TEACHING AND LEARNING ALGORITHMS WITH APPS

O. Herden

Baden Wuerttemberg Cooperative State University (GERMANY)

Abstract

It is very important that undergraduate students get deep knowledge in the topics algorithms and data structures because these are central terms in computer science. So there is a course “Algorithms and Data Structures” in each undergraduate computer science curriculum.

With the widespread distribution of smart phones and tablets there is a new option of learning and teaching algorithms in the form of mobile apps. This way should be understood as supplement and not as a replacement of existing lectures. Due to different approaches of using apps and a diversity of existing apps it is important to have a criteria catalogue. This catalogue serves as basis for decision-making.

In this contribution we first give an overview about different ways of teaching algorithms. Then we sketch different scenarios of using animated algorithm apps. Herefrom we derive a criteria catalogue for this kind of apps. Subsequently the final set of criteria is applied to some apps exemplarily. The paper concludes with a summary and an outlook to future tasks.

Keywords: Higher Education, Undergraduate, Computer Science, Algorithm, Apps, Criteria Catalogue.

1 INTRODUCTION

Algorithms and data structures are central terms in computer science. Hence it is necessary that undergraduate students in computer science get deep knowledge in these issues. Moreover, it is very important not only to learn a set of algorithms but also to understand important design principles and properties of algorithms. In practice this is realized by a course “Algorithms and Data Structures” in each computer science curriculum.

Over the years, the way of teaching algorithms has been changed due to technological conditions. The latest innovation are apps for mobile devices for teaching and learning algorithms. These apps can be used in different manner und existing apps differ extremely in their basic approach and their realization. Therefore, it is important to have a certain set of criteria as basis for decision-making which app can be used for which purposes.

The reminder of the paper is organized as follows: The following section sketches the classical ways of teaching algorithms, section 3 shows different use cases of integrating apps into lectures. Section 4 describes the criteria catalogue. Application examples are shown in section 5. The contribution closes with a summary and an outlook.

2 ALGORITHMS AND THE WAY OF TEACHING THEM

During the first or second semester a fixed set of algorithms, data structures as well as design principles and properties has to be learned by undergraduate students of computer science [3, 8, 13, 14]. Figure 1 summarizes these aspects.
To master this task different didactical approaches were applied in the past, all approaches were influenced by the available technology [6]. The oldest method is the classical lecture with chalk and sponge at the blackboard. The step-by-step execution of algorithms is realized by writing and wiping at the board. The next step were overhead projectors were tokens and knobs can be used to demonstrate the dynamic aspects of algorithms. Today, document cameras are a popular substitution of overhead projectors. With the distribution of presentation tools like Power Point in the 1990ies this task can be realized by animations within these tools. With the emerging web technology the next step were small simulations or animations of algorithms embedded into websites, see e.g. [5].

The next technological development step is the massive distribution of smart devices, figure 2 shows the number of smartphone users in Germany on the left hand side and on the right hand side a prediction of worldwide smart phone users. We can see that the number of users is increasing steadily and we can assume that every student has its own device. So we can follow a BYOD (Bring your own device) strategy [2] for integrating mobile devices into lectures.

**Data Structures:**
Array, List, Stack, Queue, Priority Queue, (Disjoint) Sets, Hash Map/Table, Tree, Graph

**Design Principles:**
Iteration, Recursion, Backtracking, Greedy, Divide and Conquer, Dynamic Programming, Brute Force

**Properties:**
Determinacy, Determinism, Termination, Effectivity, Efficiency, Time/Space Complexity

**Sorting:**
- Bubble Sort
- Selection Sort
- Insertion Sort
- Shell Sort
- Merge Sort
- Heap Sort
- Quick Sort
- Counting Sort
- Bucket Sort
- Radix Sort
- External Sorting

**Searching:**
- Binary Search
- Interpolation Search
- Trees
- Hashing

**Graphs:**
- Traversal
- Topological Sort
- Shortest Path
- Minimal Spanning Tree
- Colouring

**String Processing:**
- Matching
- Retrieving

**Figure 1. Important algorithms and design principles. [7]**

**Figure 2. Distribution of smart devices. [15]**

8751
As a consequence, it is important to analyze in which way smart devices can be integrated into lectures. The following section addresses this topic. In so doing, we do not want to substitute the existing lecture material but use both technologies in parallel.

3 USE CASES IN LECTURES

This section describes possible use cases of integrating smart devices into a lecture. We assume a "typical" lecture in computer science: it is based on power point presentations and each single lecture is supplemented by an exercise. These exercises can be pen and paper based as well as practical programming tasks.

The classical situation is depicted in figure 3. There is no preparation by the students. The lecturer holds his talk with the help of his slides, he has the active role all over the time, the students become active not before the exercises.

**Figure 3. Scenario classical lecture.**

Use cases 1 and 2 (see figure 4) make use of the smart device during the lecture. While in the first approach the lecturer uses the mobile device as replacement for animations in the slides, in the second one the lecturer interrupts his talk and the students use the smart device for exploring a new algorithm on their own. Afterwards the algorithm and its properties can be discussed in the plenum. So, the difference of these use cases is the activity degree of the students.

**Figure 4. Use in the classroom.**
Using the smart device for post processing is described in use cases 3 and 4 being depicted in figure 5. The difference is the point of time. While use case 3 addresses the use in exercises, use case 4 refers to the learning for exams.

**Use Case 3: Post processing, students use app in exercises**

![Use Case 3 Diagram]

**Use Case 4: Post processing, students use app for exam preparation**

![Use Case 4 Diagram]

Figure 5. Use for post processing.

Figure 6 shows the use cases 5 and 6 where the smart device is used for preparation. While use case 5 is characterized by preparing single algorithms, use case 6 describes the flipped classroom scenario.

**Use Case 5: Preparation, students use app for single algorithms**

![Use Case 5 Diagram]

**Use Case 6: Preparation, flipped classroom**

![Use Case 6 Diagram]

Figure 6. Use for preparation.
Different use cases have different preparation effort for the lecturer. The use cases 1 and 2 have low effort because only appropriate apps have to be selected. Use cases 3 to 5 need higher preparation time because special exercises for using the apps must be designed. The highest effort has use case 6 because the complete course material has to be redesigned.

As we can see, the use cases differ with respect to the point in time and place of use, students’ activity and preparation effort for the lecturer. Figure 7 summarizes these facts in an overview.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Time</th>
<th>Classroom</th>
<th>Students’ Activity</th>
<th>Preparation effort</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Before</td>
<td>During</td>
<td>After</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Low</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Middle</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Middle</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Middle</td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>High</td>
</tr>
</tbody>
</table>

*Figure 7. Use cases and their properties.*

At this point it is important to state that the use cases are not mutual exclusive, e.g. smart devices can be used during the lecture as well as during exercises.

The different use cases and different frequency and intensity of use lead to different requirements of apps concerning algorithms and their animations. For distinguishing them and to get a good choice for the right system we need certain criteria for describing and selecting this kind of apps. So a criteria catalogue will be presented in the following section.

4 CRITERIA CATALOGUE

This section describes the criteria catalogue for algorithmic learning apps. First we give an overview, followed by handling the two areas usability of animations and general learning app issues in detail.

4.1 Overview

The catalogue is divided into six sections sketched in figure 8.

*Figure 8. Areas of criteria catalogue.*

Basic data concern the fundamental information about an app like name, platforms, name of vendor, etc. The area content contains a list of the algorithms and data structures handled in the app. Technical
issues comprise criteria of apps independent from the learning environment, e.g. questions about the storage model or security issues [4, 10, 12]. The area general usability deals with usability of apps in general, concerning topics like colours, zooming, scrolling and ergonomic use [1, 11].

The two last areas consider issues relevant for learning apps. One considers usability with respect to algorithms and data structure, esp. with the animation and stepwise execution [10]. The other one handles learning app features in general, not specific to the use case of learning algorithms. Examples are the possibility of review questions or of tracking the learning progress.

Since the two latter areas are the most important ones they will be described in detail below.

4.2 Algorithm Animation

The execution of the algorithms is visualized and animated. Figure 9 is a schematic presentation of the desirable functionalities.

The animation can be executed step by step. This can be realized by a back and next button, with the buttons first and last the user can jump to the end resp. initial situation. The animation can be played automatically, the speed can be regulated by a slider. The automatic play can be paused or stopped by the user. Additionally to the animation the pseudocode of the algorithm and a textual explanation of each step can be displayed. Due to the fact of limited screen size these two features should be showed and hidden by user control. In the case of displaying pseudocode the actual code line should be highlighted.

Typically, the apps have a few predefined examples for each algorithm. Here we can distinguish the number and the variety of the examples. Moreover, the size of the algorithms' input can be differentiated. The built-in examples can be fixed or parametrizable. Another feature is user defined input offering the possibility to define own examples.

The app can be completed by explanations of the algorithms with a text or video, a glossary and search function.

4.3 Learning App Features

The apps can be distinguished by some typical learning app features. These features can be grouped by lecturer activity, reviews, individualization and communication. Lecturer activity means the functionality of controlling the use of the app by the lecturer, i.e. issues like configuration and approval of learning units or inspection of passed units. Reviews offer the learners the possibility to check their understanding of the learned and the learning progress. Individualization sketches topics like configuring individual learning paths or selecting between examples of different degree of difficulty. The area communication covers the integration of the connection of the users. This communication can either be
between student and lecturer or in their peer group between students. Orthogonally, we can differentiate whether the communication is realized within the app or by connecting to an existing social network.

5 EXAMPLES

For working with the criteria list we have developed the tool ToCALA (Tool for Comparing Apps for Learning Algorithms). Initially, ToCALA offers the criteria list from the former section. The user can customize the criteria list by selecting existing criteria, optionally add additional criteria and change the order of criteria within their area. After configuring the report by defining the information about the apps (textual, yes/no, numeric) and the number of compared apps, ToCALA generates a page for filling in the information. When the user has finished this task, the tool checks the completeness and correctness of the data and a pdf file as output can be generated.

We have evaluated some apps from the Google Play Store, Algorithms: Explained and Animated (by Moriteru Ishida), AlgoPrep (by Pranit Krishna Kulkarni), Algorithm Visualizer (by Naman Dwidedi) and Algorithm Learn Easy (by Drunk Developer). The screenshots in figure 10 give an impression of the use interfaces.

We can state that the evaluated apps are based on different philosophies and each has its own pros and cons. None of the apps covers the desired content (all algorithms to be taught in the course) completely. Moreover, we made the observation that different algorithms (with different properties or principles) can not all be animated with one app in the best way. The integration into the classroom (use cases 1 and 2 from section 3) is definitely an added value in teaching algorithms. Also students found the apps helpful for preparation of the written exam.

6 SUMMARY AND OUTLOOK

This contribution has handled the use of mobile apps for teaching and learning algorithms. After an introduction and motivation we have sketched different use cases about the way of using apps. As a core result we have developed a criteria catalogue to compare different apps and to have an assistant for deciding which app to use. For applying the criteria catalogue in a useful manner we have implemented the tool ToCALA. In ToCALA the criteria catalogue is predefined, can be adopted and different apps can be rated.

As our work’s main results we can summarize so far:

- Mobile devices become more and more widespread and they can be used following the BYOD approach
- There are several meaningful use cases where classical lectures based on power point presentations can be enriched by using mobile apps
- Because of the different use cases and the great offer of mobile apps there is a need for tool as basis for solid decision making which app to use. Therefore we have designed a detailed criteria catalogue

Figure 10. Example apps.
• Due to different use cases and different individual needs and/or constraints it is necessary that the criteria catalogue can be adopted and individualized. Our tool ToCALA fulfills this requirement
• The evaluation of some existing apps led to the result that each of them has different advantages and disadvantages and a use of more than one app seems to be necessary

As future tasks we have identified among others:
• The tool ToCALA is a prototype at the moment and some requirements for more intuitive use should be identified and implemented
• The comparison of different apps was a little bit arbitrary so far, we want to continue this work by producing a systematic overview
• Besides learning the algorithms it is also an important skill to be learned by computer science students to get practice, i.e. to implement algorithms. For this reason, it can be a future task to design and realize an own app

REFERENCES