

CPSC 490 : Problem Solving in Computer Science - Course Proposal

1 Course Rationale

We would like to propose a Computer Science Problem Solving Course. Problem solving is an important part of Computer Science, and we feel that it needs to be emphasised more than it is in courses at UBC. Most tech companies today (eg. Google, Microsoft, Facebook, etc.) have problem solving components to their interviews and require applicants to come up with algorithms and implement them in a short amount of time. Our course aims to bridge the gap between the theoretical algorithms taught in class and practical implementations of these algorithms in standard programming languages. We will focus on many important programming techniques and concepts (eg. dynamic programming, graph algorithms, advanced data structures, etc.) with problems that illustrate their applications. Some of those concepts are taught in upper level algorithms courses (namely CPSC 320 and CPSC 420), but those courses are mostly theoretical. Students do not get a chance to try and implement the algorithms themselves.

This type of implementation based problem solving course has been offered in other universities with great success, such as CS 97SI at Stanford 15-295 at CMU. Every offering of CPSC 490 in the past as a problem solving course has been very successful in both attendance and interest.

2 Prerequisites

The only prerequisite for this course is CPSC 320. However, students with sufficient familiarity with C++ or Java or Python (those without the prerequisites can email the coordinator for more details). We intend to introduce all algorithms from scratch, focusing on implementation and application and leaving theoretical aspects to courses such as CPSC 420.

3 Format

As with previous offerings of CPSC 490, the course will be offered in the format of a seminar. Each week, we (the coordinators) will present a topic for discussion

(eg. dynamic programming, network flow, etc.) and outline the algorithms pertaining to the topic. The discussion may be led by the coordinators or the students themselves. Following the discussion, the coordinators will present a few problems related to the topic. The problems presented will be non-trivial, in that several reductions will often be needed before one of the outlined algorithms can be used. Each student will also have a chance to present problems of their choosing along with their solutions, as long as it pertains to the weekly topic. At the end of every week, we will assign homework problems in the format described in Section 4. The maximum enrollment of this course will be 15 students.

4 Homework Problems

The most novel part of the course is the homework submission system. Unlike most courses at UBC, students will submit their solutions to a particular problem via an online judge system that we operate. The automated online judge system will receive code from the students, compile and execute the program, and reply appropriately (one of 'Accepted', 'Time Limit Exceeded', 'Runtime Error' or 'Incorrect Output'). Since a problem may have many different solutions with varying time complexities, we enforce a strict time limit on the number of seconds a student's program is allowed to run. This allows us to accept solutions with the roughly the correct complexity, and emphasize the importance of implementing efficient algorithms. Furthermore, students benefit from getting immediate results to their submissions, and can then review their code and attempt to correct mistakes.

In addition to implementation problems, we may give occasional written problems (for cases where the implementation of the solution is extremely demanding).

Some problems can be challenging, so teamwork will be encouraged. However, the judge system also has plagiarism detection that attempts to maintain academic discipline. The coordinators will report all cases of plagiarism to the Faculty.

As an example that this judging system is technically feasible, we note that such a system is already utilized by the UBC ACM team in their weekly practices. Both the judging system and the typical style of problems we assign can be seen by exploring various links at <http://www.cs.ubc.ca/~acm-web/practice/>

5 Syllabus

The seminar will attempt to cover the following topics (in order) with * denoting an advanced topic that may or may not be covered (depending on the speed of the class). Topics may be added or removed depending on the class background and interests.

1. **Dynamic Programming** - Longest common subsequence, Longest increasing subsequence, Space saving tricks*, DP Optimizations*, Four Russians*
2. **Graph Algorithms** - Breadth-first search, Depth-first search, Shortest paths on weighted graphs (single source or all pairs, with positive and/or negative edge weights), Minimum spanning trees, Euler tours*
3. **Advanced Data Structures** - Segment trees, Binary indexed trees, Square root decomposition
4. **Combinatorial Optimization** - Bipartite matching, Network Flow, Linear Programming*
5. **String Algorithms** - Tries, Suffix trees*, Suffix array*
6. **Computational Geometry** - Basic geometrical primitives (line intersection, segment intersection, etc.), Convex hull, Line sweeps*

6 Potential Faculty Sponsor

1. Will Evans: Associate Professor, 604-822-0827, will@cs.ubc.ca

7 Qualifications of the Coordinators

We have two confirmed coordinators and one potential coordinator (who may only be present for a fraction of the sessions due to scheduling conflicts):

Raunak Kumar: 4th year Major in Computer Science One year experience on UBC ACM team, Two year experience as UTA for Computer Science

Jason Chiu: 5th year Double Major in Computer Science and Mathematics Three years experience on UBC ACM Team, qualified for 2017 ACM ICPC World Finals