Assessment of Introducing Algorithms with Video Lectures and Pseudocode Rhymed to a Melody

*Anonymous*

**ABSTRACT**

We envision using video to introduce various topics in computing to students outside of the classroom.  In this paper, we discuss two video series, each with a sequence of three videos.  The first series presents binary search, and the second series covers selection sort.  Each series begins with an overview of the algorithm with step-by-step examples, proceeds to a video about the associated complexity analysis, and concludes with a video of a song with algorithm pseudo code as lyrics and uses the lyrics as the skeleton of the actual code.  These video series were piloted among a set of introductory courses at two colleges. We assess the effectiveness of each series in terms of conceptual understanding and changes in student attitudes.

**Categories and Subject Descriptors**

K.3.m [**Computers and Education**]: Miscellaneous

**Keywords**

Flipped classroom, videos, music, song, tools for novices.

# INTRODUCTION

Music has been used as a way to help people to remember ideas deemed important enough to compose a song about. From the Iliad and the Odyssey [Mitchell], to the Alan Alda’s recent Flame Challenge to explain science to pre-teens [Ames], song has been exploited as a way to help people, and students in particular, to remember and increase understanding. Song has been demonstrated to support recall for a diverse population such as young, novice students [Yeoh] as well as students with disabilities Claussen]. For example, in computer science education, music has been used extensively to understand complexity in courses ranging from introductory [**Dougherty**] to analysis of algorithms [Knuth, Chavey]. For a more extensive overview about the use of songs in learning, especially with computing, see [**Schreiber**].

SAMPLE: Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*SIGCSE’17*, March 8–11, 2017, Seattle, WA, USA.

Copyright 2017 ACM 1-58113-000-0/00/0010 …$15.00.

DOI: http://dx.doi.org/10.1145/12345.67890

We assume faculty members have full schedules.  Thus, any tool or approach that can be administered outside of class meetings and has the potential to increase learning is useful and desired as described in work such as [Maher].  This project outlines such a set of readily available videos to provide a means for students to engage with two introductory algorithms; specifically, binary search and selection sort.  In each series of three videos, there is a song describing the algorithm presented with captioned lyrics about the algorithm.  These lyrics are then “cut and pasted” directly into a program and used as comments for the associated lines of code they represent.  We expect this approach will be of great value to novice computing students where such a “musical scaffold” will be a balance of accessibility, utility, and a form of entertainment to some degree.

We started to test this expectation with a preliminary investigation in the Fall of 2015, assigning the Binary Search video series in conjunction with two sections taught by two different professors at an undergraduate college [**Schreiber**].  This papers reports on an expanded investigation in the Spring of 2016.

## Environment of this Investigation

The video series were used among the following three distinct computing courses:

1. A CS1 course at *College A* taught by three different instructors in corresponding sections.
2. A new CS1 course that provided emphasis with programming, including imperative and object-based approaches, that serves an alternate entry into the CS major at *College B*, taught by a fourth professor distinct from the above three professors.
3. A CS2 course at *College B* taught by the same professor at *College B* above.

Of the courses described above, #1 and #2 used both video series during the course.  Course #3 only viewed Selection Sort for this report as these students had already viewed the Binary Search video series as part of [**Schreiber**].

Each professor was asked to assign, or at least “strongly encourage,” the students in their course to review a given video series just before the same algorithm was introduced in the course.  We had hoped that students reviewed the material prior to the class where the corresponding algorithm was discussed, but this hope was not always possible.

Before each video series was a request to complete a brief attitude survey (*i.e.,* pre-survey).  Next, each student would complete the three videos in sequence for the given algorithm.  Finally, they would complete a second attitude survey (*i.e.,* post-survey) with some follow up questions on the initial survey, as well as a few conceptual questions, and more questions about their experience and takeaways from watching the videos.

## The Video Series

In this study we examine the effectiveness of two video series, one on binary search and the other on selection sort. Both of the series employ a three-part approach to teaching algorithms: intuition, analysis, and then song and code. The intuition video explains the idea behind the algorithm and how one could perform it using pen and paper. The instructor in the video first covers a high level explanation of the algorithm and then performs it step by step on an interactive blackboard where the result of each iteration are visible to the viewer. Next is the analysis video where the idea of asymptotic analysis is first introduced to the viewer. The instructor does this by showing how an algorithm can be broken down into a number of steps and then boiling this value down by eliminating constants and terms that are dominated as the input size grows. After going through a few simple examples of how to perform an analysis, the viewer sees how to synthesize the topic algorithm into a generic number of steps (in terms of a list of size *n*) and then how to distill that expression by removing constants and insignificant terms. Next after covering the analysis, the instructor performs the “*anonymous*” which is the given algorithm’s pseudo code rhymed with melody and piano accompaniment. During this performance, the lyrics of the song are visualized in a code editor as the instructor sings them. Lastly, the instructor takes each line of pseudo code and shows how it can be translated into equivalent Python code. No prerequisite Python knowledge is assumed, but students can visit the website *ananomous.url* for introductory explanations for Python syntax and concepts.

## Study Methodology

We administered both a pre- and post- survey, each to assess the effectiveness of a given video series used to introduce a particular algorithm. Specifically, students would answer questions that gauged their familiarity with programming, music and the given algorithm itself. Each student provided an (anonymous) ID on each of the surveys that would allow us to pair with their survey responses. Since objectives are stated at the very beginning of each video, students were told that if they felt that they had a strong grasp of the objective of a video they could move onto the next. The post survey consisted of questions that gauged any change in confidence level reported after watching the video series, but also assessed each student’s knowledge by asking conceptual questions about the algorithm and its analysis. We offered a section where students could optionally provide their gender, race/ethnicity, and whether they had a disability because we were also interested in measuring responses across different demographics. Lastly, students had the option to write feedback of any nature about the videos.

# RESULTS

We first describe the makeup of the students in our study as reported by the students themselves. Next, we present results of the students understanding of the conceptual materials obtained from the post-survey. Finally, we try to gauge any change in confidence and attitude reported by the students.

## Demographics

The classes surveyed for this study came from three different courses between two liberal arts colleges. For both algorithms there were five different sections among the three courses. For both algorithms we recorded the distributions for race/ethnicity, course/section, disability and gender. Please note that students were invited to share with an open-ended request (*e.g.,* gender?), and not from a fixed set of choices.

We present the student reported demographics for the study in the following three tables. In addition to these characteristics we also invited students to share any disability. Of the 22 that responded for binary search one student reported ADHD, and of the 28 students that responded for selection sort, again one reported ADHD.

Since not all participants participated in both the pre- and post-surveys, for some analyses we removed pre-responses that did not have corresponding post-responses. The results in Tables 1 – 3 below depict the entire post response samples.

Table . Racial Distribution

|  |  |  |
| --- | --- | --- |
|  | **Binary Search** | **Selection Sort** |
| Hispanic | 2 | 2 |
| Asian | 6 | 11 |
| Black | 3 | 3 |
| White | 14 | 22 |
| Mixed | 1 | 2 |
| North African | 0 | 1 |
| No Response | 9 | 12 |
| **Totals** | **35** | **53** |

Table . Gender Distribution

|  |  |  |
| --- | --- | --- |
|  | **Binary Search** | **Selection Sort** |
| Female | 10 | 19 |
| Male | 18 | 25 |
| No Response | 7 | 9 |
| **Totals** | **35** | **53** |

Table . Course and Section Distribution

|  |  |  |
| --- | --- | --- |
|  | **Binary Search** | **Selection Sort** |
| CS1-A.1 | 0 | 11 |
| CS1-A.2 | 17 | 16 |
| CS1-A.3 | 12 | 1 |
| CS1-B | 6 | 15 |
| CS2 | 0 | 10 |
| **Totals** | **35** | **53** |

## Technical Knowledge

Our primary technical goals for what the students would take away from the videos were 1) being able to identify the time complexity of the algorithm, 2) answer questions that require simulation of the algorithm, and 3) remember fundamental facts about the algorithm. With this in mind, we evaluated students’ baseline knowledge of the algorithm in the pre survey by asking if the students were familiar with big-O notation and if they were confident coding a search or a sorting algorithm. After watching the series of videos in order we then asked one question per each of our goals in a multiple choice format where 1 of 5 possible selections was correct.

We now present a set of tables, each providing the student responses for the conceptual questions asked on the post-survey.

Below we can see that students were relatively unfamiliar with big-O notation with 7% knowing for binary search and 15% knowing for selection sort. However, after watching the videos students showed significant improvement in their knowledge of the algorithms: 85.7% of students were able to identify the runtime of binary search and 84.9% students were able to identify the runtime of selection sort compared to 14.9% initially. Additionally we can also see that the other questions were all answered correctly at least 80% of the time.

Table 4. What is the time complexity of binary search?

|  |  |
| --- | --- |
| O(2) | 0 |
| O(log n) | 30 |
| O(2n) | 0 |
| O(n2) | 3 |
| O(2n!) | 0 |
| I don’t know | 2 |

Table 5. In binary search, if the value at the mid position is less than the key (e.g., say your key was 9 and the value at the middle position is 5) …

|  |  |
| --- | --- |
| You've found the key so return the mid index | 0 |
| This must mean that the key does not exist in the list. | 1 |
| Set low to mid+1 and let the high position be. | 29 |
| Set high to mid-1 and let the low position be. | 5 |
| This cannot possibly happen | 0 |

Table 6. Which statement is true about the binary search algorithm?

|  |  |
| --- | --- |
| It is better than linear search in all cases | 4 |
| It cuts the list in half until it finds the key's place. | 28 |
| The list does not need to be sorted for it to work properly. | 1 |
| When the low index passes the high index, it has found the right key. | 2 |
| It is used to sort a list of values. | 0 |

Table 7. What is the time complexity of selection sort?

|  |  |
| --- | --- |
| O(1) | 3 |
| O(log n) | 3 |
| O(n) | 2 |
| O(n2) | 45 |
| O(n log n) | 0 |
| I don’t know | 2 |

Table 8. What guarantee could be made of the list to be sorted after the initial iteration of the outer loop of the selection sort algorithm?

|  |  |
| --- | --- |
| The list would be sorted | 2 |
| The largest element is at the end of the list | 0 |
| The smallest element would be at the start of the list | 43 |
| Either the largest will be at the end of the smallest will be at the start | 0 |
| The largest will be at the end and the smallest will be at the start | 3 |
| You can’t guarantee anything after the initial pass | 3 |
| I don’t know | 2 |
| Other | 0 |

Table 9. Which statement is true about the selection sort algorithm?

|  |  |
| --- | --- |
| The algorithm stops when the list is sorted | 1 |
| With each pass of the outer loop the number of inner loop iterations increases | 6 |
| Duplicate values can cause selection sort to return an unsorted list | 0 |
| With each pass of the outer loop the number of inner loop iterations decreases | 43 |
| I don’t know | 1 |
| Other | 2 |

## Self-Gauged Competency

Since the majority of students who would watch these videos are beginners in computer science, it was a major goal of ours to measure students’ confidence with concepts such as complexity and big-O and as well as to reinforce some underlying principles involved with algorithm design (*e.g.,* input parameters and preconditions, simple typing, control structures).  The video series was presented approximately midway in each course, but was an introduction to the ideas of algorithmic searching and sorting for many of the students. When asked to grade their confidence in being able to code a sort the average response was 2.41 out of 5 and for coding a search the average response 1.77 out of 5. Likewise, when asked about big-O, students were also not very familiar with the concept with an average response of2.14 for selection sort and 1.54 for binary search**.** These values were understandably low, but we wanted to see how much students’ mentalities improved as a result of following the video series. In order to track their performance students would provide an anonymous ID (last four digits of phone number + last three digits of zip), which kept the responses anonymous but allowed us to pair each person’s pre responses with their post response. Note that if a student didn’t provide their ID in both their pre and post surveys we did not consider their responses in the calculation of average improvement. Below are the results of the study.

We can see that after watching the binary search video series the average response difference was +1.58 for being able to code the algorithm and +1.67 for confidence with big-O, and after watching the selection sort series the average response difference was +1.33 for being able to code the algorithm and +1.31 for confidence with big-O. To verify whether these results were statistically significant we conducted paired *t*-tests for each question, where the results are shown below (will format these into tables)

Table 10. Student-Reported Confidence Increases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ***df*** | ***p*** | **Mean increase** | **Percent increase** |
| Selection Sort coding | 50 | 2.30x10-12 | 1.33 | 55.3% |
| Selection Sort complexity | 50 | 3.18x10-8 | 1.31 | 61.4% |
| Binary Search coding | 30 | 5.89x10-12 | 1.58 | 89.6% |
| Binary Search complexity | 30 | 2.41x10-11 | 1.68 | 109.7% |

For each question we can see that there is a significantly positive difference between pre and post responses for the entire sample. Moreover, since we were interested in how people from different demographics responded, we investigated if there was a difference in how much confidence gain there was between male and female students. To verify these results, we conducted a t-test for difference in means between two samples (male and female students) and did not consider anyone who was missing an ID or did not provide their gender on the survey. Below we see the mean difference in confidence per question per gender. For each question the mean confidence increase for women was higher, but none of the questions yielded a large enough spread between men and women for us to conclude there is a statistically significant difference in their confidence boosts.

## Critique of Videos

Students evaluated the videos using a Likert-type scale, where students answer on a discrete 1-5 scale where 1 means strongly disagree, 3 means neutral and 5 means strongly agree. Our main objectives in this section were to gain insight on the students’ experiences watching the videos, their attitudes after watching the videos, their responses to the music, and how they planned to utilize the videos. In order to ensure that students were thoroughly reading the questions before answering we added two questions with negative sentiments, in which “strongly agree” meant that they did not like a certain aspect of the videos. In addition to supplying the mean and standard deviation of each response, we also include p values for tests against the null hypothesis that the mean response was neutral.  Below are the results for both algorithms per question.

**[JD to insert table here once confirmed]**

We can see that in both series students felt that the flow of the videos was logical and natural, the lyrics of the songs translated well to pseudo code, the videos overall made the algorithms less intimidating, and students felt that the videos would be a good study resource for exams (all have p values below ???). These questions had responses that were significantly more positive than neutral for both algorithms, but even if questions do render a close or even less than average response there can still be merit to the information they offer. For instance, when asked if they wanted more videos to help them learn other algorithms and concepts, selection sort had an average response rate of 3.30 and binary search had an average of 3.14, but for both algorithms over 41% of students indicated a score of 4 or 5, and under 21% of students indicated a score of 2 or 1. Although we did not expect them to memorize it after the first listen we still asked students if the song would help them memorize the algorithm and over 20% reported that they would for both algorithms.

**2.5 Free Responses**

# CONCLUDING DISCUSSION

Educators at all levels are under intense, often increasing, pressure to cover a complete curriculum inside of a limited number of lectures. This tension makes it difficult to develop interactive and meaningful learning opportunities as opposed to lecture-only delivery of material. There is just too much information to cover in too little time.

Another problem that we in computer scienceare facing today is diversity throughout all levels of study. We want to provide equity of access to the material. Music, we believe, provides an almost universal means to reach students and engage them. This project uses music to promote learning about basic algorithms.

We propose a video series used to teach algorithms that incorporates the emotional and mnemonic power of music, and employs audio-visual demonstrations to teach concepts and guide students through such topics as sorting, searching, and complexity analysis.

The study covered in this paper, which was held at two colleges across three different courses with five different instructors, has provided evidence to that students gained a strong conceptual grasp of the material which they had little to no understanding before by watching these video series. Moreover, when testing students’ attitudes of the material, results show that their baseline level of confidence increased significantly. This increase in confidence was seen in both males and females, but was actually higher for female students.

The bottom line is that there is nothing to lose from using these video series and so much to gain. Having 80% of students walking into a classroom understanding concepts such as big-O, searching and sorting to some degree prior to the meeting can open up class time for teachers. This will enable them to include creative learning activities such as in-class problem solving time and peer instruction. These video series are freely and readily available, allow students to work at their own pace, are accessible to anyone with Internet access, and even incorporate the power of music to illustrate code. Thus, we encourage other educators to explore these videos and to offer the series to their students, especially novice students, as a fun, effective way to begin to develop their appreciation and understanding of computer science.

# ACKNOWLEDGMENTS

Our thanks to the professors who piloted the videos series in their respective courses in the Spring of 2016.

# REFERENCES

1. Ames, B.J. 2012. What is a flame? The Flame Challenge: Explaining Science to an 11 Year Old. *Alan Alda Center for Communicating Science, Stony Brook University*. Accessed 20 August 2016, <http://www.centerforcommunicatingscience.org/what-is-a-flame/>.
2. Bransford, J.D., Brown, A.L. and Cocking, R.R., 1999. *How people learn: Brain, mind, experience, and school*. National Academy Press.
3. Chavey, D. 1996. Songs and the analysis of algorithms. In *Proceedings of the Twenty-seventh SIGCSE Technical Symposium on Computer Science Education*, Karl J. Klee (Ed.). ACM, New York, NY, USA, 4 – 8.
4. Claussen, D. W., and M. H. Thaut. 1997. “Music as a mnemonic device for children with learning disabilities.” *Canadian Journal of Music Therapy* 5.1: 55 – 66.
5. **Dougherty, J. P. 2008. “Using lyrics and music to reinforce concepts.” *Journal of Computing Sciences in Colleges*, 23, 3 (Jan. 2008), 106 – 113.**
6. Knuth, D. 1984. The complexity of songs. *Commun. ACM* 27(4), 344 – 346.
7. Maher, M.L., Latulipe, C., Lipford, H., and Rorrer, A. 2015. “Flipped Classroom Strategies for CS Education.” *In Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 218 – 223.
8. **Schreiber, B., and Dougherty, J. P. 2016. “Embedding Algorithm Pseudocode in Lyrics to Facilitate Recall and Promote Learning.” Accepted, *Consortium for Computing Sciences in Colleges Eastern Conference* (Frostburg State University, Frostburg, MD, USA, October 28 – 29).**
9. Yeoh, M.P. 2013. “Musical mnemonics to facilitate learning of matriculation biology: The Calvin Cycle.” *Molecules* 3: 3C.
10. Zull, J.E., 2004. The art of changing the brain. *Educational Leadership*, *62*(1), pp.68-72.

Columns on Last Page Should Be Made As Close As Possible to Equal Length