Robust Automatic Testing of Student Programs

Guanwen Wang[1] (gai25@wildcats.unh.edu), R. Daniel Bergeron[1]

[1]Department of Computer Science, University of New Hampshire, Durham, NH 03824, USA

Graduate Research Conference 2016, Durham, NH, April 11 – April 12, 2016

Background

- Time-consuming to grade student programs.
- Tame (Tool for Assignment Management and Evaluation) [Sun, 2014]
  - Instructor-defined rubrics
  - Automatically calculates deductions
  - Helps compare student output to sample output.
  - Grader must determine other deductions
- Semi-automated partial credit grading [Rossi, 2016]
  - Compare student program output to a sample solution
  - Sample can produce partially correct results also

Weaknesses

- Little support for handling "runaway" student programs
  - Infinite looping
  - Excessive output
- Little support non-trivial data structures such as lists and trees.

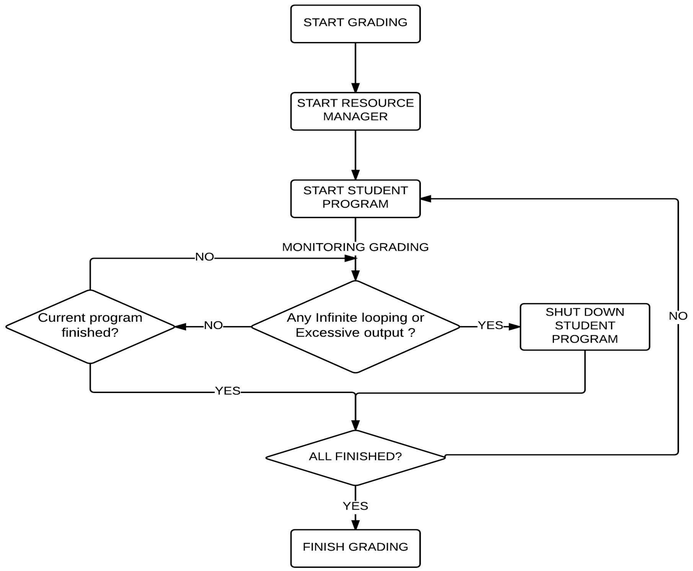
Project Goals

- Handle runaway student programs more effectively
- Compare complex data structures

Methodology

- Runaway programs
  - Test code catches student programs that use excessive time or output
- Equality and similarity of two Data Structure objects
  - Compare content : object in the data structure
  - Compare how objects organized in data structure
  - Create library to support instructor-defined comparisons

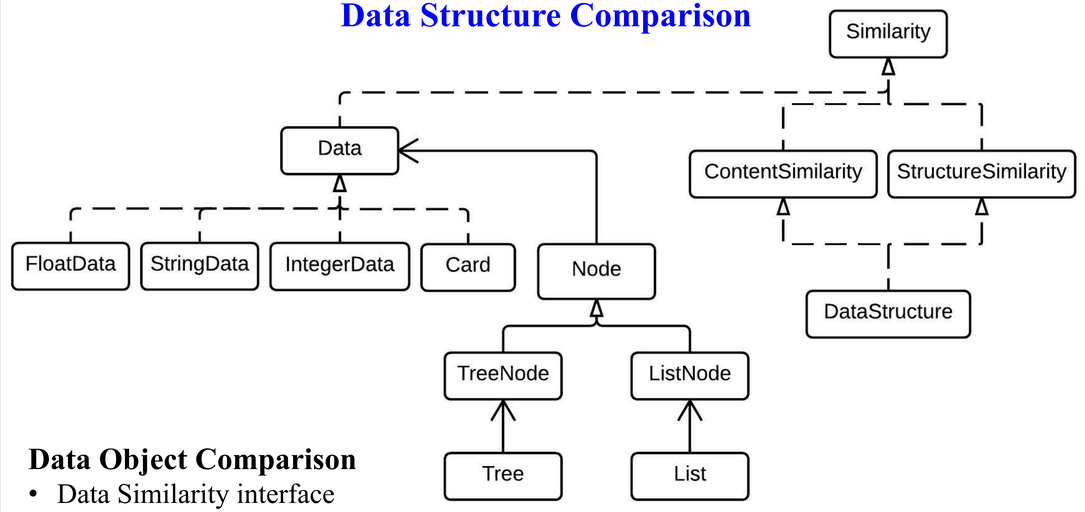
Runaway Programs



Excessive Time and Output Examples

- Excessive Time
  - For Bubble sort algorithm 2 seconds for a certain amount of numbers
  - For Sum-Up algorithm 500 ms for a certain range
  - The grading manager will check the student's code and shut it down if it excessed the preset time limit.
- Excessive Output
  - For Bubble sort algorithm 1kb limit
  - For Sum-Up algorithm 500b limit
  - Student program will be shut down when it print out more than the output limit.

Data Structure Comparison



Data Object Comparison

- Data Similarity interface
  - dynamically defined semantics

Data Structure Comparison

- Interface to compare content similarity
  - use Data Similarity interface
- Interface to compare structure similarity
- Build library of common DataStructure classes: list, stack, queue, binary tree, quadtree

Data Example

- Card class comparison: Club < Diamond < Heart < Spade

0 1 … 12 13 14 … 25 26 27 … 38 39 40 … 51

2C < 3C < … < AC < 2D < 3D < … < AD < 2H < 3H < … < AH < 2S < 3S < … < AS

Data Structure Comparison Summary

- Implement multiple instructor specified metrics based on content similarity and structure similarity
- Cluster students based on these metrics using different instructor-specified weights
- Good class design should significantly improve the grading efficiency
- Instructor can use and extend the library easily and conveniently

References

- Sun, Xiaobo, *Tame (Tool for Assignment Management and Evaluation)*. August 2014
- Rossi, Thomas, *Semi Automated Partial Credit Grading of Programming Assignments*. May 2016