ORIE 3120 Spring 2020
Recitation 6
Discrete Event Simulation

Learning Objectives

- ☐ Understand how simulation is used in medicine and healthcare logistics.
- ☐ Understand how simulation works.
- ☐ Be able to analyze simulator output.
- ☐ Be able to create a new simulator that has novel functionality, based on a starting simulator.
- ☐ Use simulation as a tool to make recommendations to a hospital administrator about purchasing decisions for equipment in the laboratory.

Hospital Laboratory Simulator

You work as a healthcare operations analyst at a hospital, and you are helping to plan equipment purchases for the hospital's laboratory, which runs medical tests for patients. You are currently focusing on one particular machine, which runs a specialized test in batches.

This machine is used to process blood samples that arrive from around the hospital. If the machine is idle when a sample arrives, the sample is put into the machine, and the machine performs the test. If, on the other hand, the machine is in use when a blood sample arrives, the sample is placed in a queue, and waits until the machine is available. As soon as the machine is available, blood samples from the queue are placed into the machine and the machine is restarted. The machine can process at most 2 samples at a time.

Figure 1: Blood gas analyzer. Photo source https://en.wikipedia.org/wiki/Arterial_blood_gas. This machine is used to run arterial blood gas (abg) tests on blood samples taken from patients. This test measures gases dissolved in blood, and is an important diagnostic tool in critical care and pulmonary medicine.

The spreadsheet Rec06_Simulation.xls simulates this process. It is currently set to simulate for 1000 minutes (about 16 hours). In this simulation, the machine takes 25 minutes to process each sample, and the time between arrivals of each sample (the interarrival time) is a random variable that is uniformly distributed between 10.5 minutes and 17 minutes.

You want to understand whether the current machine is fast enough to process all of the tests that the hospital needs to handle in a timely fashion. If it isn't, then the number of tests waiting to be processed can grow large. This is a problem for several reasons:

- First, if doctors have to wait too long for patients' test results, this can delay important medical decisions. This is particularly important for tests involved in critical care, such as the blood gas analyzer from Figure 1, because a test result may be a critical piece of information in figuring out which course of action is most likely to save a patient's life, and because there is a narrow time window in which to take action.
- Second, some tests depend on the freshness of the sample for accuracy. For example, in the blood gas test context, if a test is delayed by more than 30 minutes after drawing a blood sample from a patient, then a different more complicated procedure involving refrigeration must be used to preserve the gas content of the blood and ensure accuracy. [Source: https://en.wikipedia.org/wiki/Arterial_blood_gas]
- Third, when there are lots of samples waiting to be tested, the lab can run out of room to store them safely.

Initial investigation and simulation variability

Let's first look at the number of samples that are waiting to be tested. We call this "Number of Items in the Queue". "Queue" is British English for what we usually call a "line" in the US, and is used in operations research to refer to a collection of items waiting to be processed.

Before running the simulator, you'll have to enable macros in Excel. First, if Excel says, "PROTECTED VIEW: Be careful --- files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View" then click "Enable Editing." Then, if Excel says, "SECURITY WARNING: Macros have been disabled," click "Enable Content."

Now you are ready to run the simulator.

Run the simulator by going to Sheet1, clicking on "Analyze and Run", then clicking "Rebuild Model", "Reset", then "Run".

This erases and then fills in the SimTrace sheet, with information about the simulation run. You can also use it to step through the simulation one event at a time by clicking "Run Next Event." This is very useful for understanding how the simulation works, and for debugging new simulations that you will build later.

The spreadsheet already calculates the number of items in the queue, and some useful statistics of that value over time: the max, the min, the mean, and the standard deviation, in column F. In particular, the entry in cell F2 is the maximum number of samples in the queue during the simulation run, and cell F4 is the average number.

Here, "average" means that if we were to look at how many items were in the queue at every millisecond during the simulation, and put these numbers into a really long list, it would be the average value of these numbers, i.e., the sum of the list of numbers divided by its length. The SimTrace does not have an entry for every millisecond during the simulation --- only at those moments when an event happened --- so we can't simply average the numbers in column F to obtain column F. Fortunately, the code that generates SimTrace does it for us.

SimTrace is not easily editable, and is erased after each run, because of the way that the simulation code works. To allow you to manipulate the numbers on this spreadsheet, there is another sheet called "Editable SimTrace" that has a reference to all of cells in SimTrace, and that is a regular Excel sheet that you can add to.

Run the simulator 3 different times and for each simulation record below or in a separate location the value of the mean number of items in the queue (on the SimTrace spreadsheet in cell F4), and the max number of items in queue (on the SimTrace spreadsheet in cell F2).

	Mean # of Items in the Queue	*Max # of Items in the Queue*
Run 1		
Run 2		
Run 3		

Randomness in the simulator causes these numbers to differ from run to run.

You should see the mean number of items in queue is usually in the range of 0.8 to 1.1, and the max is usually 2 or 3. This is a good value for these numbers --- if the number of samples in queue gets to be more than 5 then this can be an issue because we run out of storage space, but it seems that this almost never happens according to the simulation.

Two queues: high and low priority

Arterial blood gas tests are used both in extremely high priority situations, where a life or death decision must be made in a short time window, and in lower priority (but still important) situations, to diagnose and monitor conditions like COPD and asthma. In the former case, it is critical to get the test done quickly, while in the second case it is not a great inconvenience to wait an hour or more for the result.

We will change the simulation so that we have two queues: a low priority queue and a high priority queue. When the machine becomes empty, in our new simulation, we will give priority to the items in the high priority queue, so that they enter the machine first, and then more items from the low priority queue are added to the machine only if there is more room, respecting the maximum machine capacity of 2. For example, if there is one high-priority test that needs to be performed waiting in the queue when the machine finishes, and three low-priority tests, we would start one high priority test and one low priority. If there are two high-priority tests and three low-priority tests, we would start both high-priority ones, and no low priority ones.

To accomplish this, we need to have a separate arrival event and delay for high priority samples, and we need to add a state variable for tracking the number of items in each queue. We will also need to change the logic in MachineIsEmpty, Initialize, and Arrival to handle the two queues.

To do this, we'll first change Sheet1 to add new shapes for the new arrival event and the new delay, and change the names of the old shapes.

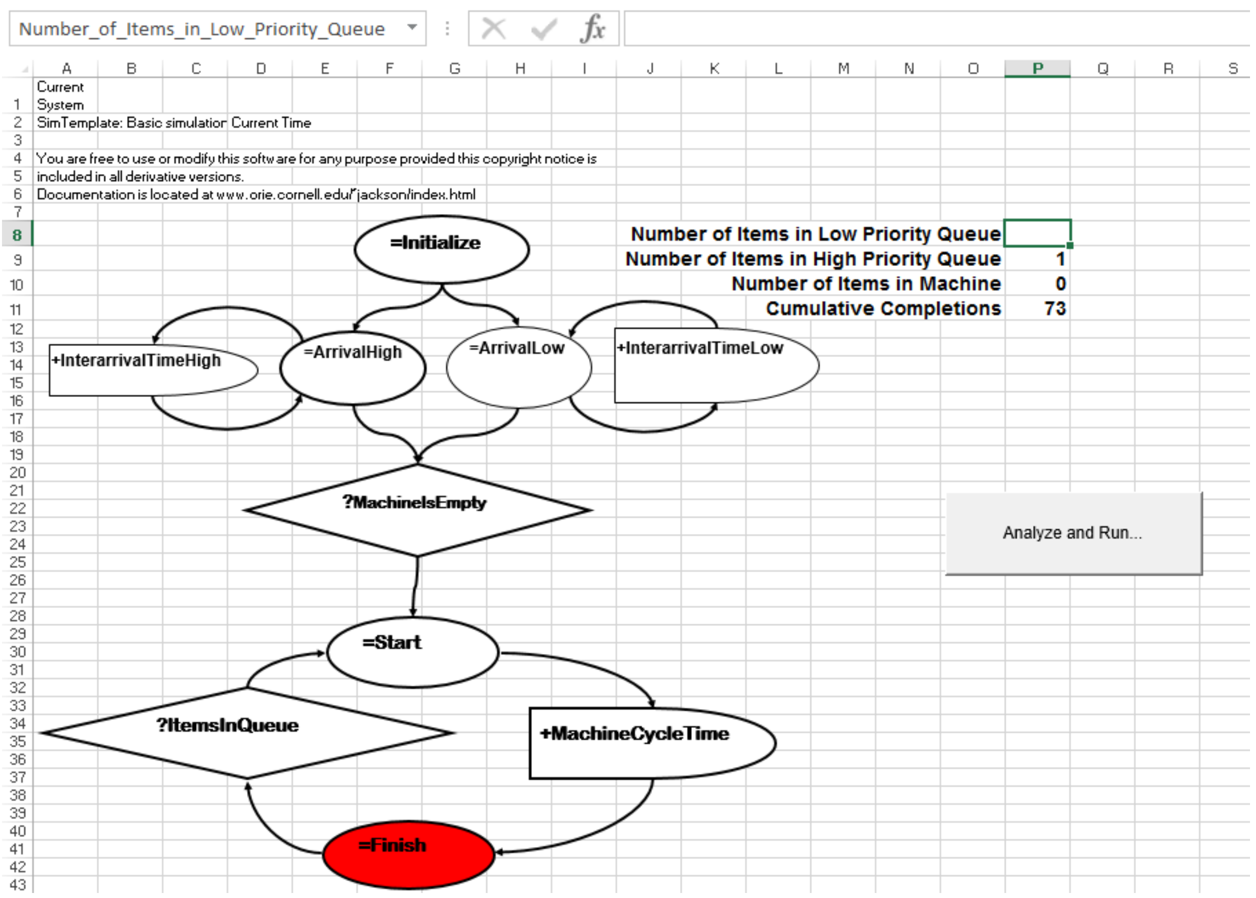
First, change the name of the oval from "=Arrival" to "=ArrivalLow". This arrival event will now represent the arrival of a low priority sample to the low priority queue. Then, change the name of the half oval / half rectangle from "+InterarrivalTime" to "+InterarrivalTimeLow". This will represent the delay between two arrivals to the low priority queue.

Second, create a new oval, with the name "=ArrivalHigh", and a new half-oval / half-rectangle with the name "+InterarrivalTimeHigh". These will represent the arrival of a high priority sample to the high priority queue, and the delay between two high priority arrivals. You can find the half-oval / half-rectangle shape under "Callouts" in the menu that lets you insert a new shape.

Warning: do not use copy & paste to create your new oval and half oval / half rectangle shapes. For reasons having to do with the way that Excel manages names of object, this will not work. Make sure to make a new shape.

Warning: don't forget the "=" and "+" in the names of these shapes. This tells the simulation VBA code whether the shape is an event or a delay.

Create arrows (it is ok to copy and paste these) that connect these shapes in the way shown in the diagram below. Make sure that you snap the arrows to the shapes, and don't simply have the ends of the arrows near the shape. You will know that it is done correctly if the arrows move with the shape when you move the shape a bit with your mouse.


Number_of_Items_in_Low_Priority_Queue

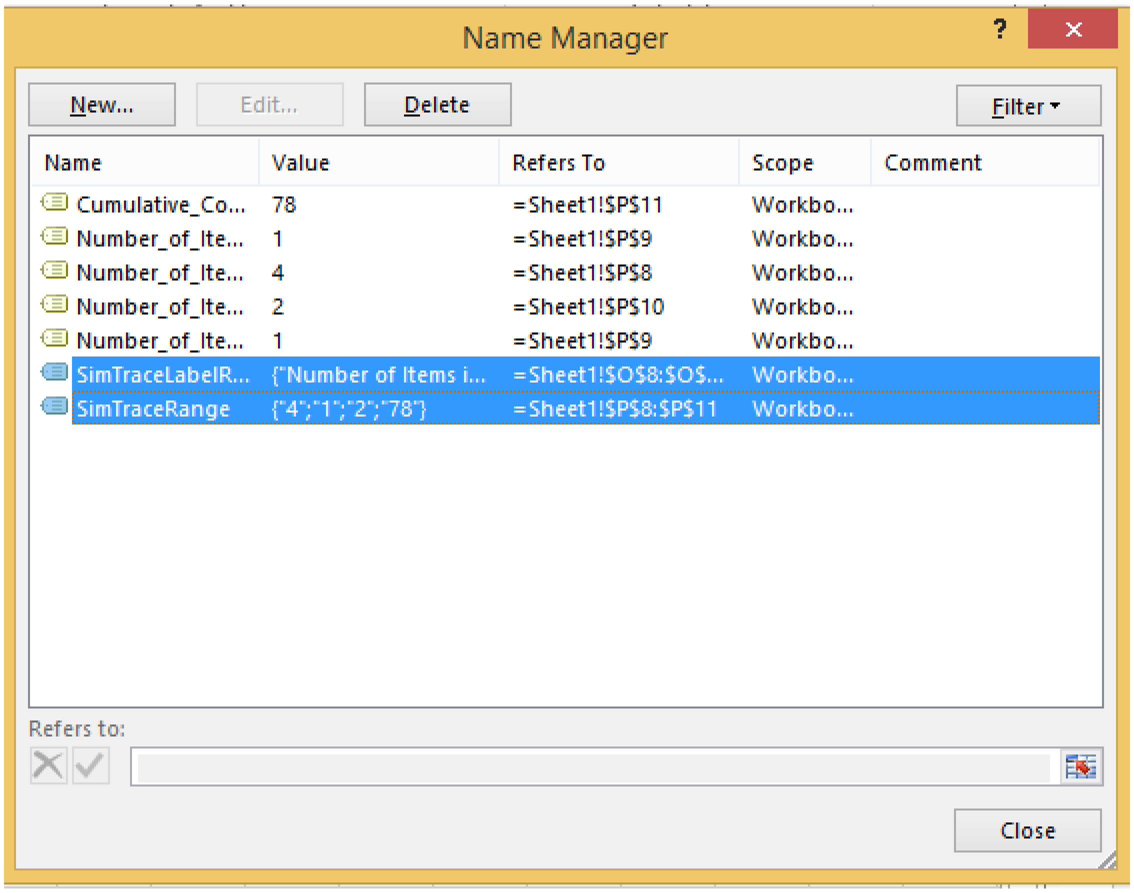
Current System
SimTemplate: Basic simulation Current Time

You are free to use or modify this software for any purpose provided this copyright notice is included in all derivative versions.
Documentation is located at www.orie.cornell.edu/~jackson/index.html

Number of Items in Low Priority Queue	
Number of Items in High Priority Queue	1
Number of Items in Machine	0
Cumulative Completions	73

You'll then need to create a new state variable, and change the name of the old state variable called Number_of_Items_in_Queue. Go over to columns O and P in Sheet1, and change the bolded text there as shown in the figure above, and change the name within cell P9 to Number_of_Items_in_Queue to Number_of_Items_in_High_Priority_Queue (by right-clicking on the cell and then clicking "Define Name"). Finally, change the name of cell P8 to Number_of_Items_in_Low_Priority_Queue, again by right-clicking on the cell and clicking "Define Name".

To let the simulation VBA code keep track of this new variable, you will also need to make two changes. Open up the Name Manager. In Excel 2013 on Windows, it is under the Formulas tab. In Excel 2017 on Mac, you can access it by right clicking a cell in Sheet1 and clicking "Define Name". In Excel 2011 on Mac, access it under the Insert tab, by clicking "Name" and then "Define." Once you open the Name Manager, change the definition of the two highlighted names below to be the highlighted values. You will be making these names to be ranges that start from O8 and P8 (the cells you just added) instead of O9 and P9 (where they were before). This will let the VBA code know the set of state variables that you want to keep track of in your trace.


Name Manager

New... Edit... Delete Filter

Name	Value	Refers To	Scope	Comment
Cumulative_Co...	78	=Sheet1!$P$11	Workbo...	
Number_of_Ite...	1	=Sheet1!$P$9	Workbo...	
Number_of_Ite...	4	=Sheet1!$P$8	Workbo...	
Number_of_Ite...	2	=Sheet1!$P$10	Workbo...	
Number_of_Ite...	1	=Sheet1!$P$9	Workbo...	
SimTraceLabelR...	{"Number of Items i...	=Sheet1!$O$8:O...	Workbo...	
SimTraceRange	{"4";"1";"2";"78"}	=Sheet1!$P$8:$P$11	Workbo...	

Refers to:

Close

Now we will edit the VBA code. Hit Alt-F11, click on "ModuleUser", and you will see the code corresponding to the simulation. (To hit Alt-F11 on most Mac laptop keyboards, you will need to hit "fn", "alt" and the "F11" key.) Each event has its own function, that governs how the state variables change when the event fires.

First, change the name of the InterarrivalTime Function to match one of the newly defined events (say, IntearrrivalTimeLow) and then make a copy of the function and set its name to the other newly defined events (InterarrivalTimeHigh). Do the same thing with Arrival, so that you have two functions that have the same code as the original Arrival function, called ArrivalHigh and ArrivalLow.

We want the low priority queue to have interarrival times uniformly distributed between 0 minutes and 45 minutes, and the high priority queue to have interarrival time distributions that are uniformly distributed between 0 and 60 minutes.

Edit the lines that determine duration in your two newly created interarrival time functions (InterarrivalTimeHigh and InterarrivalTimeLow) to create these distributions. These are both one line changes.

Also, in InterarrivalTimeHigh and InterarrivalTimeLow, we need to make sure that the correct values are returned. The way that VBA returns a value from a function is by setting a variable name with the same name as the function equal to the value that you want to return. You copied and pasted the code InterarrivalTimeHigh and InterarrivalTimeLow from the original InterarrivalTime function code, and so the final line, that was

```
InterarrivalTime = duration 'this is how you return a value
```

needs to be changed to

```
InterarrivalTimeLow = duration 'this is how you return a value
```

for the InterarrivalTimeLow function, and to

```
InterarrivalTimeHigh = duration 'this is how you return a value
```

for the InterarrivalTimeHigh function.

Now, we must take the variable Q, defined in the line:

```
Global Q As Integer 'number of items in queue
```

and make two copies of it, one for tracking the number of items in the high-priority queue, and one for tracking the number in the low-priority queue. Let's call these variables L and H respectively.

Now, we must go through the code, and wherever the code currently does something to the variable Q, we should instead do something to the variable L and H. You will need to make changes to Initialize(), ArrivalHigh(), ArrivalLow(),MachineIsEmpty(),ItemsInQueue, and OutputVariables.

When you make these changes, make sure that you implement these intended changes:

- Initialize should set the number of items in each queue to 0.
- ArrivalHigh should increment (increase by 1) the number of items in the high-priority queue.
- ArrivalLow should increment (increase by 1) the number of items in the low-priority queue.
- Start should look at the number of items in each queue, and move items from the high and/or low queue as appropriate into the machine, by decreasing the state variable for each queue size (Number_of_Items_in_High_Priority_Queue and Number_of_Items_in_Low_Priority_Queue) by the number of items removed, and increasing the state variable for the number of items in the machine by the number added.
- ItemsInQueue should a Boolean (true / false) value that is true if there is at least one item in either queue (requiring the MachineIsEmpty event to fire), and false otherwise.
- OutputVariables should output all of the state variables: L, H, P, and CumulativeCompletions.

All of these changes can be done by copying and pasting within the code a small number of lines, and by changing Q to either L or H, and should be understandable even if you have not coded before. The two exceptions are the change to Start, which requires you to write a more complicated statement, and OutputVariables, which requires some understanding of how Excel is storing its values.

To help you with Start, some example code with places to fill in is pasted below:

```
Function Start()
'this function represents the start of the machine cycle
If H > 2 Then
  'We have more than two items in the high priority queue.  Start only high priority items.
  'FILL IN THIS CODE, setting P and H.  L will not change.
ElseIf L + H > 2 Then
  ' We have 2 or fewer high priority items, and more than 2 items overall.
  ' We will start H high priority items (all of them), and 2-H low priority ones.
   'FILL IN THIS CODE, setting P, L and H.
Else:
  ' We have 2 or fewer items overall.  Start all of them.
   'FILL IN THIS CODE, setting P, L and H.
End If
OutputVariables
End Function
```

Here is what OutputVariables should look like:

```
Function OutputVariables()
Worksheets("Sheet1").Range("Number_of_Items_in_Low_Priority_Queue").Value = L
Worksheets("Sheet1").Range("Number_of_Items_in_High_Priority_Queue").Value = H
Worksheets("Sheet1").Range("Number_of_Items_in_Machine").Value = P
Worksheets("Sheet1").Range("Cumulative_Completions").Value = CumulativeCompletions
End Function
```

Each line puts a state variable (L, H, P, or CumulativeCompletions) into sheet 1, where the cell into which it is placed is determined by the name that the cell has in the sheet, rather than it's alphanumeric coordinate (such as A1, or C9). For example, the name Number_of_Items_in_Low_Priority_Queue is the name that you defined earlier in this lab.

When all of this is complete, click "Analyze and Run", then "Rebuild model". If no errors happen, and if SimLog says "Model structure is valid.", then you are part of the way there. If instead you have an error, see if you can figure out why.

Instead of just clicking "Run", click "Run Next Event" several times, and watch what your simulation is doing, in Sheet1 and SimTrace. Does it make sense? Watch the sequence of events that fire, and the state variables. Are they behaving as you intended? If things look good, hit "Run" to run your code all the way through, again asking yourself if the behavior is as expected.

If there is anything strange, see if you can figure out what is going on, by looking at your code. Even if your code doesn't report an error, a bug in your code may cause it to go into an infinite loop, or produce output that isn't what you intended. You may run into challenges getting your code to be correct. Ask a TA for help if you have trouble figuring out why your code isn't working.

Otherwise, congratulations, your code is working!

Use your simulation to calculate the average time in queue for items in the low priority queue, and items in the high priority queue. Take a look at the mean value in the SimTrace sheet of Number of Items in High Priority Queue, and Number of Items in Low Priority Queue. Which one is larger? Which one should be larger based on the way the simulation was designed?

If you believe that your code is correct, run the simulation a few times, and you may see that the mean number of items in the low priority queue varies a lot from run to run. Let's run our simulation for a longer period of time to average out across time.

Run your simulation, but change the "Simulate Until Time" in the dialog box that you use to run the simulation from 1000 to 10000. Run it 3 times and write down the numbers that we get for mean number of items in the low priority queue, and mean number of items in the high priority queue.

	Mean # of Items in the Low-Priority Queue	*Mean # of Items in the High-Priority Queue*
Run 1		
Run 2		
Run 3		

Our hospital administrator would like the mean number of items in the high priority queue to be consistently less than 1, and the mean number of items in the low priority queue to be consistently less than 10.

Does your simulation show that the current system achieves this goal?