ABSTRACT
This paper attempts to explore students’ problem solving strategies by using differential mining. Students’ problem-solving activities on multiple choice questions were collected from a semester-long computer science programming course in 2016 Fall semester. Based on each question’s correctness, complexity, topic, and time, the frequent behavioral patterns were extracted to build the problem solving sequences. Seven distinct learning behaviors were discovered based on these patterns between high and low performing students, which provided insight into students’ meta-cognitive skills and thought processes.

Keywords
Behavioral mining, behavioral modeling, programming learning, problem solving, differential mining, multiple choice questions

1. INTRODUCTION
Practice is an essential component of learning and skill acquisition [5], yet not all practice is created equal. In large, introductory computer science lecture courses, it is commonplace to assign a handful of programming projects throughout the semester. Each assignment may cover a range of topics rather than targeting individual concepts, and these practice opportunities may not be temporally aligned with lectures (i.e., a concept introduced in Week 2 may not be practiced until a project assigned in Week 4). Students’ only other opportunity to assess their own performance may be an exam. Thus, typical practice activities in lecture-dominant courses may violate principles of distributed retrieval practice [3]. Several online programming self-assessment tools are developed to support coding practices, such as WebCAT [4], QuizJET [12], QuizIT [1] etc. Such tools are especially welcome and made accessible for introductory programming classes, where the programming novices could practice the necessary concepts to solve problems. However, in many introductory programming classroom studies, students were found they often appear to have limited fundamental knowledge or superficially organized knowledge, and fail to apply relevant knowledge [17]. Without solid fundamental knowledge, many of the novices may develop a trial-and-error approach to fix code errors [16], which some maintain the skill throughout more advanced courses, and some may even result in creating more errors than it solved. We begin to see researchers have found that students with similar “problem-solving genomes” perform relatively similarly with their peers in the context of parameterized quizzes. Such stability could allow to predict unproductive behavior in the problem solving process [7]. In this work, we aim to examine students’ problem-solving tactics in small programming practices, which we define them as bite-size practices.

First of all, we adopt multiple choices questions (MCQs), which is one of the commonly implemented assessments and practices. Essentially, the simplicity and structural nature of questions-and-answers not only largely improves the evaluation efficiency, but also provides greater flexibility in content authoring and construction. Due to the correct solution of the problem is presented with a fix set of distractors (incorrect answers), thus, to achieve a successful problem solving, one has to analyze the question, overcome the distractions and identify the correct answer. Nevertheless, one can also obtain a successful attempt by eliminating incorrect answers, or apply a combination of elimination and/or other tactics (such as trial-and-error approach to experiment the correctness of the answers) to reach the final correct solution. To capitalize the pedagogical values on how students approach problems and attain the solutions, we model their problem solving strategies. We hypothesize that there are individual differences in the problem solving sequences to reach to the correct solutions state. Specifically, we hypothesize the higher performing students will approach problems and reach solutions differently from lower performing ones. We aim to research What are the possible learning strategies adopted by a student working on programming problems in the context of MCQs?

2. LITERATURE REVIEW
2.1 Distributed Retrieval Practices in Learning
Researchers have consistently found that distributed practice, practice that is incremental and spaced over time, is superior to practice that is conceptually or temporally compressed [3][20]. The classic example of ineffective massed practice is "cramming", when students conduct all of their studying and exam preparation into a few days (or even hours) before an assessment [20]. Mass practice leads to fragile and isolated knowledge that may be quickly lost or forgotten. In contrast, distributed practice helps to periodically re-instantiate and reinforce key memories, concepts, and skills, which in turn facilitates acquisition, recall, and transfer. Another fundamental aspect of effective practice is that it affords opportunities for retrieval and self-assessment [9][18]. Students benefit from being tested, which provides cues and structures to retrieve and reflect on their current knowledge and performance [19, 13]. Importantly, retrieval practice and distributed practice are highly complementary, as frequent practice activities can be combined with assessments to enable well-spaced opportunities for self-testing.

### 2.2 Sequential Pattern Mining in Behavioral Modeling

Several sequential mining algorithms have been adopted in modeling cognitive skills. For instance, a modified version of the generalized sequential mining (GSM) [22] algorithm was applied to mine the frequent patterns of students' action sequences in an online collaborative software development environment. Patterns of the system interactions were extracted and distinguished by stronger and weaker performance groups. Good individual practices were found and the results encouraged poor practices identification in the future. Another common sequential pattern mining technique is SPAM [2], which uses a vertical bitmap data structure with depth-first-search pruning algorithm for identifying frequent patterns among long sequential patterns. Joshua Ho et al. [11] developed a generalized version of the SPAM algorithm, called the Pex-SPAM, which incorporates gap constraints in between patterns. Due to the action sequences can be noisy, for example, students may perform additional or irrelevant tasks in-between actual learning activities. Therefore, fusing the constraints becomes an important step in mining the logs from the online learning systems. This technique was applied to analyze the stability of students' learning behavior (AKA: problem solving genome) and found that student peers appeared to perform relatively similarly in the context of parameterized quizzes [7]. Students' action sequences were built using two alphabets, $s$ and $f$ to denote success or failures respectively. These alphabets in lowercase and uppercase denoted short and long duration while solving subsequent problems. The researchers used the Pex-SPAM algorithm to find frequent patterns and termed these "problem solving genomes". The genomes were then employed to analyze the stability of the learning behavior of students at the temporal and spatial levels. The researchers also analyzed the effect of the difficulty level of a problem on the genome. The results of that research cannot indicate whether problem-solving genomes in other subject domains might show similar properties.

Kinnebrew et al. [15] developed an ITS called Betty’s Brain [14] and created action sequences of each student based on Betty's Brain logged details. The researchers used gap constraint-based SPAM to find common patterns among all students. To identify the most interesting patterns, they developed a novel technique called differential sequential mining. This technique uses t-tests to find differentially frequent patterns and examines whether there is a statistically significant difference in a pattern's frequency for each sequence, for higher and lower performing students. This analysis distinguishes the learning behaviors of high and low performing students. Kinnebrew et al. also analyzed the reading behaviors of high and low performing students during their productive and unproductive phases of work [14]. Herold et al. [10] collected students' actions which were handwritten using a digital pen. Variables such as ink color and the duration of strokes were also logged. Action sequences were developed from the raw data, with each action representing an assignment number, topic type, and duration. Differential sequential mining technique was used to find patterns for strong and weak students, and a linear model was proposed to predict students' performance in the course based on these patterns.

### 3. METHODOLOGY

#### 3.1 Research Platform

**QuizIT** is a daily quiz system that generates a “quiz of the day” for programming concepts. It was developed and adopted in practice at Arizona State University. Students are not required to use QuizIT but are encouraged to do so and are awarded extra credits if they do. Each day, QuizIT generates a random question with a multiple-choice answer; only one of the options presented is correct. There is no time limit on choosing the answer. The QuizIT system evaluates the student’s response and reports whether the answer is correct or incorrect. Figure 1 shows the user interface of QuizIT. The left panel shows the question of the day, and the right panel shows the student discussion board related to questions that were previously attempted by other students. The discussion board also provides a platform to post comments. The student has the freedom to attempt the quiz of the day as well as previous questions (which were either attempted or not by the student). Each question is marked as a difficulty level of Easy, Moderate, or Difficult.

**Figure 1:** QuizIT daily programming system interface

#### 3.2 Data Collection

The data of students enrolled for CSE 110 in Fall 2016, and who used QuizIT, were collected. CSE 110 is an introductory Object-Oriented Programming course at undergraduate level at Arizona State University. QuizIT has 110 exercises organized across 18 topics (such as strings, expression, and
method), each of which is labeled as either easy (71 exercises), moderate (30), or difficult (9). In 2016, 375 students were enrolled in CSE 110 but QuizIT was used by 187 students; 5963 correct attempts were recorded, and 4094 incorrect attempts. Exercises at the easy level were attempted 5562 times, moderate questions were attempted 3150 times, and difficult questions were attempted 1345 times. The timestamp at which the questions were attempted was also recorded.

3.3 Building Action Sequences
The creation of action sequences was crucial because they represented discrete actions, which are suitable for differential pattern mining. Each action is a set of alphabets representing the particular event. The event is characterized using the difficulty level, correct or incorrect attempts, and duration between successive actions.

We build students’ sequences of actions by aggregating the same student ID, sorted timestamp and the overall duration. Each event segment was labeled using alphabets and numbers denoting a particular action. Each action segment was labeled with a triple \{(L, F, D)\} where \(L \in \{\text{Easy, Moderate, Difficult}\}\) represents the difficulty level, \(F \in \{1, 2\}\) represents the correctness flag where 1 denotes a correct attempt and 2 denotes an incorrect one; and \(D \in \{F, S, M, L, VL, XL\}\) represents the duration of the action. If the student attempted a particular question for the first time, it was denoted as first occurrence (F). For successive re-attempts, the terms S, M, L, VL and XL were used to indicate an action of small, medium, long, very long, and extremely long durations, respectively. For instance, <Easy-1-F> represents a question of easy difficulty that was answered correctly on the first attempt.

Most of the researchers in their prior work on behavioral mining divided the raw data into sessions to form action sequences. However, it was interesting to examine the changes in learning behavior over a semester, considering the intermittent breaks between the action sequences of a student. Therefore, a different way of labeling the durations was devised. To determine the cut-off points for each duration category, univariate k-means clustering in 1D using dynamic programming was used [23]. Among the 10057 data entries, 5344 entries were non-zero (i.e., there were 5344 sequential time data points which were not a first occurrence). Among these, 5299 data points fell below 1 hour; the rest of the data were treated as being of too long a duration and were excluded as outliers to achieve better convergence in the k-means.

The threshold was set at 3600 seconds to exclude more points as outliers and to prevent overfitting and retain the cut-off boundaries. The Bayesian information criterion (BIC) [21] was used for model selection, with models having lower BIC being preferred. The best BIC for four cluster centers was calculated at -39192.96. The resulting thresholds were 6.3287 seconds for small duration, 121.0022 seconds for medium duration, 394.4931 seconds for long duration, and 914.5216 seconds for very long duration. Finally, to create action sequences from these action segments for each student, the data were first sorted by timestamp and then grouped by student IDs.

Students received percentages and were graded on their performance in assignments, midterms, and finals in CSE 110. Using these data, each student’s action sequence was assigned to a performance group. An action sequence was assigned to the high-performance group if the student’s percentage fell above the median for the class percentage. Similarly, the action sequence was assigned to the low-performance group if the student’s percentage fell below the median of the class percentage. The data were divided into these two groups because the differential sequential mining that would be employed uses two databases as inputs. Hence the division would help to highlight the difference between the learning patterns of higher and lower performing students.

3.4 Differential Pattern Mining
To identify patterns that were distinctive to either higher or lower performing students, the differential pattern mining technique was used. This technique was developed by Kinnbrew and Biswas [15] and it uses two sequence databases, known as the left database and the right database. The order of the left and right groups did not matter. The algorithm uses two important metrics to measure the support of patterns: \(s\)-support and \(i\)-support. \(s\)-support of a pattern is defined as the number of sequences that contain a particular pattern. Patterns meeting the \(s\)-support threshold are known as \(s\)-frequent patterns; \(i\)-support of a pattern is defined as the maximum number of times the pattern appears within a sequence without any overlap. For example, a sequence database has 5 sequences, with the first 3 sequences showing 1 instance of a specific pattern, and the last 2 sequences showing 4 instances of the same pattern. In this case, the pattern has an \(s\)-support of 5 and an \(i\)-support of 1 and 4 in the first and last sequence respectively. The mean \(i\)-support of the pattern for this sequence database will be 2.2.

The algorithm begins by finding all the patterns in the left and right sequence databases that meet the \(s\)-support constraint. To find the initial set of \(s\)-frequent patterns, an open-source data-mining library called SPMF [6] was used, which takes into account the gap constraint. A maximum gap constraint of 2 was used in this work.

In the next step, to compare frequent patterns across both databases, the mean \(i\)-support of each pattern was computed in the left and right databases. Computation of \(i\)-support also considers the maximum gap constraint of 2 to allow for noise interspersed between the action sequences. After this, the t-test was performed for all the \(s\)-frequent patterns. This step determined whether an \(i\)-support value for a particular pattern in the left database differed significantly from the \(i\)-support value of the same pattern in the right database. If the p-value of the t-test result was below the set p-value threshold (based on confidence intervals), that pattern was considered to be differentially frequent. Although this approach employs multiple t-test comparisons between the databases, it is important to note that the t-test was not used to statistically prove that the left and right databases differ. Also, neither Bonferroni or other corrections were used to determine the p-value threshold for rejecting the null hypothesis. Rather, it was used as an exploratory-analysis heuristic to extract more interesting patterns for specific characteristics that were relevant to the two student
groups. After that, the mean values for i-support for the left and right databases were compared to see which patterns emerged more in one group. This comparison yielded four types of differentially frequent patterns:

- s-frequent in both databases but mean i-support higher in left database ($ptrns_{both\left\rangle\left\rangle}$)
- s-frequent in both databases but mean i-support higher in right database ($ptrns_{both\left\rangle\rangle}$)
- s-frequent only in left database ($ptrns_{left\left\rangle\rangle}$)
- s-frequent only in right database ($ptrns_{right\left\rangle\rangle}$)

In this work, we only considered the last two cases ($ptrns_{left\left\rangle\rangle}$ and $ptrns_{right\left\rangle\rangle}$) as they were clear distinctive patterns to separate high-performing students from the weaker ones. The high-performing students were used as the left database and low-performing students as the right one. The s-support was set at 0.3 (i.e., for a pattern to be s-frequent, it must appear in at least 30% of the sequences in a database). The p-value threshold was set at 0.05, or 95% confidence level.

4. RESULTS

We used differential sequential mining and extracted 35 s-frequent patterns. Among these, 23 patterns belonged to higher performing students and 12 to lower performing students. To remove patterns that were strongly correlated with each other and to avoid overfitting, an algorithm named correlation-based feature selection (CFS) [8] was used with 10-fold cross-validation. The CFS algorithm helps in identifying feature subsets having the most weights. Features selected by CFS in more than 5 out of 10 folds were included in the final model. Eventually, there were total 10 patterns extracted based on CFS. Table 1 shows all the patterns with their p-values and mean i-support values. Based on the extracted 10 patterns, seven learning behaviors were observed, 8 were belonged to high-performing patterns and 2 to low-performing patterns. Each pattern is labeled as a learning behavior and is further described in detail in the following:

1. **Persistent-practicing** is shown as $<Easy-1-F> <Easy-1-F> <Easy-1-F>$. When a student repeatedly solved one or more problems of the same difficulty level correctly, all on them were the first attempt and attempted successfully. This behavior aligns with QuizIT’s design rationale, which is to help students space their learning opportunities, by practicing simple problems distributively.

2. **Jump-forward-progression** behavior, such as $<Easy-1-F> <Easy-1-F> <Medium-1-F>$, is seen that students repeatedly solved one or more problems of the same difficulty level correctly; then they progressively solved a problem of higher difficulty level, all of them were the first attempts. Such behavior demonstrates the testing effects of QuizIT in preparing students to solve harder problems in future after practicing simpler ones. Among 10 observed patterns, three jump-forward-progression patterns represented this behavior.

3. **Steady-progression**, $<Easy-1-M> <Medium-1-F>$, illustrates that students took their time when attempting a question, although they did not answer it correctly on the first try. Eventually they solved the problem correctly and moved on to another more complex problem.

4. **Experimental-progression**, $<Easy-1-F> <Medium-2-F> <Medium-1-M>$, occurs when a student correctly solved a question of a certain difficulty level but made mistakes when attempting a question of a higher difficulty level. In this case, the student re-attempted the incorrectly solved question, took a while to comprehend the question, and finally solved the question correctly. This behavior is particularly interesting as it shows the student’s effort to learn from mistakes while solving a tougher problem and eventually solving it.

5. **Jump-backward-progression** behavior, such as $<Medium-1-F> <Easy-1-F> <Easy-1-F>$. The student correctly solved a relatively difficult question and then correctly solved one or more problems of a lower difficulty level, all of them were the first attempts. This behavior suggests that some student may not understand the medium level problem. Their first attempt for medium problem can be a guess, so they jump backward to restart practicing. This pattern is a good evidence to show that these students are practicing on a same set of concepts until they are ready to move on, and the system can adapt to such different learning paces.

6. **Struggling**, $<Medium-2-F> <Medium-2-S> <Medium-1-S>$, expresses the behavior in which the student solved questions of same difficulty level repeatedly. Among all the attempts, there were only limited correct ones in the middle. This behavior suggests that the student struggled to get the answer right, and may have applied trial-and-error strategy to get the limited correct answer. This is because the student may have taken too short of the time to re-attempt the questions. Such strategy can be an indicator of an alarming learning behavior, which the student should have spent longer time to think and work out the problem, instead of rushing to submit answers.

7. **Withdrawal**, $<Easy-2-S> <Medium-2-F>$, is a behavior where the students ventured too fast to more challenging problems without getting a correct attempt on easier level of question. Such behavior is especially undesirable as it shows that students simply give up on the problem if they are unable to solve it. This can be a real performance harming pattern.

The distribution of learning behaviors across the two performance groups is shown in Figure 2. All learning strategies are used by students in both groups. However, the strategies of persistent-practicing, jump-forward-progression, steady-progression, experimental-progression, and jump-backward-progression are mainly used by higher performing students. Struggling and withdrawal are strategies used by lower performing students. The most observed behavior among high performers is persistent-practicing, which was used by 63.29% of top students. Jump-backward-progression and jump-forward-progression were used by 50% and 47.25% of high performers, respectively. This finding shows that half of the top
Table 1: Patterns filtered by CFS algorithm

<table>
<thead>
<tr>
<th>Pattern Category</th>
<th>Pattern</th>
<th>Mean i-support</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Easy-1-F&gt; &lt;Medium-2-F&gt; &lt;Medium-1-M&gt;</td>
<td>HIGH</td>
<td>0.0107</td>
<td>0.0001</td>
</tr>
<tr>
<td>&lt;Medium-1-F&gt; &lt;Easy-1-F&gt; &lt;Easy-1-F&gt;</td>
<td>HIGH</td>
<td>0.0072</td>
<td>0.0002</td>
</tr>
<tr>
<td>&lt;Easy-1-F&gt; &lt;Easy-1-F&gt; &lt;Easy-1-F&gt;</td>
<td>HIGH</td>
<td>0.0221</td>
<td>0.0001</td>
</tr>
<tr>
<td>&lt;Easy-1-F&gt; &lt;Easy-1-F&gt; &lt;Difficult-1-F&gt;</td>
<td>HIGH</td>
<td>0.0101</td>
<td>0.0005</td>
</tr>
<tr>
<td>&lt;Easy-1-F&gt; &lt;Easy-1-F&gt; &lt;Medium-1-F&gt;</td>
<td>HIGH</td>
<td>0.0093</td>
<td>0.0005</td>
</tr>
<tr>
<td>&lt;Easy-1-F&gt; &lt;Easy-1-F&gt; &lt;Easy-1-F&gt;</td>
<td>HIGH</td>
<td>0.0154</td>
<td>0.0008</td>
</tr>
<tr>
<td>&lt;Easy-1-F&gt; &lt;Easy-1-F&gt; &lt;Easy-1-F&gt;</td>
<td>HIGH</td>
<td>0.0286</td>
<td>0.0002</td>
</tr>
<tr>
<td>&lt;Easy-1-F&gt; &lt;Easy-1-F&gt; &lt;Medium-1-F&gt;</td>
<td>HIGH</td>
<td>0.0259</td>
<td>0.0041</td>
</tr>
<tr>
<td>&lt;Medium-2-F&gt; &lt;Medium-2-S&gt; &lt;Medium-1-S&gt;</td>
<td>LOW</td>
<td>0.0086</td>
<td>0.0082</td>
</tr>
<tr>
<td>&lt;Easy-2-S&gt; &lt;Medium-2-F&gt;</td>
<td>LOW</td>
<td>0.0086</td>
<td>0.0452</td>
</tr>
</tbody>
</table>

students tended to correctly solve an easier question after attempting a tougher question, which suggests a confidence gain regarding specific topics. Withdrawal was the behavior seen most often among low-performing students, at 39.24%, followed by struggling, at 34.17%. Interestingly, the third most used strategy in the lower performing group was persistent-practicing, at 29.11%. This finding supports the design rationale of QuizIT regarding students’ need to practice solving simple questions.

Figure 2: Learning behavior distribution across the two performance groups

To verify high and low performing students’ problem solving strategies and their performances, we looked at their final course performances. The distribution of learning behaviors with respect to final grades is shown in Figure 3. The students’ grades were classified according to four main grade lists, namely A, B, C and D. The bar chart shows that 64.47% of A-grade students adopted persistent-practicing. Jump-backward-progression and jump-forward-progression were used by 50% and 48.68% of A-graders respectively. This result shows that most A-graders were using five strategies pertaining to the highly performing group. Among the B-graders, 37.7% of students showed persistent-practicing. However, withdrawal was the most prominent behavior in this group (41.5%) and struggling was the third most prominent behavior (32%). This finding shows that more B-grade students employed negative behaviors than positive ones. Hence, most of them were average students and fell into the group of lower performing students. Among the C-graders, 50% of the students who withdrew also struggled, which means that both these behaviors contributed negatively to the students’ performance. Interestingly, more C-graders showed experimental-progression behavior than B-graders. With regard to the D-graders, most both withdrew and struggled, although one D-grader had adopted jump-backward-progression as a learning strategy. This suggests that student should have followed QuizIT’s progression instead of venturing too fast too soon, especially for lower-performing students.

Figure 3: Learning behaviors by performances

5. DISCUSSIONS

This work aims to explore students’ learning strategies to solve multiple choices programming questions by using differential mining technique. We hypothesized high and low performing students would have distinct problem solving sequential patterns. The results showed that there are seven distinguishing learning behaviors. Among these, five behaviors were prominent in the high-performing group and two were prominent in the low-performing group. By identifying these patterns between high and low performing groups permits to build predictive models to estimate their performances and to provide interventions accordingly. For instance, breaking the repeated actions of the students who keep working on same difficulty level of questions and to result in struggling behavioral pattern. In addition, the potential behavioral features can be used to provide information to help instructors to address the issues faced by their students and to suggest a better learning path. Most importantly, the preliminary results suggested that given the generic type of problem-solving questions (MCQs), stronger and weaker students would still apply different strategies and approach the problems differently.

6. REFERENCES