INVESTIGATING DIFFICULT TOPICS IN A DATA STRUCTURES COURSE USING ITEM RESPONSE THEORY AND LOGGED DATA ANALYSIS

Eric Fouh	Lehigh University
Mohammed F. Farghally	Virginia Tech
Sally Hamouda	Virginia Tech
Kyuhan Koh	Virginia Tech
Clifford A. Shaffer	Virginia Tech

Friday, July 1, 2016

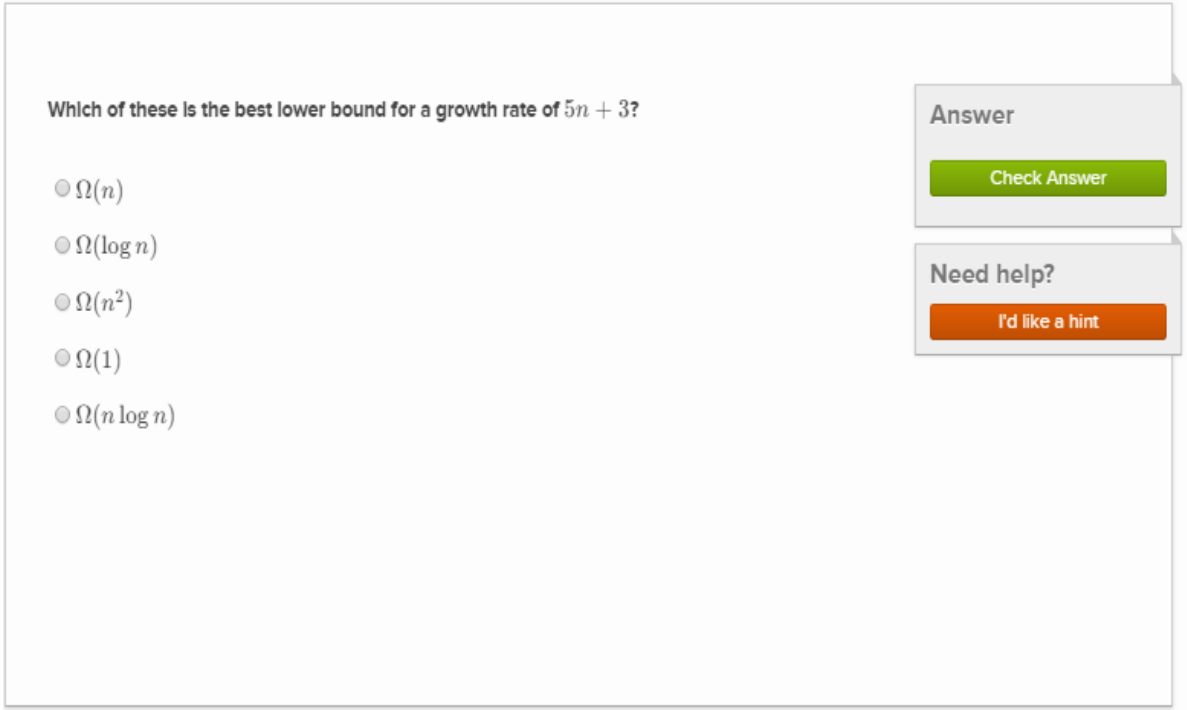
Educational Data Mining 2016 Raleigh, North Carolina

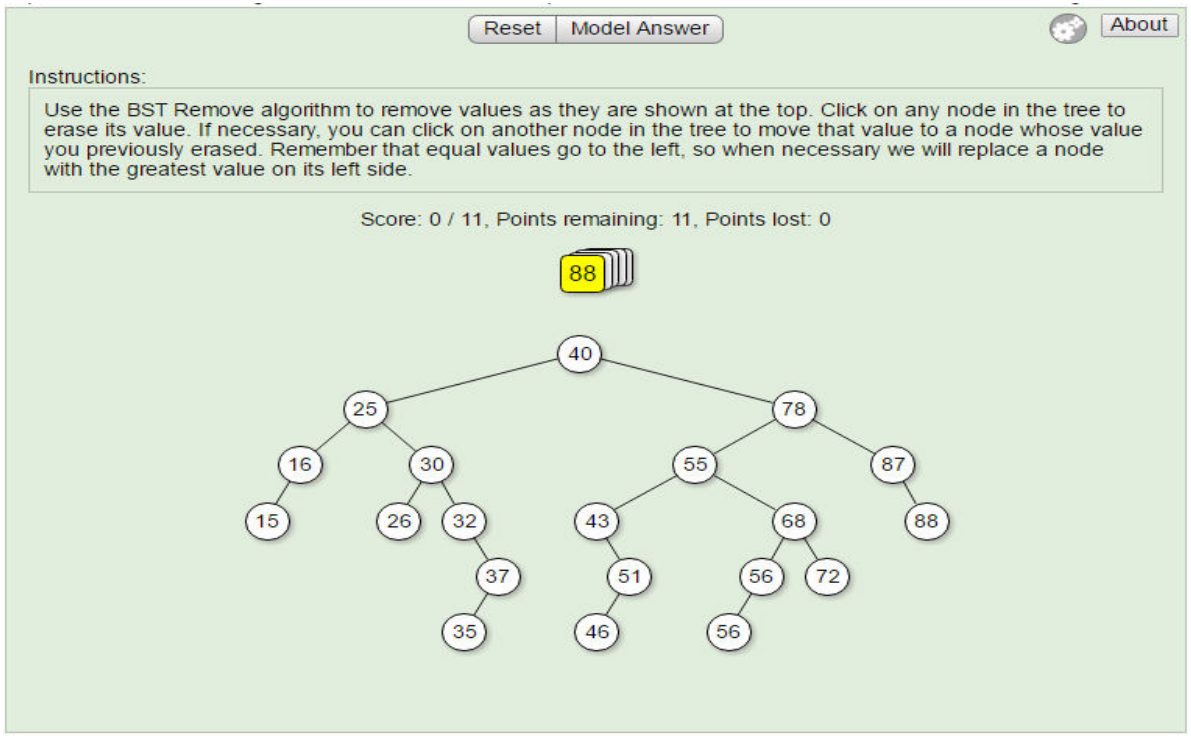
Agenda

- The OpenDSA System
- Identifying Difficult Topics
  - Through logged Data Analysis
  - Through Item Response Theory
  - Through Instructor Survey
- Why Algorithm Analysis is Hard?
- Future Work

The OpenDSA System

- Interactive eTextbook infrastructure and a body of content.
- Automatically graded interactive exercises and visualizations.





Identifying Difficult Topics

- To develop appropriate interventions and better allocate course resources.
- Analyzing exercise answers:
  - OpenDSA log data
    - Exercise correct ratio
    - Using hints and guessing
  - Item Response Theory
- 143 students enrolled in a CS3 course in Virginia Tech during Fall 2014.
- OpenDSA exercises accounted for 20% of total grade.

Analysis of Correct Answer Ratios

- OpenDSA adopts a mastery level approach.
- A Student's performance on an exercise is assessed by:

$$r_i = \frac{\#\ of\ correct\ attempts}{\#\ of\ total\ attempts}$$

- Each exercise's difficulty level is assessed by:

$$dl = 1 - \frac{\sum_{i=1}^{n} r_i}{n}$$

Analysis of Correct Answer Ratios (Cont.)

- Min: 0, Max: 0.72

Quartile	Difficulty Level Range	Topics
Fourth	$dl > 0.25$	22 / 26 Algorithm Analysis
Third	$0.13 \leq dl \leq 0.25$	14 /25 Algorithm Mechanics 10 /25 General course concepts
Second	$0.05 \leq dl < 0.13$	23 / 25 Algorithm Mechanics
First	$dl < 0.05$	Algorithm Mechanics

Analyzing Hint Use and Guessing

- Harder exercises are expected to show a higher rate of hint use and trial and error.
- Hint ratio is defined as:

$$hr = \frac{\#\ of\ hints\ used}{\#\ of\ total\ attempts}$$

- Incorrect ratio is defined as:

$$ir = \frac{\#\ of\ incorrect\ answers}{\#\ of\ total\ attempts}$$

Analyzing Hint Use and Guessing (Cont.)

- Fourth Quartile:

Topic	Hint Ratio	Incorrect Ratio
Algorithm Analysis	0.24	0.77
Tree Overhead Analysis	0.78	0.73
Quicksort Analysis	0.32	0.67
Mathematical Background	0.25	0.63
Shellsort	0.16	0.61
List Overhead Analysis	0.93	0.60
Quicksort partition	0.27	0.58

- First Quartile exercises are related to linear structures.

Item Response Theory Analysis

- A theory of testing that relates student performance on an individual item to his/her overall ability.
- IRT curves:
  - Item Characteristics Curve (ICC).
  - Item Information Curve (IIC).
  - Test Information Function (TIF).

Item Response Theory Analysis (Cont.)

- Each chapter was treated as a test with exercises as items.
- Exercise difficulty ratios were dichotomized.
  - $r \geq 0.75$ ---------- 1
  - $r \leq 0.75$ ---------- 0

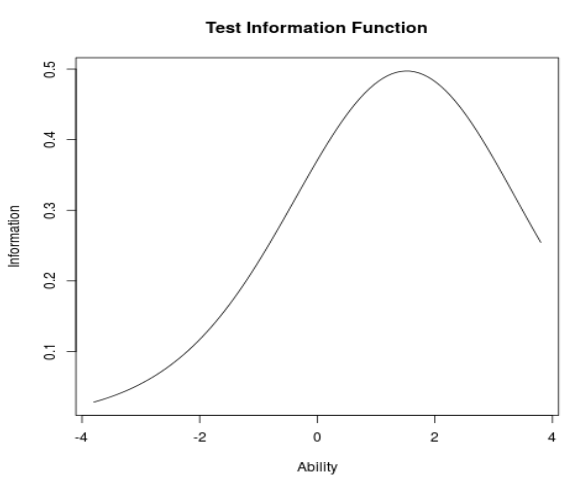
$$r = \frac{\#\ of\ correct\ attempts}{\#\ of\ total\ attempts}$$

- 1PL model was adopted.

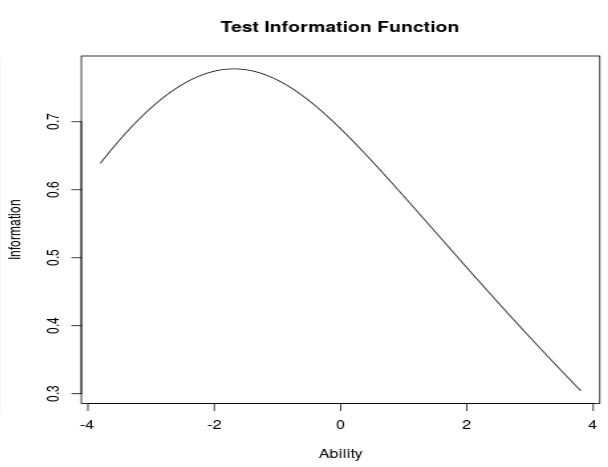
$$p_i(\theta) = \frac{e^{(\theta - b_i)}}{1 + e^{(\theta - b_i)}}$$

Item Response Theory Analysis (Cont.)

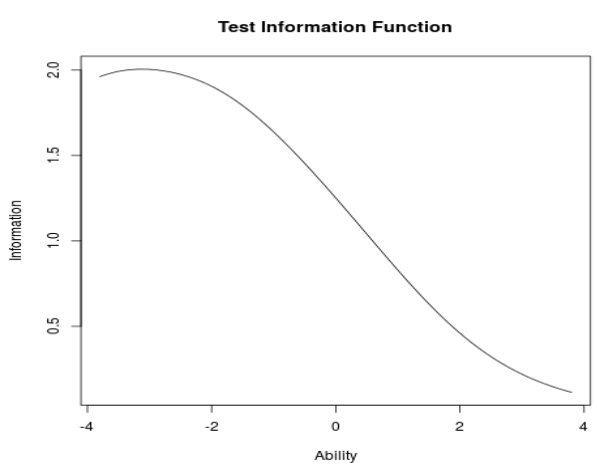
Algorithm Analysis



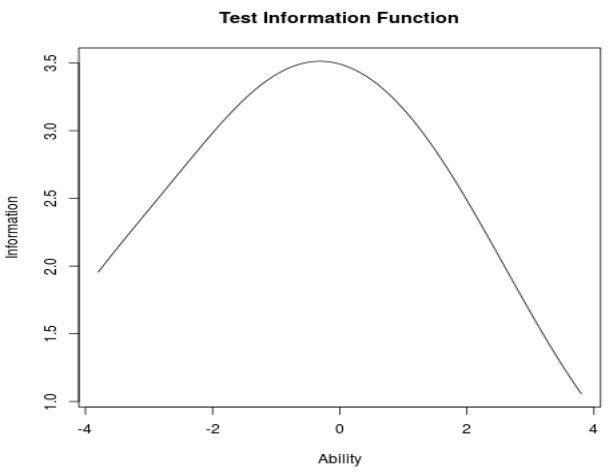
Binary Trees



Linear Structures



Sorting



- Overall 21 / 100 exercises discriminates between above average students.
- *19 of these 21 are related to Algorithm Analysis.*

Instructor Survey Results

- A survey was sent to CS3 instructors through SIGCSE mailing list.
- 23 instructor responses were analyzed.

Topic	# of instructors	Percentage
Dynamic Programming	7	18
Algorithm Analysis	6	15
OOP and Design	6	15
Recursion	4	10
Trees and Heaps	3	7
Proofs	3	7

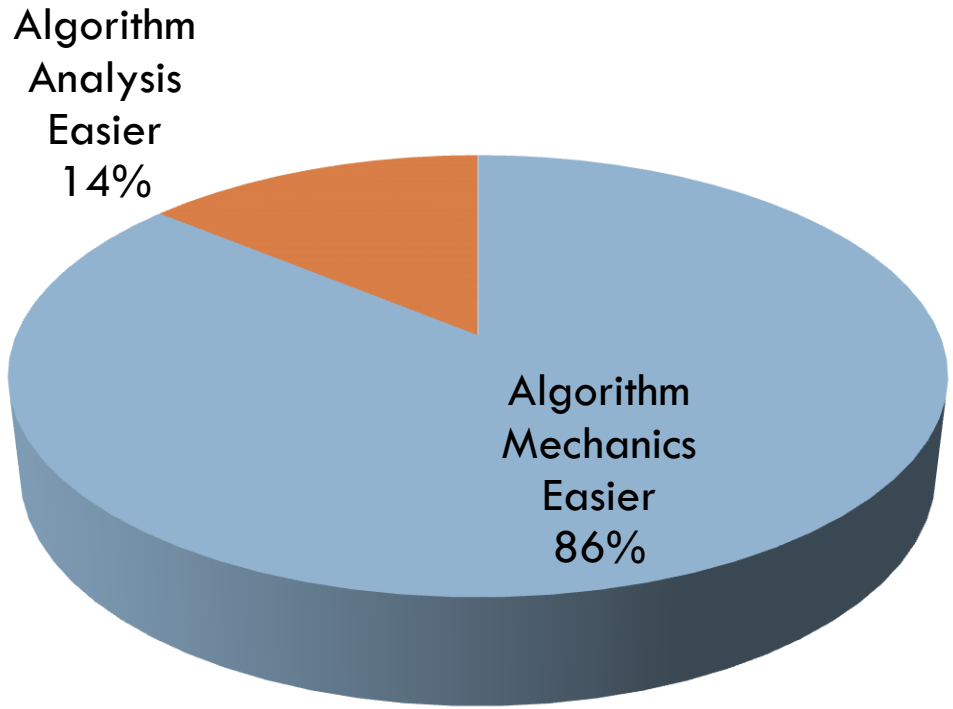
Why Algorithm Analysis is Hard?

- OpenDSA logs were analyzed for 3 universities to see the time spent in Algorithm analysis content for the sorting chapter.

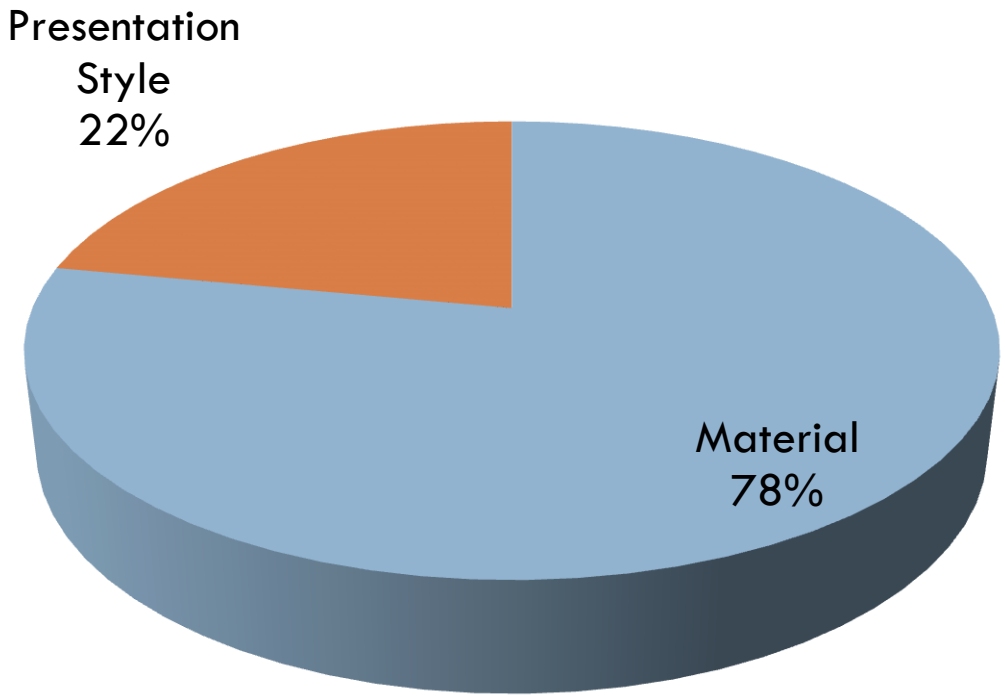
University	Module	N	Mean (Sec)	% < 1 min
Virginia Tech	Insertionsort	98	63.57	74.48
	Mergesort	96	39.79	78.12
	Quicksort	92	64.71	73.91
Texas El Paso	Insertionsort	26	49.84	80.76
	Mergesort	22	41.45	77.27
	Quicksort	16	16.18	93.75
Florida	Insertionsort	53	40.39	84.90
	Mergesort	44	18.63	95.45
	Quicksort	39	26.12	92.30

Why Algorithm Analysis is Hard? (Cont.)

- 84 students were surveyed at the end of Fall 2014 in Virginia Tech:

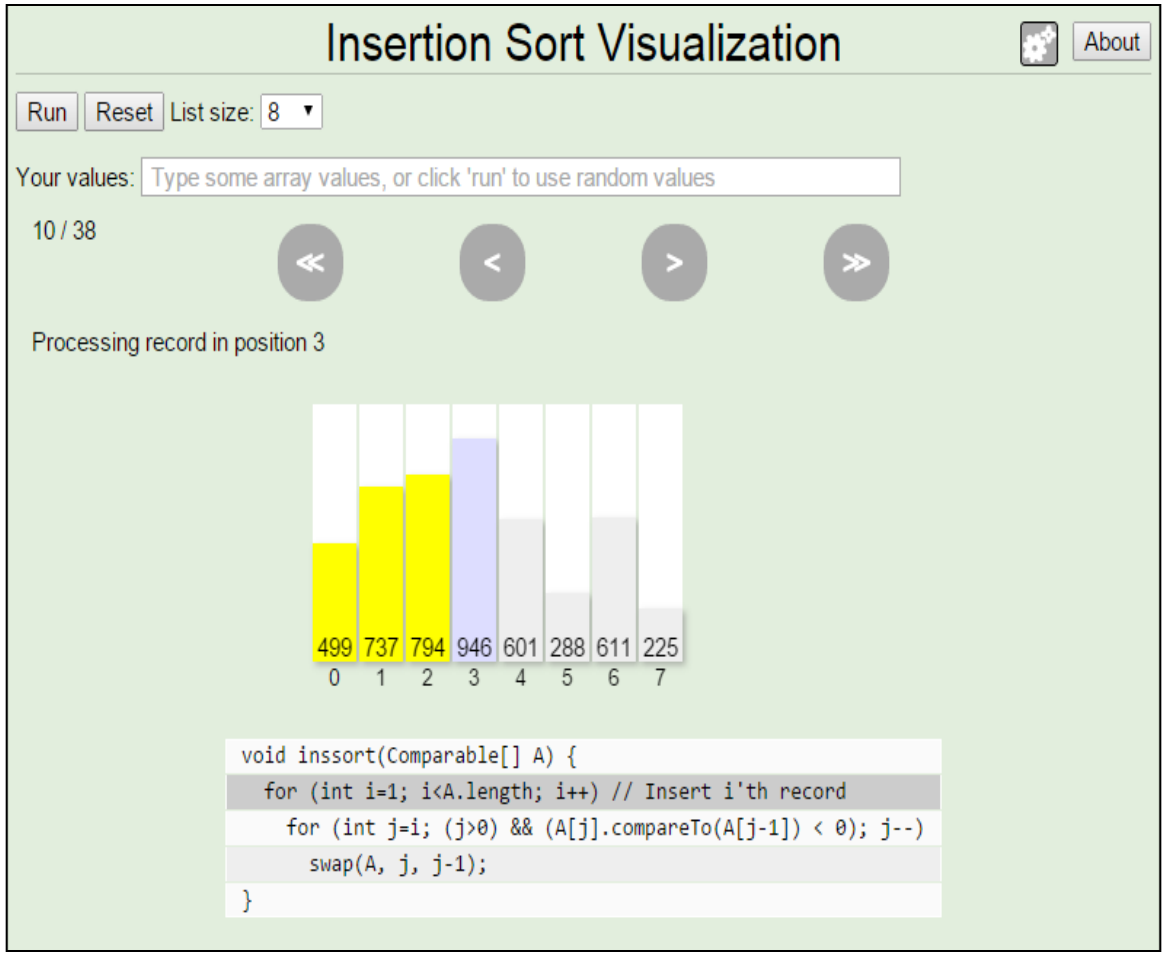


Which is easier?



Reason

Why Algorithm Analysis is Hard? (Cont.)



The body of `inssort` consists of two nested `for` loops. The outer `for` loop is executed $n-1$ times. The inner `for` loop is harder to analyze because the number of times it executes depends on how many records in positions 0 to $i-1$ have a value less than that of the record in position i. In the worst case, each record must make its way to the start of the array. This would occur if the records are initially arranged from highest to lowest, in the reverse of sorted order. In this case, the number of comparisons will be one the first time through the `for` loop, two the second time, and so on. Thus, the total number of comparisons will be

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} \approx n^2/2 = \Theta(n^2).$$

In contrast, consider the best-case cost. This occurs when the values occur in sorted order from lowest to highest. In this case, every test on the inner `for` loop will fail immediately, and no records will be moved. The total number of comparisons will be $n-1$, which is the number of times the outer `for` loop executes. Thus, the cost for Insertion Sort in the best case is $\Theta(n)$.

What is the average-case cost of Insertion Sort? When record i is processed, the number of times through the inner `for` loop depends on how far "out of order" the record is. In particular, the inner `for` loop is executed once for each value greater than the value of record i that appears in array positions 0 through $i-1$. For example, in the slideshows above the value 14 is initially preceded by five values greater than it. Each such occurrence is called an *inversion*. The number of inversions (i.e., the number of values greater than a given value that occur prior to it in the array) will determine the number of comparisons and swaps that must take place. So long as all swaps are to adjacent records, 14 will have to swap at least six times to get to the right position.

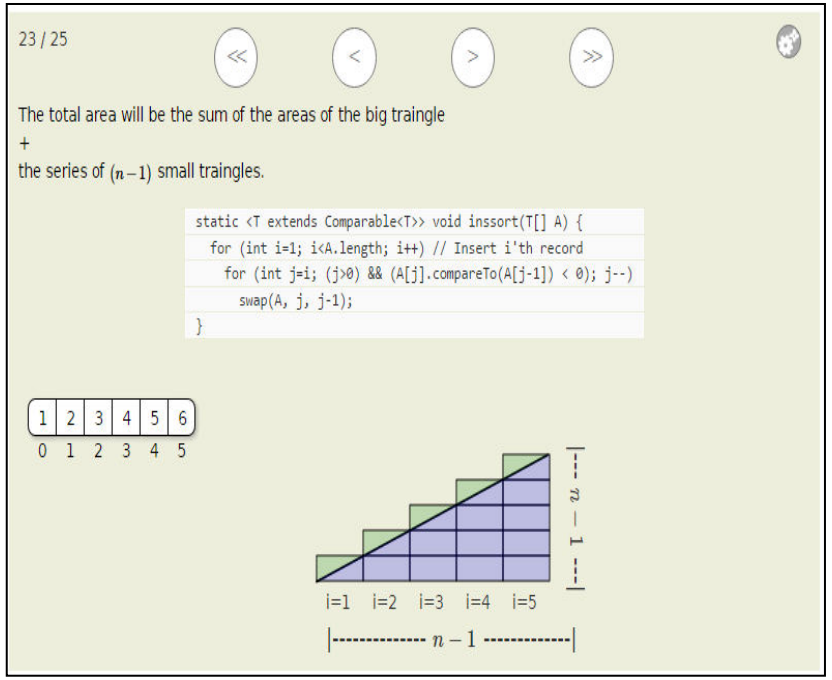
To calculate the average cost, we want to determine what the average number of inversions will be for the record in position i. We expect on average that half of the records in the first $i-1$ array positions will have a value greater than that of the record at position i. Thus, the average case should be about half the cost of the worst case, or around $n^2/4$, which is still $\Theta(n^2)$. So, the average case is no better than the worst case in its growth rate.

While the best case is significantly faster than the average and worst cases, the average and worst cases are usually more reliable indicators of the "typical" running time. However, there are situations where we can expect the input to be in sorted or nearly sorted order. One example is when an already sorted list is slightly disordered by a small number of additions to the list; restoring sorted order using Insertion Sort might be a good idea if we know that the disordering is slight. And even when the input is not perfectly sorted, Insertion Sort's cost goes up in proportion to the number of inversions. So a "nearly sorted" list will always be cheap to sort with Insertion Sort. Examples of algorithms that take advantage of Insertion Sort's near-best-case running time are *Shellsort* and *Quicksort*.

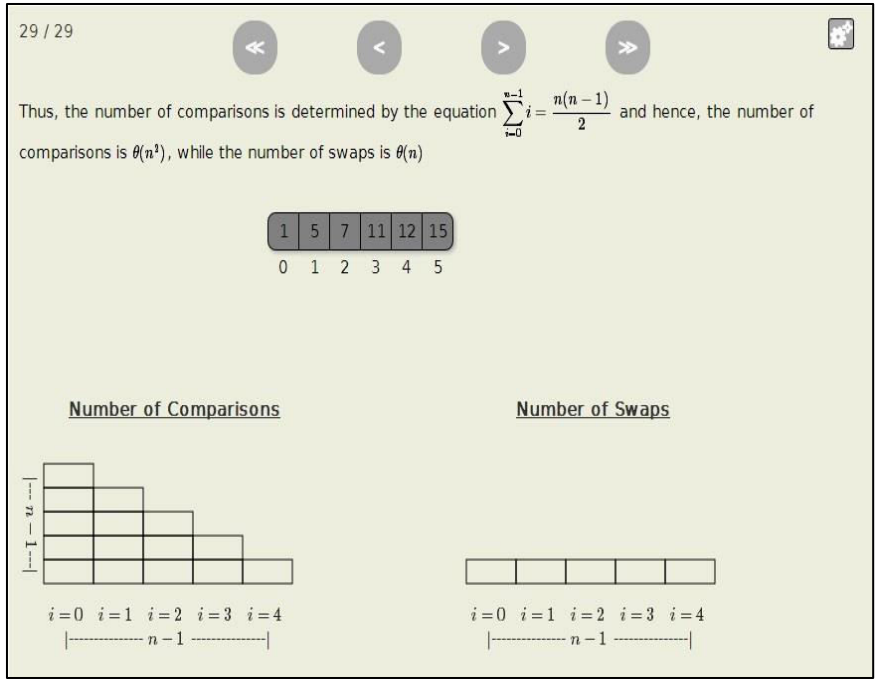
Future Work

- Towards Visualizing Algorithm Analysis content in OpenDSA.
- 28 visualization were developed for the Algorithm Analysis Introduction and Sorting chapters.
- Preliminary results indicate higher student engagement and better student performance.

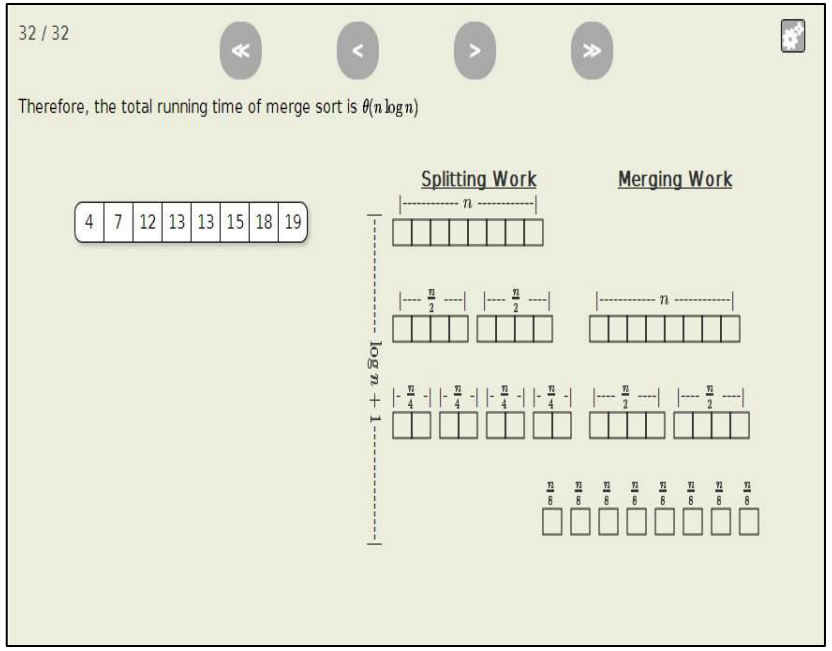
Worst case
Insertionsort



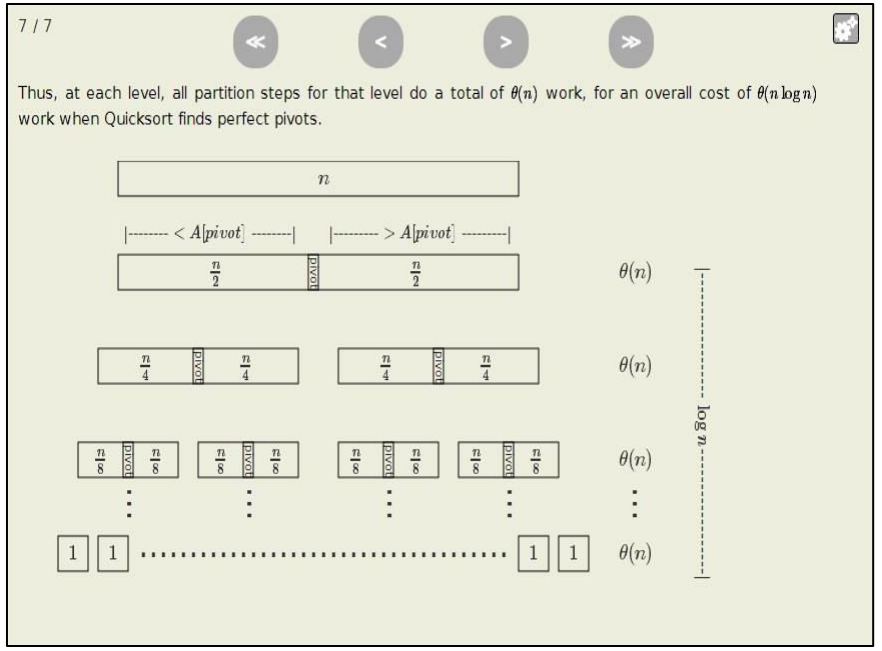
Selectionsort



Mergesort



Best case
Quicksort



Questions