Department of Computer Science

Computer Science 177: Modelling and Simulation

MIDTERM EXAMINATION — Spring 1999

1. Since event list manipulation takes time, some programmers have tried using the following ‘‘trick’’ for avoiding the need for explicit *customer arrival events* in cases where their models contain a queue with limited waiting room. The idea is based on ‘‘looking into the future’’ to keep the waiting room full. That is, the waiting room is ‘‘filled’’ once in the initialization routine, and thereafter is kept ‘‘full’’ by replacing each departing customer with a new one in the departure routine. The pseudocode for accomplishing these two tasks is as follows:

   In the INITIALIZE routine:

   ```
   LastArrival = Now = 0.0
   for i = 1 .. QueueLimit
        LastArrival += InterArrivalTime
        Add customer to waiting room; ArrivalTime=LastArrival
   end for
   ```

   In the DEPARTURE routine:

   ```
   Remove departing customer from waiting room
   loop
        LastArrival += InterArrivalTime
        if (????) then
             new customer was lost (waiting room was full)
        else
             Add customer to waiting room; ArrivalTime=LastArrival
             exit from loop
        end if
   end loop
   ```

   a. What is the missing condition in the `if` statement?

      In general, the arrival will be lost if it occurred before there is room in the waiting area to accept it. Since there is a departure `Now`*, the customer will be lost if* `(LastArrival < Now)`.

   b. The event-reduction idea can be extended to eliminate all *start service events* from the program too. Indicate what pseudocode needs to be added to the INITIALIZE and DEPARTURE routines for accomplishing this.

      The system is similar to the automated banking machine you did in the first program. In a “normal” event-driven simulation, the start service event would take place as soon as the customer reaches the front of the waiting line, which happens immediately if the system was empty when the customer arrived (where it would be triggered by the customer arrival event), or at the departure of the previous customer otherwise (where it would be triggered by the departure routine). What we are trying to do now is to extend the “trick” to eliminate all start service events in addition to eliminating the customer arrival events. This is done by scheduling the first DEPARTURE in the INITIALIZE routine, at the sum of the first customer’s arrival time plus its service time. The remaining customers will be handled by the DEPARTURE routine, where the next DEPARTURE will take place at the sum of the maximum of `Now` *and the next customer’s arrival time, plus its service time.*

2. Consider the problem of modelling a traffic signal that controls a pedestrian crosswalk, similar to the one next to the UCR campus near the corner of University Ave and Canyon Crest Dr, between the baseball field and Bannockburn Village. To simplify the problem, we will consider only the people crossing the road and ignore the cars. In addition, we assume the light runs on a fixed schedule and pedestrians do not push a button to activate the light.

   Assume that the state of the signal (as seen by the pedestrians) is one of "walk" (which lasts for 1 minute), "caution" (which lasts for 30 seconds), or "don't walk" (which lasts for 3 minutes). The state will be stored in a global variable `crosswalk` that is controlled by a single CSIM `traffic_light` process and read by each `pedestrian` process. In addition, `traffic_light` process sets an event `walk_now` each time the state of the signal changes from "don't walk" to "walk" so that any waiting pedestrians will know it is time to start crossing the street.

   Also assume that pedestrians only begin crossing the street when the state of the signal is "walk", and that the crosswalk is wide enough to allow all pedestrians to cross at the same time, so there is no need for the pedestrians to communicate between themselves.

   a. Suppose a pedestrian arrives when the state of the signal is not "walk". In this case, it cannot begin crossing the street until the state changes to "walk", at which time the `traffic_light` process will set the `walk_now` event. Should the pedestrian use the CSIM `wait()` or `queue()` construct to wait for the `walk_now` event? Explain your answer.

      The pedestrian process should use the CSIM `wait()` *construct, because all waiting pedestrians are allowed to start walking together when the crosswalk changes to state "walk". If the CSIM* `queue()` *construct were used, then only one pedestrian process would see the* `walk_now` *event, which would be incorrect.*

   b. Explain how and when the `walk_now` event should get cleared. Consider all cases, including 0, 1, or more than 1 pedestrian waiting when the state changes to "walk".

      The `walk_now` *event may get cleared right away, if one or more* `pedestrian` *processes were waiting for the state of the crosswalk to change. However, if no* `pedestrian` *processes were waiting, then the* `walk_now` *event would remain indefinitely unless the* `traffic_light` *process clears the event. This could be done immediately or delayed until just before the* `traffic_light` *process changes the state of the crosswalk to "caution".*

   c. Write the function definition for the CSIM `traffic_light` process.

```
void traffic_light()
{
        create ("traffic_light");

        while(1)                    // loop forever
        {
                crosswalk = walk;
                walk_now.set();
                hold(60);
                walk_now.clear();
                crosswalk = caution;
                hold(30);
                crosswalk = dont_walk;
```

```
                hold(180);
        }
}
```

d. Write the function definition for a pedestrian.

```
void pedestrian()
{
        create ("pedestrian");
        if (crosswalk != walk)
           walk_now.wait();
        hold(crossing_time);
}
```

3. The 7:20am express bus always seems to arrive late at your stop. Over five consecutive days last week, it arrived 1 min, 5 min, 6 min, 4 min, and 9 min late, respectively.

   a. Find the *sample mean* and *sample variance* for its lateness.

$$\bar{X}(5) = \frac{1+5+6+4+9}{5} = 5.0$$

$$S^2(5) = \frac{(1-\bar{X}(5))^2 + (5-\bar{X}(5))^2 + (6-\bar{X}(5))^2 + (4-\bar{X}(5))^2 + (9-\bar{X}(5))^2}{4} = 8.5$$

   b. Use the Student's *t*-distribution to construct an approximate 95% Confidence Interval for the average lateness. What assumption(s) did you have to make about the underlying distribution for the *coverage* of the confidence interval to be exact?

   Since we have 5 samples, we need the t-distribution with 4 degrees of freedom and one-sided critical point $\gamma = 0.9750$ *from the attached table, i.e.,* $t_{4,0.9750} = 2.776$. *Thus, using the final formula on the exam page, we have that the required confidence interval is*

$$\bar{X}(n) \pm t_{4,0.9750}\sqrt{S^2(5)/5}$$

   or

$$5.0 \pm 3.62$$

   The Student's t-distribution is based on two assumptions. First, the individual data samples have a normal distribution. Second, we use $S^2(n)$ *calculated from the data samples instead of* σ^2 *in the confidence interval. Since the t-distribution is defined to use the sample variance, the only real assumption we have made is that the individual lateness measurements are approximately normal.*

4. Find the *maximum likelihood estimators* for fitting a *uniform distribution* to the lateness data in the previous problem.

   This is a direct application of lecture material, where we showed that the likelihood function for the uniform(a, b) *distribution is maximized when* $a = \min\{X_i\} = 1$ *and* $b = \max\{X_i\} = 9$.

5. Consider a regular 12-sided die that has the number 1 written on 3 faces, the number 2 written on 2 faces, the number 4 written on 6 faces and the number 7 written on 1 face. Let X be a discrete random number whose value is set by tossing this die.

a. Find its density function $f_X(x)$.

$$f_X(x) = \begin{cases} 1/4 & \text{if } x = 1 \\ 1/6 & \text{if } x = 2 \\ 1/2 & \text{if } x = 4 \\ 1/12 & \text{if } x = 7 \\ 0 & \textit{otherwise} \end{cases}$$

b. Find its cumulative distribution function $F_X(x)$.

$$F_X(x) = \begin{cases} 0 & \text{if } x < 1 \\ 1/4 & \text{if } 1 \le x < 2 \\ 5/12 & \text{if } 2 \le x < 4 \\ 11/12 & \text{if } 4 \le x < 7 \\ 1 & \text{if} x \ge 7 \end{cases}$$

c. What is the probability that the number 4 comes up twice in a row?

Since the two rolls of the die are independent, the probability of getting the number 4 twice is product of the probabilities of getting the number 4 once, which is

$$f_X(4) \cdot f_X(4) = 1/4.$$

Formulas:

$$\bar{X}(n) = \frac{1}{n} \sum_{i=1}^{n} X_i$$

$$S^2(n) = \frac{1}{(n-1)} \sum_{i=1}^{n} (X_i - \bar{X}(n))^2$$

$$Prob[abs(\bar{X}(n) - \mu)/\sqrt{\sigma^2/n} < Z_{(1-\alpha/2)}] = 1 - \alpha$$

$$Prob[abs(\bar{X}(n) - \mu)/\sqrt{S^2(n)/n} < t_{(n-1),(1-\alpha/2)}] = 1 - \alpha$$

 END OF EXAMINATION