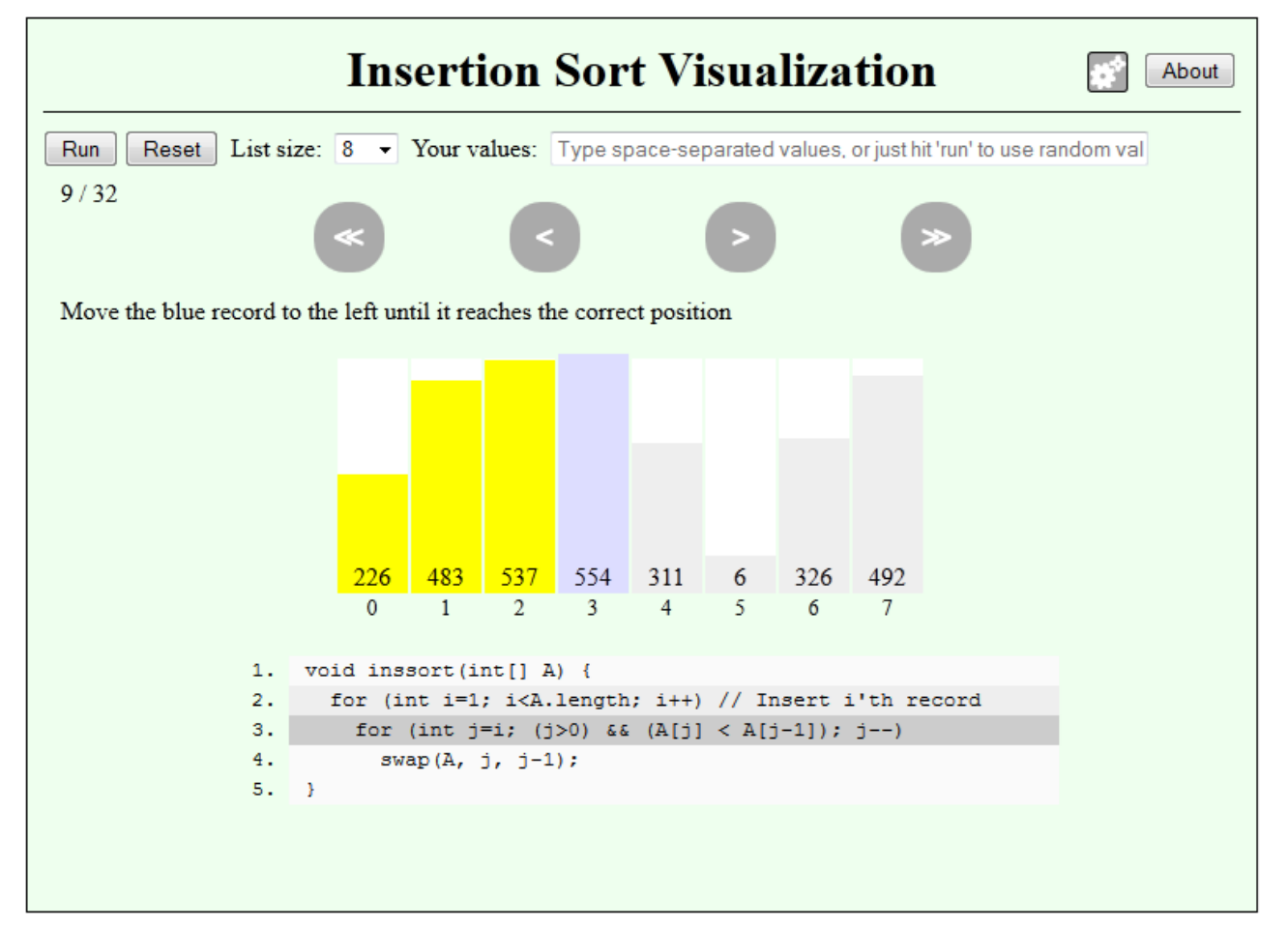
Initial Evaluation for OpenDSA: Interactive Tutorials for Data Structures and Algorithms

Clifford A. Shaffer[1], Eric Fouh[1], Simin Hall[2], Daniel Breakiron[2], Mai Elshehaly[2], and Ville Karavirta[3]
[1]Department of Computer Science, Virginia Tech, Blacksburg, VA, USA
[2]Department of Mechanical Engineering, Virginia Tech, Blacksburg, VA, USA
[3]Department of Computer Science and Engineering, Aalto University, Helsinki, Finland
{shaffer|efouh|simin.hall|breakid}@vt.edu, ville.karavirta@aalto.fi

OpenDSA

OpenDSA is an open source, online collection of interactive tutorials combining textbook-quality content with algorithm visualizations and interactive exercises. An OpenDSA module corresponds to one section in a textbook or part of a class lecture. Each has these components:

- **Text and images** for the exposition.
- **Presentation of dynamic process** (algorithms) through "slideshows".
- **Proficiency exercises** where students demonstrate proficiency by showing algorithm steps
- **Other interactive exercises**
  - Multiple Choice, T/F, short answer
  - Data structure manipulation exercises
  - Active equations and calculators


Insertion Sort Visualization

About
Run
Reset
List size: 8
Your values: Type space-separated values, or just hit 'run' to use random val
9 / 32
226
483
537
554
311
6
326
492
0
1
2
3
4
5
6
7

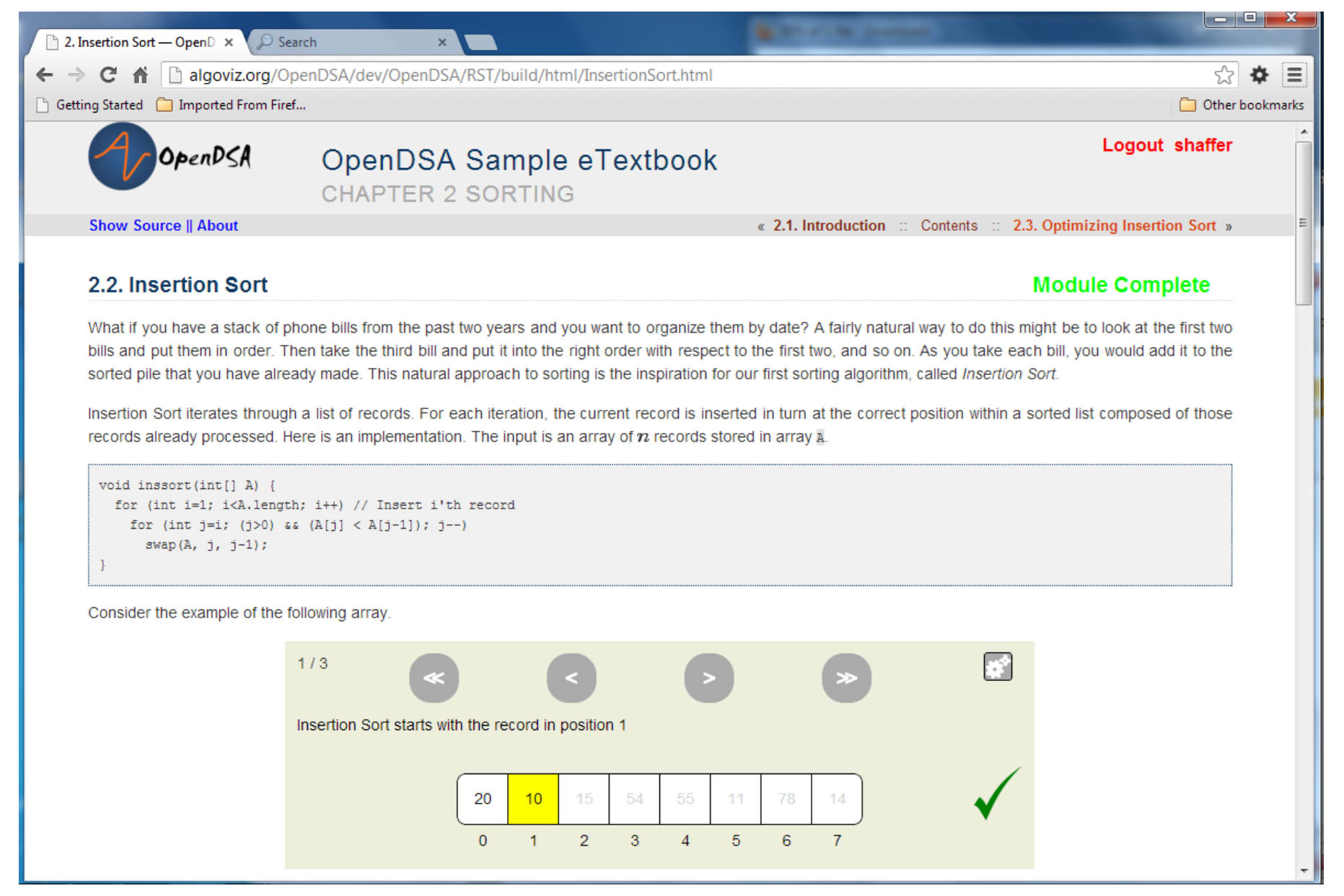
Move the blue record to the left until it reaches the correct position

```
void inssort(int[] A) {
  for (int i=1; i<A.length; i++) // Insert i'th record
    for (int j=i; (j>0) && (A[j] < A[j-1]); j--)
      swap(A, j, j-1);
}
```

Study Objectives

We present a preliminary study to evaluate the effectiveness of OpenDSA. Study questions:

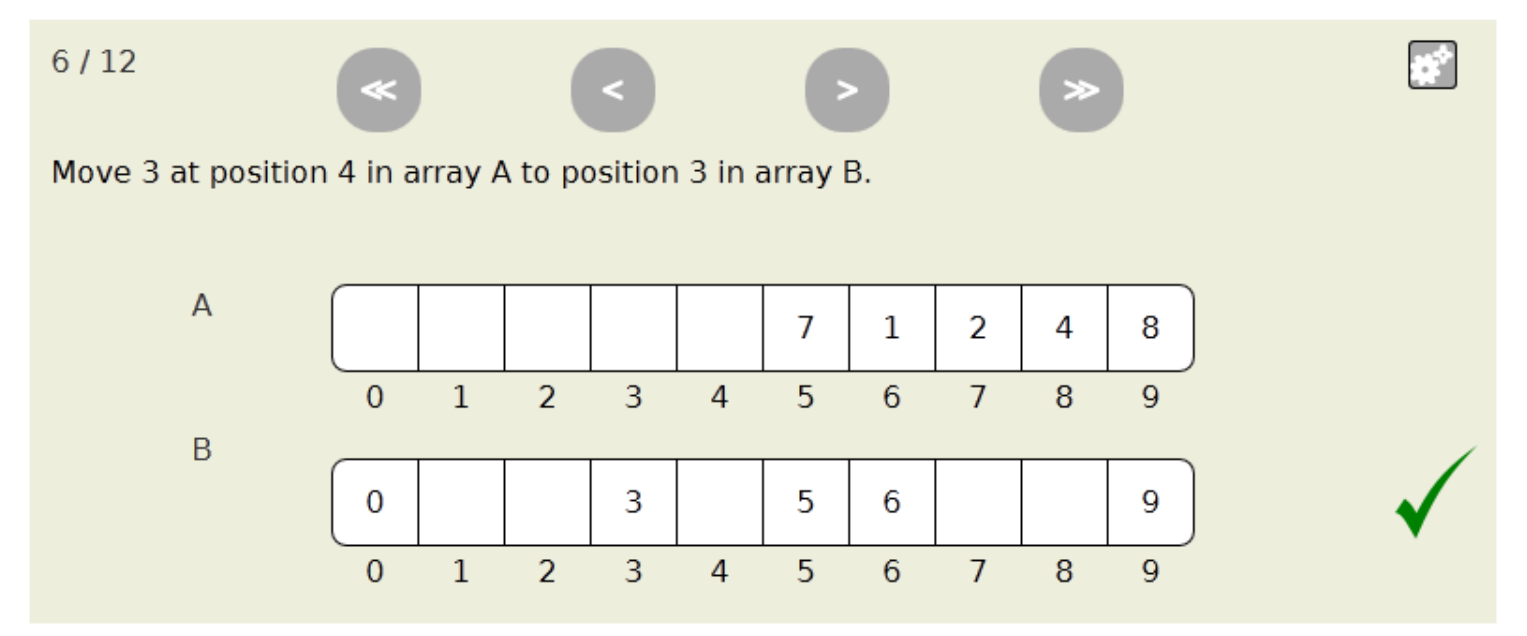
- Can students learn as well or better with interactive tutorials compared to traditional lecture and textbook?
- Will students accept a class based on interactive tutorials rather than traditional lecture and textbook?
- Will our client/server infrastructure adequately support classroom use?
- Gather feedback from students about using interactive tutorials in courses



Study Methodology

Quasi-experimental design with control and treatment course sections:

- Control group received standard lecture and textbook for three weeks
- Treatment section used OpenDSA to work through the content
- Treatment section sometimes received lecture or group discussion
- OpenDSA activities and exercises constituted a "homework" grade worth 5% of the total class score
- Same test was administered to both sections after intervention



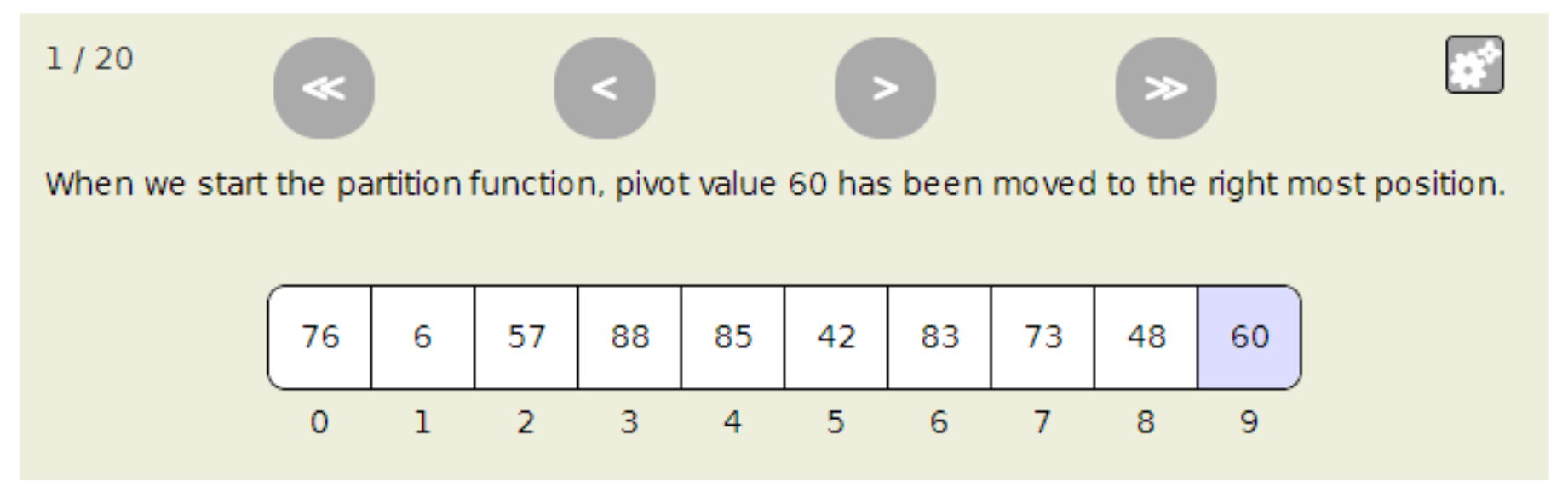
Move 3 at position 4 in array A to position 3 in array B.

Population and Data Collection:

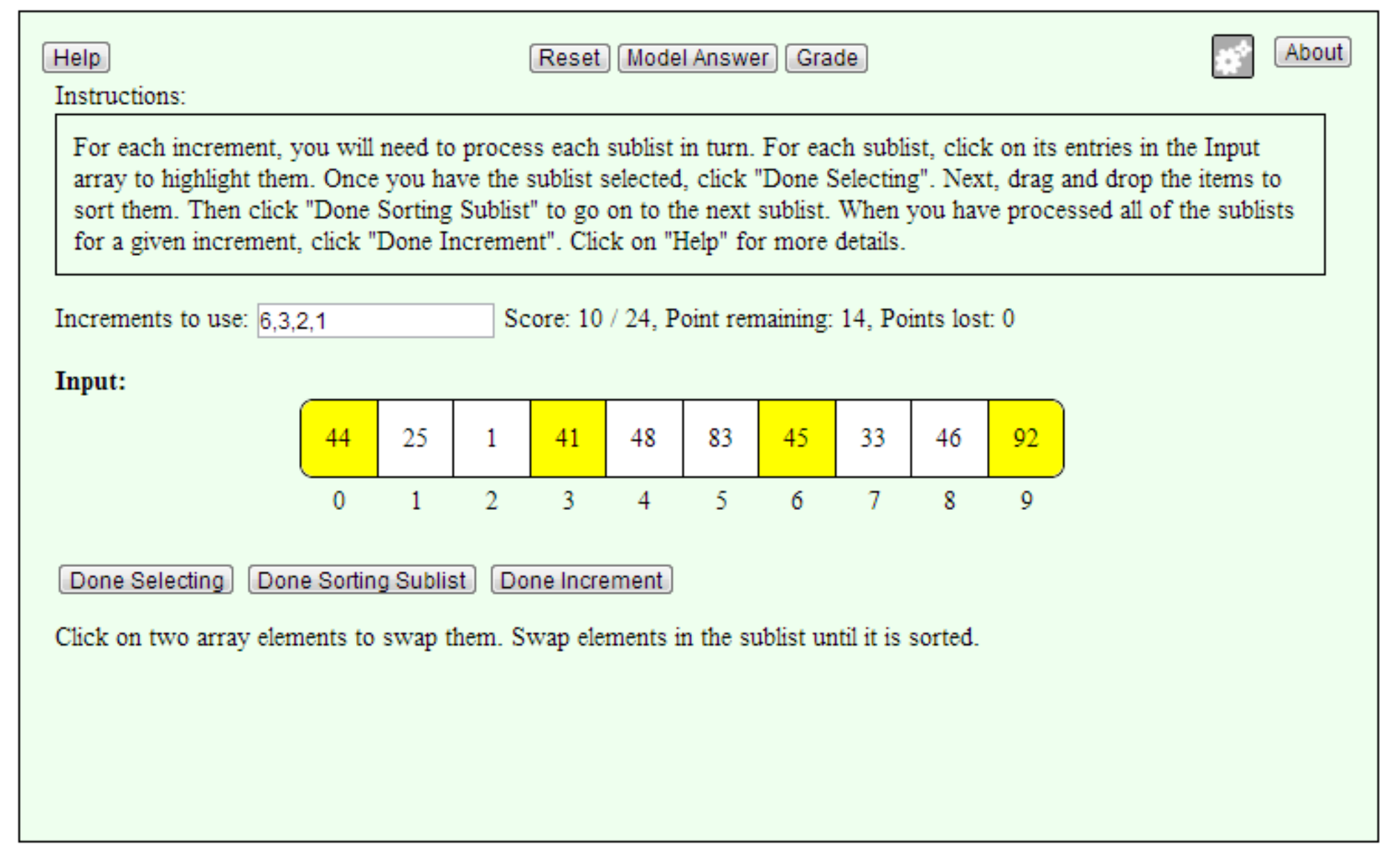
- Undergraduates students: 55 in control group and 57 in treatment group
- Pre-treatment surveys, identical for both groups measured:
  - Experience with online tools
  - Perceptions of face-to-face course vs online instruction
  - Use of technology or e-textbook in class
  - Preference for lecture type or lab setting
- Different post treatment surveys to each group
- Observation of the treatment group
- Collected extensive interaction logs
- Interviewed three students from treatment group

Study Results

- No significant difference on test scores
- Almost all students had prior experience with online courseware
- Treatment students started with a positive attitude about online courseware; after treatment their opinion of OpenDSA was higher than their initial attitude toward generic online tutorials
- Students preferred having lecture during class and homework using the OpenDSA modules over working modules in class
- Students ranked OpenDSA first for learning gains over lecture, projects, course notes, and textbook
- Students support concept of daily OpenDSA homework assignments



When we start the partition function, pivot value 60 has been moved to the right most position.

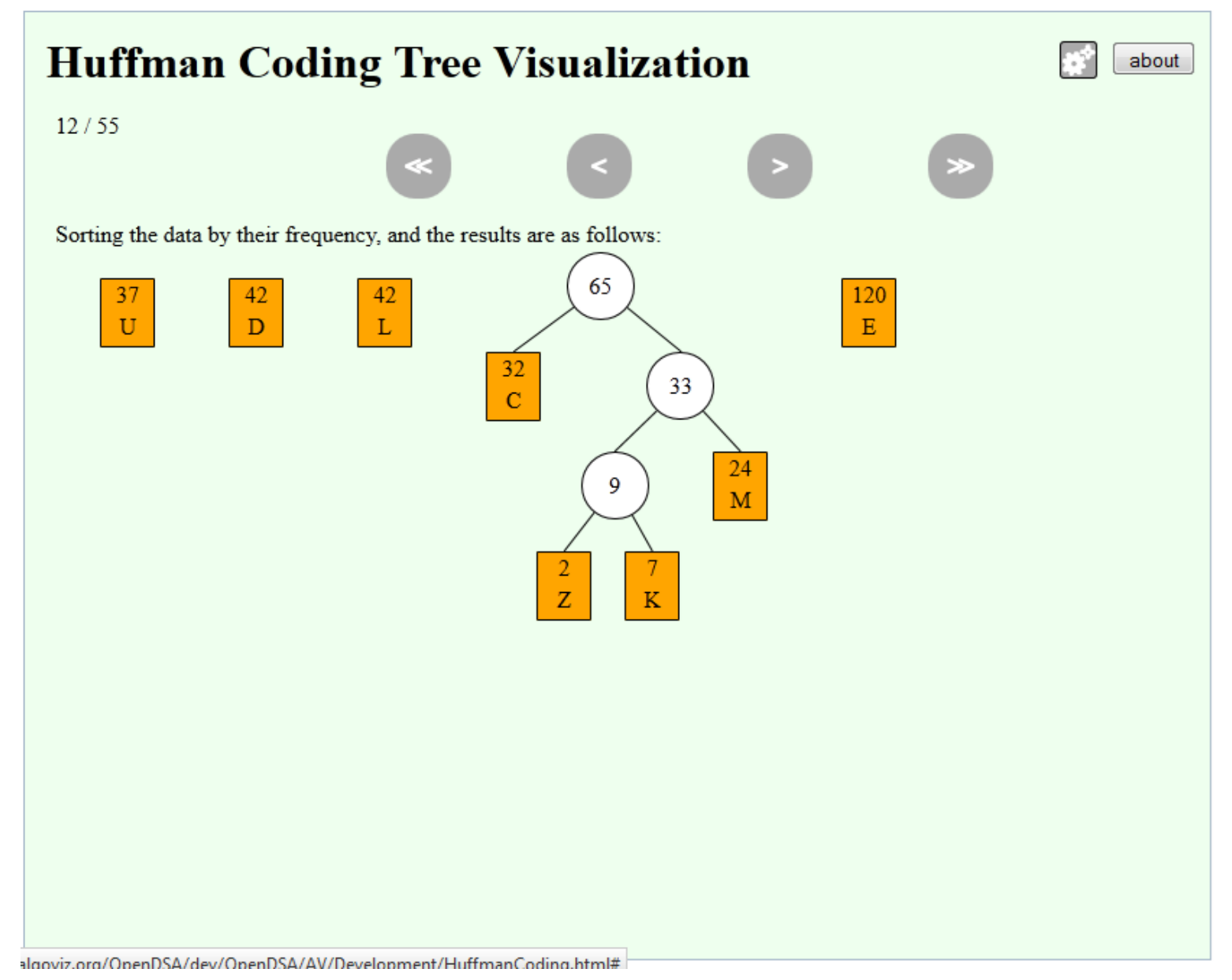


Instructions:

For each increment, you will need to process each sublist in turn. For each sublist, click on its entries in the Input array to highlight them. Once you have the sublist selected, click "Done Selecting". Next, drag and drop the items to sort them. Then click "Done Sorting Sublist" to go on to the next sublist. When you have processed all of the sublists for a given increment, click "Done Increment". Click on "Help" for more details.

Increments to use: 6,3,2,1 Score: 10 / 24, Point remaining: 14, Points lost: 0

Click on two array elements to swap them. Swap elements in the sublist until it is sorted.

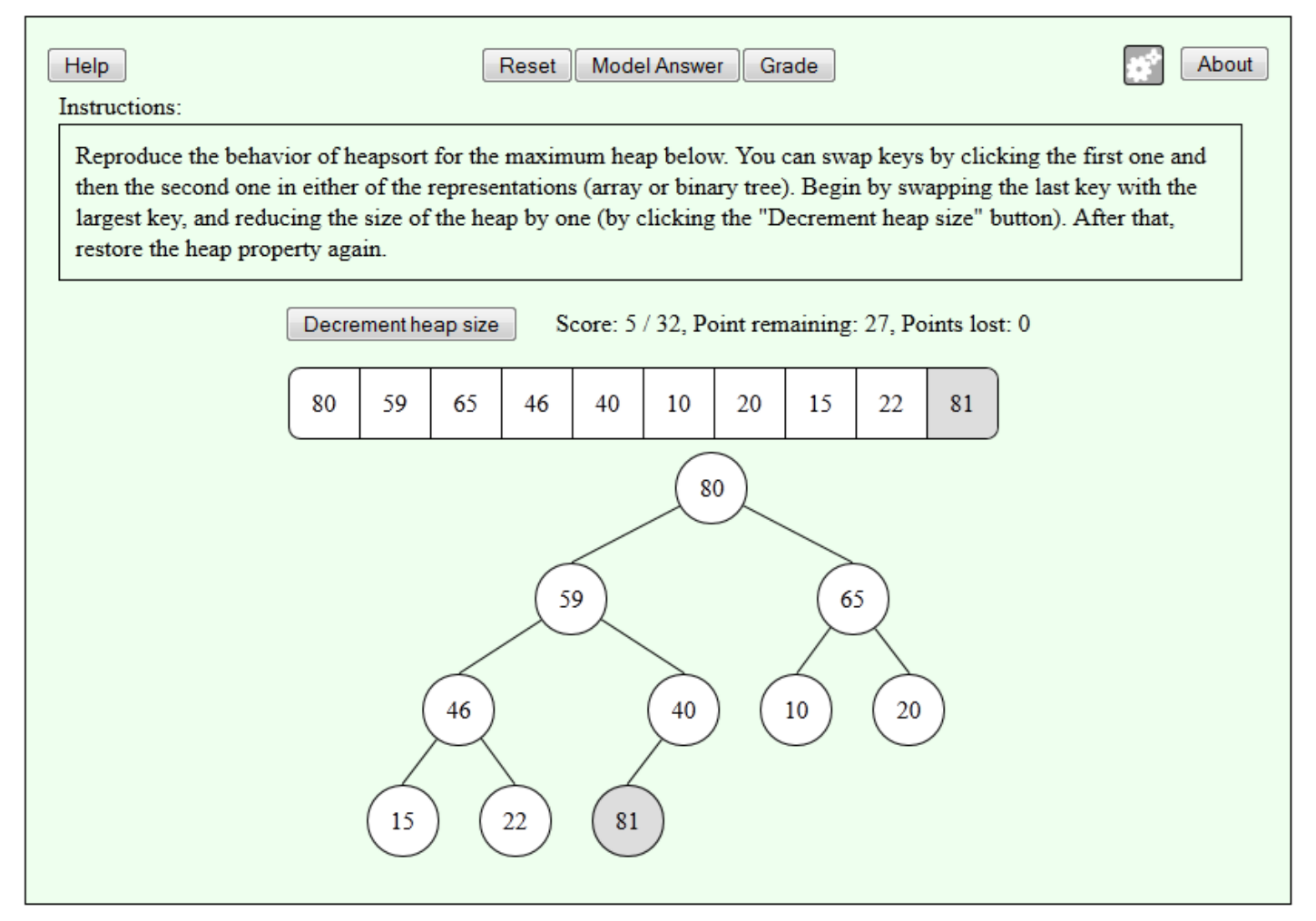

Huffman Coding Tree Visualization

Sorting the data by their frequency, and the results are as follows.

OpenDSA Infrastructure

JavaScript AV (JSAV) Library: JSAV provides development tools for interactive AVs and other dynamic components of the system using JavaScript/HTML5. JSAV features:

- Dynamic slideshows
- Layout of standard data structures and animation elements
- "Proficiency exercises" where students simulate the steps of an algorithm
- Pseudocode display
- Flexibility: existing functionality can be overriden



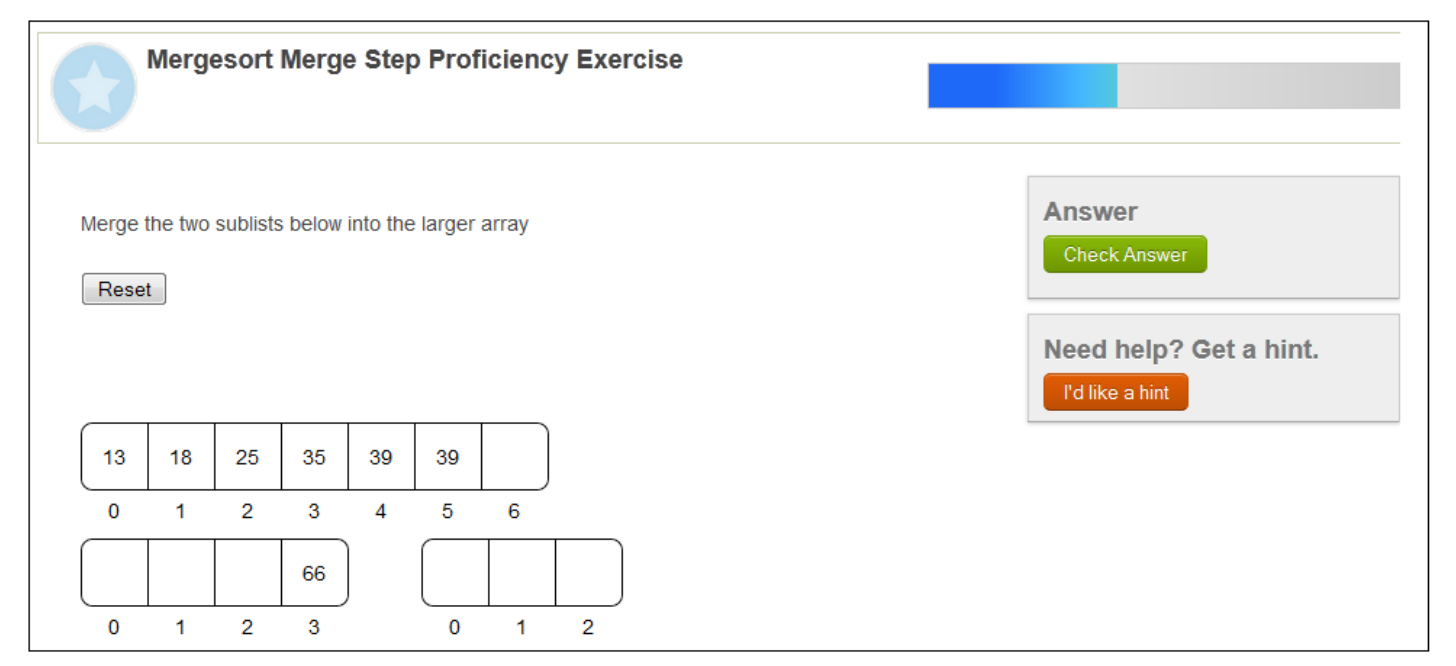
Instructions:

Reproduce the behavior of heapsort for the maximum heap below. You can swap keys by clicking the first one and then the second one in either of the representations (array or binary tree). Begin by swapping the last key with the largest key, and reducing the size of the heap by one (by clicking the "Decrement heap size" button). After that, restore the heap property again.

Score: 5 / 32, Point remaining: 27, Points lost: 0

Frontend:

- ReStucturedtext and Sphinx for authoring content
- HTML5, CSS and JavaScript for dynamic and interactive pages
- Khan Academy exercise framework used for many exercise types



Backend:

- REpresentational State Transfer (REST) design for client/server interaction: Decouples client and server.
- Python Django framework and MySQL for storing student responses and progress
- Flexible API to store student scores and interaction data
- Support for managing separate classes with separate textbook instances