======= Course Format ========

There is no exam and no formal curriculum, other than being able to program a sufficient amount (approximately 20-25) of tasks on various topics (see below) and get them accepted at various competition servers (for an example, http://uva.onlinejudge.org/) during the semster. You will also have to participate on at least 2 out of 3 competitions arranged during the semester. For topics that we expect to cover, see below. The programming will be done in Java or C++, your choice.

======= Optional textbooks ========

Programming Challenges, by Steven S. Skiena and Miguel A. Revilla

[Optional means that it is not necessary to have the book; we will loosely refer to the topics there and use some problems, so you simply might find it useful.]

Any algorithms textbook, for an example:

Algorithm Design (Kleinberg and Tardos)

Introduction to Algorithms (Cormen Leierson Rivest and Stein)

Algorithms (Papadimitriou)

======= Expected topics covered ========

These are the topics covered Spring 2013. This year I expect to do a ligher / easier variant of the course, covering a proper subset of the subjects below. Exactly which will be thrown out is yet to be decided.

Basic stuff:

(1) Introduction to programming techniques during contests.

(2) Reminder of basic graph algorithms: Graph Representation, DFS, BFS

(3) Find&Union

(4) Tricky linear-time algorithms using queues and stacks

(5) Reminder of qsort, mergesort, binary search; binary search on the expected result.

Data structures:

(1) Static binary trees; point-interval, interval-point, maybe interval-interval queries.

(2) Reminder of heap and Dijkstra algorithm; all-to-all shortest paths

(3) Usage of map and set from STL

Dynamic programming:

(1) The principle; how to recognize a problem solvable by DP.

(2) A lot of examples: DP on intervals, DP on trees, etc.

Exponential time algorithms:

(1) Strategies for implementing brute-force; branching.

(2) DP on subsets.

(3) Meet-in-the-middle technique.

Greedy algorithms:

(1) The principle; how to recognize that a greedy approach works.

(2) Formal proofs that greedy is correct.

Basic number theory and combinatorics:

(1) Reminder of binomial coefficients, computing binomial coefficients quickly.

(2) Strategies for counting various objects; counting using recurrences.

(3) Generating primes, implementation of factorization.

(4) Field F_p - main properties.

(5) Extended Euclid's algorithm.

Equations and matrices:

(1) Gauss elimination, in an abstract field.

(2) Fast matrix power computation.

(3) Solving recurrence using matrix multiplication.

(4) Applications in graphs: counting triangles, counting C_4-s

Matchings and 2-SAT:

(1) Maximum matching in a bipartite graph, classical algorithm (not Hopcroft-Karp)

(2) Linear-time algorithm for 2-SAT

(3) Examples of applications

Flows:

(1) Max-flow algorithm (implementation of Edmonds-Karp)

(2) Max-flow-min-cost (Maybe, probably not)

(3) Examples of applications

Computational geometry:

(1) Storing data, basic geometrical operations: scalar and vector product, projections, counting distances, intersecting two lines, intersecting two intervals, intersecting circles

(2) Point-in-a-polygon, point-in-a-convex-polygon (maybe)

(3) Tricks: (x,y) -> (x+y,x-y) trick, computing area of a polygon

(4) Horizontal sweeping algorithms using static binary trees

(5) Convex hull, intersection of semiplanes, other examples of radial sweeping

(6) Divide-and-conquer: closest pair

String algorithms:

(1) KMP, other examples of using the P-table

(2) Manacher algorithm

(3) Trie trees