CS130A Programming Assignment 1
Due 2017-02-08 by 11:59PM

Priority Queues and Their Uses

One of the common uses for the priority queue discussed in class (implemented with a heap) is its use in discrete event simulation. You may be familiar with another form of simulation, such as what occurs in a video game. In most real-time video games, every time interval, behavior is extrapolated (and interpolated) based on the state of the simulation and agent behavior. This can be rather costly, particularly if "important" behavior is not regular. If important behavior is clustered, it may make sense to slow down the simulation to expend CPU on higher fidelity analysis than in periods when the important behavior is not occurring. This is what a Discrete Event Simulation allows. Time in the simulation is allowed to skip to the next event that must be analyzed. During the analysis, other events may be scheduled.
So what is a Discrete Event Simulation (DES)? A DES consists of an event queue, a simulated environment, and the main simulation loop. The main simulation loop pulls the next event out of the queue, facilitates the response in the environment, and advances the simulation clock to the next event. The environment will consist of one or more agents that can respond to events and schedule events of their own. In our case, the event queue will be a priority queue implemented as a binary heap (your own array-based implementation). The environment will simulate a network under attack by an outside agent. The agents will consist of computers in the network, an outside computer, an intrusion detection system, an attacker, and a sysadmin. Note that the environment will impose certain rules on the simulation. Events will be visible only on connected computers, and events will have a minimum latency that impacts when they can be scheduled.

The Network

We are going to model a simple island-hopping attack on a small corporate network. The attacker will compromise a computer in the network and use that as the launching point for other attacks. Our attack model is simplified so that each attack takes a set amount of time and succeeds with some probability. Periodically, the attacker and each compromised machine will attempt to compromise a random machine in the network. Attacks crossing the intrusion detection system will have a certain percentage chance of being caught. The sysadmin will react (with some delay) to fix machines with 100% certainty.
The stopping conditions of the main simulation:

1. If more than 50% of the network is compromised, the simulation ends with the attacker winning.
2. If the sysadmin fixes all machines, we stop the simulation and declare the sysadmin the winner (note, this has some chance of happening right away).
3. If the simulation goes on for 100 days, we stop the simulation and declare the situation a draw.

Events

There are a minimum number of event types that must be implemented by the simulation. Remember, events are objects that are stored in the event queue. All events have a scheduled time > now.

attack(time, source, target)

Schedule an attack event for time from source to target.

fix(time, target)

Schedule a computer to be fixed. It can be compromised again.

notify(time, attack)

The sysadmin will be notified by the IDS that an attack was detected. This gives the sysadmin both the source and the target via the attack object. The sysadmin will schedule fixes for both. The sysadmin can schedule fixes 10 seconds apart, and it takes 10 seconds from notification to first fix.

Agent Types

These are agents that respond or produce events (or both)

Computer

This is the default node in the network, divided evenly into two subtrees under the IDS. Every 1000ms, if a computer is compromised, it launches an attack at a random computer in the network. While a computer will not attack itself, it is fine for it to attack an already compromised computer. Each computer is attacked with equal probability (a random number between 0 and number_of_computers - 1, with a re-roll on itself is a fine implementation).

IDS

This is not a valid target, but can generate a notify event if the source and target of an attack cross over it (either from attacker-to-computer, or from computer-to-computer if the two computers are in different subnets).

Attacker

An agent that serves as a primary event source. Every 1000 ms, he chooses a computer at random and launches an attack. The latency from the attacker to each machine is 100ms. At the end of that 100ms, a machine might be compromised and the IDS might have noticed.

Sysadmin

The sysadmin only responds to notify events, and generates fix events for both the source and the target machine (note, the sysadmin cannot fix the attacker).

Topology

The topology of the network is a tree. At the root of the tree is the IDS, with all connected components as children. The IDS is also the network gateway.
Two switches (not agents) are direct children on the IDS.
The remaining computers are split evenly as children between the two switches.
Every event from the attacker crosses the IDS. Only attacks from computers under one switch to computers under the other switch can be detected by the IDS
The sysadmin is an agent in the simulation that is not attached to the network. It can only receive simulation notification from the intrusion detection system.

Mental Caveats

It is important to remember that when doing simulation, your experiment does not typically yield results of the form "The hypothesis was correct." Simulations help determine model consistency. This becomes important as the model is simplified. Optimal behavior in the simulation may not be optimal behavior in the environment being modelled.
Consider that the problem I have proposed can be analyzed without the simulation. However, as variables in the simulation increase, it becomes more and more practical to modify the simulation and less practical to work out the results mathematically.

Your Program Inputs

Your program will take three inputs from the command line, separated by spaces:

Number_of_computers

The number of computers in the simulation. The computers are logically split into two subgroups. Attacks from compromised machines within a subgroup cannot be detected by the IDS. Attacks from one subgroup to the other will be caught with Percent_detect chance.

Percent_success

An integer between 0-100 representing the chance of an attack succeeding in the simulation. Attacks from the attacker source have the same chance of succeeding as attacks from other computers.

Percent_detect

An integer between 0-100 representing the chance of an attack being detected if it crosses the IDS. Only attacks originating from the attacker or attacks that cross from one subgroup to the other can be detected.

Constants

All latencies are 100ms, and all events are executed at the end of the event latency (so schedule it for when it happens, not when it is generated).

Your Program Outputs

Every time an event is processed, print to standard out the event type, time, and any fields using constructor syntax (i.e. “Attack(1500, 1, 2)” is fine). Computers can be referred to by their ordinal number. When the simulation terminates, print out “Attacker wins,” “Sysadmin wins,” or “Draw”

Testing hints

If Percent_success is 100 and Percent_detect is 1, I strongly suspect the attacker wins. The draw condition is unlikely to occur, but is included for completeness.
Example execution (the above scenario with 50 computers):
Your_program 50 100 1

Random Complications

1. You must implement incrementKey and decrementKey in your Priority Queue implementation. This is unlikely to be useful in a real simulation engine, and I removed my awkward concoction to require it.
2. You must resize the heap if the size grows to the current capacity. Note that this can be implemented as a doubling in the array size followed by copying the previous elements over verbatim (they already satisfied the heap order property).
3. Part of your grade will be determined by the TA looking at your code. If it is too hard to read to determine correctness of the binary heap, your grade will suffer. Comments can be helpful to you and your grade.
4. Ties in priority definitely occur in this setup. Break ties in a consistent manner to avoid oddities.