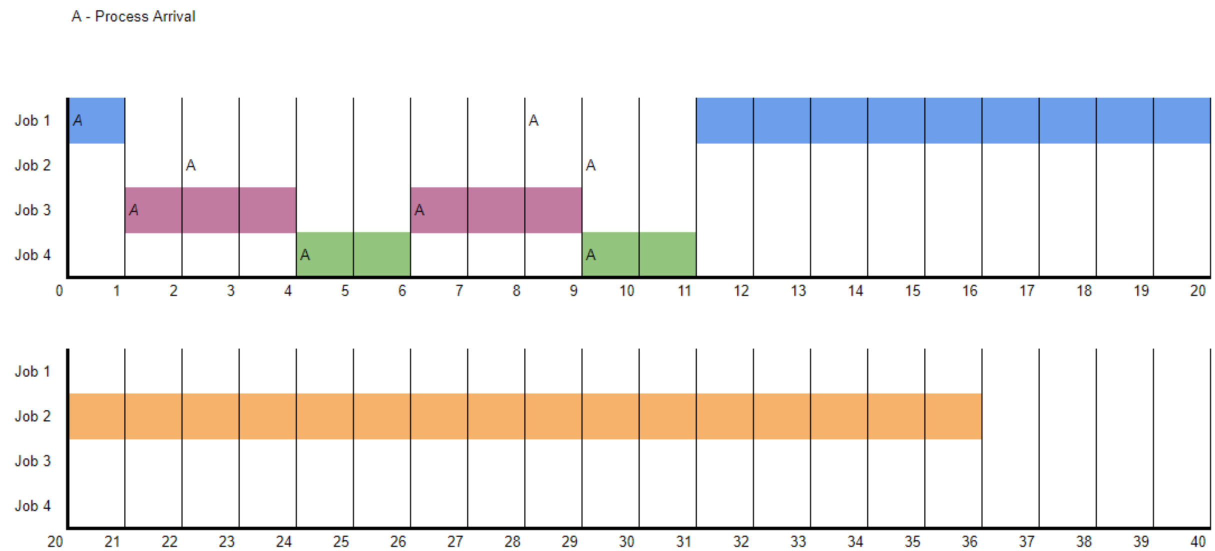
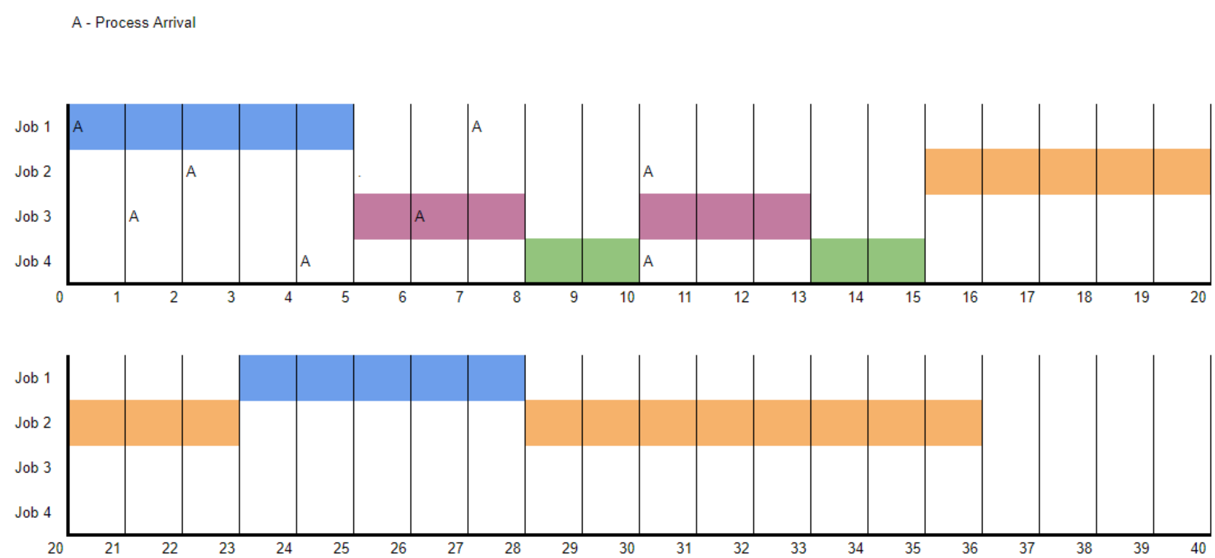
CS 350 – Fall 2020, Assignment 5

Answer 1

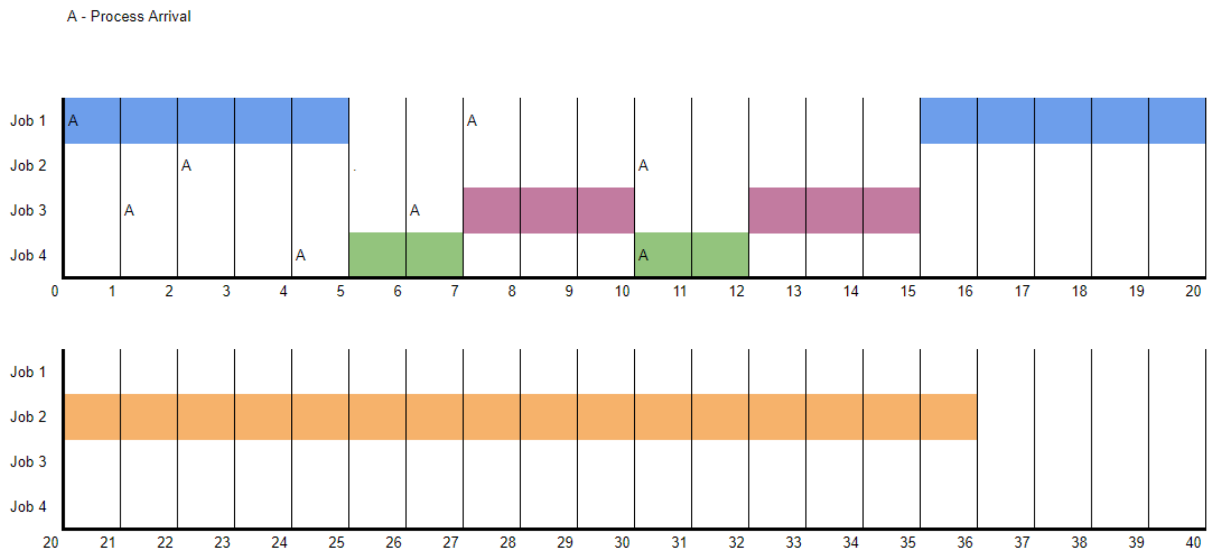
a)



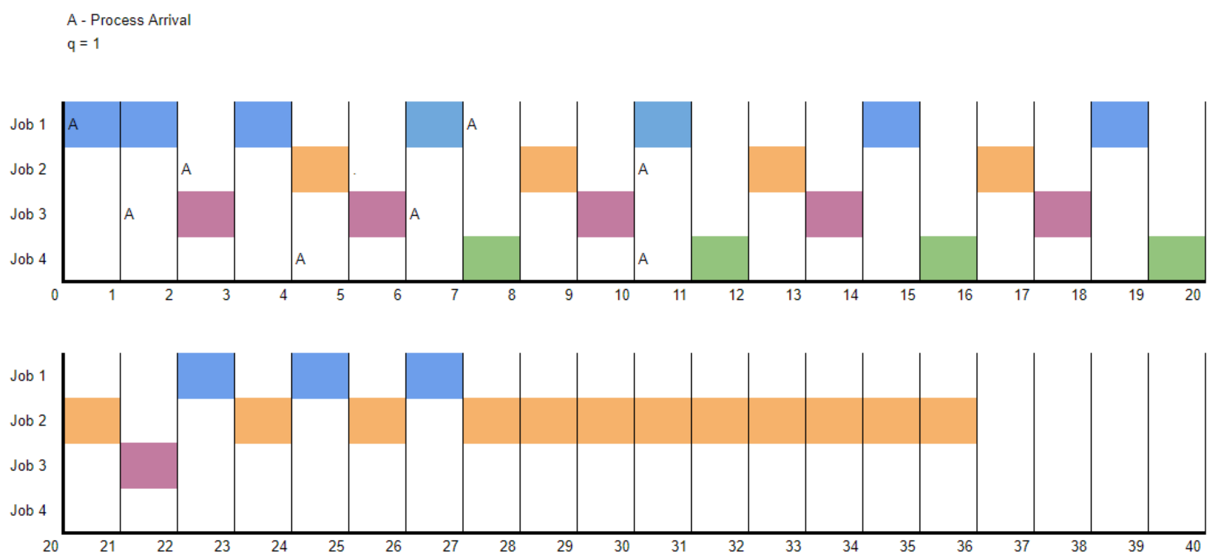
b)



c)



d)



e) The algorithm with better performance in terms of average response time is the HSN. The average response times are, $SRT = 11.125$, $HSN = 11$, $SJN = 11.625$, $Round - Robin = 15.375$, so as you can see, HSN has the lowest average response time. *(TABLE 1.1, PAGE 7)*

f) If we take for fairness how much each algorithm slows down each execution, and calculate the slowdown as response time over process time, then we need to check the standard deviation of the slowdowns of each algorithm.

If we do the calculations we get the following slowdown standard deviations: $STD_{SRT} = 1.09$, $STD_{HSN} = 0.9$, $STD_{SJN} = 0.99$, $STD_{RR} = 0.88$. From these values, we can assume that, as per the definition of fairness stated above, Round-Robin is the fairest algorithm. *(TABLE 1.2, PAGE 7)*

Answer 2

a) If at steady state, the server is receiving 326600 transactions per hour, then it is, on average, receiving a transaction every 11ms. Since the average time it took for a resource to complete on our observation is smaller than 11ms, then I can assume that the scheduler is work-conserving, and is running every time it has a task to complete.

   Furthermore, we can see that it is rare for us to have a transaction far above 11ms, which means the processor will be very likely preemptive. This is because, if it wasn't as soon as we would find a single transaction above 11ms, all of the following tasks would be delayed and display higher response times.

b) If we assume at steady state 326600 transactions are being processed per hour, then we have an average response time of each transaction to be 11.02ms. With this in mind, the average response time for our observations was 8.9ms, then we can assume that the steady-state system is averaging 2.12ms of latency per request.

c) If we take the confidence interval formula $E = Z_{\alpha/2} \times \sqrt{\frac{S^2}{N}}$, and work our way back based on our observations **(TABLE 2.1, PAGE 7)**, we get that:

$$E = Z_{\alpha/2} \times \sqrt{\frac{S^2}{N}} \equiv 1.5 = Z_{\alpha/2} \times \sqrt{\frac{9.2}{30}} \equiv \frac{1.5}{\sqrt{\frac{9.2}{30}}} = Z_{\alpha/2} \equiv \frac{1.5}{0.30(6)} = Z_{\alpha/2} \equiv Z_{\alpha/2} = 2.71$$

   From here, we can find that $Z_{\alpha/2} = 2.71 \equiv 1 - \alpha/2 = 0.9966 \equiv \alpha = 0.0068$, so our confidence interval that the per transaction service time is 8.9 +/- 1.5 milliseconds is $1 - \alpha = 1 - 0.0068 = 0.99$

d) If we use the same equation above, we know our cofidence interval is 0.99, so $\alpha = 1 - 0.99 = 0.01$, $1 - \alpha/2 = 0.995$, so our $Z_{\alpha/2} = 2.575$.

   Now we need to find the variance, to do this, let us once again look at our observations. If we take them as an isolated system, the average amount of requests at the server is 1.07, and the variance is 0.06 **(TABLE 2.2, PAGE 7)**. Then we get that:

$$E = Z_{\alpha/2} \times \sqrt{\frac{S^2}{N}} = 2.575 \times \sqrt{\frac{0.06}{30}} = 0.12$$

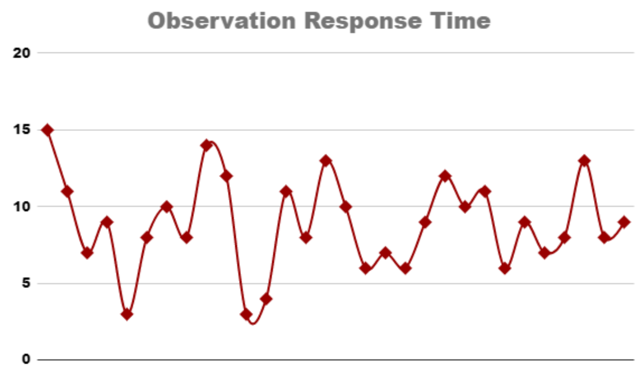
   If we assume that these observations were taken at steady state, and we know that the average amount of request in the actual system is 326600, we can calculate the E value proportionally to be 36628 for the whole system ($\frac{326600}{1.07} = \frac{x}{0.12} \equiv x = 26628$).

e) If we go back to that formula, we know that our $E = 1$, and that our confidence interval is 99.96% so $\alpha = 1 - 0.9996 = 0.0004$, so $1 - \alpha/2 = 0.9998$, so $Z_{\alpha/2} = 3.49$, so now if we plug this in using our observations **(TABLE 2.1)**, we get that:

$$E = Z_{\alpha/2} \times \sqrt{\frac{S^2}{N}} \equiv 1 = 3.9 \times \sqrt{\frac{9.2}{N}} \equiv \sqrt{\frac{9.2}{N}} = \frac{1}{3.9} \equiv \sqrt{\frac{9.2}{N}} = 0.256 \equiv \frac{9.2}{N} = 0.066 \equiv \frac{9.2}{0.066} = N = 139.394$$

With that, and assuming the 30 transactions already observed, we'd need to observe at least 110 more transactions before we can achieve a confidence interval of 99.96%. With that said, and assuming each transaction takes an average of 11ms response time, then we'd need to spend about 1210 milliseconds more observing the occurring transactions.

f)



If we look at the graph above, which graphs the observed values, we can quickly see that, despite the fact that we have no major rigorous patterns within our own observation, there is a certain up/down pattern in what regards response time, with every 6 nodes from the first low point us having a valley, and every 6 nodes from the first peak us having another peak.

In addition, it should be pointed out that the trending value is approximating the mean.

Answer 3

a)

Job ID	Job Length	Arrival Times
1	3	0, 10
2	5	1
3	3	3, 11
4	6	5
5	2	7

b) For this, we are going to figure out what the algorithm is by process of elimination. First and foremost, it isn't FCFS, since it interrupts jobs, on that same thought process, it also cannot be SJN nor HSN, since both don't interrupt jobs.

This leaves us with Round-Robin and SRT, at least from the ones we've studied thus far. From these, it's easy to eliminate Round-Robin, since, if it was the algorithm, then it would need a $q \leq 5$, since there is a clear task break at 5.

From here we can eliminate any value of q, since if it was 1 or 2, at step 3 it'd execute job 2 and not job 1. If it was 4, at step 3 it would execute job 2 and not job 3. Finally, it it was 3 or 5, then at step 15 it'd execute job 4, not job 3.

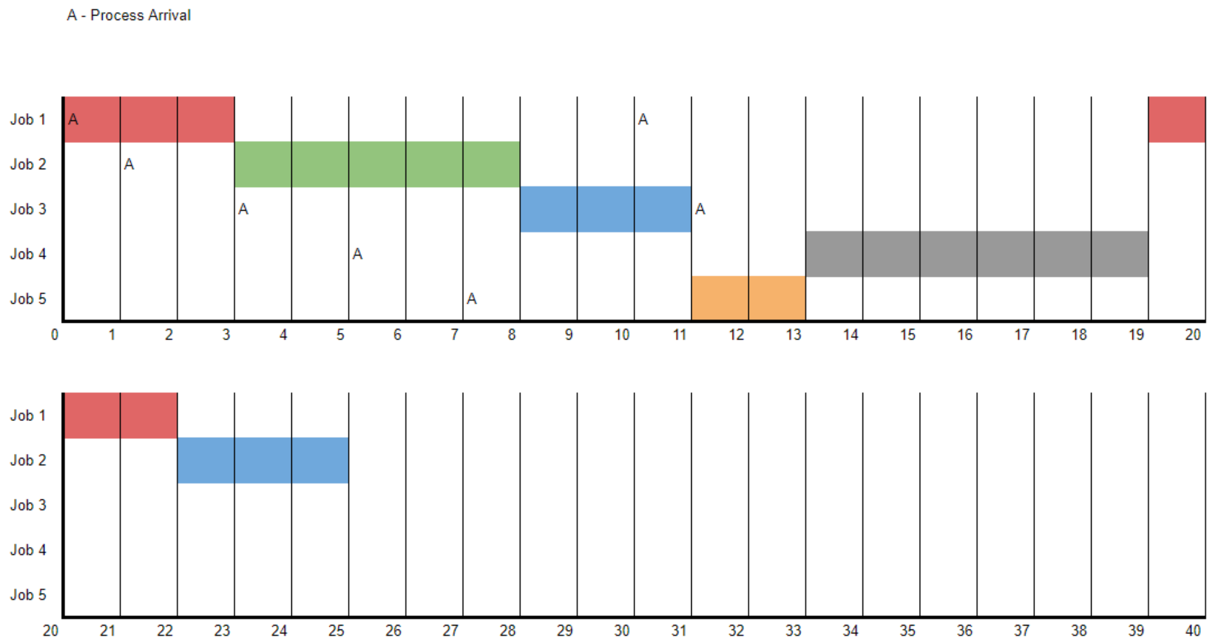
This leaves us with SRT.

c) If we assume the usage of SRT, then at Step 11, the algorithm received a new job, so it must calculate which job to execute, so it'll check the remaining execution times and pick the shortest.

So it looks at the execution time of each remaining job and checks how many units are left, and it gets: $J_1 = 2$, $J_2 = 2$, $J_3 = 4$, $J_4 = 6$, Job 5 is no longer listed.

From here we have a tie, and we can assume the procedure to untie a tie is to pick the job that arrived first, so it picks Job 2.

e)



f) If we use the definition of fairness as:

Fairness: how different is the quality of service provided to different classes of jobs. Suppose that the scheduler is serving requests from two class of users in the system. A fair scheduler would treat the two classes equally.

This to me means that, the average slowdown for each Job is about the same, so they are all slowed down about the same time.

With that in mind, it'd be impossible to produce a set of parameters for which HSN behaves unfairly with respect to shorter jobs, as the algorithm, by design, puts preference on whatever job has the next highest slowdown, and since the smaller jobs gain more slowdown faster, regardless of where we'd put them, the algorithm would always end up benefiting them by the simple math behind slowdown calculations.

All table calculations obtained via Google Spreadsheets, you can find the spreadsheet here: `https://bit.ly/35nUWNA`

TABLE 1.1

SRT

	Instance 1	Instance 2
Job 1	15	12
Job 2	26	26
Job 3	3	3
Job 4	2	2

AVRG	11.13
STD	10.4
VAR	108.13
RNG	24

HSN

	Instance 1	Instance 2
Job 1	5	21
Job 2	21	16
Job 3	7	7
Job 4	6	5

AVRG	11
STD	7.11
VAR	50.57
RNG	16

SJN

	Instance 1	Instance 2
Job 1	5	13
Job 2	26	26
Job 3	9	9
Job 4	3	2

AVRG	11.63
STD	9.56
VAR	91.41
RNG	24

Round-Robin

	Instance 1	Instance 2
Job 1	11	20
Job 2	26	26
Job 3	13	9
Job 4	8	10

AVRG	15.38
STD	7.52
VAR	56.55
RNG	18

TABLE 1.2

SRT

	Instance 1	Instance 2
Job 1	3	2.4
Job 2	3.25	3.25
Job 3	1	1
Job 4	1	1

AVRG	1.99
STD	1.09
VAR	1.18
RNG	2.25

HSN

	Instance 1	Instance 2
Job 1	1	4.2
Job 2	2.625	2
Job 3	2.333333333	2.333333333
Job 4	3	2.5

AVRG	2.5
STD	0.9
VAR	0.82
RNG	3.2

SJN

	Instance 1	Instance 2
Job 1	1	2.6
Job 2	3.25	3.25
Job 3	3	3
Job 4	1.5	1

AVRG	2.33
STD	0.99
VAR	0.98
RNG	2.25

Round-Robin

	Instance 1	Instance 2
Job 1	2.2	4
Job 2	3.25	3.25
Job 3	4.333333333	3
Job 4	4	5

AVRG	3.63
STD	0.88
VAR	0.77
RNG	2.8

TABLE 2.1

AVG	8.9
STD	3.03
VAR	9.2
MAX	15
MIN	3
RNG	12

TABLE 2.2

AVG	1.07
STD	0.25
VAR	0.06
MAX	2
MIN	1
RNG	1