**OS-Algo Visualization App**

Top-level Options:

1. CPU Scheduling Algorithms
2. Deadlock Avoidance Algorithm
3. Deadlock Detection Algorithm
4. Contiguous Memory Allocation Strategies
5. Page Replacement Algorithms
6. Disk Scheduling Algorithms

Tentative title of paper: An App to Visualize Algorithms Used in Operating Systems

Tentative journals for publication: IEEE Transactions of Learning Technologies

British Journal of Educational Technology

Computer Applications in Engineering Education

**CPU Scheduling Algorithms**

<The module should look like a form.>

<Only the fields applicable to the selected algorithm will be displayed.>

Information about the Processes –

Number of processes: <Max: 25>

Processes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Process <The processes will be named P1, P2, P3, and so on> | Arrival time (ms) | CPU burst time (ms) | Priority (0 being the highest) <Only in case of priority scheduling> | Queue <Only in case of multilevel queue scheduling> |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Information about the Scheduling Algorithm –

Algorithm: First Come First Served Scheduling

Shortest Job First Scheduling

Priority Scheduling

Round Robin Scheduling

Multilevel Queue Scheduling

Multilevel Feedback Queue Scheduling

Variant: Non-preemptive

Preemptive

<Only in cases of shortest job first scheduling and priority scheduling>

Aging: Promote the priority of a process by 1 unit each time after waiting ms until its value reaches . <Only in case of priority scheduling>

Time quantum (ms): <Only in case of round robin scheduling>

Number of queues: <Only in cases of multilevel queue scheduling and multilevel feedback queue scheduling>

Intra-queue scheduling algorithms: <Only in case of multilevel queue scheduling. The queues will be numbered 1, 2, 3, and so on. Preemptive priority scheduling will be used for inter-queue scheduling.>

|  |  |  |
| --- | --- | --- |
| Queue | Intra-queue scheduling algorithm <Only first come first served scheduling and round robin scheduling may be used> | Time quantum (ms) <Only in case of round robin scheduling> |
|  |  |  |
|  |  |  |
|  |  |  |

Time quanta of the queues: <Only in case of multilevel feedback queue scheduling. The queues will be numbered 1, 2, 3, and so on. Queues 1, 2, 3 … *n*-1 will use round robin scheduling. Queue *n* will use first come first served scheduling. Preemptive priority algorithm will be used for inter-queue scheduling.>

|  |  |
| --- | --- |
| Queue | Time quantum (ms) |
|  |  |
|  |  |
|  |  |

Dispatch latency (ms): <Default: 0>

<The visualize button will start the visualization process. After changing some information the visualize button can be pressed again. This will allow comparison of results between multiple sets of information.>

Visualize

Visualization Results –

A brief static description of the scheduling algorithm used.

Gantt chart: <Use 27 predefined colors, 25 for the processes, 1 for dispatch latency, 1 for idle time>

Timeline: <Log of all events, in text form for simplicity>

Turnaround time: <for each process, average, and standard deviation>

Waiting time: <for each process, average, and standard deviation>

Response time (assuming that the first output appears immediately after a process starts executing): <for each process, average, and standard deviation>

Throughput (processes/ms):

CPU utilization (%):

**Deadlock Avoidance Algorithm**

<Banker’s algorithm>

Number of processes: <Max: 25>

Number of resource types: <Max: 25>

Information about the current state –

Available:

Max:

Allocation:

Information about the request –

Process making the request:

Request:

Visualize

Use resource-request algorithm to check if the request can be granted.

Use safety algorithm to check if the resultant state will be safe.

Display the matrices at each step.

Display a safe sequence, if any.

**Deadlock Detection Algorithm**

Number of processes: <Max: 25>

Number of resource types: <Max: 25>

Available:

Max:

Allocation:

Request:

Visualize

Determine if there is a deadlock.

Display the matrices at each step.

If there is a deadlock, then display the deadlocked processes.

**Contiguous Memory Allocation Strategies**

<Dynamic continuous memory allocation>

Total memory (KB):

Number of processes: <Max: 25>

Memory requirements of the processes:

|  |  |
| --- | --- |
| Process | Memory requirement (KB) |
|  |  |
|  |  |
|  |  |

<Memory requirement of a process should not be more than the total memory.>

Sequence of events:

|  |  |
| --- | --- |
| Process | Arrives / Terminates |
|  |  |
|  |  |
|  |  |

<Each process must arrive and terminate once. A process must arrive before it terminates.>

Memory allocation strategy: First Fit

Best Fit

Worst Fit

Visualize

Pseudocode

While (all processes are not terminated)

{If (next event == process arrives)

{ Print “Process Pn arrives.”

If (contiguous memory available)

{Allocate memory

Print “Process Pn allocated m KB.”

Show memory map

}

Else

{Print “Memory cannot be allocated because of fragmentation.”

Add Pn to pending list

}

Else If (next event == process terminates)

{Deallocate memory

Print “Process Pn terminates.”

Show memory map

If ((pending list not empty) && (first process in pending list can be allocated))

{Allocate memory

Print “Process Pn allocated m KB.”

Show memory map

}

}

}

Sample memory map:

|  |  |
| --- | --- |
|  | 500 |
|  | 300 |
| P2 | 200 |
|  | 100 |
| P1 | 0 |

Or

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 |  | P2 |  |  |
| 0 | 100 | 200 | 300 | 500 |

**Page Replacement Algorithms**

Number of frames allocated: < Max: 25>

Reference string:

<List of page numbers.>

Page replacement algorithm: First In First Out Algorithm

Optimal Algorithm

Least Recently Used Algorithm

Additional-Reference-Bits Algorithm

Second-Chance Algorithm

Enhanced Second-Chance Algorithm

Least Frequently Used Algorithm

Most Frequently Used Algorithm

Read-only memory accesses:

<Only for Enhanced Second-Chance Algorithm>

Visualize

Show memory frames stepwise

Number of page faults:

**Disk Scheduling Algorithms**

Number of cylinders in disk:

<Cylinders to be numbered 0 to n-1.>

Current position of read/write head:

<Should be between 0 and n-1.>

Disk queue:

<List of cylinder numbers. Each cylinder number should be between 0 and n-1.>

Disk scheduling algorithm: First Come First Served

Shortest Seek Time First

Scan (Elevator Algorithm)

Circular-Scan

Look

Circular-Look

Current direction of movement of read/write head: Inward

Outward

<For Scan, Circular-Scan, Look and Circular-Look algorithms only.>

Visualize

Diagram using solid and dashed lines.

Total number of cylinder travelled: