Programmeren (FEB22012)

8. Exercise

Deadline for submission: 2011-10-02 23:59 CET

Instructions

Queues as a data structure are very useful in simulating various systems where events or objects arrive according to a certain distribution, and are processed according to another. We could implement queues as objects in Matlab, but in this exercise we will explicitly store the required information in separate variables queue, head, and tail.

Exercise, part 1: implement the data structure

Implement the required functions for queue handling: creation of a queue, enqueuing and dequeueing values:

```
function [queue, head, tail] = createQueue(arraySize)
function [queue, head, tail] = enqueue(element, queue, head, tail)
function [queue, head, tail] = dequeue(queue, head, tail)
```

There is no need to explicitly access the next element of the queue as that is trivially available in `queue(tail)`. `createQueue` needs to create a queue stored in an array of a certain size. `Enqueue` adds the given element to the queue, and `dequeue` removes one element from the queue. Note that all these functions have pre-conditions: `createQueue` should have positive `arraySize`, `enqueue` should be allowed only on queues with free slots in the array, and `dequeue` should not be called on empty queues. Document and assert all the pre-conditions.

Exercise, part 2: simulate your grading time

As you've probably noticed, it takes the teaching assistants a reasonable time to grade the assignments. To analyze this, we will make a simple queue simulation model. The amount of solutions submitted in the first 5 exercises is:

Exercise	1	2	3	4	5
Count	90	93	68	92	86

Let us make a simplifying assumption that the exercises arrive daily according to a Poisson distribution (with mean daily arrival time mean(count) / 7). We also assume a perfectly just world where the solutions are graded according to first-come first-served principle. We have three teaching assistants correcting the exercises. Each of them have been allocated 5h of grading / week (and yes, they work 7 days / week). The time needed to grade a single exercise is exponentially distributed with mean time 15 minutes.

Implement a script for simulating the grading process. Our intention is to keep track of time it takes to grade each exercise so that in the end of the simulation we can compute statistics of the grading times. Also, after the 5 weeks have passed the grading should continue until all the remaining exercises in the queue have been graded (no matter how long it takes). The simulation should look more or less like the following pseudocode:

```
initialize required constants (arrival rate, grading time, time to grade)
initialize queue of max size 1024
initialize day
while(queue is not empty or 5 weeks have not yet passed)
        if (5 weeks have not passed yet)
                sample amount of exercises to arrive today
                add exercises to the queue
        endif
        initialize time left to grade
        while(teaching assistants have time to grade
                        and there are assignments left to grade)
                t = sample grading time
                reduce time left to grade by t
                store exercise submission time (queue(head)) and grading time (day)
                dequeue one exercise
        endwhile
        increase day
endfor
```

After the simulation has finished you should have a matrix of values indicating for each exercise the day it was submitted and the day it was graded. Add to the script computation and printing (to screen, with `fprintf`) of the following statistics:

- Maximum time it took to grade an exercise
- Median time it took to grade an exercise
- Maximum amount of exercises there were in the queue at the end of a day

Add also plotting with a bar plot (see function `bar`) the queue size at the end of each day of the simulation period.