Student Learning Assessment

Computer science educators have handled enrollment growth, in part, by automating student assessment. Our field is nicely suited to this, as student code can be automatically run against a set of test cases to determine how well it performs on a given task. More importantly, this automation provides students with useful feedback on their code prior to submission, so that they can fix and learn from their mistakes.

Unfortunately, in computer theory, scaling assessment is a much harder task, as students are asked to provide formal mathematical proofs; automating proof verification is a hard problem that is itself an active area of computer science research. Additionally, students need direct feedback on their attempts at a proof, which further increases the work load. Since beginning at Mason, I have been exploring new ways to scale student assessment on homework, quizzes and exams, while still providing students with sufficient feedback to help them improve. Below is a description of some of the efforts I have taken.

Resubmission of assignments: In order to provide students with sufficient feedback to help them improve, I have experimented with various policies around re-submission of homework and exams.

Resubmission of half of homework assignments. When teaching CS 600 in Fall 2015, I allowed students to re-submit 3 of their 6 homework assignments. Allowing students to choose which assignments they would resubmit meant allowing them to wait until the end of the semester to make a decision. For students that had struggled in the class, the long deadline policy did not have the desired impact of encouraging them to engage with me and with the material until they truly understood it. Rather, after waiting until the end of the semester, most struggling students attempted to gain a few extra points without demonstrating real improvement.

Resubmission of all homework assignments. When I taught the course again in 2018, I imposed a 2 week deadline for any resubmission, and I allowed resubmissions on all of the homework. Recognizing that this would greatly increase my own workload, I discouraged students from chasing small point increases by stating that each point earned on a re-submission was worth only .8 points. These changes helped significantly, and the policy seemed to have the desired impact. Unfortunately, as course sizes continue to grow in our field, allowing the resubmission of every assignment does not scale well.

Resubmission of Midterm Exams. When teaching CS 499 in Spring, 2020, I allowed students to re-submit their midterm as frequently as they liked, up until the final week of class. To facilitate quick feedback, and to encourage students to begin early, I had each student create an Overleaf project that I could edit in real time. This allowed them to request feedback while they worked, rather than after completing the full exam. This approach was a real success: students demonstrated remarkable improvement, and the use of Overleaf helped me to smoothly provide multiple rounds of interaction. By focusing on the midterm exam, rather than every homework assignment, my own workload was reduced.

Conclusion. To balance the student need for repetition, and the need to scale the instructor workload, it is important to recognize that not all assignments are equally valuable. In most courses, focusing on 2 key concepts for student repetition will provide most of the benefit, while limiting the instructor workload.

Incentivizing Practice: Students learn through practice, and the more problems they do, the better they understand the material. Practice problems also help with the matter of scale, since they do not require instructor attention (beyond the time required for reviewing the problems in class or online). However, it is hard to encourage students to do ungraded work. I have experimented with a few approaches to motivating student engagement with practice problems.

Use of quizzes. The first time I taught an undergraduate class (CS 330), all homework was ungraded, and quizzes on the homework were worth 20% of student grades. This policy was very stressful to the students. Quizzes stretched on longer than I wanted them to, using precious lecture hours. Since then, I reduced the value of quizzes to 10%, and made them much simpler, testing only that students have reviewed course material, but not whether they can solve challenging problems. Homework problems and exams are now the primary indicator of whether students in my classes can solve problems.

Grading random problems. In Spring, 2018, when teaching CS 600, I used a policy of grading only a random subset of the assigned problems. The remainder were worth some nominal number of points, just to reward student attempts. This motivated students to do all of the problems, but students were not excited about allowing a random process to influence their grade.

Use of Piazza. In Spring, 2020, when teaching Intro to Cryptography, I tried to motivate engagement with practice problems by encouraging students to help each other solve these problems on Piazza. I also informed students that at least 1 practice problem would appear on the exam. Unfortunately, engagement through Piazza was minimal: a few strong students posted solutions, and most students were silent. I am still exploring ways of motivating engagement with ungraded problems.