AP-ASD1: An Indonesian Desktop-based Educational Tool for Basic Data Structure Course

Lucky Christiawan[#1], Oscar Karnalim[#2]

Information Technology, Maranatha Christian University
Bandung, Indonesia
[1]lucky.christiawan@gmail.com
[2]oscar.karnalim@gmail.com

Abstract— **Although there are so many available data structure educational tools, it is quite difficult to find a suitable tool to aid students for learning certain course [1]. Several major impediments in determining the tool are teaching preferences, language barrier, confusing terminologies, internet dependency, various degree of material difficulty, and other environment aspects. In this research, a data structure educational tool called AP-ASD1 is developed based on basic algorithm and data structure course (ASD 1). Since AP-ASD1 is developed following course materials and not vice versa, this educational tool is guaranteed to fit in our needs. The feasibility of AP-ASD1 is evaluated based on two factors which are functionality correctness and survey. All features are correctly functioned and yield expected output whereas survey yields fairly good result (84,305% achievement rate). Based on our survey, AP-ASD1 meets eligibility standard and its features are also successfully integrated. Survey also concludes that this application is also quite effective as a supportive tool for learning basic data structure.**

Keywords—educational tool; algorithm visualization; basic data structure; dynamic algorithm tracking

I. INTRODUCTION

Teaching data structure in computer science class requires several illustrations about how the data structure works in order to improve student's understanding. Illustrations are usually drawn manually on the board, stored as animations in presentation file, or visualized with educational tools [2]. Hand-drawn illustration is the most conventional representation which let lecturer illustrates the flow of data structure by drawing its steps on the board. This mechanism may be the most comfortable one since lecturer can adjust his/her illustration by adding, removing, or updating several specific details based on students' feedbacks. However, this mechanism consumes a lot of time and may yield faulty steps especially when the lecturer is in a rush. Storing illustrations as animations in presentation tools (e.g. Microsoft PowerPoint or Prezi) is a better approach since it is more efficient in terms of time and its steps are definitely error-free. However, most animations in presentation tools are built hardcoded which is quite difficult to incorporate dynamic test cases.

There is a long tradition among Computer Science lecturers of using educational tool software to teach algorithms since it may handle dynamic test cases efficiently in terms of time (lecturer does not require to write/draw algorithm steps on board) [3, 4, 5]. Although many data structure educational tools have been developed and can be easily found on the internet (e.g. AlgoViz [6], VisuAlgo [7] and OpenDSA [8]), these tools may have different terminologies with class material since each university may have its teaching preferences due to student characteristics. Language used in educational tools also affects student understanding since some of them are only fluent in their native language and forcing them to learn foreign language and data structure at the same time may lower their chance to understand.

In this research, an Indonesian desktop-based data structure educational tool has been developed based on Algorithm and Data Structure 1 course. This tool is called AP-ASD1 (Aplikasi Pembelajaran Algoritma dan Struktur Data 1) which is an acronym of "Educational tool for Algorithm and Data Structure 1 course" in Indonesian language. Algorithm and Data Structure 1 (ASD 1) is a computer science course in Maranatha Christian University which consists of basic data structure materials such as stack, queue, linked list, and priority queue. To fulfill our needs, this tool is developed as Indonesian-language desktop-based application with no internet requirement. It also has several additional features such as dynamic algorithm tracking mechanism and embedded course materials.

II. RELATED WORKS

A. AlgoViz

AlgoViz which is the successor of AlgoViz Wiki [9] is an algorithm visualization (AV) portal that contains links to various AV websites [6]. It is intended to connect educators, developers, and users for AVs. This portal homepage can be seen in Figure 1. It is built on Drupal [10] (an open-source content management system written in PHP) and can be accessed from http://algoviz.org/.

AlgoViz portal does not only enlist all available AV websites but also provides its quality assessment information such as ranking given by catalog editor, user rating, description, review information, field reports, and user comments [11]. These quality assessments can be used as user

consideration before exploring the target site. Since this portal is greatly depends on AV community, several mechanism are applied in order to keep the members in touch such as AlgoViz awards, forum, email notification, RSS feeds, and social-media-based notifications.

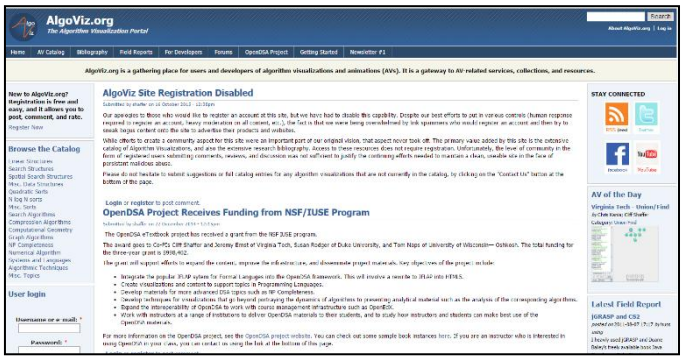

Figure 1. AlgoViz Homepage

B. *VisuAlgo*

VisuAlgo is a web-based educational tool which visualizing algorithm and data structure through animation [7, 12]. This website was conceptualized by Steven Halim in 2011 with the aims of improving teaching of data structures and algorithms through dynamic interactive visualization. VisuAlgo can be accessed from http://visualgo.net/ where its homepage can be seen in Figure 2. This website is utilized in several computer science class at National University of Singapore (NUS) and covers various algorithm and data structure materials such as sorting, bitmask, linked list, hash table, binary heap, binary search tree, graph, segment tree, and suffix tree.

Figure 2. VisuAlgo Homepage

VisuAlgo improves learning and teaching of algorithms in the following ways [2]:

1. VisuAlgo visualizes algorithm flow through animation and highlighted pseudocode which improves student understanding. These steps can be repeated as many as needed so that students can focus on algorithm part that has not been understood.
2. VisuAlgo applies dynamic visualization which let students use their own examples and see the result.
3. VisuAlgo covers various algorithm and data structure so that students have a "one-stop" place for all their learning needs.

C. *OpenDSA*

OpenDSA aims to build a complete open-source, online eTextbook for algorithm and data structure courses that integrates textbook text with a rich collection of interactive activities (including algorithm visualizations) [8, 13, 14, 15]. OpenDSA can be accessed from http://algoviz.org/OpenDSA/ (this tool is registered in AlgoViz portal) and its homepage can be seen in Figure 3. This website is built with HTML 5 technology which enables course modules to be available for use in many modern browsers with no additional plugins or software needed.

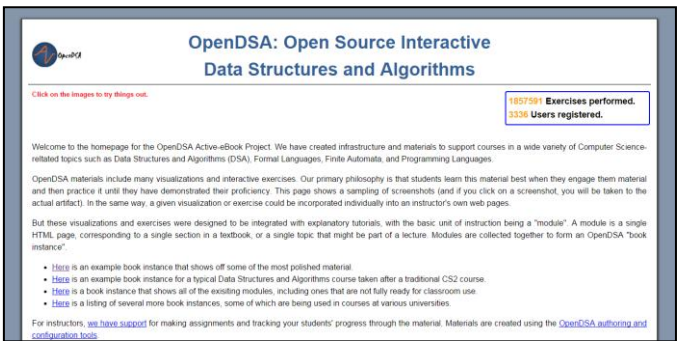

Figure 3. OpenDSA Homepage

OpenDSA is designed with the following educational goals [16]:

1. To become a leverage technology that present course material in a way that makes learning easier.
2. Engaging and motivating students through interactive activities.
3. To evaluate student performance easier through exercises and self-assessments.

A sample of eTextbook generated by OpenDSA can be seen in Figure 4 [8]. Each eTextbook is formed as a series of HTML pages which consists of algorithm and data structure materials [16]. Lecturer for a given course have control over the content, such as reuse, alter, remove, or add existing materials on it. Each material is converted as an interactive module which algorithm visualization is supported by JSAV, a JavaScript algorithm visualization library [17]. It also may consist of interactive activities such as interactive exercises, algorithmic calculator, and performance simulator [16]. Exercises can be conducted in several ways which are simple questions, proficiency exercises, and programming exercises. Proficiency exercises aim to verify student understanding about how a given algorithm works by requiring them to simulate its behavior whereas programming exercises aim to verify student understanding about how to implement an algorithm or using a certain data structure. Several minor features are also embedded such as algorithmic calculator and performance simulator. Algorithmic calculator is conducted to calculate

complex part such as hashing function whereas performance simulator is conducted to "measure" how many processes needed for executing an algorithm.

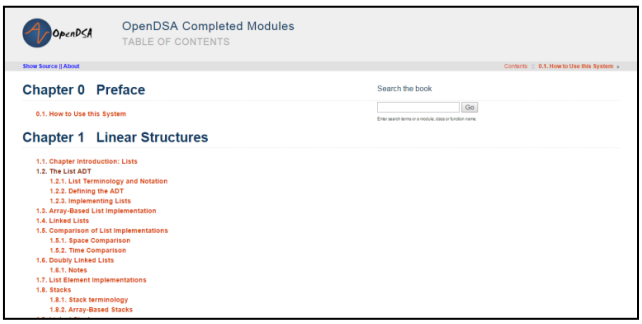

Figure 4. A Sample of E-Textbook Generated by OpenDSA

D. ViLLE

ViLLE is a desktop-based program educational tool for teaching programming to novice programmer where student can see the dynamic behavior of program execution through variable changes between each instruction, function calls through stack view, and even the description of certain code line [18]. Function call through stack view is utilized to let students understand recursion whereas code line description is required to tell specific detail of certain line (which is frequently used in advanced algorithm). ViLLE sample user interface can be seen in Figure 5. Current code line is highlighted whereas user gains flexible control of the visualization (both forwards and backwards).

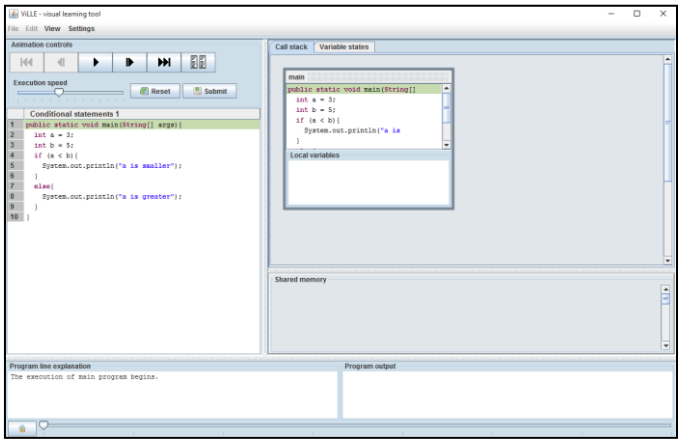

Figure 5. ViLLE Sample User Interface

ViLLE also has additional benefits such as language-independency, comparing execution of two programming languages, pop-up questions, and breakpoints [18]. Language-independency let instructor to utilize ViLLE in many programming languages by defining its equivalence with Java syntax. Comparing execution of two programming language is visualized as two parallel code editor with different highlight mark but points to similar semantic steps. This feature is sometimes needed when students are required to know translation between two programming language. Pop-up questions are designed to maintain student interaction towards the application. It also helps the lecturer to measure student understanding about the code. Breakpoints are utilized to let this application run similar to standard debugger.

E. JHAVE

JHAVE (Java-Hosted Algorithm Visualization Environment) is a support environment for AV systems where users can design their AV on it [19]. It provides several user benefits such as a standard set of VCR-like controls, graphical representation, information and pseudo-code window, input generator, pop-up question mechanism, and even several supportive content generation tool. Sample user interface of AV developed in JHAVE can be seen in Figure 6. Lecturers may design their AV including what information is represented during that phase. JHAVE is designed in client-server architecture for dynamic material update.

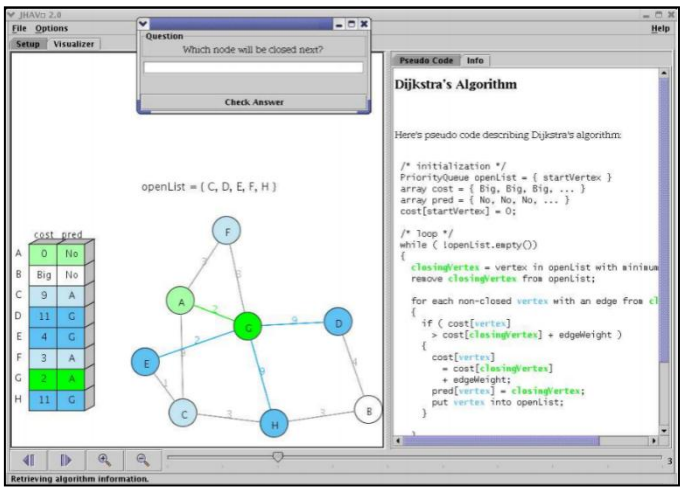

Figure 6. JHAVE Sample User Interface

F. Research Contribution

Feature comparison of existing CS educational tools and AP-ASD1 can be seen in Table I. AV represents AlgoViz, VA represents VisuAlgo, OD represents OpenDSA, V represents ViLLE, JH represents JHAVE, and AP represents AP-ASD1. Question symbol ("?") represents uncertainty where the existence of that feature is greatly depend on its AV designer.

TABLE I
FEATURE COMPARISON OF EXISTING CS EDUCATIONAL TOOLS AND AP-ASD1

Feature	AV	VA	OD	V	JH	AP
Algorithm and Data Structure 1 course-material compability	×	×	×	×	×	√
Consistent UI	√	×	√	√	√	√
Consistent terminologies	√	×	×	√	×	√

Feature	AV	VA	OD	V	JH	AP
Python algorithm tracking	×	?	×	×	×	√
Detailed data structure explanation	√	?	√	×	?	√
No-internet access dependency	×	×	×	√	×	√
Data structure visualization	√	√	√	×	√	√

Since AP-ASD1 is developed following course materials and not vice versa, this educational tool is guaranteed to fit in our needs. As we know, it is quite difficult to find and integrate suitable data structure educational tools in a course [1]. Although AlgoViz has access to various AV websites, it cannot be simply integrated in course materials since each AV websites has different UI and terminologies. Inconsistent UI and terminologies may lower student understanding since they need to adapt on each AV website. AP-ASD1 is quite similar to VisuAlgo except that it has two additional features which are fully synchronized with ASD 1 course materials. They are synchronized Python algorithm tracking and detailed data structure description. Synchronized Python algorithm tracking is utilized for easy adaptation with algorithm language (Python) whereas synchronized detailed data structure description is required to assist students while learning.

OpenDSA may fulfill our requirements but it requires internet access which is not suitable in Indonesia. Internet access in Indonesia is still relatively slow and its coverage is limited. To overcome this impediment, AP-ASD1 is developed as standalone application which does not rely on internet connection. Furthermore, since OpenDSA is a web based application which has browser-dependent UI, OpenDSA may yield inconsistent output in several minor browsers (this UI problem is also a reason why AP-ASD1 is built as desktop application). Although both ViLLE and AP-ASD1 do not rely on internet access, ViLLE only handles algorithm tracking and does not integrate data structure visualization. JHAVE also have similar impediment with other internet-based application to fit in our case which is internet access requirement.

In this research, an Indonesian-language desktop-based data structure educational tool named AP-ASD1 has been developed. It is fully synchronized with ASD 1 course materials in Maranatha Christian University which guarantees the fulfillment of our course needs. Several main features such as data structure visualization, flexible input, algorithm tracking, instruction log, detailed data structure description, and tutorial are also embedded. Furthermore, all instructions and descriptions in AP-ASD1 are represented in Indonesian language to overcome language barrier.

III. Analysis, Design, and Implementation

A. Feature Analysis

Based on our course needs and implementation environments, AP-ASD1 is built with these following features:

1. All instructions and descriptions in AP-ASD1 is represented in Indonesian language to overcome language barrier. Most AP-ASD1 users are Indonesian student whose native language is Indonesian.
2. Data structure terminologies are synchronized with course materials to avoid confusion through various terminologies. As we know, several terms may refer to the same thing (e.g. node and vertex).
3. Data structures are visualized and animated through graphical representation to improve student understanding.
4. AP-ASD1 is featured with dynamic algorithm tracking to show how the algorithm works. Variable states and instruction description for each execution are also shown to clarify its execution sequences.
5. Detailed data structure materials are embedded in AP-ASD1 as an additional feature to help students recall course materials.
6. AP-ASD1 require no internet access since internet access in Indonesia is still limited and relatively slow.
7. AP-ASD1 is developed as desktop-based application to yield consistent UI and functionality. Web-based application may not work properly since it depends on browsers as its interpreter (especially on minor browsers).
8. AP-ASD1 have consistent UI between modules for ease of use.

Since AP-ASD1 aims to improve student understanding about basic data structure in ASD 1 course, AP-ASD1 modules are based on ASD 1 course materials which correlation can be seen in Table II. All data-structure-related course materials except array are implemented. Array is excluded since it is only revisited in ASD 1 for learning abstract data type (in our curriculum, array is also introduced in basic programming course which must be taken before ASD 1). Several minor data structures are also excluded such as queue in naive real world array implementation, queue in moving head array implementation, and circular linked list. These data structure are excluded since they are not efficient and seldom used in real programming tasks.

TABLE II
THE CORRELATION BETWEEN ASD 1 COURSE MATERIALS AND AP-ASD1 MODULES

Course Material	AP-ASD1 Module
Introduction about data structure	-
Array : Revisited	-
Stack (array)	Stack in array implementation
Queue (array)	Queue in circular array implementation
Standard linked list	Standard linked list
Stack and queue in linked list implementation	Stack in linked list implementation Queue in linked list implementation

Course Material	AP-ASD1 Module
Priority queue	Priority queue in linked list implementation
Linked list variations	List that record head and tail Double-pointer linked list
Recursion	-
Advance Sorting	-

B. *Implementation*

The main window of AP-ASD1 can be seen in Figure 7 whereas its view when visualizing an operation of data structure can be seen in Figure 8. It consists of several components which are module selection, current module bar, module operation panel, visualization panel, algorithm panel, instruction log, and operation history. They are represented in Figure 7 as A to G respectively.

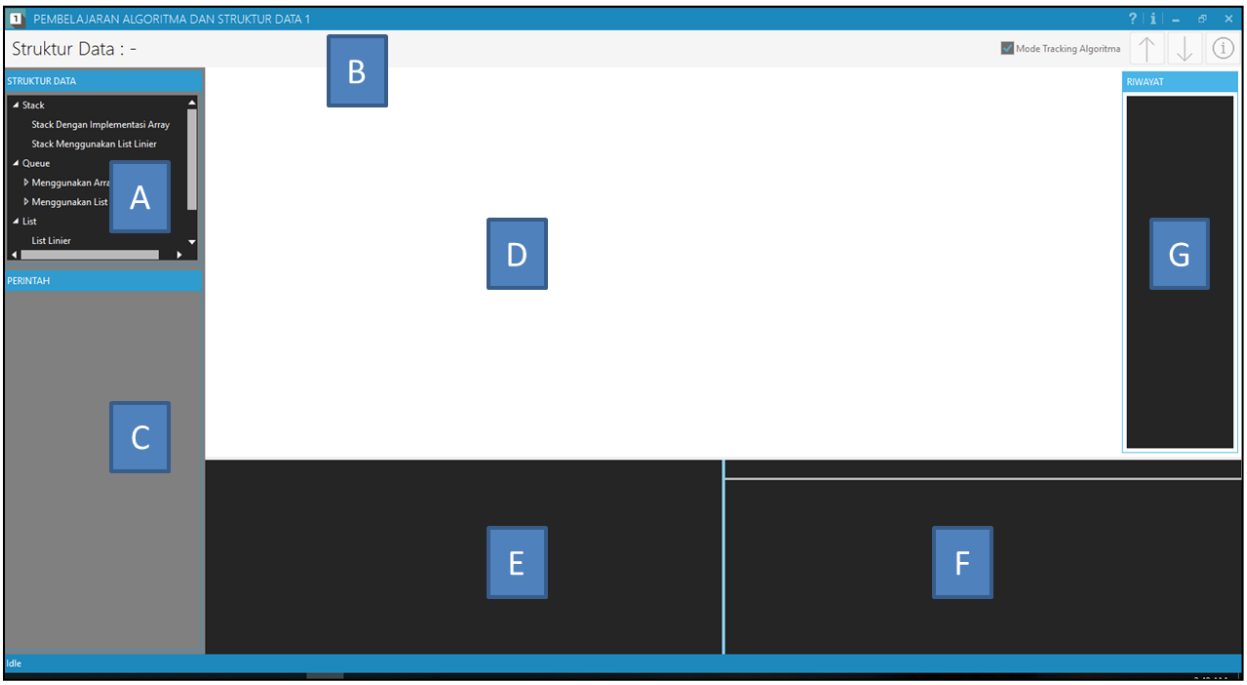

Figure 7 AP-ASD1 Main Window

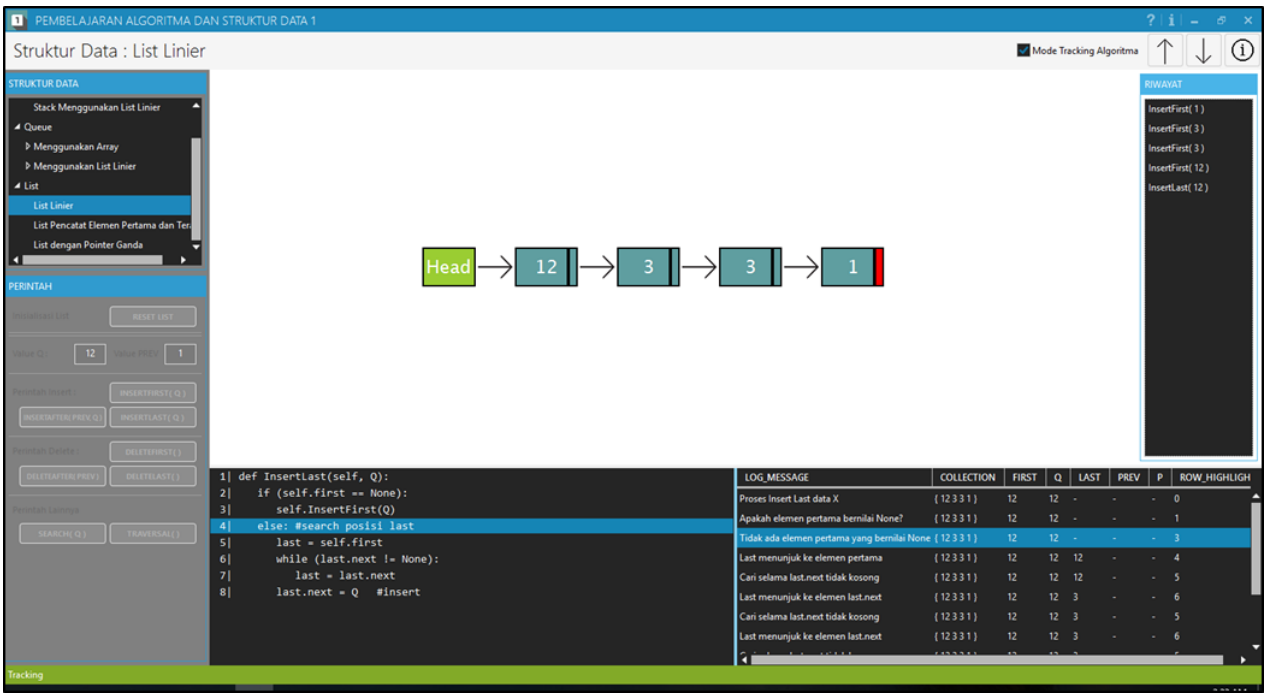

Figure 8. AP-ASD1 Main Window when Visualizing Insert Last on Standard Linked List

Module selection panel which can be seen in Figure 9 enables user to select course modules. To simplify module selection, modules are categorized based on its data structure name and grouped using tree view. User can expand or shrink each menu by clicking it. Current module bar consists of several information and setting about the current module. Beside showing current course title, this bar is also featured with tracking mode checkbox, next and prev button for algorithm tracking, and detail button. Tracking mode checkbox is checked if user want to see algorithm tracking and its algorithm tracking can be controlled through next and prev buttons. Detail button let user access detailed data structure material of the current module which example view can be seen in Figure 10. Each module is described based on course materials and featured with relevant graphics.

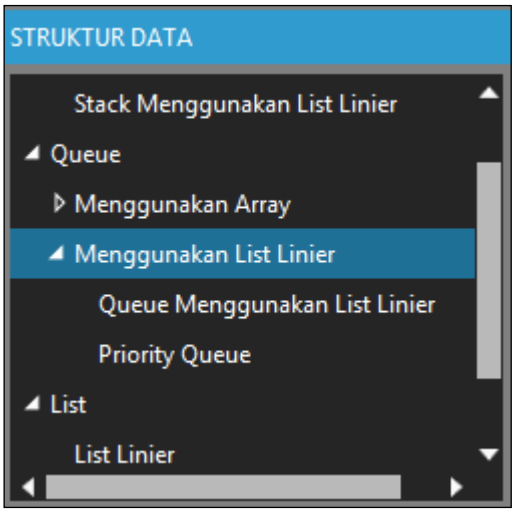

Figure 9. Module Selection Panel

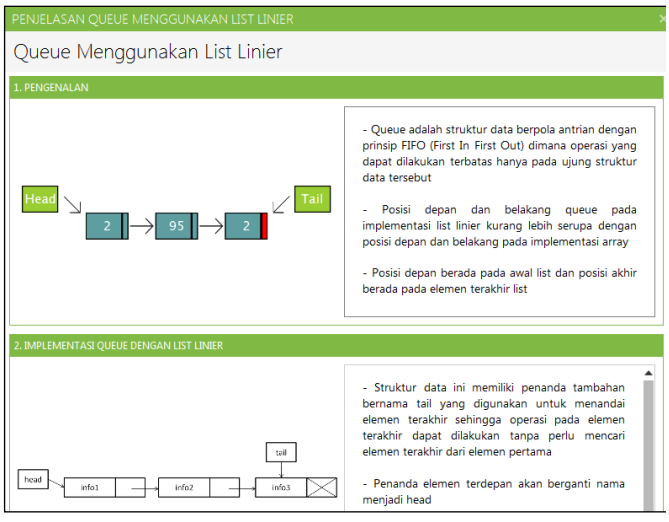

Figure 10. Detailed Data Structure Material Panel Example: Queue in Linked List Implementation

User may instruct current data structure to do some operations by utilizing module operation panel. The example of module operation panel can be seen in Figure 11. User can choose any operation by clicking selected button and give several input needed in given textfields. Standard textfields are used as input mechanism to keep AP-ASD1 as simple as possible which let user prepare all the input before executing an operation. Each operation affects visualization panel, algorithm panel, instruction log, and operation history.

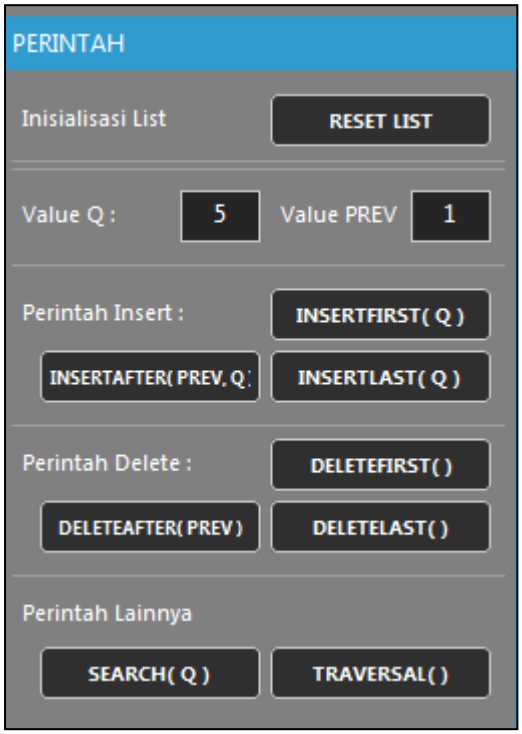

Figure 11. The Example of Module Operation Panel: Standard Linked List

The example of data structure visualization on visualization panel can be seen in Figure 12. All graphical representations are based on course materials to maintain consistency between AP-ASD1 and course materials. Before animation of data structure operation occurs, step by step of algorithm tracking is presented in algorithm panel. Current line is highlighted and all of its variables are recorded in variable log. The example of algorithm panel can be seen in Figure 13 whereas its instruction log can be seen in Figure 14. User can go forward and backward for each instruction to enable user check operation flow based on algorithm given and instruction log. The movement of current instruction in operation flow is controlled using keyboard key (enter and backspace) or current module bar's control buttons. Instruction log stores not only variables but also descriptive information about each instruction execution. All instruction execution log is recorded in instruction log as a list to enable user see variable changes between instructions. Animation and algorithm tracking are

automatically skipped if user have already unchecked tracking mode checkbox in current module bar. Each operation instructed by user are recorded in history which example can be seen in Figure 15. History panel is expected to help user remember which operations have been executed to yield certain state of data structure (User can see the state of data structure in data structure visualization panel).

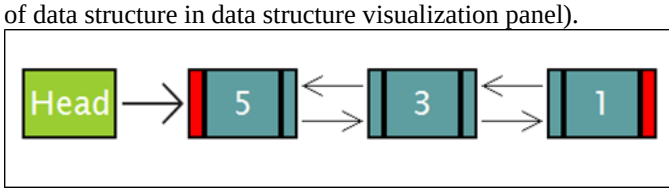

Figure 12. The Example of Data Structure Visualization: Double-Pointer Linked List

```
1| def DeleteLast(self):
2|     Q = self.tail
3|     if (self.head == self.tail)
4|         self.head = None
5|         self.tail = None
6|     else:
7|         prevLast = self.head
8|         while (prevlast.next != self.tail):
9|             prevlast = prevlast.next
10|        prevlast.next = None
11|        self.tail = prevlast
12|    return Q.info
```

Figure 13. The Example of Algorithm Panel: Delete Last on List that Record Head and Tail

LOG_MESSAGE	COLLECTION	FIRST	Q	LAST	PREV	P	ROW_HIGHLIGHT
Proses Delete Last pada list	{1 1}	1	-	1	-	-	0
Q menunjuk ke self.tail	{1 1}	1	1	1	-	-	1
Apakah head dan tail node yang sama	{1 1}	1	1	1	-	-	2
Head dan tail bukan node yang sama	{1 1}	1	1	1	-	-	5
Node prev menunjuk ke head	{1 1}	1	1	1	1	-	6
Cari selama prev.nex t tidak sama dengan tail	{1 1}	1	1	1	1	-	7
Prev.next diisi None	{1 1}	1	1	1	1	-	9
Last menunjuk ke node prev	{1}	1	1	1	1	-	10
Node Q berhasil dihapus	{1}	1	1	1	1	-	11

Figure 14. The Example of Instruction Log: Delete Last on List that Record Head and Tail

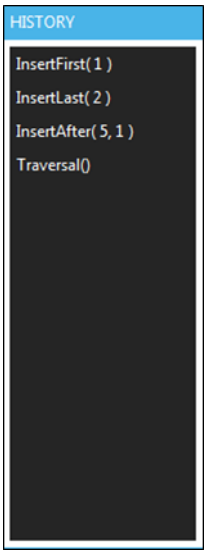

Figure 15. The Example Operation History: Standard Linked List

C. *Dynamic Algorithm Tracking Implementation*

Since algorithm tracking is generated based on operation flow and data structure state, it is quite tricky to enable user to move forward and backward through each instruction during the execution of an operation. However, this mechanism can be implemented using pre-calculated list approach. Algorithm is translated to "embedded" development code which stores states for each instruction execution in a list. Therefore, algorithm tracking can be conducted by traversing each element in the generated list.

Selected algorithm can be translated to "Embedded" development code by following several steps described in Figure 16. Each line in selected algorithm is translated to target source code with one-to-one line equivalence where line i in algorithm is equivalent with line i in target source code. Log storage initialization instruction is embedded at the beginning of target source code whereas logger instruction is embedded at the end of each target source code's line. Log records all useful information such as data structure state, variable state, line of code, and natural language description about that line. In our case, algorithm is written in Python since Python is both algorithmic and implementation language in ASD 1 course. It is translated to C# because C# is AP-ASD1 development programming language.

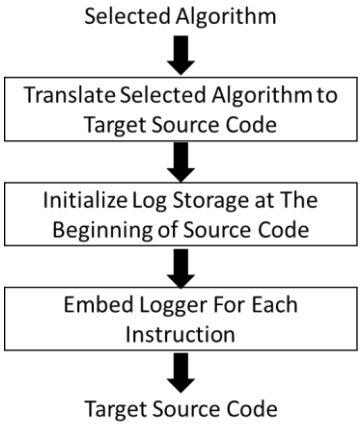

Figure 16. Translation Steps from Algorithm to "Embedded" Development Code

Before algorithm tracking starts, selected algorithm which had been translated into "embedded" source code is executed and its log is stored as a list. The movement between each instruction is implemented as a traversal between elements in list. As mentioned before, log consists of several information such as data structure state, variable state, natural language description, and line of code. Variable state and natural language description are stored and shown as a list in instruction log panel whereas line of code is used to determine which line is highlighted in algorithm panel. Data structure state is used to modify

current data structure state in visualization panel and memory.

IV. EVALUATION

The feasibility of AP-ASD1 is evaluated based on two factors which are functionality correctness and survey. Functionality correctness is measured using black-box testing whereas survey is conducted by distributing questionnaires on ASD 1 students which have used AP-ASD1 in the class.

A. Black-box Testing

Black-box testing is conducted to validate functionality correctness of AP-ASD1 features. A feature is considered correctly functioned if and only if that feature produces similar result with its expected result. Based on our evaluation, All AP-ASD1 features are correctly functioned and yield expected results. Validated AP-ASD1 features are categorized as follows:

1. Module change through module selection It also includes content consistency in current module bar, module operation panel, visualization panel, instruction log, and operation history.
2. Data structure state and visualization (including animation sequence) for each operation.
3. The flow of dynamic algorithm tracking which involves algorithm panel and instruction log.
4. Miscellaneous form components which are not evaluated in other categories.

B. Survey

Survey aims to measure AP-ASD1 subjective effectiveness from the user perspective. It is conducted by asking ASD 1 students to answer several questions based on their experience toward AP-ASD1. These questions are divided into two parts which are standard effectiveness measurement questions and feature-impact measurement questions. Each question in both parts are marked with unique ID to be easily referred in other sections. Students can answer each question with a number ranged from 0 to 5 inclusively where 0 means this objective is not achieved at all and 5 means it is completely achieved.

Standard effectiveness measurement questions can be seen in Table III. All questions given in Table III are generated based on general measurement objectives which is used to measure AP-ASD1 in general. Beside general aspects, several major features are also evaluated to validate its impact through feature-impact measurement questions. Feature-impact measurement questions which are shown at Table IV are generated based on AP-ASD1 major features in one-to-one correlation. For clarity, major features are features which described explicitly in analysis section (Indonesian-language instruction and description in AP-ASD1, course-synchronized data structure terminologies, data structure visualization, dynamic algorithm tracking, detailed data structure materials, no dependence to internet access, desktop platform for consistent UI and functionality, and consistent UI for ease of use).

TABLE III
STANDARD EFFECTIVENESS MEASUREMENT QUESTION

Question ID	Measurement Objective	Question
A1	Ease of use	How easy can you use AP-ASD1?
A2	Intuitive UI	How intuitive is the UI of AP-ASD1?
A3	UI and functionality consistency	How consistent are all UI components and functionalities between modules in AP-ASD1?
A4	User Convenience	How convenient is AP-ASD1 in terms of use?
A5	Declarative tutorial	How declarative is application tutorial given in AP-ASD1?
A6	Course material synchronization	How good is AP-ASD1 in sync with data-structure-related ASD 1 course materials?
A7	Learning impact	How helpful is AP-ASD1 to learn basic data structure, especially in ASD 1 course?

Survey are given to students in ASD 1 class for short odd semester of academic year 2015/2016. There are 15 students of 2015 which take ASD 1 course for the first time. They are instructed to utilize AP-ASD1 as a supplementary tool to understand basic data structure during ASD 1 course. When students fill up the survey, they are instructed for not writing their identity on survey page. This mechanism is conducted to let they measure the impact of AP-ASD1 objectively. All questions and instructions in survey are translated in Indonesian language due to language problem (most student only understand their native language which is Indonesian language). Survey was given at the end of semester when all materials have been taught.

TABLE IV
FEATURE-IMPACT MEASUREMENT QUESTION

Question ID	Feature	Question
B1	Indonesian-language instruction and description in AP-ASD1	How good do you understand the semantic of all instructions and descriptions given in AP-ASD1?
B2	Course-synchronized data structure terminologies	How similar data structure terminologies used in AP-ASD1 with course material?
B3	Data structure visualization	How helpful is data structure visualization to understand certain data structure?
B4	Dynamic algorithm tracking	How helpful is dynamic algorithm tracking to understand the algorithm of data structure operation?
B5	Detailed data structure materials	How declarative is detailed data structure material in AP-ASD1?

Question ID	Feature	Question
B6	No dependence to Internet access	How adaptive is AP-ASD1 in different signal level of Internet connection?
B7	Desktop platform for consistent UI and functionality	How big the impact of desktop platform for consistent UI and functionality in AP-ASD1?
B8	Consistent UI for ease of use	How big the impact of consistent UI for "ease of use" aspect?

The statistic of our survey can be seen in Table V which are considered as a good result since most objectives are rated 4 or more (nearly completely achieved). Average values are resulted from averaging all scores for selected question whereas mean average is resulted from averaging all average values. Although A4 is an exceptional case since a student did not answer it on his/her survey, its average value still can be calculated based on remaining responders.

In Table V, lowest average value is red-marked and its highest one is blue-marked. A2 has the lowest objective rate since several students found the UI of AP-ASD1 is less intuitive. They expect more detailed explanation in tool tip and algorithmic information. Furthermore, dynamic algorithm tracking is the most helpful feature since B4 scores the highest among all survey questions. Based on our responders, we can conclude that most students who experience difficulty when understanding how certain operation works can be aided by algorithm tracking in AP-ASD1. In our survey, a student misinterprets B6 question which guide him/her to rate 0 on that question. This student considers B6 question as an unrelated one since this application does not require internet connection.

AP-ASD1 meets eligibility standard of a published application since it scores 84.463% on standard effectiveness survey (question A1-A7). Furthermore, all major features which are based on ASD 1 course needs and implementation environments has been successfully integrated since it scores 84.167% on feature-impact survey (question B1-B8). Since question A7 yield 4.4 of 5, it can be concluded that AP-ASD1 is quite effective as a supportive tool for learning basic data structure, especially in ASD 1 course.

TABLE V
SURVEY STATISTICS

Question ID	Survey Statistic							
	0	1	2	3	4	5	Average	Average (%)
A1				2	7	6	4,267	85,333
A2			1	2	10	2	3,867	77,333
A3				3	9	3	4,000	80,000
A4					8	6	4,429	88,571
A5				3	6	6	4,200	84,000
A6				2	5	8	4,400	88,000
A7			1	1	4	9	4,400	88,000
B1				3	8	4	4,067	81,333
B2			1	1	7	6	4,200	84,000
B3				3	7	5	4,133	82,667
B4					6	9	4,600	92,000
B5				1	11	3	4,133	82,667
B6	1			2	1	11	4,333	86,667
B7				3	7	5	4,133	82,667
B8				4	6	5	4,067	81,333
Mean Average of Standard Effectiveness Survey							4.223	84.463
Mean Average of Feature Impact Survey							4.208	84.167
Mean Average in General							4,215	84,305

V. CONCLUSIONS

Based on our research, several conclusions can be stated which are:

1. AP-ASD1 is quite effective as a supportive tool for learning basic data structure (especially in ASD 1 course) since learning impact aspect is rated quite high (4.4 of 5)
2. All major features which are based on ASD 1 course needs and implementation environments has been successfully integrated since survey result of feature-impact measurement achieves 84.167%. These feature-based aspects are Indonesian-language instruction and description in AP-ASD1, course-synchronized data structure terminologies, data structure visualization, dynamic algorithm tracking, detailed data structure materials, no dependence to internet access, desktop platform for consistent UI and functionality, and consistent UI for ease of use.
3. AP-ASD1 meets eligibility standard of a published application since it scores 84.463% on standard effectiveness survey. Standard effectiveness survey involves ease of use, intuitive UI, UI and functionality consistency, user convenience, declarative tutorial, course material synchronization, and learning impact.

4. AP-ASD1, an Indonesian-language desktop-based educational tool for basic data structure course has been successfully developed in C# platform.
5. All features on AP-ASD1 are correctly functioned since it yields expected results. This statement is concluded based on black-box testing result which validates module change through module selection, data structure state and visualization for each operation, flow of dynamic algorithm tracking, and miscellaneous form components.

VI. FUTURE WORK

In next research, we will integrate dynamic algorithm typing in AP-ASD1 which let user write his/her algorithm as an operation on certain data structure. User can modify algorithm given in the course and see how his/her modified algorithm works. This feature is expected to improve student understanding based on two facts which are:

1. Several students are low-paced which require lecturer to show how the algorithm works in more detailed implementation.
2. Several students are curious with the implementation of their logic and how unexpected result is produced based on their logic.

Besides dynamic algorithm typing, log filter will be also embedded since several students are quite overwhelmed with complete information at instruction log section. They used to try to understand how algorithm works by only focusing on several less understandable parts.

REFERENCES

[1] C. A. Shaffer and A. J. D. Alon, "SIGCSE 2010 AlgoViz Survey Results," March 2010. [Online]. Available: http://algoviz.org/news/2010-03-25-sigcse-2010-algoviz-survey-results.

[2] S. Halim, Z. C. Koh, V. B. H. Loh and F. Halim, "Learning Algorithms with Unified and Interactive Web-Based Visualization," *Olympiads in Informatics*, vol. 6, pp. 53-68, 2012.

[3] T. L. Naps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger and J. A. Velázquez-Iturbide, "Exploring the role of visualization and engagement in computer science education," in *ITiCSE-WGR '02 Working group reports from ITiCSE on Innovation and technology in computer science education*, New York, 2002.

[4] C. A. Shaffer, M. L. Cooper, A. J. D. Alon, M. Akbar, M. Stewart, S. Ponce and S. H. Edwards, "Algorithm Visualization: The State of the Field," *ACM Transactions on Computing Education (TOCE)*, vol. 10, no. 3, 2010.

[5] E. Fouh, M. Akbar and C. A. Shaffer, "The role of visualization in computer science education," *Computers in the Schools: Interdisciplinary Journal of Practice, Theory, and Applied Research*, vol. 29, no. 1-2, pp. 95-117, 2012.

[6] "AlgoViz.org: The Algorithm Visualization Portal," [Online]. Available: http://algoviz.org/. [Accessed 7 12 2015].

[7] S. Halim, "VisuAlgo," [Online]. Available: http://visualgo.net/. [Accessed 12 5 2015].

[8] "OpenDSA," [Online]. Available: http://algoviz.org/OpenDSA/. [Accessed 5 12 2015].

[9] "Data Structures and Algorithm Visualization Wiki," Virginia Tech, 2006. [Online]. Available: http://algoviz.cs.vt.edu. [Accessed 5 12 2015].

[10] "Drupal," 2010. [Online]. Available: http://drupal.org,.

[11] C. A. Shaffer, M. Akbar, A. J. D. Alon, M. Stewart and S. H. Edwards, "Getting Algorithm Visualizations into the Classroom," in *Proceedings of the 42nd ACM technical symposium on Computer science education*.

[12] E. T. Y. Ling, Teaching Algorithms with Web-based Technologies, Singapore: B.Comp. Dissertation, Department of Computer Science, School of Computing, National University of Singapore, 2014.

[13] C. Shaffer, V. Karavirta, A. Korhonen and T. Naps, "OpenDSA: Beginning a community hypertextbook project," in *The Eleventh Koli Calling International Conference on Computing Education Research*, Finland, 2011.

[14] C. Shaffer, T. Naps and E. Fouh, "Truly interactive textbooks for computer science education," in *The Sixth Program Visualization Workshop*, Germany, 2011.

[15] S. Hall, E. Fouh, D. Breakiron, M. Elshehaly and C. Shaffer, "Education innovation for data structures and algorithms courses," in *ASEE Annual Conference*, Atlanta, 2013.

[16] E. Fouh, V. Karavirta, D. A. Breakiron, S. Hamouda, S. Hall, T. L. Naps and C. A. Shaffer, "Design and architecture of an interactive eTextbook–The OpenDSA system," *Science of Computer Programming*, vol. 88, pp. 22-40, 2014.

[17] V. Karavirta and C. A. Shaffer, "JSAV: the JavaScript algorithm visualization library," in *The 18th ACM conference on Innovation and technology in computer science education*, New York, 2013.

[18] T. Rajala, M.-J. Laakso, E. Kaila and T. Salakoski, "Effectiveness of Program Visualization: A case study with the ViLLE Tool," *Journal of Information Technology Education: Innovation in Practice*, vol. 7, 2008.

[19] T. L. Naps, "JHAVE: Supporting algorithm visualization," *IEEE on Computer Graphics and Applications*, vol. 25, no. 5, pp. 49-55, 2005.