Programming Assignment 4

CMPE 250, Data Structures and Algorithms, Fall 2014

Instructor: A. T. Cemgil
TA's: Barış Kurt, Atakan Arıkan

Due: 04 January 2015, 23:59

Discrete Event Simulation

Assume that we want to simulate a hypothetical factory that processes some jobs. The factory is composed of units, where each unit has a job queue and a worker. The worker is the main part which processes a job, and the job queue is the place where the next jobs are waiting to be processed. Once a unit processes a job, it pushes the job to one of the next possible units according to the factory layout. Each job may follow a different path in the factory and once a job is processed in an output unit, it's gone out of the system.

Factory Layout

Figure 1 shows a sample factory with 5 units. In each factory, *Unit 0* is the single unit that act as the input unit. It accepts jobs and starts their processing. The units with incoming and outgoing connections are the intermediate units, such as *Unit 1* and *Unit 2* in this example. There may be many output units, which does not have an outgoing connection to another unit. In this example *Unit 3* and *Unit 4* are the output units. The layout of each factory is going to be represented with graph structure.

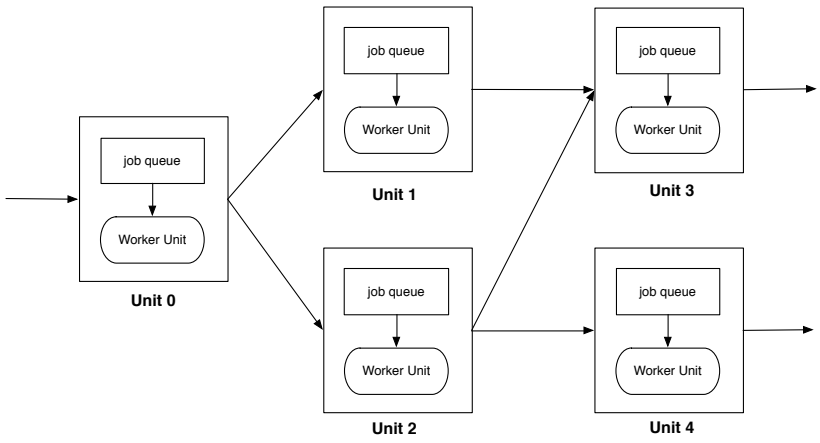

Figure 1: A sample factory.

How factory works?

The factory start working by waiting for the jobs which arrive sequentially but not at the same time. The units may be either idle, waiting for jobs, or busy processing them. We assume that each unit has a constant processing time for each job. After the processing is complete, the job is passed to one of the next units in the factory by two different ways.

1. Randomly: Next unit is selected uniformly randomly. That's if a unit has n outgoing connections, the job will be assigned each of the next units with probability $1/n$.
2. Unit with shortest queue first: The unit with the shortest queue length will be get the job. If the queue lengths are equal, assign the job to the unit that's first mentioned in the adjacency list.

What to report?

You are expected to simulate a given factory and a list of jobs with two different job assignment schemes and report the following for each simulation:

1. Total running time of the factory.
2. Utilization of each unit
3. Turnaround time of each order
4. Maximum length of each queue

$$\text{Unit Utilization} = \frac{\text{Busy time of the unit}}{\text{Total running time of the factory}}$$

$$\text{Turnaround Time} = \text{Finish time of the job} - \text{Arrival time of the job}$$

Generating a Random Process

In this project you are going to use random numbers to decide which units will be chosen to process jobs. In reality, there's no random number generation mechanism in any software or hardware system. Instead, we use pseudo-random number sequences: a list of numbers generated by a deterministic algorithm that seem like random to us. In *C++*, you can try this code, which sets the seed to system time in the begining and generates different sequence of numbers every time it's executed:

```
#include <cstdlib>
#include <ctime>
#include <iostream>
int main(){
  std::srand(std::time(0));
  for(int i=0; i<10; i++){
    std::cout<<std::rand()<<std::endl;
  }
  return 0;
}
```

Hint: If you want to generate the same random numbers each time you run your proram, try giving the system the same seed each time.

Input and Output Details

You're going to read one input file, and create two output files. Your program will be executed **exactly** with the following command:

```
./project4 [input_file] [output_file_1] [output_file_2]
```

where *input_file* contains the factory layout and job arrival times, and *output_file_1* contains the output of the factory with random unit choice, and *output_file_2* contains the output of the factory with shortest queue unit choice.

The input file format:

1. First line is the number of units in the system, let's say N.
2. The next N lines are the factory layout represented as an adjacency list. Each line has the following format:

   ```
   UnitNumber ProcessingTime [List of preceeding units] ...
   ```

3. The next line is the number of jobs, let's say J.
4. The next J lines are the arrival times of jobs. They are positive values in double precision.

The output file format:

1. First line is the total running time of the factory.
2. The next N lines are the utilization of units and max queue lengths.
3. The next J lines are the turnaround time of jobs.

You can see sample input/output files among the test cases.

Evaluation of Outputs

The first output files that you generated will most probably be different in each case, but the utilization times and total run time of the factory will be similar. The second output files are created by a deterministic process, so they should exactly the same.

Please do not just implement the program. Also try to understand and compare the outputs. Try to see what makes the difference. Is selecting the unit with shortest queue really an optimal choice? What can be a better unit selection algorithm?

Submission Details

You are supposed to use the Git system provided to you for all projects. No other type of submission will be accepted. Also pay attention to the following points:

- All source codes are checked automatically for similarity with other submissions and exercises from previous years. Make sure you write and submit your own code.

- You are expected to use C++ as powerful, steady and flexible as possible. Use mechanisms that affects these issues positively.
- Make sure you document your code with necessary inline comments, and use meaningful variable names. Do not over-comment, or make your variable names unnecessarily long.