Summary of topics covered in class

In class we elaborated on the newspaper delivery example, and computed the estimated profit per day using (including a confidence interval) using replication. This is the sample code sc3c.m.

We looked at FEL implementations and described in detail 4 algorithms:

1. simple linked list \( (O(n)) \)
2. heaps (also called balanced trees) \( (O(\log(n))) \)
3. calendar queue’s \( (O(#l) \text{ with } #l \text{ number of events in each bucket}) \)
4. skip lists \( (O(\log(n)) \text{ on average, } O(n) \text{ worst case}) \)

See the additional handout (eventListAlgorithms.pdf) for details on these. (There is more in that paper, but you can skip that.) On the lecture notes slide I mention the Van Emde-Boas trees for your interest, but we didn’t cover them. I also put the original paper on skip lists on the web. This is optional reading material.

After introducing the Discrete Event Simulation algorithm, I presented a way to implement the 1-server 1-queue problem with more (redundant) events. The sample code OneServer2.java had a (deliberate) error, namely on enqueue there is no way to generate a ”begin service” event. The solution is to generate a ”Begin Service” event when enqueuing while the server is idle. The pseudocode in the lecture notes is correct. In addition the FEL implementation would drop events if you try to add an event with the same time as an event already in the list. It happens that this would never occur in this code, but it could lead to hard to detect bugs. From the next example on (DumpTruck) we have a correct implementation of the FEL using the TreeSet from the Java collection classes.

The DumpTuck.java example was extended to a real-time simulation by adding a “wait” call in the core discrete event simulation loop. The sample code DumTruckAn.java and associated files adds a simple animation and sound effects to this simulation. Because the travel from the weigher to the loader is an activity when this activity starts we can simply create an independent animation thread that will move the truck over a trajectory. To compile and run the code add jass.jar to your classpath, e.g., javac -classpath jass.jar *.java

The sample code OneServerPI.java implements the 1-queue 1-server system using a real-time process interaction method. The delays are implemented in each entity by the java Thread call sleep(). Unfortunately sleep() is quite inaccurate so in actual production code you would instead interact with a simulation clock which needs to be implemented to support process interaction. This would also allow the creation of an offline simulation with this methodology.