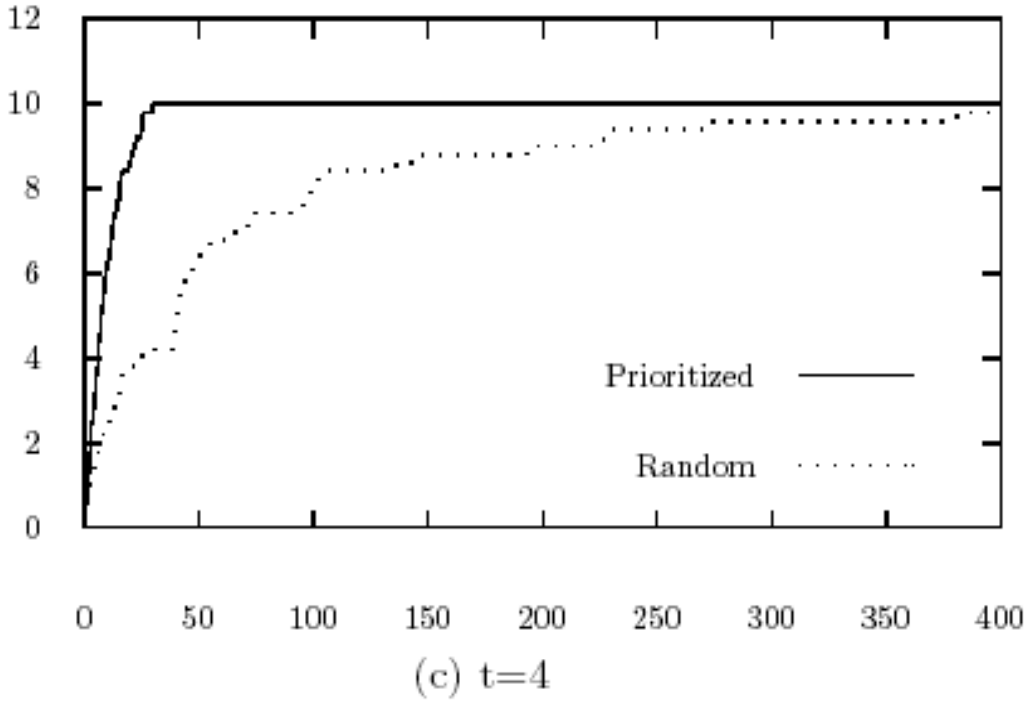
Sample results

% of test suite run	Most frequent requests	No. of Requests Long to short	No. of Requests Short to long	PVs Long to short	PVs Short to Long	1-way	2-way	Random
10	**85.28**	78.17	75.14	83.53	16.38	83.79	83.72	48.63
20	**88.52**	80.34	77.76	88.77	25.6	87.78	90.8	57.55
30	**89.4**	81.77	80.27	88.77	26.44	91.54	91.72	64.51
40	89.86	84.58	81.39	92.71	28.76	94.79	**95.64**	69.19
50	91.04	85.58	82.95	92.71	30.33	94.79	**95.64**	73.03
60	91.58	87.14	84.44	94.26	34.64	94.79	**95.64**	75.37
70	92.1	87.74	85.15	94.26	39.15	94.79	**95.64**	77.37
80	92.35	88.27	86.21	94.26	39.58	94.79	**95.64**	78.24
90	92.37	88.3	86.31	94.26	42.18	94.99	**95.64**	78.45
100	92.45	88.36	86.35	94.26	43.09	94.99	**95.64**	78.49

Test prioritization by interaction coverage

- Test suite prioritization
  - GUI-based testing



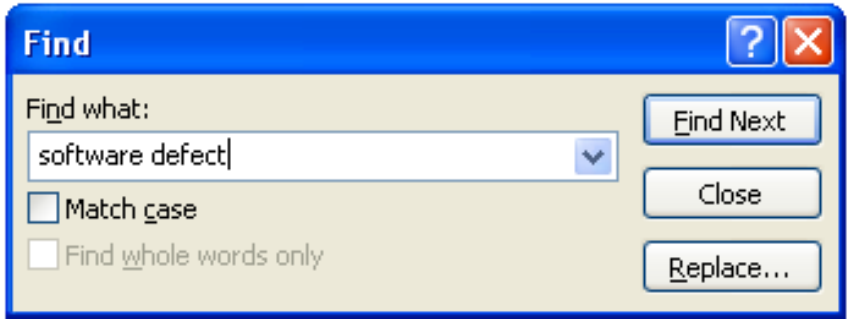


(c) t=4

Empirical Studies

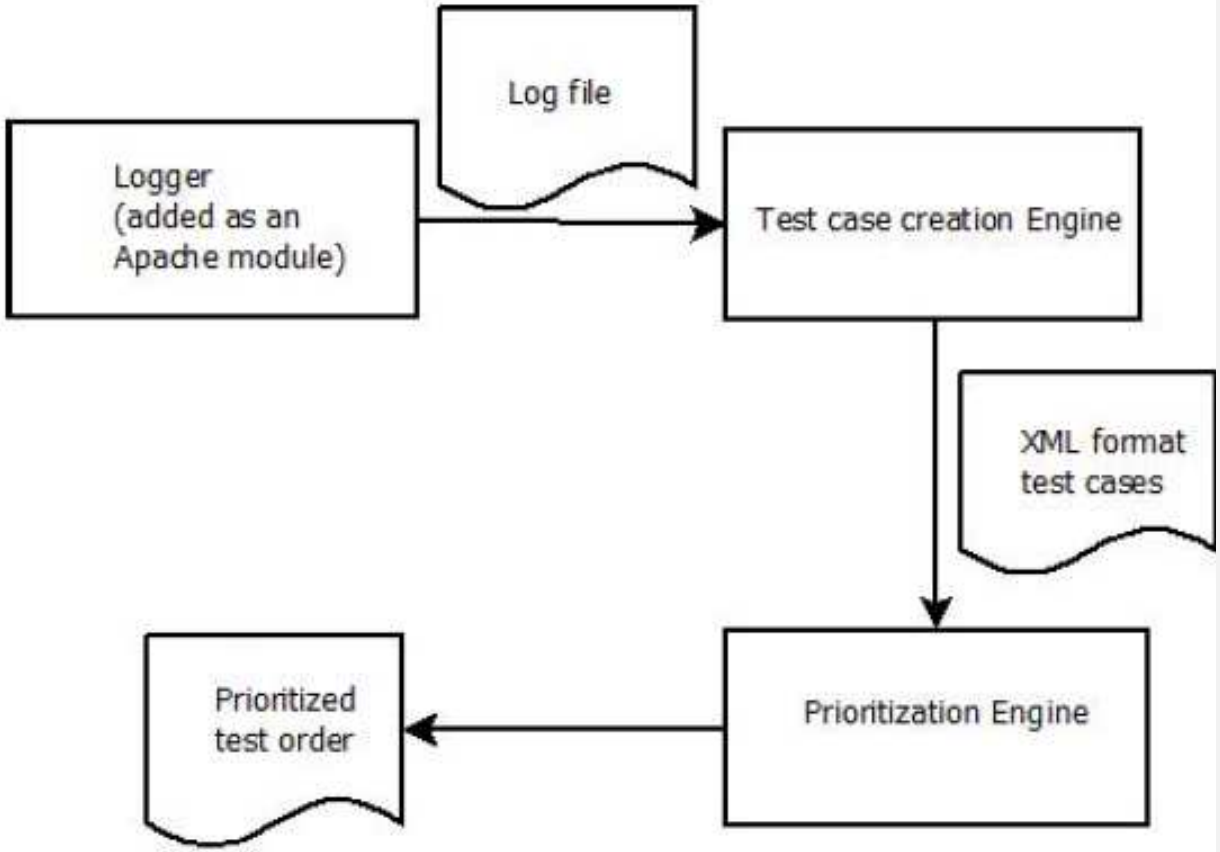
- Traffic Collision Avoidance Syst

- GUI-based Testing
  - Word processor
  - Spreadsheet
  - Paint
  - Calculator



- Web application Testing
  - Bookstore
  - Course Project Manager
  - Conference Management Software

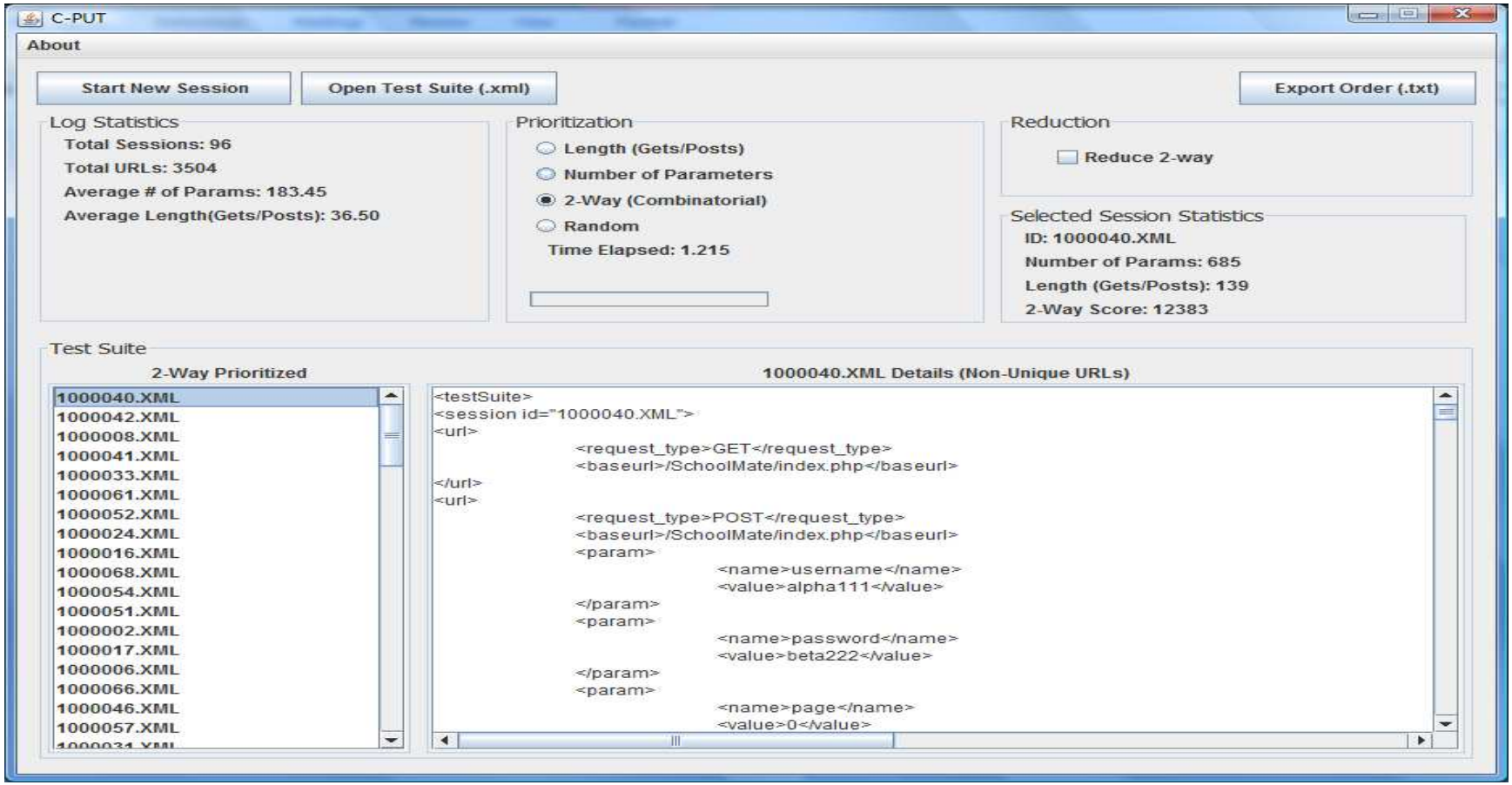
Transfer of Work



Potential users that have contacted NIST to use our tool:

- AT&T
- BBC (for Winter Olympics website)
- Booz Allen Hamilton
- Angel.com
- U.S. Army Test and Evaluation Research Laboratory, Aberdeen Proving Ground
- A2Z Research and Development
- NASA IV&V

Transfer of Work (Demo)



[1] S. Sampath, R. Bryce, S. Jain, S. Manchester. A Tool for Combinatorial-based Prioritization and Reduction of User-Session-Based Test Suites, *International Conference on Software Maintenance (ICSM) - Tool Demonstration Track*, Williamsburg, VA, September 2011

Next steps

- Methodologies
  - Examining issues with RIAs
- Algorithms
  - Hybrid techniques
- Empirical Studies
  - “Real” studies
  - RIA studies

Test Suite Reduction

- Problem: Given T, a test suite with test cases { $t_1, t_2, \ldots, t_m$ }, a set of testing requirements, { $r_1, r_2, \ldots, r_n$ }, that must be satisfied to provide the desired test coverage of the program, and subsets { $T_1, T_2, \ldots, T_n$ } of T, one associated with each of the r_is such that any one of the tests belonging to t_j Ti satisfies r_i. Find the *minimal cardinality* subset of T that exercises all of the requirements exercised by the original test suite T.

Original Test Suite (Too large for our budget)

Reduced Test Suite (Fits into budget)

Reduction Example

- Original Test Suite
  - $\{t_1, t_2, t_3, t_4\}$
- Requirements covered by the test suite
  - $\{r_1, r_2, r_3, r_4\}$
- Problem: Reduce the test suite such that it maintains coverage of these requirements

Test Suite Reduction Example

T	Requirement	T_i
1	1	{t_3,t_4}
2	2	{t_4}
3	3	{t_1, t_2, t_3, t_5}
4	4	{t_1, t_2, t_3}

In this example, there are three possible solutions. We highlighted 1: {t_1, t_4}

Test Suite Reduction Example

T	Requirement	T_i
~~1~~	~~1~~	~~$\{t_1, t_5\}$~~
~~2~~	~~2~~	~~$\{t_5\}$~~
~~3~~	~~3~~	~~$\{t_1, t_2, t_3\}$~~
~~4~~	~~4~~	~~$\{t_3, t_6\}$~~
~~5~~	~~5~~	~~$\{t_1, t_4\}$~~
~~6~~	~~6~~	~~$\{t_1, t_6\}$~~
~~7~~	~~7~~	~~$\{t_3, t_4, t_7\}$~~
~~8~~	~~8~~	~~$\{t_2, t_3, t_4, t_7\}$~~

HGS Algorithm

3. T_4 is of cardinality 2, there is a tie between t_3 and t_6, so we look at sets of size cardinality (m+1). We choose t_3.

Reduced Test Suite:

$\{t_5, t_1, t_3\}$

Exercise

- Reduce this test suite using the HGS algorithm:

T	Requirement	T_i
1	1	$\{t_1, t_5\}$
2	2	$\{t_5\}$
3	3	$\{t_1, t_2, t_3\}$
4	4	$\{t_3, t_6\}$
5	5	$\{t_1, t_4\}$
6	6	$\{t_1, t_6\}$
7	7	$\{t_3, t_4, t_7\}$
8	8	$\{t_2, t_3, t_4, t_7\}$

Test Suite Reduction Example

T	Requirement	T_i
~~1~~	~~1~~	~~$\{t_1, t_5\}$~~
~~2~~	~~2~~	~~$\{t_5\}$~~
~~3~~	~~3~~	~~$\{t_1, t_2, t_3\}$~~
~~4~~	~~4~~	~~$\{t_3, t_6\}$~~
~~5~~	~~5~~	~~$\{t_1, t_4\}$~~
~~6~~	~~6~~	~~$\{t_1, t_6\}$~~
~~7~~	~~7~~	~~$\{t_3, t_4, t_7\}$~~
~~8~~	~~8~~	~~$\{t_2, t_3, t_4, t_7\}$~~

HGS Algorithm

3. T_4 is of cardinality 2, there is a tie between t_3 and t_6, so we look at sets of size cardinality (m+1). We choose t_3.

Reduced Test Suite:

$\{t_5, t_1, t_3\}$