Journal of Engineering Education Transformations ,
Volume 32 , No. 4, April 2019, ISSN 2349-2473, eISSN 2394-1707

OSAVA: An Android App for Teaching a Course on Operating Systems

Pinaki Chakraborty[1], Udit Arora[1], Namrata Mukhija[2], Vipra Goel[1], Priyanka[2], Siddarth Shikhar[1], Rohit Takhar[1]
[1] Division of Computer Engineering, Netaji Subhas University of Technology, New Delhi 110078, India
[2] Division of Information Technology, Netaji Subhas University of Technology, New Delhi 110078, India
[1]pinaki_chakraborty_163@yahoo.com

Abstract: We have developed an Android app named Operating System Algorithms Visualization App (OSAVA) to visualize different types of algorithms used in operating systems. We have used it to teach a course on operating systems in the Spring semester of 2016, 2017 and 2018. The course was attended by 243 undergraduate students and 84% of them said that OSAVA helped them in understanding the algorithms. The students scored 6% more marks in the exam than the students of the previous year who were taught without the app. We feel that implementing the tool as a mobile app allowed the students to use it during lectures and proved particularly helpful in its integration in the course.

Keywords: Educational software, Android app, operating system, algorithm visualization.

1. Introduction

Operating systems are used in computers, and in other programmable devices like smartphones, because they provide an efficient way of transforming the hardware into usable computing systems. Because of the importance of operating systems, a course on them is included in most undergraduate curricula in computer science. This course is taught in different ways in various universities. Some professors recommend that students study the code of a simple operating system and reengineer parts of the same. This 'build an operating system' approach of teaching the course helps students to know the details of the working of operating systems (Krishnamoorthy, 2002). Accordingly, operating systems purported to be used as an instructional aid, like XINU (Comer, 2015) and MINIX (Tanenbaum and Woodhull, 2006), have been developed. However, many professors prefer to teach the course more traditionally using the 'chalk and board' approach (Desnoyers, 2011). In either case, the course can be enhanced using a visualization tool to illustrate the algorithms.

The Operating System Algorithms Visualization App (OSAVA), as we call it, can visualize different types of algorithms used in operating systems, viz. CPU scheduling algorithms, deadlock avoidance algorithm, deadlock detection algorithm, contiguous memory allocation strategies, page replacement algorithms and disk scheduling algorithms. These algorithms play an important role in operating systems. OSAVA helps students to understand the internal working of operating systems by visualizing these algorithms. OSAVA covers the entire curriculum and can visualize more algorithms than the instructional tools developed earlier to teach the course on operating systems.

Pinaki Chakraborty
Division of Computer Engineering,
Netaji Subhas University of Technology, New Delhi 110078, India
pinaki_chakraborty_163@yahoo.com

2. Related Work

Over the years, professors and, as in some cases, hobbyists have developed tools to visualize and animate the algorithms used in operating systems. Some of these tools can be downloaded from the Web. However, most of them lack academic rigor and experience of using them for teaching have not been reported in the literature. Nevertheless, there exist a few tools to visualize CPU scheduling algorithms and page replacement algorithms that are worth mentioning (Table 1). No satisfactory tool to visualize deadlock handling algorithms and disk scheduling algorithms are available however.

CPU scheduling algorithms are perhaps the most important algorithms used by operating systems. A few sophisticated tools to visualize and teach them have been developed. Khuri and Hsu (1999) developed a tool to visualize the multilevel feedback queue scheduling algorithm. The tool is interactive and has been used to teach at San Jose State University. Students used the tool to study the effects of parameters like time quantum and dispatch latency on the performance of the algorithm. Suranauwarat (2007) developed a tool to visualize first-come first-served, shortest job first, round robin and multilevel feedback queue algorithms for CPU scheduling. The tool supports animation and the visualizations are mostly self-explanatory. The tool takes care to visualize the state of a process and the events that change the state. Fischbach (2013) developed a tool to visualize CPU scheduling algorithms of various difficulty levels from first-come first-served to multilevel feedback queue scheduling. The tool has been used to teach at Widener University where it helped students to study CPU scheduling in detail.

Page replacement algorithms have been also taught successfully in several universities using purpose-built visualization tools. Khuri and Hsu (1999) developed a tool to visualize first-in first-out, optimal, least recently used and second chance page replacement algorithms, and used it to teach at San Jose State University. The tool developed by Fischbach (2013) can also visualize page replacement algorithms like optimal, least recently used and second chance algorithms, and has been used to teach at Widener University. Garmpis (2013) developed a Web-based interactive visualization tool to teach first-in first-out, optimal, least recently used and second chance page replacement algorithms. The tool can generate random reference strings that students may use to simulate the algorithms. The tool has been used to teach at Technological Educational Institution of Messolonghi where students reported that it helped them in their study in a survey conducted at the end of the course.

Table 1. Algorithms visualized by tools developed to teach the course on operating systems.

	CPU Scheduling Algorithms	Page Replacement Algorithms	Platform	Language of Implementation	Important Feature
Khuri and Hsu (1999)	Multilevel feedback queue scheduling algorithm	First-in first-out, optimal, least recently used and second chance algorithms	Desktop -based	Java	Allows stepwise simulation
Suranauwarat (2007)	First-come first-served, shortest job first, round robin and multilevel feedback queue scheduling algorithms	—	Desktop -based	Java	Students can customize the algorithms
Fischbach (2013)	First-come first-served, multilevel feedback queue scheduling, and other algorithms	Optimal, least recently used and second chance algorithms	Desktop -based	Java	Students can design their own algorithms
Garmpis (2013)	—	First-in first-out, optimal, least recently used and second chance algorithms	Web-based	Visual Basic	Can be used to test the knowledge of students
Our tool	First-come first-served, shortest job first, priority, round robin, multilevel queue and multilevel feedback queue scheduling	First-in first-out, optimal, least recently used, second chance, enhanced second chance, least frequently used and most	Mobile app	Python	Can visualize algorithms for deadlock handling and disk scheduling also

Yuan et al. (2008) used algorithm visualization tools to teach the course on operating systems at North Carolina A&T State University and corroborated that visualization tools help students to learn about algorithms used in operating systems. Although we appreciate the visualization tools developed so far to teach the course on operating systems, we think that such tools can be improved in the following ways.

- Increasing the coverage. The available tools can visualize only a few CPU scheduling algorithms and page replacement algorithms. If a tool is able to visualize all the important algorithms taught in the course on operating systems then it will be far more beneficial to students.

- Enhancing ubiquity. The visualization tools developed so far have been implemented as desktop- and web-based applications which students typically use in a lab session or during self-study after a lecture. Alternatively, if a visualization tool is implemented as a mobile app then students can also use it during the lectures, just like they use calculators in mathematics lectures.

- Improving dissemination. The tools developed so far have been made available on the websites of the universities where they were developed (Khuri and Hsu, 1999; Suranauwarat, 2007; Garmpis, 2013). We think that tools should also be disseminated through online software stores and their source code should be made public. The developers should use their tools to teach in classroom and report their empirical findings in the literature.

3. Operating System Algorithms Visualization App

OSAVA visualizes the different types of algorithms used in operating systems as given in Silberschatz et al. (2012). OSAVA has six modules each dedicated to a particular type of algorithm. A module asks the user to select one of the algorithms available in it and enter relevant information. The module displays a brief description of the selected algorithm and then simulates its working. Output is provided typically as a combination of text and illustrations. OSAVA can visualize a total of twenty-four algorithms (Table 2).

A. CPU Scheduling Algorithms

A computer using a multitasking operating system has multiple processes loaded in its memory at a given time. The operating system uses a CPU scheduling algorithm to decide which of those processes will be executed next. OSAVA can visualize the CPU scheduling algorithms commonly used by operating

Table 2. Modules of OSAVA.

Module	Algorithms Visualized	Illustration	Output	Pedagogical Utility
CPU scheduling algorithms	First-come first-served, shortest job first, priority, round robin, multilevel queue and multilevel feedback queue scheduling algorithms	Gantt chart	Turnaround time, waiting time, response time, throughput and CPU utilization	Can be used to solve numerical questions involving large number of processes
Deadlock avoidance algorithm	Banker's algorithm	—	Safe sequence	Can be used to solve numerical questions involving large number of processes and resource types
Deadlock detection algorithm	Variant of Banker's algorithm	—	List of deadlocked processes	Can be used to solve numerical questions involving large number of processes and resource types
Contiguous memory allocation strategies	First fit, best fit and worst fit strategies	Memory map	—	Can be used to solve numerical questions involving large number of processes
Page replacement algorithms	First-in first-out, optimal, least recently used, second chance, enhanced second chance, least frequently used and most frequently used page replacement algorithms	Memory map	Number of page faults and page fault ratio	Can be used to solve numerical questions involving large number of page faults and frames
Disk scheduling algorithms	First-come first-served, shortest seek time first, SCAN, C-SCAN, LOOK and C-LOOK disk scheduling algorithms	Trace of the path followed by read/ write head	Number of cylinders traversed by read/ write head	Can be used to solve numerical questions involving large number of disk accesses

systems. The user has to enter the arrival time and the CPU burst time of the processes. The user then needs to select the CPU scheduling algorithm to be visualized. When shortest job first and priority scheduling algorithms are selected, the user needs to specify whether non-preemptive or preemptive scheduling is to be performed. If priority scheduling is used, then the user also needs to enter the priority of each process. In case of priority scheduling, OSAVA can also visualize the concept of aging in which the priority of a process is incremented every time it waits in the ready queue for a predefined amount of time. If round robin scheduling is used, then the user needs to specify the time quantum. When multilevel queue scheduling and multilevel feedback queue scheduling are selected, then first-come first served and round robin algorithms are used for intra-queue scheduling and preemptive priority scheduling algorithm is used for inter-queue scheduling. Details of the queues are to be entered by the user. Finally, the user has to enter the dispatch latency which is considered to be negligible if the user chooses not to enter a value. OSAVA simulates the selected CPU scheduling algorithm and displays a Gantt chart showing the time intervals when the different processes executed on the CPU. OSAVA then displays a timeline representing

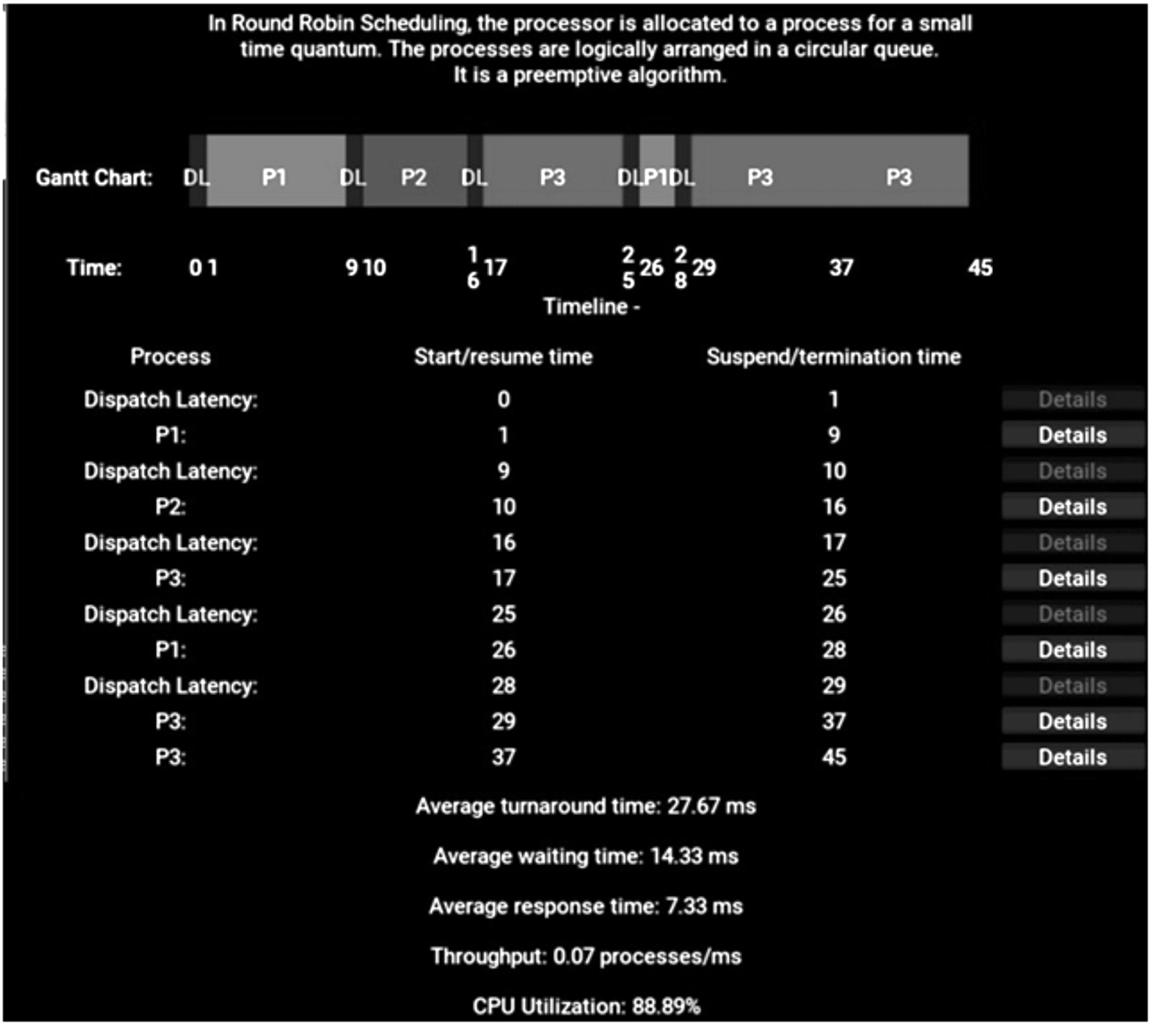

Fig. 1. Sample output of the module to visualize CPU scheduling algorithms. Here the module is visualizing round robin scheduling with a time quantum of 8 ms and dispatch latency of 1 ms.

the execution of the processes. The turnaround time, waiting time and response time of a particular process can be viewed by tapping on the 'Details' button next to it. To calculate the response time of a process, it is assumed that a process generates its first response to the user as soon as it starts executing. The average turnaround time, the average waiting time and the average response time of the processes are then

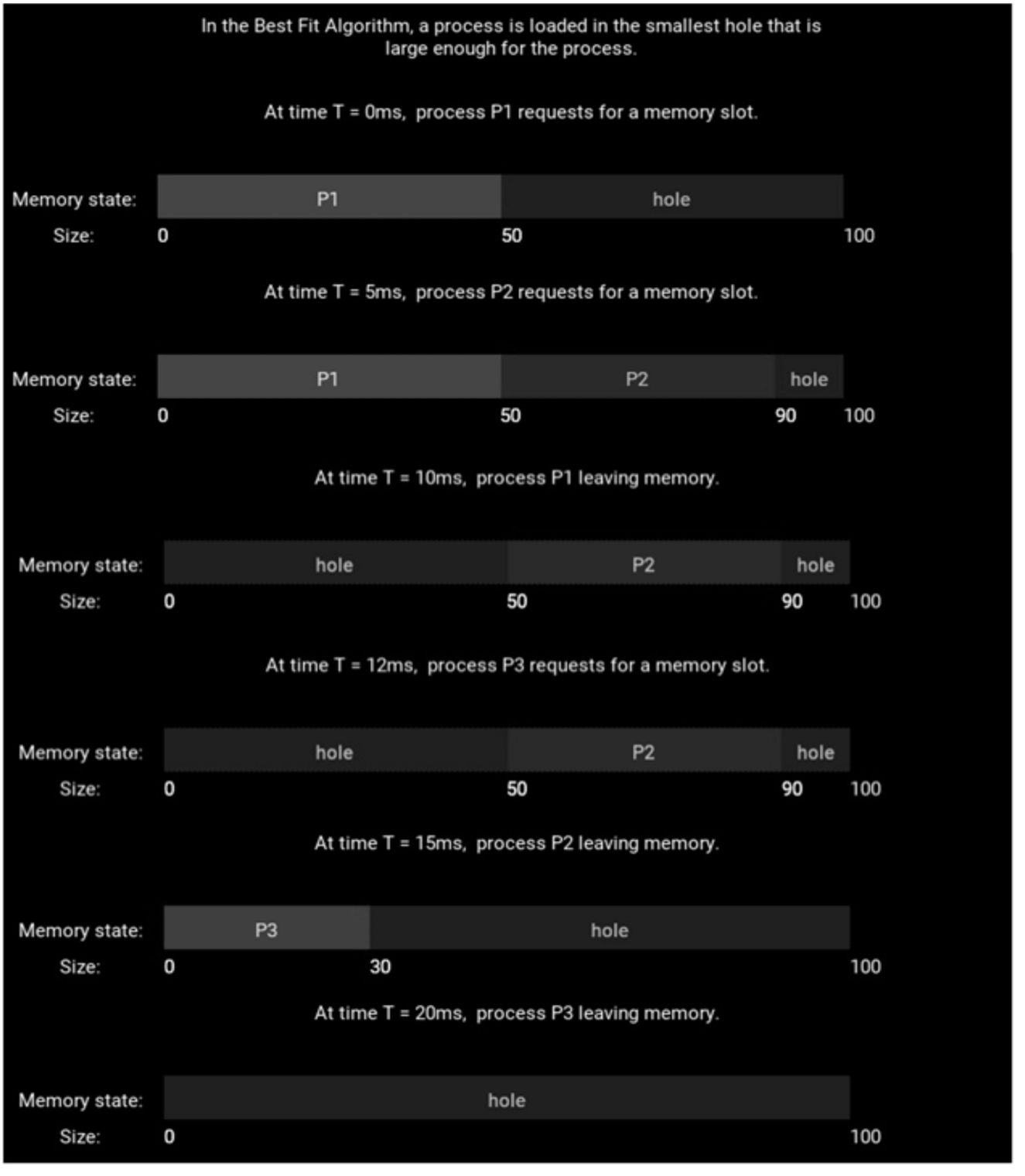

Fig. 2. Sample output of the module to visualize contiguous memory allocation strategies. Here the module is visualizing the best fit strategy.

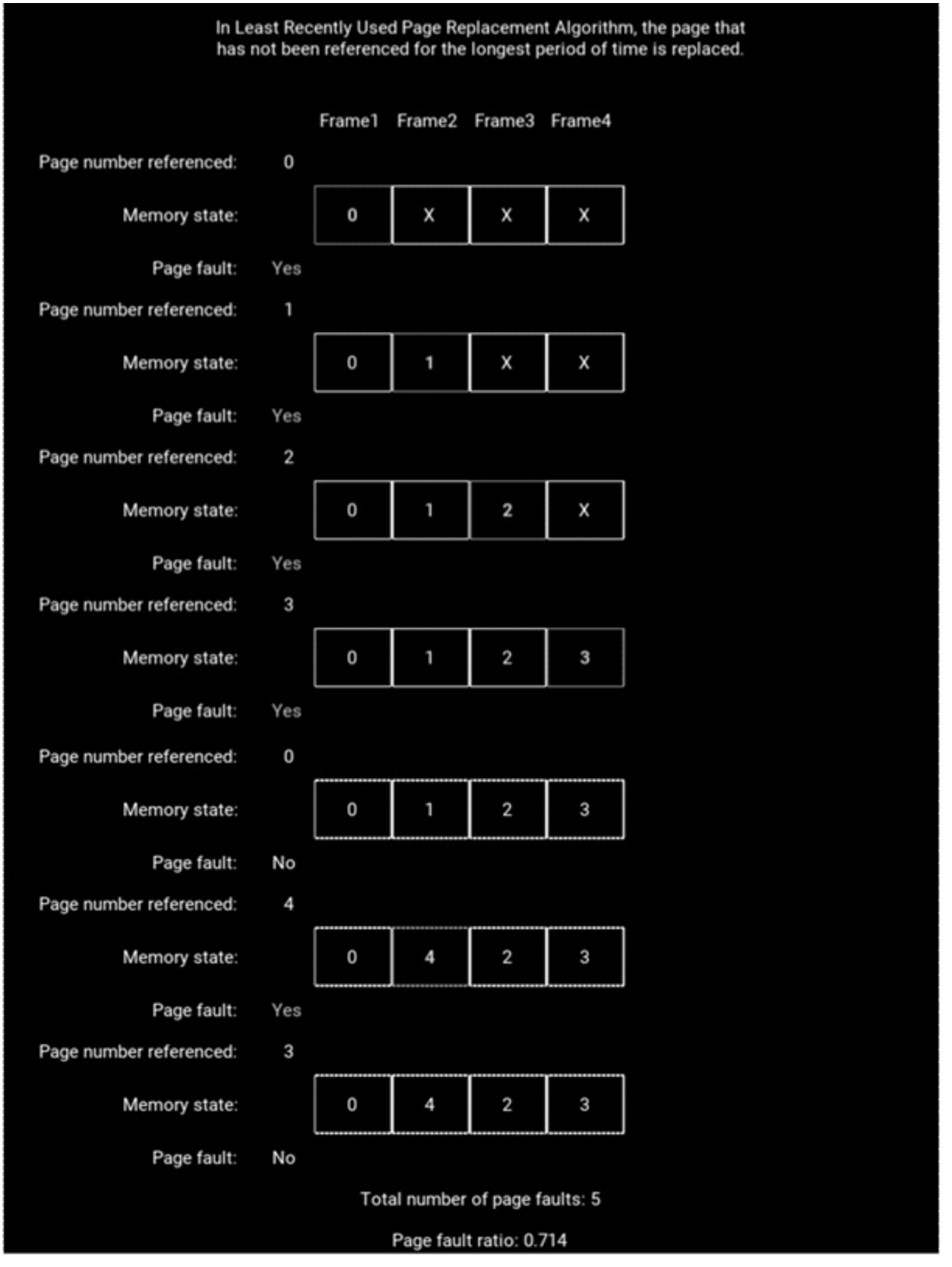

Fig. 3. Sample output of the module to visualize page replacement algorithms. Here the module is visualizing the least recently used algorithm with 4 memory frames allotted to a process.

calculated. The throughput of the system and the CPU utilization are also displayed (Fig. 1). OSAVA displays more information about the processes than the earlier tools (Khuri and Hsu, 1999; Suranauwarat, 2007; Fischbach, 2013) but in a more compact format.

B. Deadlock Handling Algorithms

In context of operating systems, a deadlock is a situation where two or more processes are waiting indefinitely because the resources they have requested for are being held by one another. Operating systems may use three techniques to handle deadlocks, viz. deadlock prevention, deadlock avoidance, and deadlock detection and recovery. The Banker's algorithm is a well known deadlock avoidance technique and can be visualized by OSAVA. The user has to enter the number of resources of different types that have been allocated to the processes and the maximum number of resources of each type that the processes can request for. The user also needs to enter the number of resources of each type that are currently free. Finally, the user needs to specify a process that is now requesting for more resources and enter the number of resources of each type the process is requesting for. It is assumed that the system is initially in a safe state, i.e. there is no possibility of a deadlock. OSAVA uses the resource-request algorithm to determine the state in which the system will be in if the requested resources are granted to the process. Then OSAVA uses the safety algorithm to find a safe sequence, i.e. a sequence in which the processes should be executed so that they can be allocated the resources they need. The matrices and vectors involved in the calculation are displayed to enhance pedagogy.

Instead of using the deadlock avoidance algorithm, operating systems may choose to detect deadlocks as they occur and recover from them. The algorithm to detect deadlocks is similar to the Banker's algorithm and can be visualized by OSAVA as well. The user has to enter the number of resources of different types that have been allocated to the processes and the number of resources of each type that are currently free. The user then needs to enter the number of resources of each type that the processes are now requesting for. OSAVA tries to find a sequence in which the processes can be allocated the resources they are requesting for without causing a deadlock. If such a sequence cannot be found, then OSAVA lists the deadlocked processes. C. Memory Management Algorithms

An operating system may use a contiguous memory allocation scheme wherein a process is loaded in a contiguous block of memory. OSAVA can visualize three common contiguous memory allocation strategies, viz. first fit, best fit and worst fit. The user has to enter the size of the memory, and arrival time, turnaround time and memory requirement of each process. OSAVA then visualizes a memory allocation strategy and displays a map of the memory whenever a process arrives or terminates. Such a memory map shows which parts of the memory are occupied by the different processes and which parts are free (Fig. 2).

Most operating systems use pure demand paging to implement virtual memory. Such operating systems must use a page replacement algorithm to select a page to be swapped out from the memory in case of a page fault. The commonly used page replacement algorithms can be visualized by OSAVA. The user has to enter the number of frames in the memory that have been allocated to a process and a reference string. OSAVA then visualizes the selected page replacement algorithm. It shows which page is residing in which frame after each memory access. OSAVA also calculates the total number of page faults and the page fault ratio (Fig. 3). The user interface of OSAVA is simpler and more intuitive to use than those of the earlier tools (Khuri and Hsu, 1999; Fischbach, 2013; Garmpis, 2013) for visualizing page replacement algorithms.

D. Disk Scheduling Algorithms

A hard disk is used as the secondary storage device of a typical computer. Operating systems use an algorithm to schedule the requests to access the different cylinders in the disk in order to decrease the average access time. The common disk scheduling algorithms can be visualized by OSAVA. The user has to enter the number of cylinders in the disk, the current position and the direction of movement of the read/write head. OSAVA simulates a disk scheduling algorithm, and displays a trace of the path followed by the read/write head and the count of cylinders traversed (Fig. 4).

OSAVA has been implemented in Python. It can be downloaded from the Google Play online app store (https://play.google.com/store/apps/details?id=org.nsit.osava&hl=en) and installed on all mobile phones that use the Android operating system. OSAVA is free

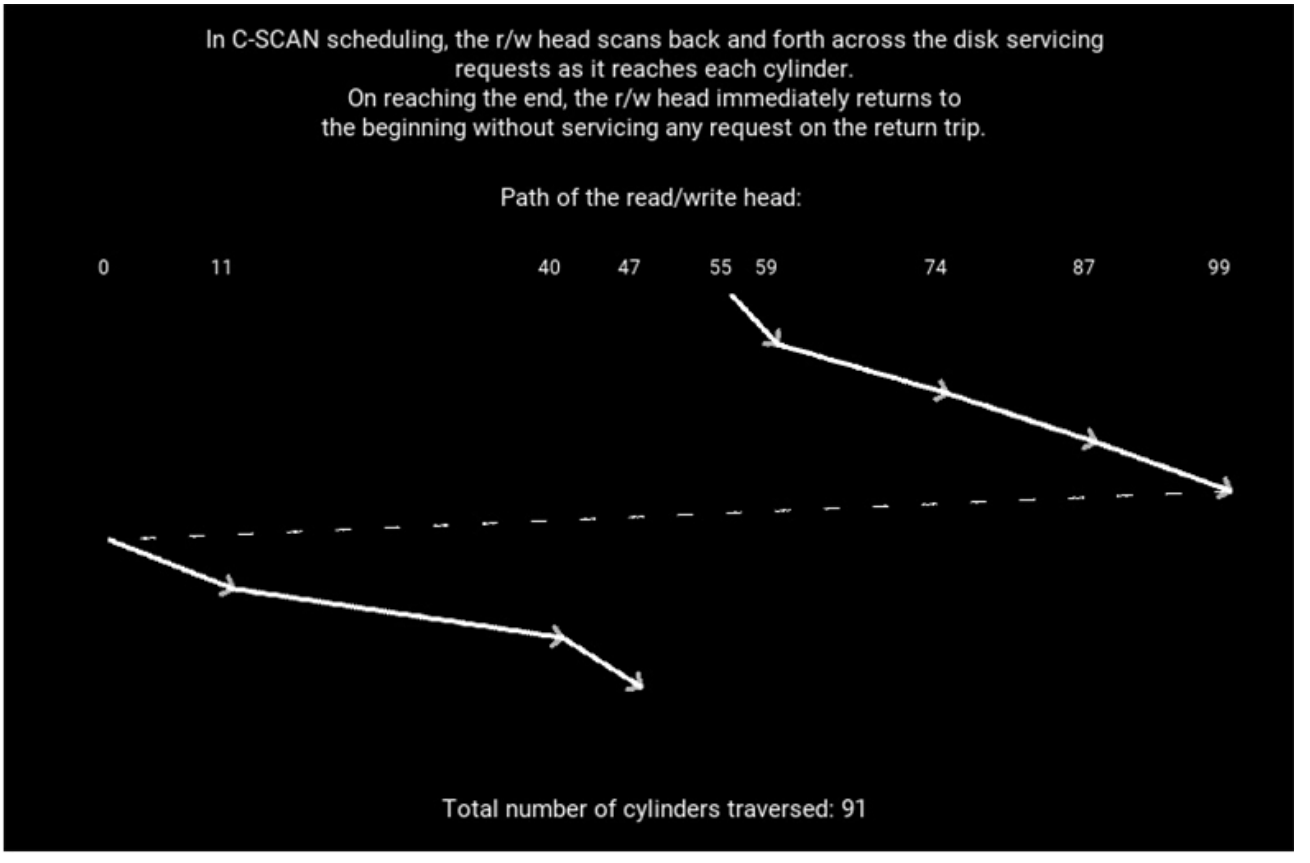

Fig. 4. Sample output of the module to visualize disk scheduling algorithms. Here the module is visualizing the C-SCAN algorithm. The dashed line represents the movement of the read/write head from the innermost cylinder to the outermost cylinder without stopping.

and open-source (https://github.com/osava-nsit/osava), and does not require connecting to the Internet.

4. Instructional Use of OSAVA

We have used OSAVA to teach the course on operating systems in Spring 2016, Spring 2017 and Spring 2018 semesters at Netaji Subhas University of Technology. On the three occasions, the course was attended by 60, 56 and 127 undergraduate students, respectively. OSAVA was distributed among the students at the beginning of the semester. The course was taught using the 'chalk and board' approach. After teaching an algorithm, the instructor (the first author) presented the students with a set of numerical questions based on the algorithm. The students were first made to solve the questions by hand and then cross-check their answers using OSAVA in the class. The answers were then analyzed in the class and the instructor used them to explain the internal working of operating systems to the students. Figs. 1-4 show some of the questions that were actually solved in the class. Following this approach, it was possible to solve numerical questions involving large number of processes and representing different conditions in the class.

Students became familiar with OSAVA when CPU scheduling algorithms were taught in the class. The instructor continued using OSAVA to teach about deadlocks, memory management, virtual memory and secondary storage management over the semester. The students were particularly keen in using OSAVA to study the comparatively more difficult algorithms like multilevel queue scheduling, multilevel feedback queue scheduling, Banker's, second chance, C-SCAN and C-LOOK algorithms. Many students later reported that they have also used OSAVA for self-study in the days before the exam when an instructor was not available to help them. We used two techniques to assess the benefits of OSAVA, viz. collecting feedback from the students and analyzing the scores they obtained in the exam.

Table 3. Student feedback (N = 243).

		Agree	Disagree	No Opinion
1.	Did OSAVA help you in understanding the algorithms?	204	30	9
2.	Is OSAVA easy to use?	214	25	4
3.	Are the outputs of OSAVA presented logically?	231	9	3
4.	Did OSAVA enhance the overall learning experience?	194	38	11

A. Student Feedback

A survey was conducted at the end of the semester. The students were asked if they felt that OSAVA helped them in understanding the algorithms, was easy to use, presented the outputs logically and augmented the overall learning experience. Out of the 243 students who attended the course, 84% felt that OSAVA helped them to better understand the algorithms used in operating systems, 88% agreed that OSAVA was easy to use, 95% felt that OSAVA produces outputs in scientific and self-explanatory formats, and 80% felt that OSAVA enhanced the course as a whole (Table 3). In their feedback, several students also remarked that they would like using similar mobile apps for studying other courses that use complex algorithms.

B. Analysis of Exam Results

The students were awarded a score out of 100 by the instructor at the completion of the course. We compared the scores obtained by the students who attended the course in Spring 2016, Spring 2017 and Spring 2018 with of those who attend the course in Spring 2015 (Fig. 5). It may be noted that the course was taught in Spring 2015 by the same instructor to a class of 60 students but without OSAVA. The instructor took care to keep the format and the level of difficulty of the questions similar in the three years. We observed that the average score of the students increased by more than 9% ($P < 0.01$) from 2015 to 2016-18. We also observed that the standard deviation fell by about 6% during the same period. There was an overall improvement in the quality of answers written by the students.

5. Discussion

In our opinion, it was possible to successfully assimilate OSAVA in the course because of its following three properties.Fig. 5. Mean and standard deviation of scores of students.

-Comprehensiveness. OSAVA visualizes all the algorithms that are typically taught in the course on operating systems. This is in contrast to the earlier tools that concentrated only on a few important algorithms (compare Tables 1 and 2). OSAVA takes into account all variants and parameters of the algorithms. OSAVA helped students to practice numerical questions, learn more about the algorithms and score better in the exam. Students often used OSAVA to compare the different algorithms.

-Fidelity to the textbook. OSAVA visualizes the algorithms as described in the textbook (Silberschatz et al., 2012). Same terminology, and similar notations and diagrams have been used. This helped in a seamless integration of OSAVA in the course. OSAVA provides a brief textual description of the algorithm before visualizing it. This helps students to quickly recollect the underlying concept of the algorithm. The modules of OSAVA present their outputs as a combination of appropriate diagrams and numeric results. They give students an idea of how to answer questions in the exam. We believe that this property gives OSAVA an edge over the earlier tools.

-Attractiveness. Smartphones are a powerful, portable and attractive technology. Most students now carry a smartphone to the class which can be used to augment the teaching-learning process. Students appreciated the idea that an educational app is being used to foster formal curriculum-based education. They used OSAVA during the class hours and beyond. A desktop version of OSAVA was also developed and made available to the students. However, the students invariably preferred to use the mobile app.

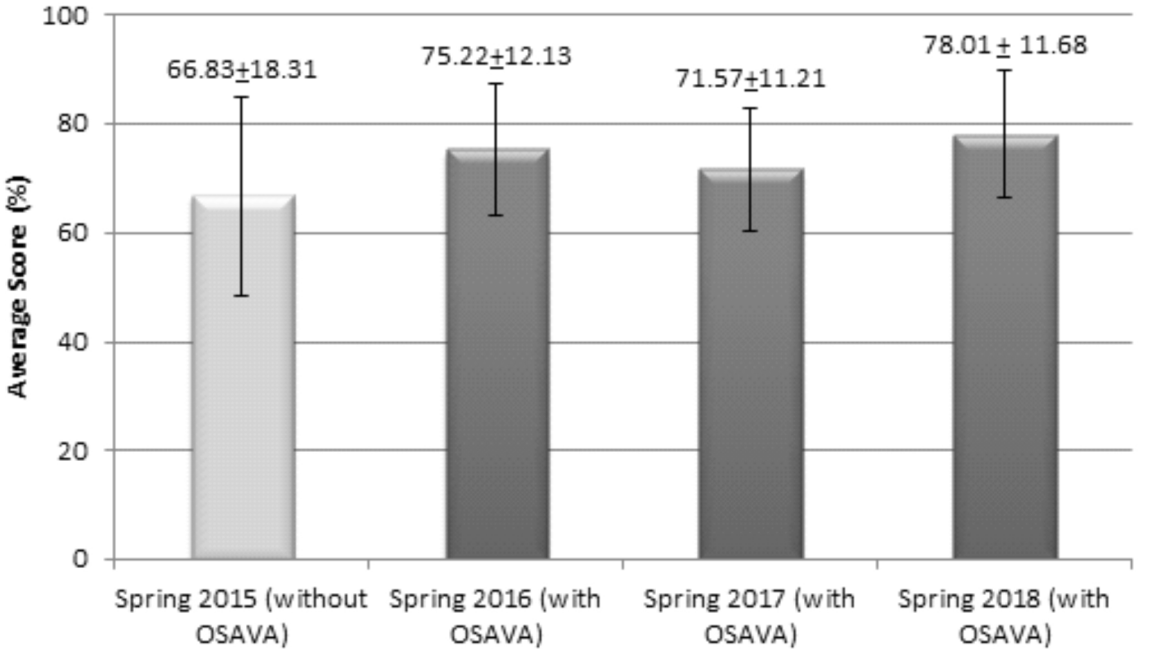

Fig. 5. Mean and standard deviation of scores of students.

6. Conclusions

We have developed a mobile app and used it to teach the course on operating systems for two consecutive years. Students attending the course felt that the app helped them in understanding the algorithms used in operating systems better. An analysis of the scores obtained by the students in the exam also showed that the app helped them in understanding the internal working of operating systems. Although we followed the book by Silberschatz et al. (2012), OSAVA can be used with other textbooks as well.

We have two observations that may be of the interest of educationists in general. First, course-specific software tools, if designed properly, can prove useful as instructional aids. Second, mobile apps are more pervasive than most other forms of software and hence, if it suits the purpose, educational software may be implemented as mobile apps.

Acknowledgement

We thank Savita Yadav who also used OSAVA to teach the course on operating systems and shared her experience with us.

References

[1] Comer, D. (2015). Operating System Design – The Xinu Approach. 2nd ed., CRC Press.

[2] Desnoyers, P. J. (2011). Teaching operating systems as how computers work. Proceedings of the Forty-second ACM Technical Symposium on Computer Science Education, 281-286.

[3] Fischbach, J. A. (2013). Visualization of student-implemented OS algorithms in Java. Journal of Computing Sciences in Colleges, 28(3), 6-13.

[4] Garmpis, A. (2013). Alg_OS – A web-based software tool to teach page replacement algorithms of operating systems to undergraduate students. Computer Applications in Engineering Education, 21(4), 581-585.

[5] Khuri, S., & Hsu, H.-C. (1999). Visualizing the CPU scheduler and page replacement algorithms. ACM SIGCSE Bulletin, 31(1), 227-231.

[6] Krishnamoorthy, S. (2002). An experience teaching operating systems course with a programming project. Journal of Computing Sciences in Colleges, 17(6), 25-38.

[7] Silberschatz, A., Galvin, P. B., & Gagne, G. (2012). Operating System Concepts. 9th ed., Wiley.

[8] Suranauwarat, S. (2007). A CPU scheduling algorithm simulator. Proceedings of the Thirty-seventh Annual Frontiers in Education Conference, F2H19-F2H24.

[9] Tanenbaum, A. S., & Woodhull, A. S. (2006). Operating Systems: Design and Implementation. 3rd ed., Prentice Hall.

[10] Yuan, X., Pioro, B., Archer, R., & Li, Y. (2008). Teaching operating systems using visualization: A comparative study, In: Iskander, M. (Ed.) Innovative Techniques in Instruction Technology, E-learning, E-assessment, and Education, Springer, 576-580.