AKASH SURTI

SHORTEST JOB FIRST

STATS

```
****************************************** Simulation Number:  1 ******************************************
-------------------------------------------------------------------------------------------------------------
| process # id | Service Time | Interarrival Time | Arrival Time | Service Start Time | Service Complete Time |
-------------------------------------------------------------------------------------------------------------
| process #  0 |    1.9741    |       2.3566      |    2.3566    |       2.3566       |         4.3307        |
| process #  1 |    0.0627    |       6.9310      |    9.2876    |       9.2876       |         9.3503        |
| process #  2 |    0.6799    |       4.9449      |   14.2325    |      14.2325       |        14.9125        |
| process #  3 |    2.7897    |       0.4418      |   14.6743    |      14.9125       |        17.7022        |
| process #  4 |    1.4366    |       0.4463      |   15.1206    |      17.7022       |        19.1388        |
| process #  5 |    0.4375    |       5.9860      |   21.1066    |      21.1066       |        21.5441        |
| process #  6 |    2.9458    |       4.9335      |   26.0402    |      26.0402       |        28.9859        |
| process #  7 |    1.2199    |       3.3073      |   29.3475    |      29.3475       |        30.5674        |
| process #  8 |    0.2220    |       3.9083      |   33.2558    |      33.2558       |        33.4778        |
| process #  9 |    0.4960    |       1.1514      |   34.4072    |      34.4072       |        34.9032        |
-------------------------------------------------------------------------------------------------------------

     Mean Service Time =   1.2264
              Varience =   0.9821

Mean Interarrival Time =   3.4407
              Varience =   4.7765

        Mean Wait Time =   0.2820
     Maximum Wait Time =   2.5816
```

TEXT

SOME CODE COMPUTING STATISTICS

```
        // Calculate the deviation of each serviceTime
        // and then square the value
        varServiceTime += pow(cpp->serviceTime - meanServiceTime, 2);

        // Calculate the deviation of each interarrivalTime
        // and then square the value
        varInterarrivalTime += pow(cpp->interarrivalTime - meanInterarrivalTime, 2);

        // Caluclate the amount of time a process has to wait
        // before being initiated
        cpp->waitTime = cpp->serviceStartTime - cpp->arrivalTime;

        // Calculates the sum of all waitTimes
        meanWaitTime += cpp->waitTime;

        // Finding the maximumWait
        if(maximumWait < cpp->waitTime){
          maximumWait = cpp->waitTime;
        }
    }

    varServiceTime = varServiceTime/N;
    varInterarrivalTime = varInterarrivalTime/N;

    meanWaitTime = meanWaitTime/N;
```

SHORTEST JOB FIRST!

- Shortest job next is advantageous because of its simplicity and because it minimizes the average amount of time each process has to wait until its execution is complete.
- A disadvantage of using SJF is that the total execution time of a job must be known before execution. While it is not possible to perfectly predict execution time, several methods can be used to estimate the execution time for a job.

SHORTEST JOB FIRST!

BEFORE QUEUE

PRIORITY QUEUE

process # id	Service Time	process # id	Service Time
process # 0	1.7971	process # 8	0.0422
process # 1	3.5894	process # 3	1.2832
process # 2	4.9480	process # 5	1.3322
process # 3	1.2832	process # 4	1.5205
process # 4	1.5205	process # 6	1.5419
process # 5	1.3322	process # 0	1.7971
process # 6	1.5419	process # 1	3.5894
process # 7	4.7008	process # 9	4.4195
process # 8	0.0422	process # 7	4.7008
process # 9	4.4195	process # 2	4.9480

SO THIS HAPPENED . . .

process # id	Service Time	Interarrival Time	Arrival Time	Service Start Time	Service Complete Time
process # 0	1.6238	1.7698	1.7698	1.7698	3.3936
process # 1	5.3989	5.6157	7.3856	7.3856	12.7845
process # 2	1.7449	4.4841	11.8696	12.7845	14.5293
process # 3	4.4776	2.7956	14.6652	14.6652	19.1428
process # 4	0.0950	1.9490	16.6143	19.1428	19.2378
process # 5	0.4252	5.9938	22.6080	22.6080	23.0332
process # 6	2.3465	3.1284	25.7364	25.7364	28.0830
process # 7	0.2115	7.6142	33.3506	33.3506	33.5621
process # 8	4.7562	2.4059	35.7565	35.7565	40.5127
process # 9	0.0051	12.5794	48.3359	48.3359	48.3410

Segmentation fault: 11

AND I FIXED IT!

process # id	Service Time	Interarrival Time	Arrival Time	Service Start Time	Service Complete Time
process # 0	2.1930	3.9003	3.9003	3.9003	6.0933
process # 1	0.0516	0.4266	4.3269	6.0933	6.1449
process # 2	6.3433	0.7724	5.0993	6.1449	12.4882
process # 3	3.8458	0.5640	5.6633	12.4882	16.3341
process # 4	0.8952	5.0929	10.7562	16.3341	17.2293
process # 5	0.8842	1.5708	12.3269	17.2293	18.1135
process # 6	2.5062	3.1987	15.5257	18.1135	20.6197
process # 7	0.9652	10.8125	26.3382	26.3382	27.3034
process # 8	2.3216	1.9507	28.2889	28.2889	30.6105
process # 9	0.0816	7.5307	35.8197	35.8197	35.9013

```
0.051607 0.081599 0.884204 0.965173 0.895234 6.343350 2.506216 3.845848 2.321610 2.192967
highest priority item =   0.0516
0.081599 0.895234 0.884204 0.965173 2.192967 6.343350 2.506216 3.845848 2.321610
highest priority item =   0.0816
0.884204 0.895234 2.321610 0.965173 2.192967 6.343350 2.506216 3.845848
highest priority item =   0.8842
0.895234 0.965173 2.321610 3.845848 2.192967 6.343350 2.506216
highest priority item =   0.8952
0.965173 2.192967 2.321610 3.845848 2.506216 6.343350
highest priority item =   0.9652
2.192967 2.506216 2.321610 3.845848 6.343350
highest priority item =   2.1930
2.321610 2.506216 6.343350 3.845848
highest priority item =   2.3216
2.506216 3.845848 6.343350
highest priority item =   2.5062
3.845848 6.343350
highest priority item =   3.8458
6.343350
highest priority item =   6.3433
```

process # id	Service Time	Interarrival Time	Arrival Time	Service Start Time	Service Complete Time
process # 1	0.0516	0.4266	0.0000	0.0000	0.0000
process # 9	0.0816	7.5307	0.0000	0.0000	0.0000
process # 5	0.8842	1.5708	0.0000	0.0000	0.0000
process # 4	0.8952	5.0929	0.0000	0.0000	0.0000
process # 7	0.9652	10.8125	0.0000	0.0000	0.0000
process # 0	2.1930	3.9003	0.0000	0.0000	0.0000
process # 8	2.3216	1.9507	0.0000	0.0000	0.0000
process # 6	2.5062	3.1987	0.0000	0.0000	0.0000
process # 3	3.8458	0.5640	0.0000	0.0000	0.0000
process # 2	6.3433	0.7724	0.0000	0.0000	0.0000

HOW I DID IT

```
QueuePointer test(QueuePointer qp){
  PriorityQueuePointer pq = createPriorityQueue(qp->length);
  QueuePointer qpp = createQueue();
  NodePointer np = qp->pointerToHead;

  for(int i = 0; i < qp->length; i++){
    ProcessPointer pp = np->processPointer;
    double input = pp->serviceTime;

    pqEnqueue(pq, input);
    np = np->pointerToPrevNode;
  }

  printf("\n");
  np = qp->pointerToHead;

  while( !isPriorityQueueEmpty(pq) ) {
    double n = pqDequeue(pq);
    printf( "highest priority item = %8.4f\n", n );

    ProcessPointer pp = sort(qp, n);
    enqueue(qpp, pp);
  } // while

  printf("\n");

  return qpp;
}
```

```
ProcessPointer sort(QueuePointer qp, double n){
  NodePointer np = qp->pointerToHead;
  ProcessPointer ppp = NULL;

  for(int i = 0; i < qp->length; i++){
    ProcessPointer pp = np->processPointer;
    double input = pp->serviceTime;

    if(input == n){
      ppp = pp;
      printProcess(ppp);
      return ppp;
    }
    np = np->pointerToPrevNode;
  }// for

  return NULL;
}
```