Zhaoxi Sun <sunz5>, Xinyan Sun <sunx15>, Tongyu Wang <wangt15>, Yueting Liao <liaoy3>

1.

By using data collection and predictions, our team decided to choose the shortest Remaining time(SRT) algorithm as the “best” algorithm out of four. Even with a different seed number, the SRT algorithm performances the best out of the four by having the least average run time. SRT is most efficient for processes to complete its CPU burst and I/O burst by allowing processes to preempt and “sort” based on the situation.

SJF	seed	average wait time	average turnaround time	average CPU burst time
1 processes	2	0.000 ms	84.312 ms	80.312 ms
2 processes	2	57.459 ms	164.486 ms	103.027 ms
16 processes	2	669.520 ms	757.823 ms	84.304 ms
8 processes	64	2889.805 ms	3797.237 ms	903.431 ms

SJF algorithm is the best-suited for CPU-bound processes, it always allows the process with the shortest CPU burst from the ready state to enter running state. Which allows the CPU bound to complete first compare to other algorithms.

SRT	seed	average wait time	average turnaround time	average CPU burst time
1 processes	2	0.000 ms	84.312 ms	80.312 ms
2 processes	2	43.162 ms	151.054 ms	103.027 ms
16 processes	2	695.521 ms	785.248 ms	84.304 ms
8 processes	64	2811.782 ms	3720.338 ms	903.431 ms

Zhaoxi Sun <sunz5>, Xinyan Sun <sunx15>, Tongyu Wang <wangt15>, Yueting Liao <liaoy3>

FCFS is best-suited for I/O-bound, for the reason of choosing FCFS is that a process always has the chance to enter its I/O burst after finishing its CPU burst. The process under the FCFS scheduling algorithm will never get preempted by other processes, which will lose the chance to enter I/O burst immediately. FCFS ensures the I/O burst for each process completes consistently.

FCFS	seed	average wait time	average turnaround time	average CPU burst time
1 processes	2	0.000 ms	84.312 ms	80.312 ms
2 processes	2	57.459 ms	164.486 ms	103.027 ms
16 processes	2	622.507 ms	710.810 ms	84.304 ms
8 processes	64	3185.499 ms	4092.930 ms	903.431 ms

Zhaoxi Sun <sunz5>, Xinyan Sun <sunx15>, Tongyu Wang <wangt15>, Yueting Liao <liaoy3>

2.

We compared the results of the RR algorithm from END to BEGINNING. We found that most of the average turnaround times for the BEGINNING algorithm are a bit smaller than that for END algorithm. Hence we think the RR algorithm will be better if the rr_add is BEGINNING.

RR	seed	average wait time	average turnaround time	average CPU burst time
1 processes	2	0.000 ms	84.312 ms	80.312 ms
2 processes	2	46.486 ms	154.378 ms	103.027 ms
16 processes	2	669.814 ms	761.492 ms	84.304 ms
8 processes	64	3253.886 ms	4161.782 ms	903.431 ms

3.

Since SJF and SRT both rely on the estimated burst time for CPU burst time, the key initial value α determines the estimate calculation. SRT has a much lower average time around time compared to the SJF in α range from 0 to 1. The lowest turnaround time appears to happen with the SRT algorithm.

4.

By changing from a non-preemptive algorithm (SJF) to a preemptive algorithm (SRT), the simulation result changes by changing the performance time and starvation problem. The starvation issue occurs when changing from SJF to SRT since there is preemption, a process with huge CPU burst time might result in starvation. Also, the SRT would likely to perform a long time due to the preemption time waste. Relatively, SJF might have a slightly better running performance compare to SRT.

Limitation 1: There is a big difference between the proportion of CPU burst time and IO burst time in our simulation and the proportion of CPU burst time and IO burst time in the real condition. The running speed is much faster than reading and writing speed in the real world. So the IO burst time will be much greater than CPU burst time. Our simulation does not have a good prediction of this proportion. We can ask the user for the proportion of CPU burst time and IO burst time. So that we can make a more accurate prediction.

Limitation 2: We cannot know if there is a process that is starved for too long times. For example, if we use the SJF algorithm and there is a process that needs much more CPU burst time than other processes, this process may starve for a very long time or it will start running after all other processes have finished running. We can create a list to store the idle time for each process so we can make the model better.

Limitation 3: Our simulation can just simulate a one-core CPU. But in the real world, most CPUs we use are multiple-core CPUs, so the simulation cannot simulate the real condition computer well now. We can change our code to let it simulate multiple-core CPUs.

Zhaoxi Sun <sunz5>, Xinyan Sun <sunx15>, Tongyu Wang <wangt15>, Yueting Liao <liaoy3>

6.

Our new scheduling algorithm will create several queues that have different priorities. Every process will be put into the lowest priority queue when they arrive. Then, the processes will be sorted by the CPU burst time. The shortest process will run first just like the SJF algorithm. But every process has a timer to record the waiting time for themselves. User will give the maximum waiting time in the beginning. If the waiting time of a process exceeds the maximum waiting time, it will be taken out from its current queue and be put into the queue which has a higher priority. The CPU will run the process in a higher priority first even if it has a longer CPU burst time.

Advantages:

If a process has been idle for a long time, it will be put into a higher priority queue. So our scheduling algorithm solves the starvation and ensures the large process completes.

Disadvantage:

If there are many larger processes that need a long CPU burst time. The processes that need less CPU burst time may run at the end instead. So the waiting time for a short process may be very long.

Our algorithm	seed	average wait time	average turnaround time	average CPU burst time
1 processes	2	0.000 ms	84.312 ms	80.312 ms
2 processes	2	53.435 ms	160.403 ms	103.027 ms
16 processes	2	646.932 ms	734.030 ms	84.304 ms
8 processes	64	3034.631 ms	3952.233 ms	903.431 ms