CS 490 Course Proposal: Problem Solving in Computer Science

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 deserves more attention than most CPSC courses offered at UBC give it. Many tech companies nowadays (such as Google, Facebook, and Amazon) have problem solving components to their interviews, and require applicants to quickly implement basic algorithms. 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 a number of important programming techniques and concepts (dynamic programming, graph algorithms, computational geometry, 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 algorithm themselves.

This type of implementation based problem solving course has been offered in other universities with great success, such as CS 97SI at Stanford and CSE 392 at Stony Brook. Furthermore, every offering of CPSC 490 in the past as a problem solving course has been quite successful in both attendance and interest.

2 Prerequisites

Our only requirement for this course is CPSC 320. However, students with sufficient familiarity with a programming language (C++ or Java) are also welcome (those without prerequisites can email the coordinator for more details). We intend to introduce all algorithms from scratch. We will focus on the practical implementation, application, and usage of these algorithms, and will leave the more abstract 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 the coordinators will present a topic for discussion (e.g. dynamic programming, search algorithms, etc.) and outline some basic algorithms pertaining to the topic. The discussions may be led by the coordinators, guest lecturers, or even the students themselves. Following the discussion, the coordinators will present a few problems illustrating the topic of the week. 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, so 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 would receive code from students, compile and execute the program, and reply appropriately (one of 'Accepted Solution', '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 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. This allows us to fairly assess students' ability to understand the concept of technique involved, while at the same time encourage them to implement the algorithms correctly.

In addition to implementation problems, we will also give occasional written problems (for cases where the solution is quite simple in theory but extremely demanding to implement).

Some problems can be challenging, so teamwork would 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*, Convex hull trick*, Four Russians*
2. **Greedy algorithms** - Interval scheduling and variants, Greedy algorithms on matroids*
3. **Basic graph algorithms** - Breadth-first search, Depth-first search, Shortest paths on weighted graphs (with positive and/or negative weights), Minimum spanning trees, Euler tours*, Random walks*
4. **Combinatorial optimization** - Bipartite matching, network flows, linear programming*

5. **Computational geometry** - Basic geometrical primitives (line intersection, segment intersection, etc.), Convex hull, Line sweeps*, Duality*

6. **String algorithms** - Tries, KMP, Suffix arrays*, Suffix trees*

7. **Advanced data structures** - Binary Indexed Trees*, Segment and interval trees*, Link-cut trees*, Splay trees*

6 Faculty Sponsors

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

2. David G. Kirkpatrick: Professor, 604-822-4777, kirk@cs.ubc.ca

7 Qualifications of the Coordinators

Bruno Vacherot: 6th Year Computer Science Student

- Related experience: Four co-op terms, hosted co-op interview workshop

Andrew Kuba Karpierz: 1st Year Computer Science Masters Student

- Related experience: UTA for CPSC 221 in 2014, GTA for CPSC 221 in 2015. URA in CS Education in 2013/14.