Exercise 2-3 Solution file from Kelton/Sadowski/Zupick, *Simulation With Arena*, 6th edition, McGraw-Hill, 2015

Depict this future-service-time attribute by representing the parts in service and in queue by a (vertical) two-vector with the first (top) entry's being the arrival time, as already done, and the second (bottom) entry's being the service requirement, as taken from Table 2-1. Also, keep the queue ranked in decreasing order of the service-requirement attribute. The resulting hand-simulation table looks like this (the shaded entries indicate differences from Table 2-2 that were caused by moving from the FIFO to the SPT queue discipline):

Just-Finished Event			Variables		Attributes		Statistical Accumulators								Event Calendar
Entity No.	Time t	Event Type	$Q(t)$	$B(t)$	Arrival Times: (In Queue)	In Service	P	N	ΣWQ	WQ^*	ΣTS	TS^*	$\int Q$	Q^*	$\int B$	[Entity No., Time, Type]
–	0.00	Init	0	0	()	–	0	0	0.00	0.00	0.00	0.00	0.00	0	0.00	[1, 0.00, Arr] [–, 20.00, End]
					()	–										
1	0.00	Arr	0	1	()	0.00	0	1	0.00	0.00	0.00	0.00	0.00	0	0.00	[2, 1.73, Arr] [1, 2.90, Dep] [–, 20.00, End]
					()	2.90										
2	1.73	Arr	1	1	(1.73)	0.00	0	1	0.00	0.00	0.00	0.00	0.00	1	1.73	[1, 2.90, Dep] [3, 3.08, Arr] [–, 20.00, End]
					(1.76)	2.90										
1	2.90	Dep	0	1	()	1.73	1	2	1.17	1.17	2.90	2.90	1.17	1	2.90	[3, 3.08, Arr] [2, 4.66, Dep] [–, 20.00, End]
					()	1.76										
3	3.08	Arr	1	1	(3.08)	1.73	1	2	1.17	1.17	2.90	2.90	1.17	1	3.08	[4, 3.79, Arr] [2, 4.66, Dep] [–, 20.00, End]
					(3.39)	1.76										
4	3.79	Arr	2	1	(3.79, 3.08)	1.73	1	2	1.17	1.17	2.90	2.90	1.88	2	3.79	[5, 4.41, Arr] [2, 4.66, Dep] [–, 20.00, End]
					(4.52, 3.39)	1.76										
5	4.41	Arr	3	1	(3.79, 4.41, 3.08)	1.73	1	2	1.17	1.17	2.90	2.90	3.12	3	4.41	[2, 4.66, Dep] [6, 18.69, Arr] [–, 20.00, End]
					(4.52, 4.46, 3.39)	1.76										
2	4.66	Dep	2	1	(3.79, 4.41)	3.08	2	3	2.75	1.58	5.83	2.93	3.87	3	4.66	[3, 8.05, Dep] [6, 18.69, Arr] [–, 20.00, End]
					(4.52, 4.46)	3.39										
3	8.05	Dep	1	1	(3.79)	4.41	3	4	6.39	3.64	10.80	4.97	10.65	3	8.05	[5, 12.51, Dep] [6, 18.69, Arr] [–, 20.00, End]
					(4.52)	4.46										
5	12.51	Dep	0	1	()	3.79	4	5	15.11	8.72	18.90	8.10	15.11	3	12.5	[4, 17.03, Dep] [6, 18.69, Arr] [–, 20.00, End]
					()	4.52										
4	17.03	Dep	0	0	()	–	5	5	15.11	8.72	32.14	13.24	15.11	3	17.0	[6, 18.69, Arr] [–, 20.00, End]
					()	–										
6	18.69	Arr	0	1	()	18.69	5	6	15.11	8.72	32.14	13.24	15.11	3	17.0	[7, 19.39, Arr] [–, 20.00, End] [6, 23.05, Dep]
					()	4.36										
7	19.39	Arr	1	1	(19.39)	18.69	5	6	15.11	8.72	32.14	13.24	15.11	3	17.7	[–, 20.00, End] [6, 23.05, Dep] [8, 34.91, Arr]
					(2.07)	4.36										
–	20.00	End	1	1	(19.39)	18.69	5	6	15.11	8.72	32.14	13.24	15.72	3	18.3	[6, 23.05, Dep] [8, 34.91, Arr]
					(2.07)	4.36										

The first time this rule has an effect is in the order of the queue after the Arrival of entity 5 at time 4.41, whose service time (4.46) is less than the service time (4.52) of entity 4, which is already in queue; entity 5 jumps ahead of entity 4 at this point.

The final output performance measures, corresponding to Table 2-3, are:

Performance Measure	**Value**	**Result from Table 2-3**	**Change**
Total production	5 parts	5 parts	No change
Average waiting time in queue	2.52 minutes per part (6 parts)	2.53 minutes per part (6 parts)	Decreased
Maximum waiting time in queue	8.72 minutes	8.16 minutes	Increased
Average total time in system	6.43 minutes per part (5 parts)	6.44 minutes per part (5 parts)	Decreased
Maximum total time in system	13.24 minutes	12.62 minutes	Increased
Time-average number of parts in queue	0.79 part	0.79 part	No change
Maximum number of parts in queue	3 parts	3 parts	No change
Drill-press utilization	0.92 (dimensionless proportion)	0.92 (dimensionless proportion)	No change

The effect of SPT rather than FIFO is for the average waiting time in queue and the average total time in system to go down (get better), while the maximum of these two measures went up (got worse), since the large jobs get stuck at the back of the queue for a long time. Thus, whether this is "better" depends on the performance measure

It would also be possible to put entities into the queue at the end and then do a search on the service-requirement attribute to decide which entity to remove; this would be somewhat more computation, though, since inserting the entity in the right place will generally require searching less than the whole queue, while picking the minimum-service-requirement entity out of an unordered queue will always require searching the entire queue to be sure we get the minimum.