Probationary Faculty Development Grant Final Report

Spring 2020

Yuan Cheng, Department of Computer Science, ECS

Title

A Web-based Interactive Algorithm Simulator for Computer Science Courses

Objectives

This project aims to enhance the Computer Science undergraduate curriculum by using visualization and animation in an active way. Visualization uses images to enhance understanding, while animation refers to the use of visual displays that change over time. We develop a web application, named ALVIS (ALgorithm VISualizer). This application generates a collection of interactive animation projects for an operating systems class (i.e., CSC 139 Operating System Principles), including CPU scheduling, memory page replacement, and disk head scheduling. Each animation will illustrate a concept and encourage students to examine the concept in depth. Students can tune the parameters to change the animations. These animations should enhance the student's understanding of the concepts and making learning more enjoyable and challenging.

Description

This section describes the expected features, the required technologies to implement those features, the expected use, and the expected users.

Description of Features

ALVIS features three families of algorithms from CSC 139 Operating System Principles.

1. CPU Scheduling
   a. First Come, First Served (FCFS)
   b. Shortest Job First (SJF)
   c. Shortest Remaining Time First (SRTF)
   d. Round Robin (RR)
   e. Non-preemptive Priority
   f. Preemptive Priority
2. Memory Page Replacement
   a. First In First Out (FIFO)
   b. Optimal (OPT)
   c. Least Recently Used (LRU)
3. Disk Head Scheduling
   a. First Come, First Served (FCFS)
   b. Shortest Seek Time First (SSTF)
   c. SCAN

d. C-SCAN
e. LOOK
f. C-LOOK

Users can select an algorithm from the dropdown menu on left side panel to practice. They can type the input parameters in the textboxes and hit the "Run" button. The application will execute the algorithm and display the output on the webpage accordingly.

For CPU scheduling algorithms, the output will be shown in the form of a Gantt chart. The average waiting time and the average turnaround time will also be calculated and displayed.

For memory page replacement, the contents of each physical memory frame will appear along with the total number of page faults.

For disk head scheduling algorithms, a graph that shows the simulated traversal of a disk head will be generated.

Required Technologies

- HTML
- CSS
- Javascript: React.js, Node.js
- Git

Expected Use

Students can use this application to practice knowledge after class and before exams. Instructors can use it to demonstrate the behavior of these algorithms in class.

Expected Users

Computer Science students and faculty. In the future, the coverage can be extended to the public.

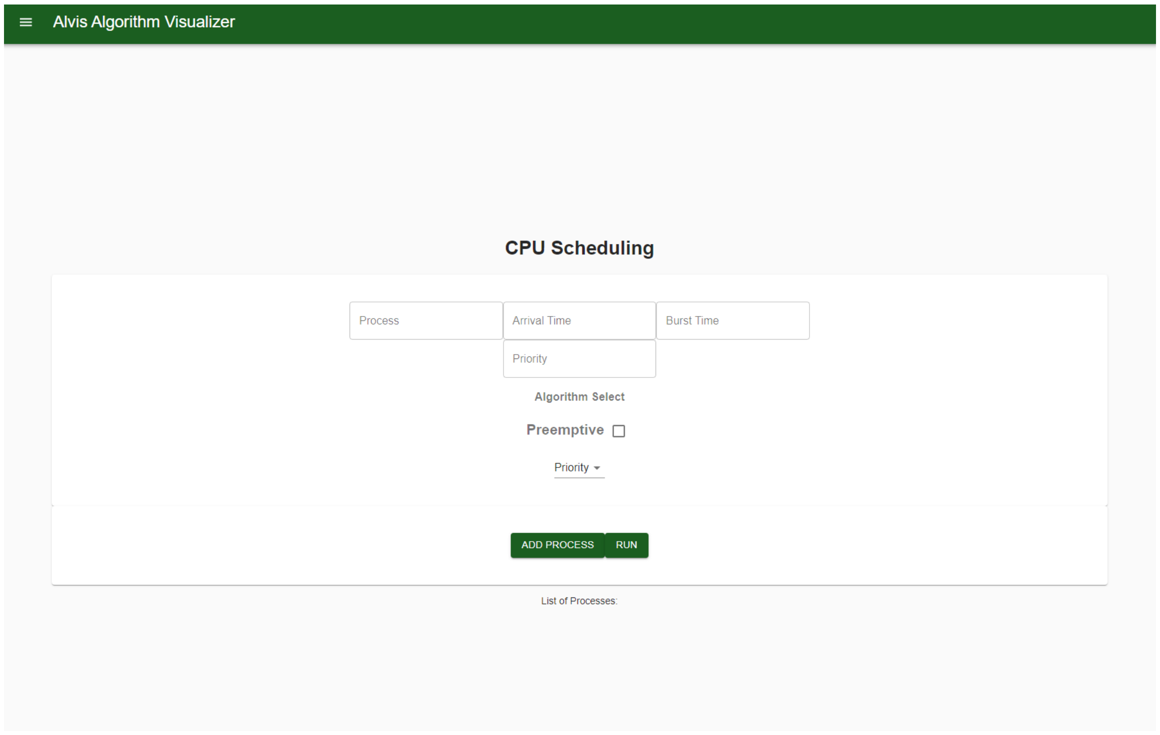
Results

As of the end of Summer 2020, a prototype system has been implemented and running on a node server provided by CSUS IRT. Users with a valid CSUS account will (eventually) be able to access the application from within the campus network or via a VPN connection.

Here are some screenshots taken from sample runs.

CPU Scheduling

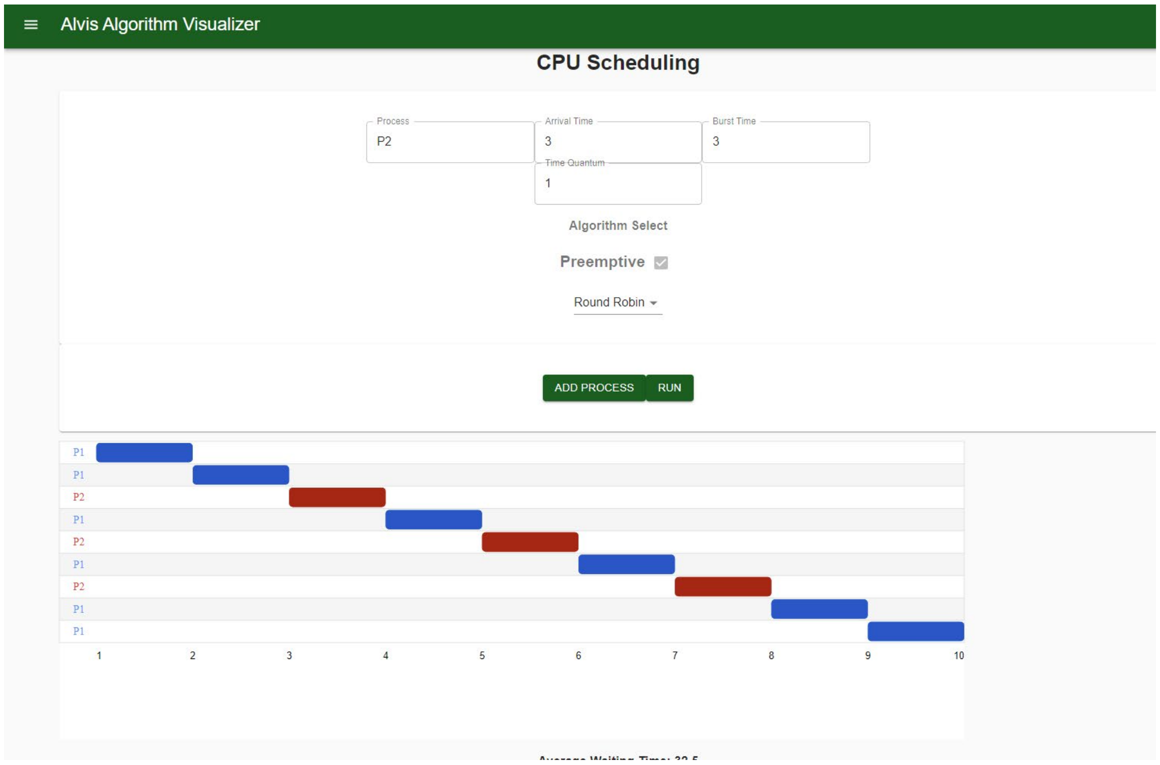
Figure 1 shows the initial view of CPU Scheduling visualization. Users can add the information of each process, such as process ID, arrival time, burst time. If it is a priority algorithm, the priority level of each process is also part of the input. The added processes will appear at the bottom of the page. Users can then choose an algorithm from the dropdown menu. For certain algorithms, they can also tick the box to indicate if the algorithm is preemptive or non-preemptive.



Preemptive ☐

Priority

Figure 1: An initial view of CPU Scheduling visualizer



Process: P2

Arrival Time: 3

Burst Time: 3

Time Quantum: 1

Preemptive ✓

Round Robin

Average Waiting Time: 32.5

Figure 2: A sample run of Round Robin algorithm

Figure 2 shows a sample run of the Round Robin scheduling algorithm. The Gantt chart clearly illustrates which process executes during which interval. Different processes are assigned with different colors.

Memory Page Replacement

Figure 3 presents the initial view of Page Replacement algorithms. Users can choose an algorithm from the dropdown menu and specify the number of physical frames along with a sequence of memory page numbers.

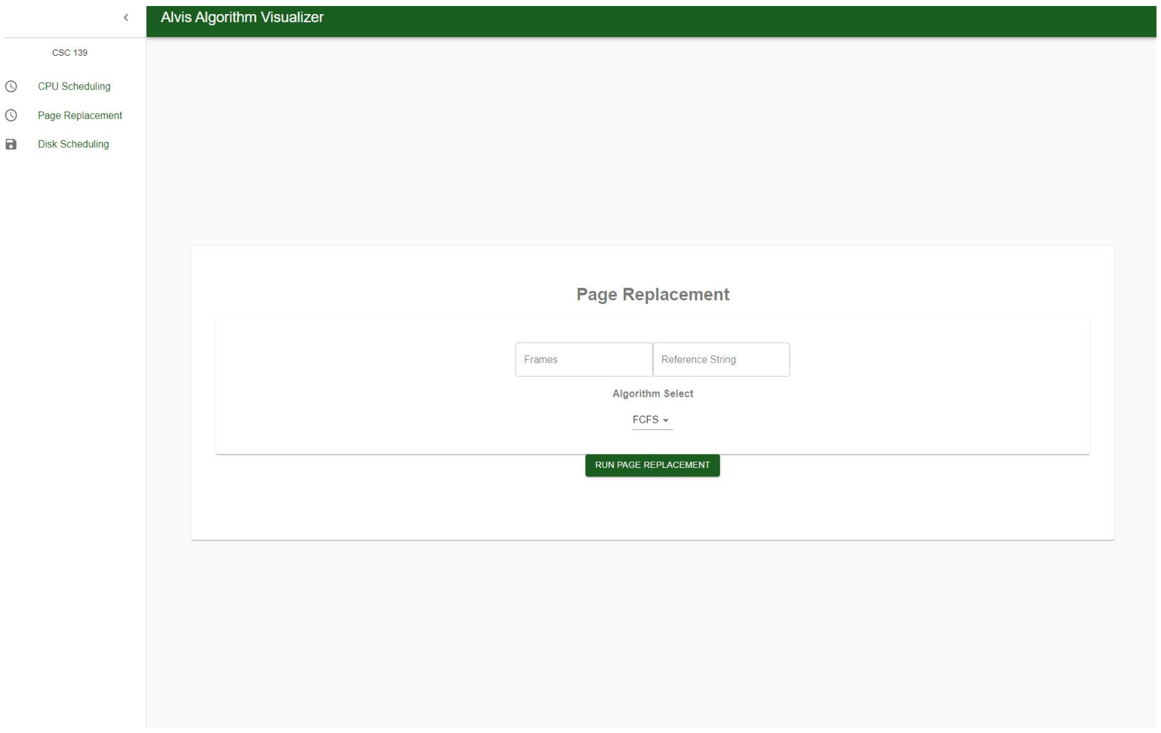
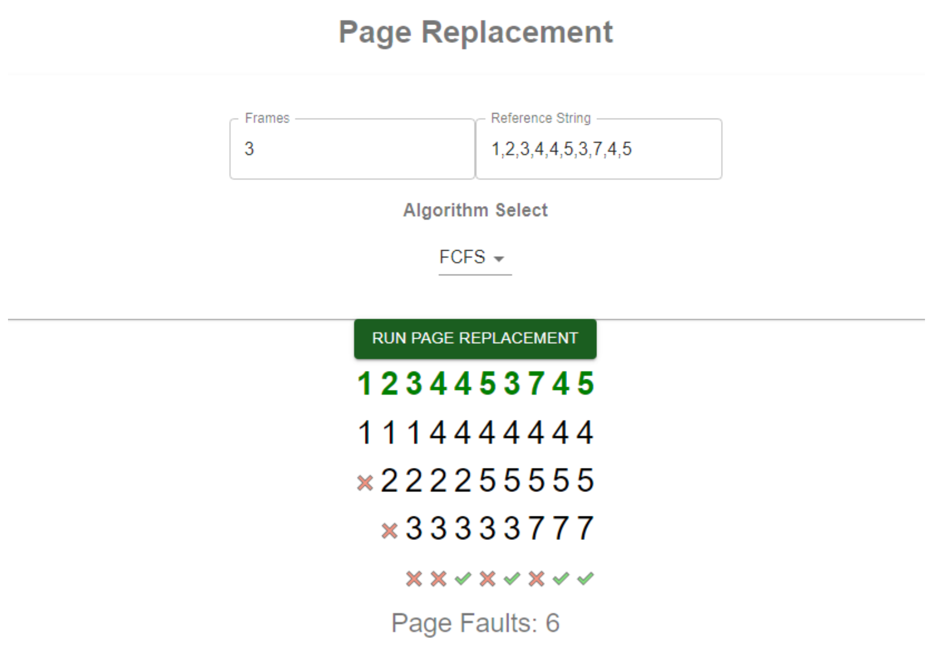


Figure 3: An initial view of Page Replacement visualizer

Figures 4 and 5 show a sample run of FCFS and OPT page replacement algorithms, respectively. The application outputs the content of each frame at each step. A page fault is indicated by a red cross, while a green checkmark implies that there is no page fault. The total number of page faults are shown at the bottom.



Page Replacement

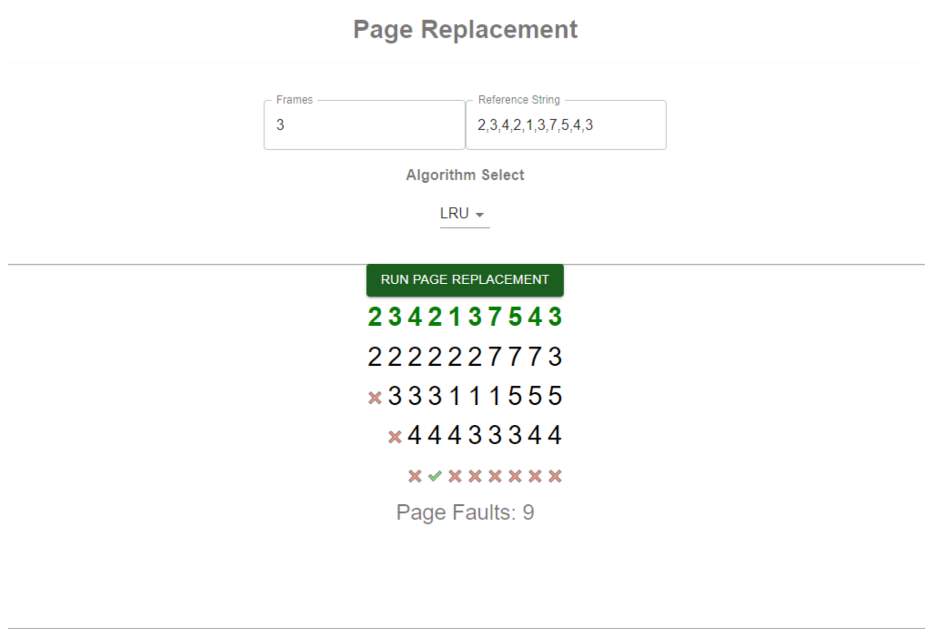
Frames: 3

Reference String: 1,2,3,4,4,5,3,7,4,5

Algorithm Select: FCFS

```
1 2 3 4 4 5 3 7 4 5
1 1 1 4 4 4 4 4 4 4
✗ 2 2 2 2 5 5 5 5 5
  ✗ 3 3 3 3 3 7 7 7
    ✗ ✗ ✓ ✗ ✓ ✗ ✓ ✓
Page Faults: 6
```

Figure 4: A sample run of FCFS replacement algorithm



Page Replacement

Frames: 3

Reference String: 2,3,4,2,1,3,7,5,4,3

Algorithm Select: LRU

```
2 3 4 2 1 3 7 5 4 3
2 2 2 2 2 2 7 7 7 3
✗ 3 3 3 1 1 1 5 5 5
  ✗ 4 4 4 3 3 3 4 4
    ✗ ✓ ✗ ✗ ✗ ✗ ✗ ✗
Page Faults: 9
```

Figure 5: A sample run of OPT replacement algorithm

Disk Head Scheduling

Figure 6 shows the initial view of Disk Head Scheduling algorithms. Users can choose an algorithm from the dropdown menu and set the initial moving direction of the disk head. The initial position of the disk head, total number of cylinders, and the sequence of cylinder numbers of disk access requests can be specified in the text boxes.

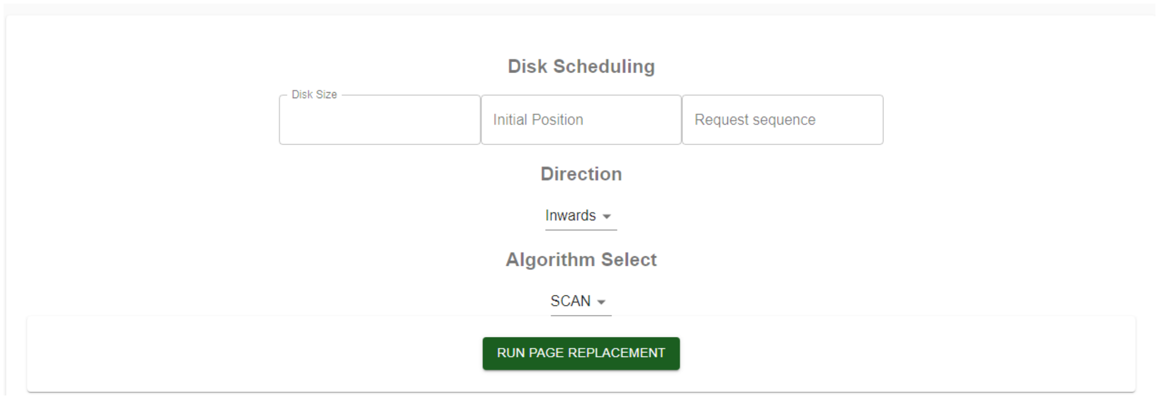
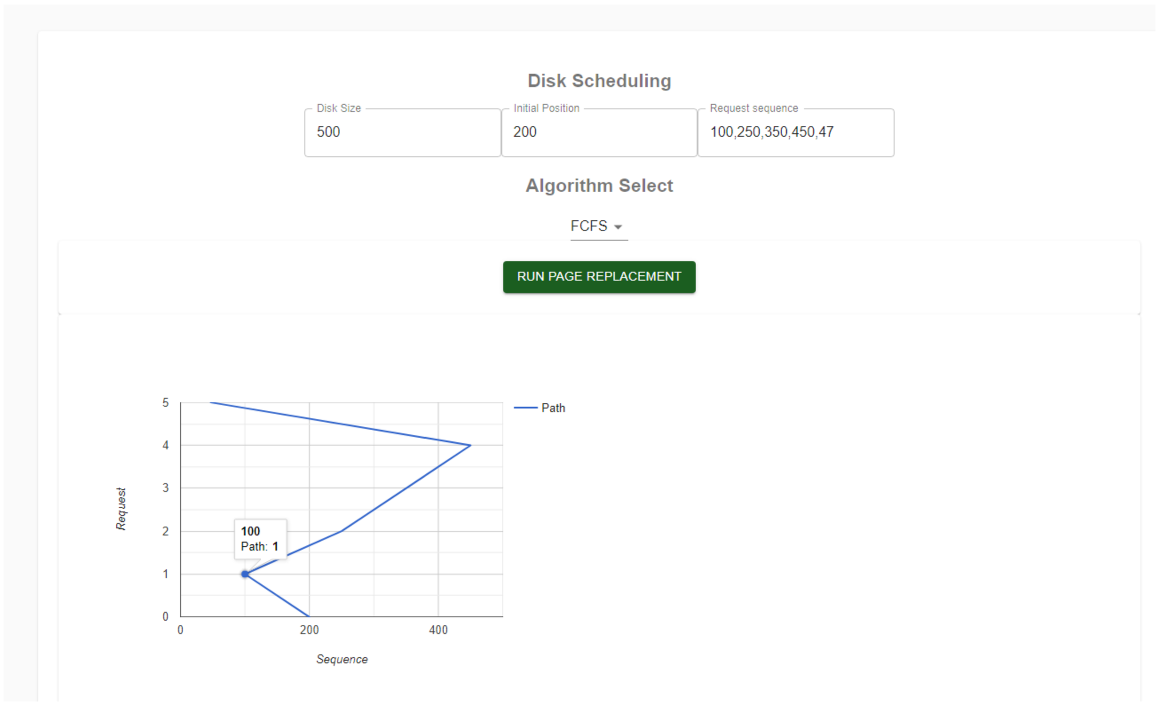


Figure 6: An initial view of Disk Head Scheduling visualizer

After the chosen algorithm is executed, the movement of the disk head will be displayed on the webpage. Figure 7 shows a sample run of the FCFS disk head scheduling algorithm, while Figure 8 exhibits the behavior of the C-LOOK algorithm with the initial moving direction set to outward.



Disk Scheduling

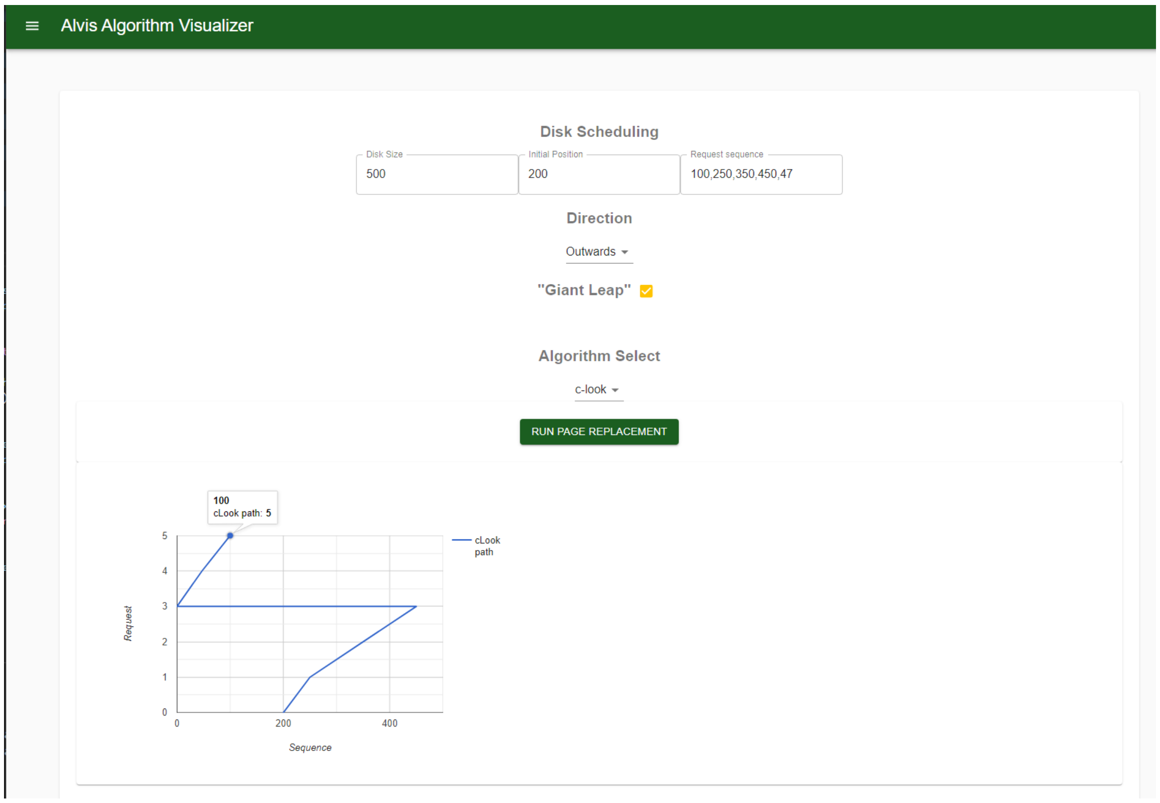
Disk Size: 500

Initial Position: 200

Request sequence: 100,250,350,450,47

Algorithm Select: FCFS

Figure 7: A sample run of FCFS disk scheduling


Alvis Algorithm Visualizer

Disk Scheduling

Disk Size: 500
Initial Position: 200
Request sequence: 100,250,350,450,47

Direction: Outwards

"Giant Leap" ✓

Algorithm Select: c-look

Figure 8: A sample run of C-LOOK disk scheduling

Future Work

The application is currently hosted on a testing server. We will migrate it to a production server soon and enable access for CSC 139 students in Fall 2020. As students try out the application throughout the semester, we will collect feedbacks and bug reports and come up with a sustainable plan for running and extending the application in the next couple of years.