# Processor Scheduling emulations

## Technology:

The programms were written using Node.JS v10. It is a single-threaded runtime environment for running javascript applications on the backend side.

## Input Data:

Each processor emulation was supplied with the same tasks. Task is a class with defined properties and methods. Each task is equipped with three properties: id, scheduled time and process duration (inner iterations). The table of processes looks like this:

| ID | Scheduled time (sched) | Duration (iter) |
|---|---|---|
| 0 | 0 | 43 |
| 1 | 2000 | 43 |
| 2 | 5 | 4 |
| 3 | 6 | 423 |
| 4 | 7 | 4789323 |
| 5 | 8 | 43452553 |
| 6 | 12 | 44433 |
| 7 | 16 | 0 |
| 8 | 17 | 43423 |
| 9 | 18 | 4323 |
| 10 | 12 | 433 |
| 11 | 16 | 2 |
| 12 | 12 | 4313 |
| 13 | 16 | 2555 |

# 1. FCFS

ID: 0 time: 0
 started: 0 ended: 43
---
ID: 1 time: 0
 started: 2000 ended: 2043
---
ID: 2 time: 0
 started: 2043 ended: 2047
---
ID: 3 time: 0
 started: 2047 ended: 2470
---

ID: 4 time: 5
 started: 2470 ended: 4791793
---
ID: 5 time: 30
 started: 4791793 ended: 48244346
---
ID: 6 time: 0
 started: 48244346 ended: 48288779
---
ID: 7 time: 0
 started: 48288779 ended: 48288779
---
ID: 8 time: 0
 started: 48288779 ended: 48332202
---
ID: 9 time: 0
 started: 48332202 ended: 48336525
---
ID: 10 time: 0
 started: 48336525 ended: 48336958
---
ID: 11 time: 0
 started: 48336958 ended: 48336960
---
ID: 12 time: 0
 started: 48336960 ended: 48341273
---
ID: 13 time: 0
 started: 48341273 ended: 48343828

---

# 2. SJF

ID: 0 time: 0
 started: 0 ended: 43
---
ID: 7 time: 0
 started: 43 ended: 43
---
ID: 11 time: 0
 started: 43 ended: 45
---
ID: 2 time: 0
 started: 45 ended: 49
---
ID: 3 time: 0

started: 49 ended: 472
---
ID: 10 time: 0
started: 472 ended: 905
---
ID: 13 time: 0
started: 905 ended: 3460
---
ID: 1 time: 0
started: 3460 ended: 3503
---
ID: 12 time: 1
started: 3503 ended: 7816
---
ID: 9 time: 0
started: 7816 ended: 12139
---
ID: 8 time: 1
started: 12139 ended: 55562
---
ID: 6 time: 1
started: 55562 ended: 99995
---
ID: 4 time: 3
started: 99995 ended: 4889318
---
ID: 5 time: 28
started: 4889318 ended: 48341871

---

# 3. Preemptive SJF

ID: 0 time: 5
started: 0 ended: 4
---
ID: 2 time: 4
started: 5 ended: 8
---
ID: 0 time: 7
started: 9 ended: 15
---
ID: 7 time: 1
started: 16 ended: 16
---
ID: 11 time: 2
started: 17 ended: 18
---
ID: 0 time: 31
started: 19 ended: 49
---

ID: 3 time: 423
 started: 50 ended: 472
---
ID: 10 time: 433
 started: 473 ended: 905
---
ID: 13 time: 1094
 started: 906 ended: 1999
---
ID: 1 time: 43
 started: 2000 ended: 2042
---
ID: 13 time: 1461
 started: 2043 ended: 3503
---
ID: 12 time: 4313
 started: 3504 ended: 7816
---
ID: 9 time: 4323
 started: 7817 ended: 12139
---
ID: 8 time: 43423
 started: 12140 ended: 55562
---
ID: 6 time: 44433
 started: 55563 ended: 99995
---
ID: 4 time: 4789323
 started: 99996 ended: 4889318
---
ID: 5 time: 43452553
 started: 4889319 ended: 48341871

---

# 4. Rotational

ID: 0 time: 0
 started: 0 ended: 43
---
ID: 2 time: 0
 started: 43 ended: 47
---
ID: 3 time: 0
 started: 47 ended: 470
---
ID: 4 time: 14
 started: 470 ended: 4789793
---
ID: 1 time: 1

started: 4789793 ended: 4789836
---
ID: 5 time: 113
started: 4789836 ended: 48242389
---
ID: 6 time: 1
started: 48242389 ended: 48286822
---
ID: 7 time: 0
started: 48286822 ended: 48286822
---
ID: 8 time: 0
started: 48286822 ended: 48330245
---
ID: 9 time: 0
started: 48330245 ended: 48334568
---
ID: 10 time: 0
started: 48334568 ended: 48335001
---
ID: 11 time: 0
started: 48335001 ended: 48335003
---
ID: 12 time: 0
started: 48335003 ended: 48339316
---
ID: 13 time: 0
started: 48339316 ended: 48341871

---

# Comparison of algorithms:

It is assumed that each algorithm starts with 0 tick and ends when the last process terminates. Moreover, each processor was supplied with the same tasks. Based on this such a comparison can be noticed:

| **Algorithm** | **Number of Ticks** | **Time taken** |
|---|---|---|
| **FCFS** | 48343828 | 74 ms |
| **SJF** | **48341871** | **46 ms** |
| **Preemptive SJF** | 48341871 | 9137 ms |
| **ROT** | 48341871 | 140 ms |

Based on the numbers, it can be assesed that for this particular dataset all but FCFS algorithms are optimal. This is because the Process 1 freezes the thread

for 1957 ticks. In case of all other processes, the processes are being chosen before execution amongst the already available ones (the ones their scheduled burst tick is less than the actual tick counter). Even though In order to assess the best algorithm, I introduced measuring time. The time is being measured in milliseconds. It starts being measured when the runtime is launched and is stopped when the last process is solved. Based on this the winner is SJF algorithm. It is a greedy algorithm which does not let the thread be blocked by pending process with late burst time. Moreover, it does not perform so many operations comparing to Preemptive SJF, which constanlty searches for a shorter process (it takes time).