# CS 3149 Competitive Learning in Computer Science

**Professor Contact Information**
Ivor Page, ECS 4.410, ivor@utdallas.edu
Office Hours: TBA

**Course Description**
This is a one-hour course that can be repeated twice for a total of three credit hours. We will learn how to solve programming problems of the kind that occur in ACM ICPC programming contests. We will start with problems requiring only basic programming techniques without need of complex algorithms and build our techniques using problems that require numerical techniques, sorting, graph traversals, shortest paths, backtracking, network flow, geometric techniques, dynamic programming and other algorithmic techniques. The idea is that, "You are only as good as the set of problems that you have solved." We will use an online judge.
We will study problems and their solutions together in class and then you will be assigned weekly problems according to your skill level and experience. During the course, all students must participate in at least three UTD programming contests. These are usually held on Saturdays, every week.

**Student Learning Objectives/Outcomes**

Ability to analyze and understand problems and use a subset of the following techniques, depending on the each students ability:

(1) Sorting
(2) String processing, searching, comparison
(3) Arithmetic, Integers, reals, root finding, huge numbers, factorials, Fibonacci numbers
(4) Combinatorics, counting sequences, binomial coefficients, permutations, combinations
(5) Number theory, primes, GCD, LCM, modular arithmetic
(6) Backtracking, maze running, constructing all permutations, all subsets
(7) Graph traversal, BFS, DFS, finding cycles, bi-coloring, shortest paths, bipartite matching
(8) Greedy algorithms, dynamic programming,
(9) Geometric problems and computational geometry, lines, line segments, line segment intersections, polygon area, Picks theorem, point containment in a polygon, convex hulls, smallest containing circle, closest point to a circle, line segment, etc. closest pair.

**Recommended Textbooks and Materials:**

Steven S. Skiena and Riguel A. Revilla, Programming Challenges, Springer. ISBN 978-0-387-00163-8

**Suggested Additional Course Materials**

Notes on eLearning web page for this course.

See audio files and notes on the website for the textbook (click on Audio lectures): http://www.programming-challenges.com/pg.php?page=index

Any good book on Java 1.8 and any good data structures text. For most students, Weiss, Data Structures and Algorithm Analysis in Java, is a good start (used in CS 3345). For the most

advanced students, the following is the best book (known as CRLS.) Its not an easy read: Cormen, Rivest, Leiserson, Stein, Introduction to Algorithms, 2ND, Edition (used in CS 4349 and CS 6363).

**Assignments & Academic Calendar**

There will be no exams, but multiple programming problems. Students must compete in at least three of our weekly contests.

**Course Content - Likely topics, maybe not in this order**

## General Principles of Problem Solving

1. Read the problem, circle the nouns and verbs.
2. Make careful note of the limits on the sizes of numbers, can any be negative, zero?
3. Note that many algorithms will have restrictions on the data values and numbers of them (Dijkstra APSP with negative edge weights).
4. Make careful note of the sizes of datasets - this could affect the choice of algorithm.
5. Consider enumerating all the cases.
6. Once you have a possible solution, work one or two if the sample datasets on paper.
7. Think of edge cases for this problem. Did you cover all of them?
8. Think of the largest values and largest data structures that could arise. Will integer overflow occur? Do you need longs, or the Big Integer Package?
9. Consider the runtime of your method. Can you estimate it from 2 and 3?
10. Code your solution using the fastest algorithm that you are familiar with and test your code on all the sample data.

1. Tricks, a look at runtimes for arithmetic calculations using ints, longs, floats, doubles, Javas Big Integer package

2. Enumerating Primes

3. For ints, Don't divide when a shift will do.

4. Shifting and masking techniques. Counting trailing zeros, etc.

5. The $x\&(-x)$ trick to isolate the lsb that equals 1. Possibly look at Fenwick Trees?

6. Bit arrays, insertion, deletion, masking, use of AND/OR/EXOR instructions (three EXORs to swap 2 values)
- UVa 562, Dividing Coins, UVa 10032 Tug of War (We will study DP solutions).

7. Java's Big Integer Package

8. Fibonacci numbers, Binets formula, Fib(n) mod m, equations relating Fibs,
- UVa 10183, How many Fibs
- Fib mod M cycles problem

9. I/O, Scanner vs. Buffered I/O.

10. Tricky Problems:
UVa 10173 The Trip, UVa 10102 Pairsumonious Numbers, UVa 136 Ugly Numbers, Vito's Family,

Prime Rings, UVa 846 Steps, UVa 10611 Playboy Chimp, UVa 10487 Closest Sums, UVa 10110 Light More Light, UVa Pairs

11. How fast are the built-in types and the collection classes? Are Array Lists to be avoided? etc.
- Look at the runtimes for Vector, LinkedList, ArrayList, Set, HashSet, TreeSet, HashMap, TreeMap, LinkedHashSet,
- http://objectissues.blogspot.com/2006/11/big-o-notation-and-java-constant-time.html

12. String processing, searching, comparison
- A look at Java's String class and the C++ equivalent.

13. DP methods for Strings - Edit Distance, Longest Common Substring, Longest Common Subsequence
- UVa 10192 Vacation, UVa 12747, Back to Edit Distance, UVa 526 - String Distance and Transform Process, UVa 10405 - Longest Common Subsequence

14. Sorting, the two Java builtin functions, Binary Search
- UVa 299 Train Swapping, UVa 10152 Shell Sort, UVa 120 Stack of Flapjacks, UVa 102 Ecological Bin Packing, UVa ? Searching Quickly, UVa ? Closest Sums, UVa 107 Cat in the Hat, UVa 10252 Common Permutations

15. Greedy algorithms, dynamic programming, Combinatorics, counting sequences, binomial coefficients, permutations, combinations

16. Number theory, primes, GCD, LCM, modular arithmetic

17. Backtracking, maze running, constructing all permutations, all subsets

18. Graph traversal, BFS, DFS, Dijkstra, Bellman-Ford, finding cycles, bi-coloring, shortest paths, bipartite matching

19. Geometric problems and computational geometry, lines, line segments, line segment intersections, polygon area,

20. Picks theorem, point containment in a polygon, closest pair, convex hull, smallest containing circle, closest point to a circle, line segment

**Grading Policy**
(including percentages for assignments, grade scale, etc.) There will be about 10 programming problems plus the results of three contests. Total grades for programming problems will be 60% and the total grade for contest performances will be 40%

**Course & Instructor Policies**
(make-up exams, extra credit, late work, special assignments, class attendance, classroom citizenship, etc.)
There will not be any exams.

**UT Dallas Syllabus Policies and Procedures** Please go to http://go.utdallas.edu/syllabus-policies
These descriptions and timelines are subject to change at the discretion of the instructor.

**UTD Honor Code: "As a Comet, I pledge honesty, integrity, and service in all that I do."**