Implementation and Evaluation of Technical Interview Preparation Activities in a Data Structures and Algorithms Course

Amanpreet Kapoor
Engineering Education
University of Florida
kapooramanpreet@ufl.edu

Sajani Panchal
Computer & Info. Science & Engineering
University of Florida
sajanipanchal@ufl.edu

Christina Gardner-McCune
Computer & Info. Science & Engineering
University of Florida
gmccune@ufl.edu

ABSTRACT

This experience report describes and evaluates the introduction of Hire Thy Gator technical interview preparation activities in a Data Structures and Algorithms (DSA) course. Our intervention included a panel on internship experiences, a role-play interview demonstration, two participatory mock interview preparation exercises where students interviewed each other first using self-selected peers and second through random pair-ups, and graded short programming problems. We (1) explain the logistics and rationale for embedding these activities, (2) describe the lessons learned and evolution of the activities beyond the intervention semester, and (3) evaluate the impact of these activities on students. We report data from 257 students who participated in our intervention and 106 students who were a part of a control group. Students found that our activities promoted awareness of the recruitment process, allowed them to self-evaluate their strengths and weaknesses, and prepared them for technical interviews. Quantitatively, the intervention cohort reported a higher average normalized confidence gain (0.42) than the control group (0.36) indicating that our activities can aid in building students' confidence.

CCS CONCEPTS

• **Social and professional topics**~Professional topics~Computing profession~Employment issues

KEYWORDS

technical interview, employment, data structures, algorithms

ACM Reference format:

Amanpreet Kapoor, Sajani Panchal, and Christina Gardner-McCune. 2023. Implementation and Evaluation of Technical Interview Preparation Activities in a Data Structures and Algorithms Course. In Proceedings of *54th ACM Technical Symposium on Computer Science Education (SIGCSE '23), March 2023, Toronto, Canada.* ACM, New York, NY, USA, 7 pages. DOI: https://doi.org/10.1145/3545945.3569755

SIGCSE 2023, March 15–18, 2023, Toronto, ON, Canada

ACM 978-1-4503-9431-4/23/03…$15.00
https://doi.org/10.1145/3545945.3569755

1 Introduction

One role of computing degree programs is to educate individuals so they can contribute to the economy by joining the workforce. This goal aligns with the majority of computing students' aspirations to secure jobs in the industry after graduation [15]. Unfortunately, most undergraduates in a computing major have to devote career preparation time for technical interviews outside of coursework as these interviews act as gatekeepers to internships and full-time jobs in the technology industry [2, 18]. This need for time outside of the curriculum is unfavorable and inequitable, especially for students of low socioeconomic backgrounds who may not have substantial time outside of the curriculum due to family or work responsibilities [14]. In addition, students have reported that technical interviews often induce anxiety and stress [3]. To solve these issues, we introduced Hire Thy Gator technical interview preparation activities in our Data Structures and Algorithms (DSA) course to familiarize students with the interview process and build students' confidence to succeed in these interviews. The content of the technical interviews have a broad overlap with DSA courses [18] and hence our activities can scaffold the transition between coursework and technical interview preparation providing an equitable way for students to prepare for these interviews.

Our intervention included a panel on internship experiences, a role-play interviewing demonstration, and two participatory mock interview exercises where students interviewed each other. In addition, students also independently solved one or two graded short programming questions similar to technical interview questions every week on an online system for programming problems. Since there is extant literature on the efficacy of short programming problems in CS Education research (CER) [6, 7, 16], in this report we focus on our rationale, execution, and evaluation of the mock interviews and associated activities. Our work contributes rich descriptions and preliminary evaluation of a scalable, collaborative, and formative professional development activity which can support students' awareness and preparation for future technical interviews.

2 Prior Work

2.1 Employment recruitment process in the US

The hiring process in the US varies for roles which span eclectic computing areas such as software engineering, data science, user experience design, etc. Industry employers hire interns and full-time employees for these roles through a multi-stage competitive recruitment process [18, 21]. The process typically has three stages: an application phase, an interview phase, and a negotiation phase. During the application phase, an applicant applies to various roles and companies by submitting their resumes and answering questions on digital applications, career fairs, or company information sessions. The applications are screened and candidates are selected for the next stage which is the interview phase based on a student's experience, GPA, and involvement in projects [21].

Companies invite applicants for one or more technical and behavioral interviews and there is variation in the number and rigor of the interviews depending on the job role and companies. A majority of companies ask students technical questions in an interview related to DSA, especially for software development and engineering positions. In these interviews, applicants are asked to either write programs on whiteboards or shared screen text editors and talk out loud about their thought processes when solving a problem. Applicants are evaluated on problem-solving skills, professional skills such as communication skills, and the ability to derive correct solutions in a limited timeframe. Finally, in the third stage, an offer is made by the company and the applicant has an opportunity to negotiate. Our intervention focuses on preparing students for the interviewing phase of software development and engineering jobs given their prominence and the overlap with our DSA course.

2.2 Technical interviews in CER

Research on technical interviews in literature spans three areas:

Employer-centric research on structure and expectations in a technical interview: Work that explored expectations of employers in a technical interview includes Ford et. al's work [8] on interviewers' expectations from potential software engineer candidates. They found that interviewers were not only interested in the technical problem-solving ability of the candidates, but also the interpersonal skills such as effective communication skills [8]. Another work assessing the structure of technical interviews by Stepanova et. al. [21] found that recruitment professionals reported differences in interview structure across companies with variations in components like coding tests, on-site interviews, team interviews, or behavioral interviews. However, it is evident from the aforementioned studies that technical interviews are used as a primary recruitment tool for securing jobs in the computing industry. Given that not all students have a considerable amount of time to prepare for these interviews outside the curriculum [14], we wanted to introduce an intervention that can provide preliminary exposure to technical interviews to our students.

Student-centric research on interview participation and factors that influence success: Studies have also explored student participation in technical interviews and factors that promote or hinder success in the interviews. This work includes Wyrich et al.'s study [23] which identified the individual characteristics of students' performance in solving coding challenges and found that students who completed coding challenges had higher grades, more programming experience, and higher happiness. Lunn et. al. [17] observed similar results and found that students who had more coding experience had positive experiences with technical interviews and a higher computing identity. Another example is Hall and Gosha's study [11] which identified African American students' participation in technical interviews and found that interview performance decreased with increasing anxiety and the anxiety decreased as students participated in more interviews. Other studies [3, 5] have also found that interviewees participating in technical interviews have experienced stress and anxiety which prohibits their performance. In short, these interviews can be stressful and higher participation may yield better outcomes. So why not use supplementary formative exercises in coursework to help students' feel more confident in their ability to excel in technical interviews? We aim to abate these issues through our exercises.

Practitioner-centric research on designing interventions for interview preparation: A few interventions have been introduced in computing classrooms [5, 22] or through academic-industry partnered programs [1] which were intended to prepare students for technical interviews. These include Urness's work [22] on the introduction of technical coding exercises in a CS2 course in the form of programming assignments. However, this intervention focused on individual problem-solving akin to a coding test which is seldom a precursor to an actual technical interview [8]. Another work by Dillon et. al. [5] incorporated and evaluated the efficacy of the inclusion of coding exercises in CS2 and Object Oriented Programming courses where students were assigned into groups of three and asked to think aloud and explain solutions using Zoom breakout rooms to their peers. They found that students received the activities positively but still showed adequate levels of anxiety. The latter intervention was introduced in a smaller course and the interview questions were provided by the instructor for the group. The setup did not consist of dyads with an interviewer and interviewee role. Our intervention is different from this intervention as we tried to mimic the more prevalent dyad interview format and we present how to scale our activities in large classrooms using peer interview approach.

3 Settings

3.1 Educational Institution

Our intervention was introduced in a DSA course at a large public university in the southeast USA in the Fall 2020. At the research site, admission in undergraduate degree programs is competitive and participation in industry internship(s) before graduation is not mandatory. Our DSA course is a required

course for CS and Computer Engineering majors and CS minors. It follows the CS1, CS2, and Discrete Mathematics courses at our institution and students have prior knowledge of programming in C++ and Java. 250-450 students enroll in the course in Spring and Fall and 100-150 in the Summer. For this paper, we use data from 257 students who consented and participated in our intervention in Fall 2020 and 106 students who were enrolled in Summer 2020 in our course and did not participate in our activities (control group).

3.2 Course Structure and Content

Our course covers different DSA-related topics such as Algorithm Analysis, Sets, Maps, Trees, Graphs, Greedy Algorithms, etc. The language of instruction is C++, and the course has an equal mix of theory and practice. For the latter, students solve short programming DSA problems on a browser-based system and work on projects. The course was worth 4 credits in Summer and Fall 2020 and students had to attend three lectures led by the instructor and one discussion every week led by a teaching assistant. The course lasts 15 weeks in Fall and 12 weeks in Summer. In both Summer and Fall 2020, the course was online due to Covid-19, was taught by the first author, and followed a hybrid format structurally consisting of two remote synchronous lectures and discussion and two remote asynchronous pre-recorded lectures. Students were tested on a weekly quiz, two individual projects, a final ill-structured and self-proposed group project, and two exams in both semesters.

4 Logistics

Our technical interview exercises were designed after taking input from the students in Week 2 of the Fall 2020 intervention semester. In the second week of our course, we added a few optional ungraded questions to the first quiz which asked students about their familiarity with technical interviews. Most students (58% or 143 of the 248 students who answered this question) were not familiar with the technical interviews. Quite a few students (30% of 248, n=75) were familiar with technical interviews but had not participated in them. The remaining students had participated in a technical interview but failed to secure an internship (6%, n=14), cleared a technical interview and had interned (6%, n=14), or were not interested in computing careers (2%, n=5). Three students selected more than one statement and hence the numbers don't add to 100%. Since the awareness of the technical interview process was quite low, we decided to incorporate two activities: a panel and a role-play exercise before asking students to participate in mock interviews (see Figure 1).

4.1 Panel

The first activity was a panel hosted in Week 5 of our course. The goal of the panel was to make students aware of the importance of internships and introduce them to the recruitment process. The panel consisted of four undergraduate TAs and was moderated by the instructor. All four TAs had worked as an intern in top tech companies in the US such as Alphabet and Microsoft. The panel revolved around the technical interview process, former participation experiences, and the strategies TAs used for successfully securing an internship. This panel was conducted outside of course hours and lasted 45 minutes.

4.2 Role play demonstration

The second activity consisted of a role-play demonstration which was organized in Week 6 of our course. During this exercise, the TAs role-play acted as an interviewer and an interviewee in a weekly discussion session to show students what they could expect in a technical interview. We emphasized an iterative approach to solving a problem and underscored the importance of asking follow-up questions as well as writing pseudocode or explaining the solution in words before writing actual code. We also highlighted that interviewers can provide hints if the interviewee is stuck for too long. The role play exercise ended with a conversation between the interviewer and interviewee reflecting on the strengths and weaknesses of the interviewee and how an interviewee can improve. Although the latter conversation is not a part of an actual technical interview, we wanted student interviewers to understand how to conduct a discussion after the mock interview for providing constructive feedback to the interviewee. The reflection and feedback components were added as our activities are formative and the feedback can better prepare students for subsequent interviews. The TAs spent three hours preparing for this exercise and the actual class discussion session lasted another 50 minutes.

4.3 Mock interviews

After the role-play demonstration, students were asked to work in pairs for two mock interviews in the middle of the semester (Week 8) and the second-last week of the course (Week 14). In the first mock interview, students were allowed to self-select peers with who they wanted to work. If they could not find a partner, the matching was facilitated by the instructor. In the second mock interview (Week 14), we randomly paired them with another student. This allowed us to scaffold the social interaction of the students and reduce the interview anxiety which is common in technical interviews [3, 5]. In each activity, the students were asked to interview each other with every student acting as an interviewer and an interviewee. We wanted students to act as an interviewer so that they can gain exposure to the recruitment side. In addition, we wanted to make the activity scalable for large classes where the course staff does not have enough resources to interview each student individually and this setup allowed minimal resources with the necessary benefits of exposure to a technical interview.

The activity descriptions were created on our learning management system (LMS), Canvas. Each student was asked to fill out two graded survey assignments for each round of interviewing: one as an interviewee, and another as an interviewer. To help students prepare as an interviewee or an interviewer, we also provided optional resources in survey descriptions such as YouTube videos from Google on what interviewers seek out from candidates in a technical interview or

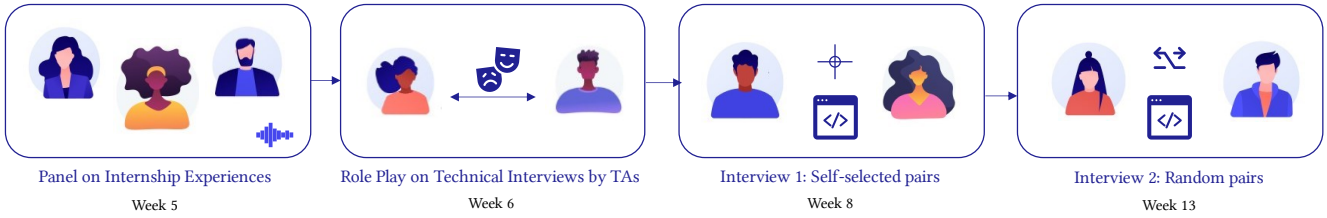


Figure 1. Logistics of Embedding Hire Thy Gator Technical Interview Exercises in a Data Structures and Algorithms Course

from Gayle McDowell, the author of Cracking the Coding Interview [18] on how to approach interviews. The descriptions also pointed to links to two sample interview questions.

The assignment descriptions also had instructions for the interviewer and interviewee. Interviewers were asked to (1) research the question they were going to ask the interviewee, (2) coordinate the time, (3) record the interview and keep track of the solution document, (4) give the interviewee hints if they are stuck for too long, and (5) provide actionable feedback to the interviewee reflecting what were the strengths of the interviewee and what can the interviewee improve on. We asked interviewers to record the interview using Zoom and provide the candidate with a Google document link to write the solution. We also suggested the interviewer a 40-45 minute window for the technical interview and a 15-20 minute session on giving actionable feedback to the interviewee and filing the survey. If students spent less than 20 minutes on the activity, we required them to ask an additional technical question or conduct the activity again. For the first set of interviews, interviewers were supposed to pick a question on Trees or Heaps, and for the second one on Graphs or Sets and Maps which are common topics that are covered in technical interviews [18]. This alignment was based on the topics covered in our course during respective times. After the interview, the interviewer was supposed to fill out a survey where they entered a link to the recorded interview, the solution document, and a few reflection questions on their as well as the interviewee's performance. They were also asked for any feedback on our activity.

The interviewee's assignment had a description of their responsibilities. They did not know the question beforehand, but they knew that the interviewer would ask questions about the covered topics. The interviewees were told to walk through their approach to solving a problem before coding the solution, ask questions to clarify constraints, improve their solution iteratively, and walk through their solution with a test case to identify any possible bugs. After the activity, the interviewees reflected in the survey their strengths and weaknesses and their experience in the activity. Instructors can find all relevant materials to incorporate our exercises here [24].

The mock interviews were graded based on participation and carried 8% of the points of the final grade (2% for acting as an interviewee and 2% for being an interviewer for each activity). Students (N=257) self-reported average time for preparation and participation in the interview was 2 hours for acting as an interviewer and 1 hour 52 minutes as an interviewee per activity.

4.4 Time requirements

To sum up, practitioners replicating the technical interview preparation activities can expect to utilize 6 hours of fixed time to introduce these activities. This time includes 1 to 2 hours of course instruction time for conducting the panel and role-play demonstration and 1 to 4 hours of preparatory time for setting up and organizing the activities. Additionally, variable time spent would cost another 1 to 2 minutes for grading each student submission per activity. The latter time would be more if an instructor wants to provide personable feedback to each student.

4.5 Lessons learned and evolution

Pairing facilitation: A problem that quite a few students (approximately 15-20%) faced was a lack of communication and scheduling issues with the assigned partner. We received several messages on our discussion tool regarding these issues which is an overhead, especially in large classes. We mitigated these problems by assigning a new partner who had a similar issue. In hindsight, we should have provided two deadlines for each interview activity: (1) communicate with the partner and set up the time for the interviews, and (2) the deadline for the actual interview and deliverables.

Alternate assignment: 4% of students did not participate in the interview activities. The instructor reached out to these students asking if they wished to justify why they didn't participate and offered them an alternate assignment as it had a non-trivial impact on student grades. Five students responded that they did not participate because of social anxiety, lack of interest in CS jobs, or lack of time. For instance, a student stated, "*There were a few reasons I didn't complete the assignment, the main ones being my social anxiety/difficulty interacting with people I don't know [...] and that I am not expecting to look for/apply for a job in this field*". The students who did not participate were given alternate coding problems. We recommend other instructors offer alternate activities for such students.

Reduction in grading weights to account for time: Our assumption was students would spend 7 to 8 hours preparing and participating in each round of an interview for both roles. However, the self-reported time spent was less than our anticipation, and students spent on average 4 hours per interview. Hence, in subsequent iterations of the course, the grading structure was reevaluated to account for the time spent

and the grading percentage was reduced from 8% to 5% for participating in the two activities.

5 Evaluation Methodology

5.1 Study Design and Participants

To evaluate our activities, we designed a survey-based study using a retrospective post then pre-design [20]. In this design, a survey is disseminated at the end of an educational activity, to gauge a participant's change in attitudes, knowledge, or confidence. Data is collected only once, and participants state their confidence level at the end of an activity (post) and retrospectively gauge their confidence level at the beginning of the activity (pre). This type of study avoids pretest sensitivity and response shift bias that results from pretest misestimation. Response shift bias occurs when participants use different frames of understanding about a question between pre and post-intervals [13]. We use data from a research survey disseminated at the end of control and intervention semesters to understand what students gained from participating in our interview exercises. In addition, we seek to answer the following research question: *How does participation in technical interview preparation activities influence students' confidence levels for programming in a technical interview?*

Our study was approved by the Institutional Review Board at our university. 345 students were enrolled in our course in Fall 2020 and 279 students consented to the study (Response rate: 80.9%). Of the 279 students, 22 students' data were discarded due to missing data. Thus, our intervention corpus consists of 257 students. In addition, 143 students enrolled in our course in Summer 2020. The data from this cohort was used as a control group as our activities were introduced after this semester. For Summer 2020, 115 students consented to research (Response rate: 80.4%). Our control group corpus consists of data from 106 students after deleting missing values. A majority of students in our corpora were CS majors (66%) and CE majors (18%), enrolled in their sophomore (Year 2, 62%) and junior (Year 3, 26%) years. Gender proportions were 71% males, 28% females, and 1% others.

5.2 Data Collection and Analysis

We use three questions from the research survey in our analysis: (S1) *Did you find the activity valuable? Should this be continued in the future?*, (S2) *How confident were you with your ability to program in programming interviews before the starting of this course?*, and (S3) *How confident are you with your ability to program in programming interviews at the end of this course?* The last two questions were 5-point ordinal scale questions (Not confident (coded as 0) – Extremely confident (coded as 4)). Note that question S1 was included only in the Intervention survey.

We use representative quotes from survey question (S1) to demonstrate what students gained from our activities. Due to space constraints, we did not focus on a more structured qualitative analysis. For answering our RQ., *How does participation in technical interview preparation activities influence students' confidence levels?*, we took a quantitative approach. We coded survey questions, S2 and S3, and applied nonparametric statistical tests to determine the significance of our results across the population of undergraduate students. A Mann-Whitney U Test was conducted to evaluate differences in pre- and post-data from independent samples as our data did not follow a normal distribution. The null hypothesis for these tests asserts that the median pre or post students confidence levels of the two samples (control and intervention) are identical and a p-value < 0.05 was used to reject the null. Additionally, we used a confidence gain metric similar to Hake's learning gain metric [4, 9] as there was a significant difference between the pre-confidence levels of our control and intervention cohorts. The confidence gain metric would account for cohorts that may have higher confidence than others when they begin the semester and it was computed as:

$$\textit{Average normalized confidence gain}, \langle g \rangle = \frac{\langle \%\ \text{Post} \rangle - \langle \%\ \text{Pre} \rangle}{100\% - \langle \%\ \text{Pre} \rangle}$$

where *<% Post>* and *<% Pre>* measures are the final and initial course averages of self-reported confidence computed as a percentage. Our confidence gains are computed at a classroom level (gain of averages method, [10, 19]).

5.3 Limitations

To evaluate our activities for confidence building, we compare data from the intervention semester with a previous semester's data. Both cohorts were taught by the same instructor. There were however two differences between the offerings. Both changes pertained to the grading structure, but the course content was the same. First, we introduced graded participatory coding exercises. The problems were available to the students in the control group but they were optional. However, in the intervention term, students could receive 5% points if they attempted 21 or more of the 55 problems. This change in the grading rubric was based on the control cohort's feedback which mentioned that students were spending significant time on these problems and found them useful for interview preparation. Second, we introduced the mock interview exercises which carried 8% points of final grades. Hence, to account for students' time on our formative activities, we made room in the grading rubric for the intervention cohort. 2-3% points were reduced from other assessment grade weights. The scope of these assessments was adjusted to make up for the increased workload. Our assessment of the intervention for building confidence could be attributed to a combination of the two activities (graded short programming problems or mock interview exercises) which pertained to technical interview preparations. In the future, the efficacy of the activities can be assessed in isolation through more structured quasi-experiments.

6 Evaluation Results and Findings

6.1 Impact of our activities

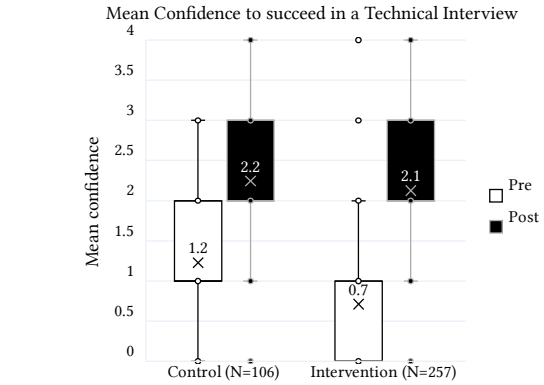
Students received our intervention positively and mentioned that the exercises aided them to understand the technical interview process, prepared them for future interviews, allowed them to apply coursework practically, built their confidence to secure a job, motivated them to apply for a job, and helped them in

knowing their strengths and weaknesses. For instance, a student described how the activities generated awareness and they believed "*that the [interview] exercises were important in getting familiarised with the programming interview process. This [was] especially true for people like [them] who had never done a live programming interview before*". Another student mentioned how the activities supported building confidence and reducing anxiety stating that "*the [interviews] massively improved [their] confidence for interviews*". They further stated "*I think these should be continued as they are great for people like me who have never touched anything remotely close to a technical interview. I think it takes away the uncertainty and fear of these interviews to an extent as it also lets you collaborate with classmates and see their point of views as well*". Students also reported that the activities allowed them to self-evaluate themselves, "*I liked them and they revealed things I need to work on before I do another technical interview*". Lastly, students described how the activities allowed them to apply coursework more practically, "*These were really good for contextualising our course content with something that is very relevant to all of us looking for jobs and internships*".

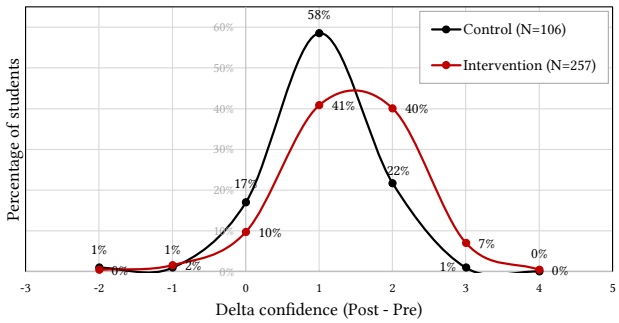
6.2 Efficacy in building confidence

To evaluate the efficacy of our interview preparation activities and graded programming problems on building students' confidence, we asked students to gauge their confidence in their ability to program in technical interviews before and after the course in both cohorts. We hypothesized that participation in our activities would improve students' confidence. However, there was a difference in the pre-confidence levels of our control and intervention cohorts. The mean confidence reported at the start of the semester (pre) by the students in the control group was 1.2 (on a scale of 0-4) while that of the intervention cohort was 0.7 (see Figure 2a). This difference was significant ($Z = 4.9$, $p < 0.001$). The students in the control group were enrolled in the first-semester post the onset of Covid-19 and most students were not participating in internships because of the pandemic. The higher confidence scores could be because of the fact that students had less fatigue and these levels were an anomaly compared to subsequent semesters. However, since we introduced the intervention and students reported significant benefits of the activities, we would prohibit students from the benefits of these activities if we were to reevaluate the efficacy using a new control group. This may violate the educational equipoise principles [12] and hence we recommend researchers who have not offered these activities to verify the efficacy of our activities. There was no significant difference in post confidence levels between the control ($Post_{Summer2020} = 2.2$) and intervention ($Post_{Fall2020} = 2.1$) group, $Z = 1.47$, $p = 0.14$ suggesting that the post confidence levels were almost similar for both cohorts.

Since there was a difference between the two cohorts' pre-measures, we computed the average confidence gain of the intervention and control cohorts. The overall average confidence gain of the control semester cohort was 0.36 while that of the intervention semester was 0.42. The change in confidence levels is shown in Figure 2b. It is evident from this figure that during the intervention semester, a higher percentage of students had a greater boost in confidence levels (40% of students had a 2-point jump in the intervention cohort compared to 22% of students in the control cohort and 7% of students had a 3 point jump in intervention cohort compared to 1% students in control cohort). This difference in confidence gains could either be attributed to our interview activities or graded programming problems. Nevertheless, the increase in confidence gains and student recommendation on the usefulness of our activities is promising.



(a)



(b)

Figure 2: (a) Box plots of Students' confidence; (b) Change in confidence levels between pre and post for the two cohorts

7 Discussion and Conclusion

In conclusion, we presented the implementation and evaluation of technical interview activities. While prior work has focused on incorporating coding exercises in the curriculum to prepare students for technical interviews [22], our work provides an intervention that is closer to an actual technical interview. Moreover, our formative activities can be used as scalable collaborative assessments in large classrooms with minimal time overheads. Similar to prior work which reported that students find interviews stressful and anxiety-inducing [3, 5], we also observed a few students describing that the interviews were stressful. However, our activities were graded based on participation and not correctness and we introduced measures to scaffold the social anxiety such as allowing them to self-select partners in the first round of interviews. Regarding evaluation, the activities were well received. Therefore, we recommend other instructors introduce these exercises, especially in DSA courses given the overlap with course content.

REFERENCES

[1] Alvarez, A., Burge, L., Emanuel, S., Gates, A., Goldman, S., Griffin, J., Keeling, H., Madda, M.J., Okafor, B., Onowho, A. and Washington, G. 2020. Google Tech Exchange: An Industry-Academic Partnership That Prepares Black and Latinx Undergraduates for High-Tech Careers. *J. Comput. Sci. Coll.* 35, 10 (Apr. 2020), 46–52.

[2] Behroozi, M., Parnin, C. and Barik, T. 2019. Hiring is Broken: What Do Developers Say About Technical Interviews? *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (2019), 1–9.

[3] Behroozi, M., Shirolkar, S., Barik, T. and Parnin, C. 2020. Does stress impact technical interview performance? *ESEC/FSE 2020 - Proceedings of the 28th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (New York, NY, USA, Nov. 2020), 481–492.

[4] Coletta, V.P. and Steinert, J.J. 2020. Why normalized gain should continue to be used in analyzing preinstruction and postinstruction scores on concept inventories. *Phys. Rev. Phys. Educ. Res.* 16, 1 (Feb. 2020), 10108. DOI:https://doi.org/10.1103/PhysRevPhysEducRes.16.010108.

[5] Dillon, E., Williams, B., Ajayi, A., Bright, Z., Kimble-Brown, Q., Rogers, C., Lewis, M., Esema, J., Clinkscale, B. and Williams, K.L. 2021. Exposing Early CS Majors to Coding Interview Practices: An HBCU Case Study. *2021 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT)* (2021), 1–4.

[6] Edwards, S.H. and Murali, K.P. 2017. CodeWorkout: Short Programming Exercises with Built-in Data Collection. *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education* (New York, NY, USA, 2017), 188–193.

[7] Edwards, S.H., Murali, K.P. and Kazerouni, A.M. 2019. The Relationship Between Voluntary Practice of Short Programming Exercises and Exam Performance. *Proceedings of the ACM Conference on Global Computing Education* (New York, NY, USA, 2019), 113–119.

[8] Ford, D., Barik, T., Rand-Pickett, L. and Parnin, C. 2017. The tech-talk balance: what technical interviewers expect from technical candidates. *Proceedings - 2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2017* (Jun. 2017), 43–48.

[9] Hake, R.R. 2002. Relationship of Individual Student Normalized Learning Gains in Mechanics with Gender , High-School Physics , and Pretest Scores on Mathematics and Spatial Visualization. (2002).

[10] Hake, R.R. 2001. Suggestions for administering and reporting pre/post diagnostic tests. (2001).

[11] Hall Jr., P. and Gosha, K. 2018. The Effects of Anxiety and Preparation on Performance in Technical Interviews for HBCU Computer Science Majors. *Proceedings of the 2018 ACM SIGMIS Conference on Computers and People Research* (New York, NY, USA, 2018), 64–69.

[12] Hersch, G. 2018. Educational equipoise and the educational misconception: Lessons from bioethics. *Teaching and Learning Inquiry.* 6, 2 SE-Articles (Sep. 2018), 3–15. DOI:https://doi.org/10.20343/teachlearninqu.6.2.2.

[13] Howard, G.S., Ralph, K.M., Gulanick, N.A., Maxwell, S.E., Nance, D.W. and Gerber, S.K. 1979. Internal Invalidity in Pretest-Posttest Self-Report Evaluations and a Re-evaluation of Retrospective Pretests. *Applied Psychological Measurement.* 3, 1 (1979), 1–23. DOI:https://doi.org/10.1177/014662167900300101.

[14] Kapoor, A. and Gardner-McCune, C. 2020. Barriers to securing industry internships in computing. *ACE 2020 - Proceedings of the 22nd Australasian Computing Education Conference, Held in conjunction with Australasian Computer Science Week* (2020).

[15] Kapoor, A. and Gardner-McCune, C. 2018. Understanding Professional Identities and Goals of Computer Science Undergraduate Students. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2018), 191–196.

[16] Kuppuswami, S. and Vivekanandan, K. 2004. The Effects of Pair Programming on Learning Efficiency in Short Programming Assignments. *Informatics in Education.* 3, 2 (2004), 251–266. DOI:https://doi.org/10.15388/infedu.2004.18.

[17] Lunn, S., Ross, M., Hazari, Z., Weiss, M.A., Georgiopoulos, M. and Christensen, K. 2021. The Impact of Technical Interviews, and Other Professional and Cultural Experiences on Students' Computing Identity. *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1* (New York, NY, USA, 2021), 415–421.

[18] McDowell, G.L. 2019. *Cracking the coding interview: 189 programming questions and solutions.* CareerCup.

[19] Normalized gain: What is it and when and how should I use it? 2016. *https://www.physport.org/recommendations/Entry.cfm?ID=93334.* Accessed: 2022-08-11.

[20] Pratt, C.C., McGuigan, W.M. and Katzev, A.R. 2000. Measuring Program Outcomes: Using Retrospective Pretest Methodology. *American Journal of Evaluation.* 21, 3 (2000), 341–349. DOI:https://doi.org/10.1177/109821400002100305.

[21] Stepanova, A., Weaver, A., Lahey, J., Alexander, G. and Hammond, T. 2021. Hiring CS Graduates: What We Learned from Employers. *ACM Trans. Comput. Educ.* 22, 1 (Oct. 2021). DOI:https://doi.org/10.1145/3474623.

[22] Urness, T. 2017. Using Interview Questions as Short-Term Programming Assignments in CS2. *J. Comput. Sci. Coll.* 32, 5 (May 2017), 170–177.

[23] Wyrich, M., Graziotin, D. and Wagner, S. 2019. A theory on individual characteristics of successful coding challenge solvers. *PeerJ. Computer science.* 5, (2019), e173. DOI:https://doi.org/10.7717/peerj-cs.173.

[24] Kapoor, A., Panchal, S., & Gardner-McCune, C. (2022). Hire Thy Gator Technical Interview Exercises (Version 1.0.0). *https://github.com/kapooramanpreet/Technical-Interview-Exercises*