Building a Dynamic Website For Data Structures And Algorithms in Computer Science.

By

No.	Student Name	RegNo	Std No
1	**AZAMUKE DENISH**	*16/U/171*	*216001004*
2	**MUTAMBUZE PAUL**	*16/U/7738/EVE*	*216012181*
3	**KUSEMERERWA ROY**	*15/U/20296/EVE*	*215020649*
4	**NSUBUGA FRANCIS**	16/U/19985/EVE	*216021708*
5	**OKOCHE GILBERT**	*16/U/20063/EVE*	*216021710*

[*]

Makerere University

march 10 2018

[*]supervisor: Ernest Mwebaze

1 Abstract

[1]Data structures are a conceptually demanding topic which confronts many Computer Science students early in their course. The topic has a strong conceptual basis and often proves difficult for many to grasp. A number of previous studies have examined that the use of interaction and visualization within the systems can motivate a student to engage in the learning process. This literature review investigates the effectiveness of these systems that were and are being used today for teaching and learning of data structures to novice Computer Science students.

In this report a web application that features the [2]visualization of commonly used data structures and their associated insertion and deletion operations is introduced.

2 Introduction

Data Structures and Algorithms is a fundamental course in Computer Science[3]. However, many students find it difficult because it requires abstract thinking. It would be very helpful if there was a visualization tool of data structures such as arrays, queues, stacks, trees and graphs for students to experiment with. The tool would allow students to see how an element is inserted into or deleted from different data structures, how a tree is traversed in different order (pre-order, in-order, post-order, level-order), etc. Moreover, this tool would provide a simple language, by which students can write their own algorithms so that the execution of the algorithm is animated. This project is intended to create such an exploration environment, in which students can learn through experimentation. This tool can be used as an effective supplement to the traditional lecture room education and textbooks for Data Structures and Algorithms courses. The web application presented in this document has the following functionality; Provides complete visualization for the widely used data structures such as array, stack, queue, tree, heap, graph, etc[4]. Provides the animation of common operations associated with the data structures, such as inserting an element into and deleting an element from array, stack, and queue.Provides animation of simple user-defined algorithms.

3 Background

The development of technologies and the evolvement of the World Wide Web have influenced education. Instructional Web sites and

courses on the Web have grown dramatically. Web based courses that consist of the syllabus, assignments and lecture notes are now widely used. Instructional Web sites that are dedicated to Data Structures and algorithms can be easily found by using Search Engines.

However, the majority of the instructional web sites explored during this project lack interactive multimedia.

One of the best sites found that does contain interactivity is a course site developed for teaching Data Structures and Algorithms in Java by the Computer Science Department of Brown University. This site has a collection of applets that demonstrate some commonly used data structures such as queues, stacks, and some famous algorithms such as merge sort, quick sort, etc. However, these applets are not complete and lack a common Graphical User Interface. Another good site in interactive Data Structure visualizations is developed by Duane J. Jarc in George Washington University[5]. This site provides animations in binary Trees, graphs, and sorting algorithms. But there is no animation available for algorithms that are defined by users.

Another algorithm animation system found is Zeus, which is developed by Digital Equipment Corporation's Systems Research Center[6]. This system is a little complicated, require from the user lots of effort to prepare animations. It is targeted at more advanced application programmers.

Since our web application is intended to the aid second year Computer Science students learning Data Structures and Algorithms, ease of use becomes our main consideration. Our approach for the algorithm animation is that the user provides inputs for the already developed algorithm and it provides an output in form of a data structure that they require in form of an animation. [7]The only effort the user needs to make is to instantiate the data structures he/she wants to observe using the observable data types provided by the software. An animation frame is created and the observable data structures are added to the frame so that the user can watch the changes made to the data structures when the algorithm is executing.

4 Problem Statement

Computing is the foundation of modern society. A proficient computing workforce is essential for maintaining our country's leadership and competitiveness in the global economy. For this reason, the recent decline in student motivation and diversity pose significant challenges to the continuation of the nation's prominent position in the global high technology arena. An immediate solution is urgently needed.

We must build excitement and enthusiasm for our discipline in order to attract a bright new generation of students early in their academic careers.

Problems faced include:

[8]The first difficulty identified by teachers and tutors, was the low motivation of students according to Mesialo, et al: What on earth do these (theoretical) algorithms and data structures have to do with my future (practical) job?

The second difficulty talked about was how tricky data structures can be and how they often remained abstract to students.

Third stated that the assignments were done individually and students were not encouraged to cooperate.

Lastly, the problem settings were closed: so for example, a student might be asked to implement the depth first search algorithm. This made the assignments distant from any research or real problem setting.

4.1 Purpose of the Course

The main purpose of this course is to provide the students with solid foundations in the basic concepts of programming: data structures and algorithms.

5 Objectives

To assess how the choice of data structures and algorithm design methods impacts the performance of programs.

To choose the appropriate data structure and algorithm design method for a specified application.

To write programs using procedure-oriented design principles.

To solve problems using data structures such as linear lists, stacks, queues, hash tables, binary trees, heaps, binary search trees, and graphs and writing programs for these solutions[9].

5.1 Learning outcome for students (Specific Objectives)

Students will:

- Be familiar with basic data structure of algorithms.
- Be familiar with writing recursive methods by using java.
- Master the implementation of linked data structures such as linked lists and binary trees.
- Be familiar with advanced data structures such as balanced search trees, hash tables, priority queues and the disjoint set union/find data structure
- Be familiar with several searching and sorting algorithms including quick-sort, Merge-Sort and Heap-Sort[10].
- Be familiar with some graph algorithms such as shortest path and minimum spanning tree
- Master the standard data structure library of a major programming language (e.g. in java)
- Master analyzing problems and writing program solutions to problems using the above techniques.

5.2 Plans for developing and improving the course

Through coursework and group assignments, students are expected to cultivate the following attitudes and dispositions:

- [11]Confidence in Programming (in OOP java) skills and knowledge.
- Desire for continuous and independent learning, analyzing and using data.
- Awareness of career opportunities in computer organization.
- [12]Appreciation for the dynamic role of solving problems and algorithms

6 Reasons for teaching/learning data structures

There were five categories of instructor rationale identified for the[13] purposes of teaching data structures (Raymond et al, 2004), but there were merged to 4 in this context:

Developing Transferable thinking: Here, data structures were described as a vehicle for developing thinking and problem solving skills. The design of a data structure is like the solution to a riddle: the process of developing the answer is as important as the answer itself.

Improving programming skills: [14]Implementation of data structures is used to improve programming skills of students, especially their dexterity with recursion and pointers. "...reading and using the code without having written something similar is like watching Olympic ping pong on TV. It sure looks easy, even somewhat repetitious; however, the level of precision is only experienced by trying to do the same."

Knowing "what's under the hood": Students often use libraries to implement data structures and algorithms. This category acknowledges a place for learning the libraries and modifying them to serve their purpose. "A graduate should be convinced that fancy technology is understandable, and adjustable; they should feel that they can be masters of the magic that the Wizard hides behind the curtain."

Component Thinking: This category puts emphasis on the importance of student learning component engineering principles, such as black-box interaction and code re-use. "Software Engineering is moving away from emphasis on the creation of code, toward emphasis on components and code reuse." This avoids the process of re-inventing the wheel.

All of the above reasons gave a strong indication on how important it was for the students to learn data structures as part of their Undergraduate Degree in Computer Science.

7 Tools and their Outcomes/Methodology

There have been a number of tools developed for the purpose of teaching and learning data structures. We will look at those specific to our problem domain.

7.1 The DSL

Tool An interactive visualization tool, the Data Structures Learning (DSL) discussed in Alhousban (2011) paper[15], was developed and used first in a short mini study that showed that, used together with visualizations of algorithms, and aural instructions, it produced faster student response times than did textual instructions. This result suggested that the additional use of the auditory sensory channel reduced the cognitive load of the student. It was then used in a study over two academic terms in which students studying data structures module were offered the opportunity to use the DSL tool with either aural or textual instructions. The collected data showed that the DSL tool was extensively used by weaker students. A comparison was made of the students' DSL use with their end-of-year assessment marks which revealed that academically weaker students had tended to use the tool most. From the evaluation of the DSL tool, it was concluded that less able students were keen to use any useful and available instrument to aid their understanding, especially of difficult concepts.

7.2 The Vedya

Tool Vedya tool discussed in Segura P et al (2008) paper was used for visualizations of data structures and algorithm schemes. [8]The pedagogical aim of Vedya was to facilitate the student?s grasp of the target procedures of education in Computer Science by means of interactive learning, in order to facilitate teamwork and communication between teachers and students. It managed the administration of the individual students? homework, including generation of exercises, tests, grading the deliverable homework, and storing achieved results. It covered the most common data structures: stacks queues, binary search trees, AVL trees, priority queues, sorting and hash tables.

It made use of interesting visualization techniques through a maze representation. For example, when the user inserted an item in a queue, the truck was shown to throw the item on top of the maze. When extracting an item, the end of the maze opened and the first item would fall down. The use of the maze illustrated that items could not jump over the previous ones and the fact that in a queue, items were extracted in the order that they came in.

[8]Vedya tool helped students benefit from complementary and interactive material, facilitating the intuitive comprehension of most typical operations of classical data structures without any restriction of time or material.

7.3 Sketch based Interfaces

We are entering a new era of teaching, learning and computing. Adamchik's (2011)[16] vision was to design a pen based computing environment in which student itself would draw data structure using tablet and stylus. For this, they had to develop an interface with four key parts:

- Stroke recognition and beautification.
- The association of strokes to an underlying domain-dependent data structure,
- The animation of the algorithms, and
- The verification of those algorithms.

The idea was to draw a particular instance of data structure using stylus and invoke an algorithm to animate over this data structure. Two study modes were identified according to students' cognitive learning style. The first mode was the initial learning of the algorithm by stepping through it either using textual description or graphical approach. In the second study mode, students actively participated in the learning process. An example would be student tested themselves in preparation for the exam.

This system was developed after much evaluation on the how students learned data structures and algorithms. The evaluations indicated that the algorithm animations had no significant effect if the students were just observers. Student must be actively engaged in order to learn. Therefore, electronic education technology integrated with a Tablet PC environment had a fundamental influence since it allowed expression and the exchange of ideas in a highly interactive atmosphere. Using pen-based gesture interface, the system promoted the students' intuition for both problem solving and algorithm thinking[17].

7.4 VIDSAA

[8]Tool Kacha and Ron (2006) reported on a project which had developed a range of learning objects1 that helped students to learn about the different data structures and the algorithms by which they were controlled, called VIDSAA (Visualization in Data Structure and Algorithms). In this tool they covered data structures like Lists, Stacks, Queues, Trees, Sorting and Searching and Graph. There were a total of 55 animations. Each animation provided an interactive representation of the algorithms and data structures which the student could control. In this, they visualized the data structure using programs and flow chart representation.

The use of play, stop, forward and backward buttons enabled the student to step through the processing and to observe the changes to the variables and the relevant data. This feature was intended to enable the students to pause and think before watching a further step of the animation and this was intended to provide an opportunity for students to become active learners. VIDSAA was designed with a capacity for installation onto a student?s computer for the use of outside the classroom activities as part of their independent learning.

7.5 Different approaches to learning Data Structures

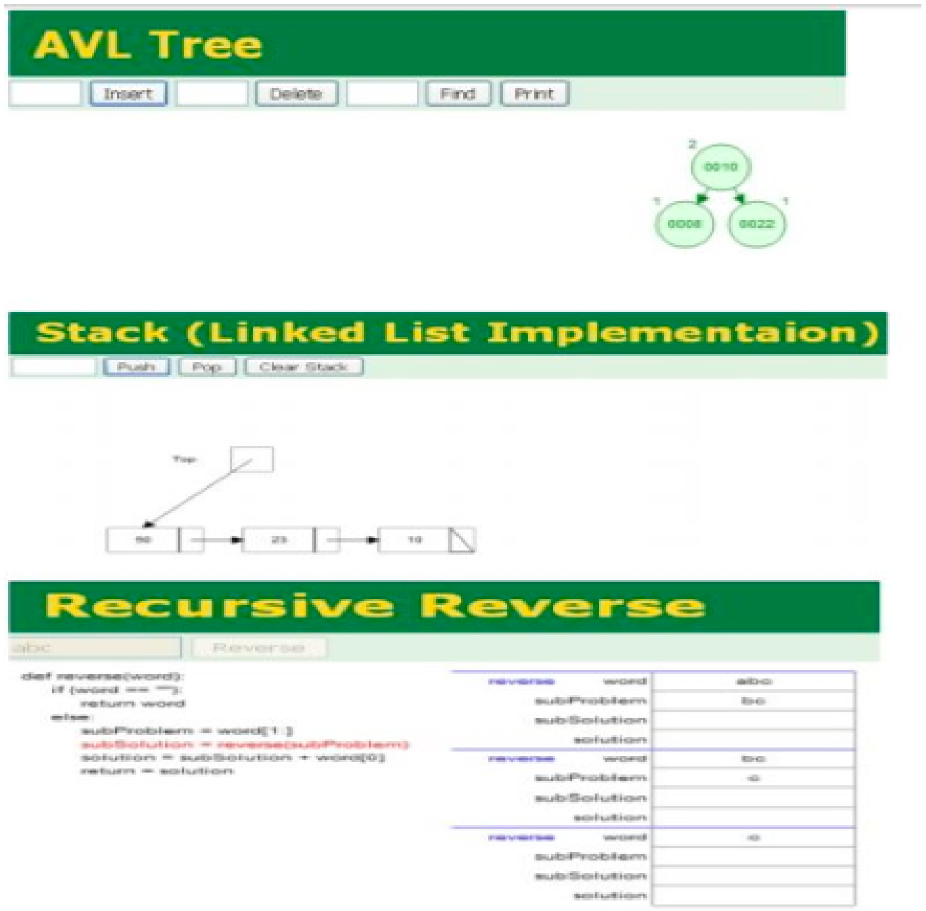
Three most popular methods were identified that are being used today to make data structures for both learning and teaching, effective and fun. The tools mentioned above integrated some or all of these methods in their systems. We will discuss how Multimedia-interactive Systems approach has been useful.

7.6 Multimedia-interactive Systems (VISUALIZATION TOOL)

[18]Andrade et al (2008) compared three different teaching approaches, namely, traditional teacher led instruction, Web pages and a multimedia-interactive system to determine which method works best. Three different groups were involved, each group using a different approach. This system was specifically designed to teach binary trees. The Web page contained hyperlinks to navigate through the information and examples. The information was shown by text and images. There was no animation and interactivity in the Web page. In this case the instructor had to check and evaluate the student's exercises. The multimedia-interactive system kept the same webpage, but there were some interaction differences: animation and sounds as explanatory sections of each topic (insertion, deletion and searching nodes). Also, this system included interactivity where students had the ability to answer interactive exercises by moving data and images. In this case

the instructor did not check and evaluate the students' exercises, the system did this activity. The first two groups of students were tested based on a written test and the multimedia group students were tested on the system. After the results were evaluated, it was concluded that multimedia could effectively be used to help students learn data structures specifically binary trees

sample data structures on Visualization Tools



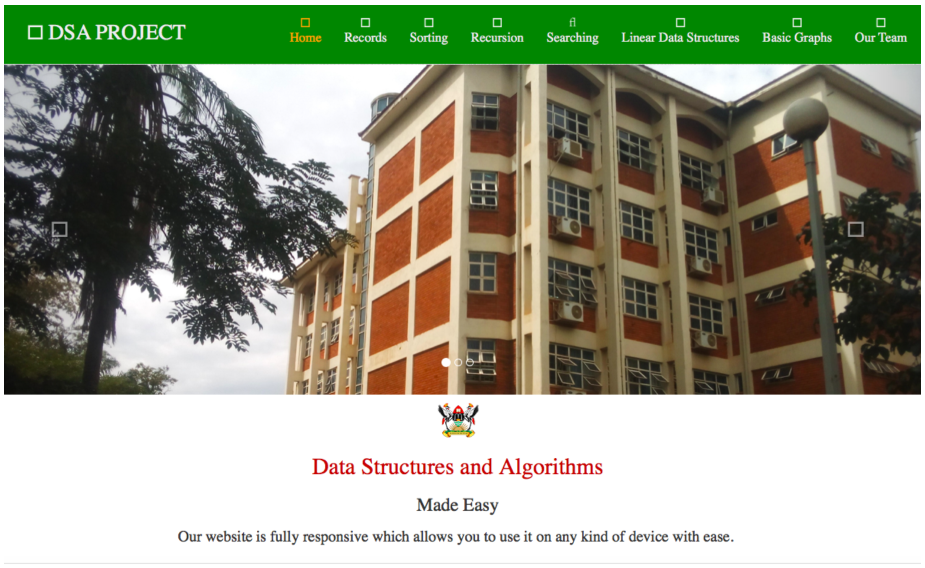
```
def reverse(word):
    if (word == ""):
        return word
    else:
        subProblem = word[1:]
        subSolution = reverse(subProblem)
        solution = subSolution + word[0]
        return = solution
```

reverse	word	abc
	subProblem	bc
	subSolution	
	solution	
reverse	word	bc
	subProblem	c
	subSolution	
	solution	
reverse	word	c
	subProblem	
	subSolution	
	solution	

fig 1

8 Link To The Implementation

Find our implementation using https://csceve17-005.000webhostapp.com

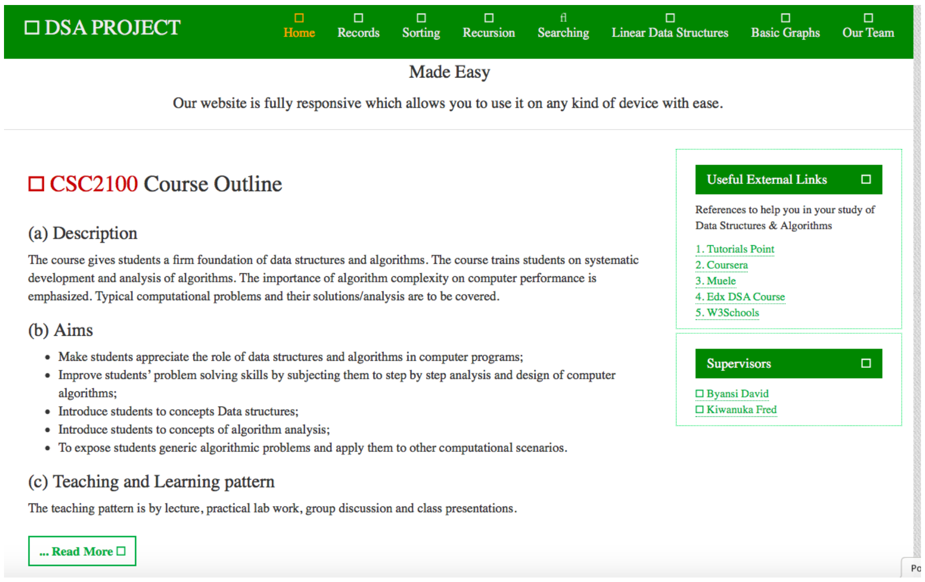


Data Structures and Algorithms

Made Easy

Our website is fully responsive which allows you to use it on any kind of device with ease.

fig 2



Made Easy

Our website is fully responsive which allows you to use it on any kind of device with ease.

☐ CSC2100 Course Outline

(a) Description

The course gives students a firm foundation of data structures and algorithms. The course trains students on systematic development and analysis of algorithms. The importance of algorithm complexity on computer performance is emphasized. Typical computational problems and their solutions/analysis are to be covered.

(b) Aims

- Make students appreciate the role of data structures and algorithms in computer programs;
- Improve students' problem solving skills by subjecting them to step by step analysis and design of computer algorithms;
- Introduce students to concepts Data structures;
- Introduce students to concepts of algorithm analysis;
- To expose students generic algorithmic problems and apply them to other computational scenarios.

(c) Teaching and Learning pattern

The teaching pattern is by lecture, practical lab work, group discussion and class presentations.

... Read More ☐

Useful External Links ☐

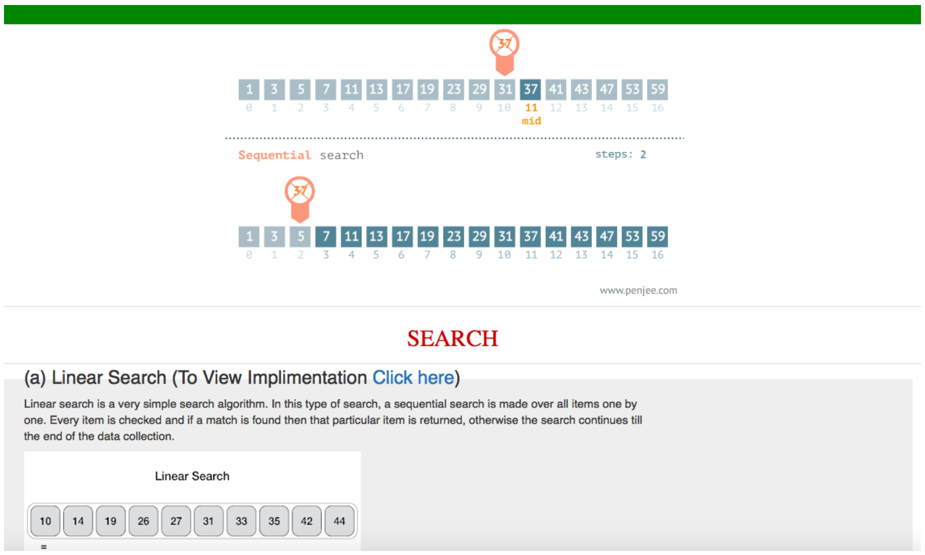
References to help you in your study of Data Structures & Algorithms

1. Tutorials Point
2. Coursera
3. Muele
4. Edx DSA Course
5. W3Schools

Supervisors ☐

☐ Byansi David
☐ Kiwanuka Fred

fig 3

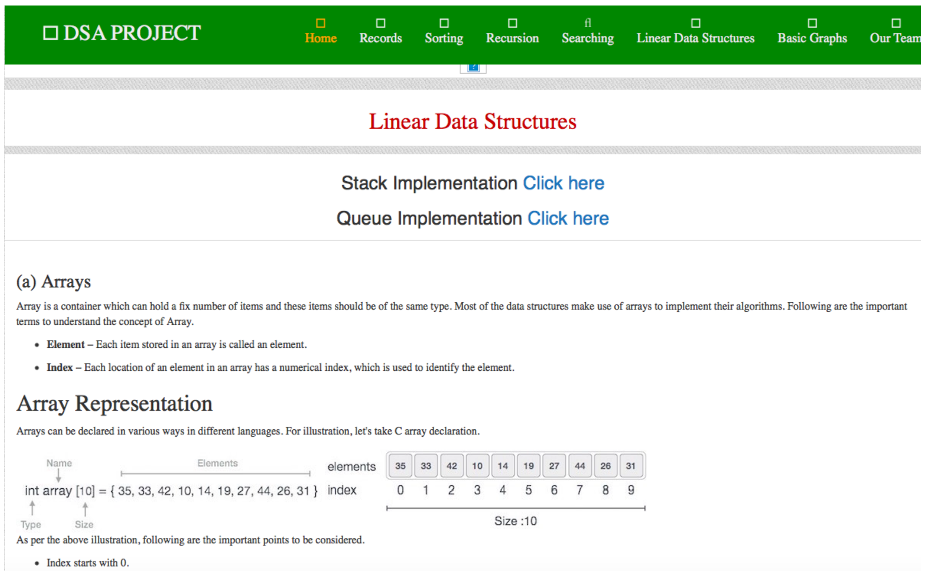


SEARCH

(a) Linear Search (To View Implimentation Click here)

Linear search is a very simple search algorithm. In this type of search, a sequential search is made over all items one by one. Every item is checked and if a match is found then that particular item is returned, otherwise the search continues till the end of the data collection.

fig 4



Linear Data Structures

Stack Implementation Click here

Queue Implementation Click here

(a) Arrays

Array is a container which can hold a fix number of items and these items should be of the same type. Most of the data structures make use of arrays to implement their algorithms. Following are the important terms to understand the concept of Array.

- **Element** – Each item stored in an array is called an element.
- **Index** – Each location of an element in an array has a numerical index, which is used to identify the element.

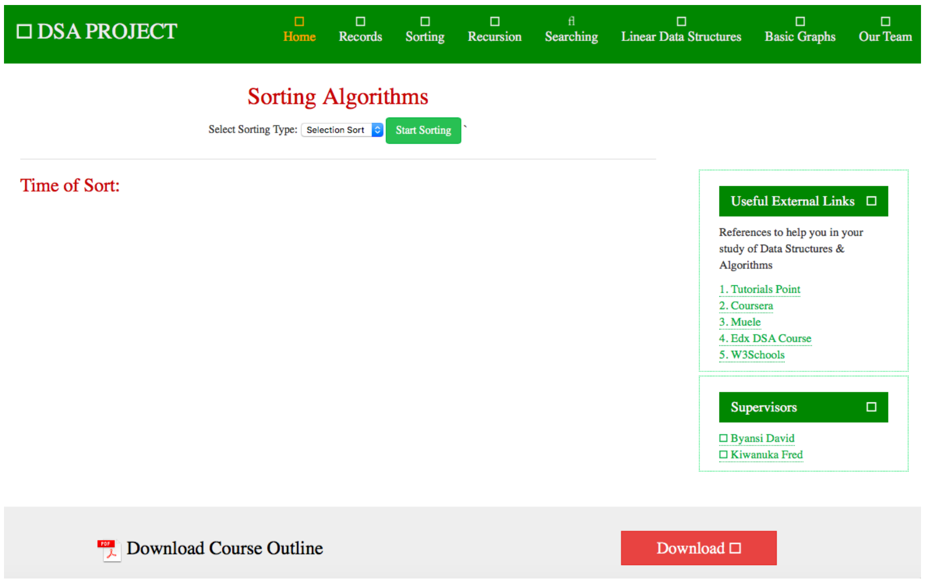
Array Representation

Arrays can be declared in various ways in different languages. For illustration, let's take C array declaration.

As per the above illustration, following are the important points to be considered.

- Index starts with 0.

fig 5



Sorting Algorithms

Select Sorting Type: Selection Sort [Start Sorting]

Time of Sort:

Useful External Links

References to help you in your study of Data Structures & Algorithms

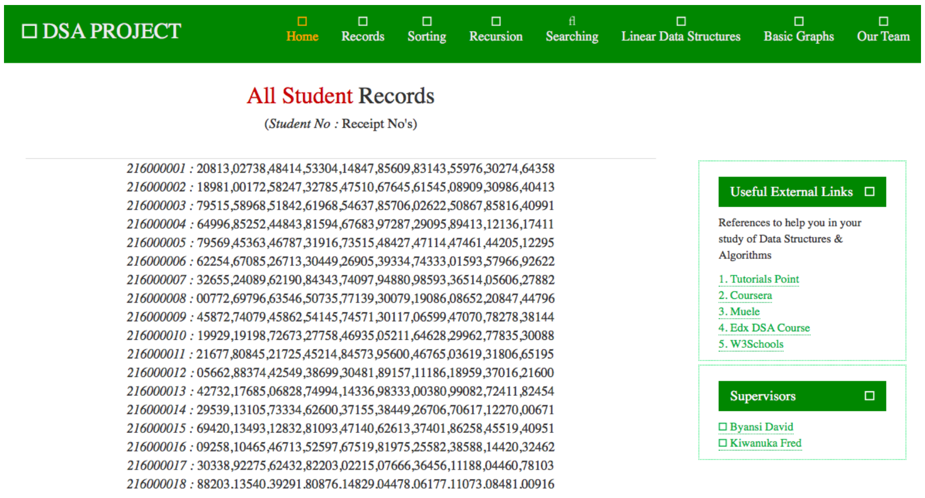
1. Tutorials Point
2. Coursera
3. Muele
4. Edx DSA Course
5. W3Schools

Supervisors

- Byansi David
- Kiwanuka Fred

Download Course Outline [Download]

fig 6



All Student Records

(*Student No :* Receipt No's)

```
216000001 : 20813,02738,48414,53304,14847,85609,83143,55976,30274,64358
216000002 : 18981,00172,58247,32785,47510,67645,61545,08909,30986,40413
216000003 : 79515,58968,51842,61968,54637,85706,02622,50867,85816,40991
216000004 : 64996,85252,44843,81594,67683,97287,29095,89413,12136,17411
216000005 : 79569,45363,46787,31916,73515,48427,47114,47461,44205,12295
216000006 : 62254,67085,26713,30449,26905,39334,74333,01593,57966,92622
216000007 : 32655,24089,62190,84343,74097,94880,98593,36514,05606,27882
216000008 : 00772,69796,63546,50735,77139,30079,19086,08652,20847,44796
216000009 : 45872,74079,45862,54145,74571,30117,06599,47070,78278,38144
216000010 : 19929,19198,72673,27758,46935,05211,64628,29962,77835,30088
216000011 : 21677,80845,21725,45214,84573,95600,46765,03619,31806,65195
216000012 : 05662,88374,42549,38699,30481,89157,11186,18959,37016,21600
216000013 : 42732,17685,06828,74994,14336,98333,00380,99082,72411,82454
216000014 : 29539,13105,73334,62600,37155,38449,26706,70617,12270,00671
216000015 : 69420,13493,12832,81093,47140,62613,37401,86258,45519,40951
216000016 : 09258,10465,46713,52597,67519,81975,25582,38588,14420,32462
216000017 : 30338,92275,62432,82203,02215,07666,36456,11188,04460,78103
216000018 : 88203.13540.39291.80876.14829.04478.06177.11073.08481.00916
```

Useful External Links

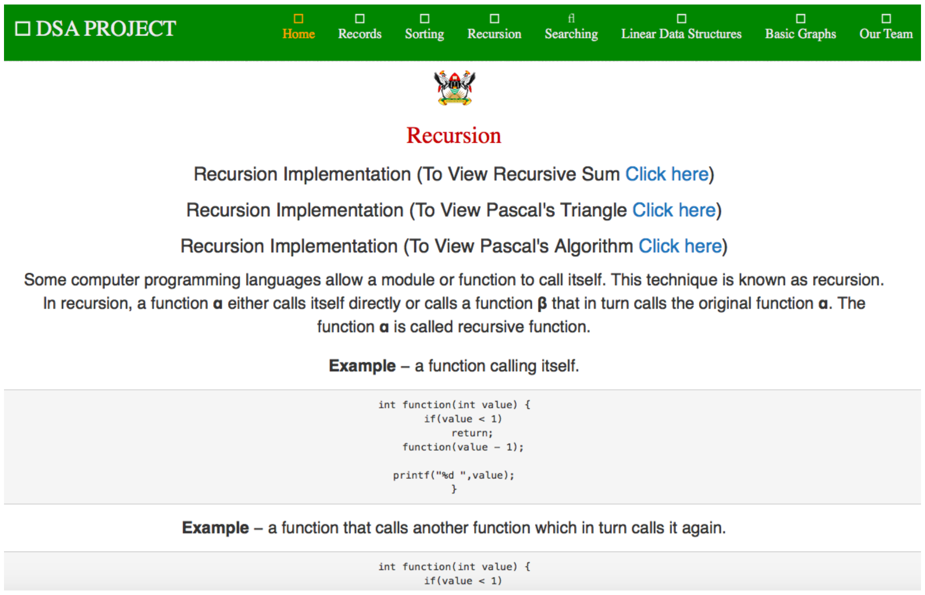
References to help you in your study of Data Structures & Algorithms

1. Tutorials Point
2. Coursera
3. Muele
4. Edx DSA Course
5. W3Schools

Supervisors

- Byansi David
- Kiwanuka Fred

fig 7



Recursion

Recursion Implementation (To View Recursive Sum Click here)

Recursion Implementation (To View Pascal's Triangle Click here)

Recursion Implementation (To View Pascal's Algorithm Click here)

Some computer programming languages allow a module or function to call itself. This technique is known as recursion. In recursion, a function **α** either calls itself directly or calls a function **β** that in turn calls the original function **α**. The function **α** is called recursive function.

Example – a function calling itself.

```
int function(int value) {
      if(value < 1)
         return;
   function(value - 1);

  printf("%d ",value);
         }
```

Example – a function that calls another function which in turn calls it again.

```
int function(int value) {
      if(value < 1)
```

fig 8

9 Conclusion

There are many factors which can influence the success of learning objects as support for student learning in university settings. Students will often need to be motivated and encouraged to use the independent learning resources. Also, it is unclear as to what forms of learning support best facilitate their use.

[19]However, it is clear from the papers considered in this review that when designing tutoring systems, interaction with the system is the most important thing to consider for engaging students in the learning process. These interactions include: the use of multimedia like text supported with aural instructions, animations like the maze representation used in the Vedya tool, and tutorial videos for familiarization with the system.

Making introductory data structures courses interesting and challenging requires projects that motivate students to enhance their programming abilities using the new data structures. This paper has discussed how a project involving competitive gaming motivates students to learn advanced game intelligence programming and improves their opinion of the course overall. The use of competition during an assignment can be used in any course where a suitable project can be developed. The competition server architecture can be used to

enable the competitive environment. The major contribution is that interactive competition during the assignment increases student effort and satisfaction compared with projects where the competition comes after the assignment is completed.

In this paper, we present a visualization tool designed to aid second-year computer science students learn Data Structures and Algorithms.

This tool not only lets students visualize the commonly used data structures, but also allows students to observe the execution of the algorithms. We believe this tool will be an effective supplement to traditional instruction. Because of the time limitation, only the most commonly used data structures are implemented in this version of the software package, which include arrays, stacks, queues, binary search tree, binary heap and priority queue.

[20]This approach will allow users to use their own observable data structures, hence add more flexibility to the software. Another possible future enhancement for the software is to highlight the executing command line of the algorithm. This would help the user to better follow the execution of the algorithm.

References

[1] C. A. Shaffer, "Data structures and algorithm analysis," *Update*, vol. 3, pp. 0–3, 2012.

[2] T. Chen and T. Sobh, "A tool for data structure visualization and user-defined algorithm animation," in *Frontiers in Education Conference, 2001. 31st Annual*, vol. 1. IEEE, 2001, pp. TID–2.

[3] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach.* Malaysia; Pearson Education Limited,, 2016.

[4] T. D. Hendrix, J. H. Cross II, and L. A. Barowski, "An extensible framework for providing dynamic data structure visualizations in a lightweight ide," in *ACM SIGCSE Bulletin*, vol. 36, no. 1. ACM, 2004, pp. 387–391.

[5] D. J. Jarc and M. B. Feldman, "An empirical study of web-based algorithm animation courseware in an ada data structure course," in *ACM SIGAda Ada Letters*, vol. 18, no. 6. ACM, 1998, pp. 68–74.

[6] M. H. Brown and M. A. Najork, "Algorithm animation using 3d interactive graphics," in *Proceedings of the 6th annual ACM symposium on User interface software and technology.* ACM, 1993, pp. 93–100.

[7] G. Gupta, *Introduction to data mining with case studies.* PHI Learning Pvt. Ltd., 2014.

[8] S. Patel, "A literature review on tools for learning data structures," *University of Capetown*, 2014.

[9] A. V. Aho and J. E. Hopcroft, *The design and analysis of computer algorithms.* Pearson Education India, 1974.

[10] C. R. Cook and D. J. Kim, "Best sorting algorithm for nearly sorted lists," *Communications of the ACM*, vol. 23, no. 11, pp. 620–624, 1980.

[11] J. Bennedsen and M. E. Caspersen, "Programming in context: a model-first approach to cs1," in *ACM SIGCSE Bulletin*, vol. 36, no. 1. ACM, 2004, pp. 477–481.

[12] J. C. Trinkle, J.-S. Pang, S. Sudarsky, and G. Lo, "On dynamic multi-rigid-body contact problems with coulomb friction," *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 77, no. 4, pp. 267–279, 1997.

[13] R. Lister, I. Box, B. Morrison, J. Tenenberg, and D. S. Westbrook, "The dimensions of variation in the teaching of data structures," in *ACM SIGCSE Bulletin*, vol. 36, no. 3. ACM, 2004, pp. 92–96.

[14] T. J. Blank, *Folk culture in the digital age: The emergent dynamics of human interaction.* University Press of Colorado, 2012.

[15] K. S. Kumar and K. Syed, "Conceptualization of data structures course using visualization techniques-a case study," *Journal of Engineering Education Transformations*, vol. 30, no. 4, pp. 34–39, 2017.

[16] T. Hammond, S. Valentine, A. Adler, and M. Payton, *The impact of pen and touch technology on education.* Springer, 2015.

[17] N. Myller, R. Bednarik, E. Sutinen, and M. Ben-Ari, "Extending the engagement taxonomy: Software visualization and collaborative learning," *ACM Transactions on Computing Education (TOCE)*, vol. 9, no. 1, p. 7, 2009.

[18] H. J. Ladegaard and W. Cheng, "Conference programme (format a)."

[19] S. De Freitas and M. Oliver, "How can exploratory learning with games and simulations within the curriculum be most effectively evaluated?" *Computers & education*, vol. 46, no. 3, pp. 249–264, 2006.

[20] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.