FIT1008 Introduction to Computer Science
Practical Session 11

Semester 2, 2014

Objectives of this practical session

To understand how to implement priority queues using an array implementation of a Heap, and use them in a simple simulation application.

Testing

For this prac, you are required to write:

(1) a *function to test* each part of the functionality of the programs you implement (*at least two test cases*). The cases need to show that your functions can handle both valid and invalid inputs.

(2) programs with a sensible level of modularity.

Background

Task 1 [3 marks]

Write a Python program that implements a Priority Queue using an array implementation of a min-Heap, and allows a user to perform the following commands on the Priority Queue using a menu:

Append: which should ask the user for an integer, check it is indeed an integer. If it is an integer, it should append it to the Priority Queue and, if not, should give the appropriate error.

Serve: which should remove the minimum value and print it.

Print: which should print all the values in the Priority Queue.

Size: which should print the number of values in the Priority Queue.

Quit: which allows the user to quit the program.

Task 2 [4 marks]

For this Task you will write a Python program to simulate a small coffee shop. The program will use a Priority Queue which contains information about events. Each event has a time (in minutes since

8am) and a label stating the type of event. The time will be used as the priority, so that events get processed in the order they occur.

(i) Create 200 "arrival" events. Each event consists of an arrival time, a random number between 0 and 600, representing when a customer arrives at the shop after 8am, and a label identifying it as an "arrival" event.

(ii) Put the 200 "arrival" events into a Priority Queue.

(iii) Write a function `convert_time` which takes an integer, `n`, between 0 and 600, and returns a string representing the time that is `n` minutes after 8am.

(iv) For each event in the Priority Queue, take them off the queue in the order they occur and do the following:

- If the event is an "arrival" event, print out a message stating that a customer has arrived and at what time they arrived. Then generate a random number between 0 and 5. This number will represent the serving time. Put a new "served" event on the queue. This event has a time, which is the sum of the arrival time + serving time, and a label identifying it as a "served" event.
- If the event is a "served" event, print out a message stating a customer has been served and at what time they were served. Then generate a random number between 0 and 30. This number represents the time a customer remains in the shop after they have been served. Put a new "leaving" event on the queue. This event has a time which is the sum of the arrival time + serving time + time spent in the shop after being served, and a label identifying it as a "leaving" event.
- If the event is a "leaving" event, print out a message stating a customer has left and what time they left.

Task 3 [3 marks]

Modify the program you wrote for the last task so that it does the following:

(i) Create 10 "status" events, one for each hour between 9am and 6pm and add them to the Priority Queue before any event is processed.

(ii) When a customer arrives at the shop they join a service queue. If the service queue was empty when they joined it, put a new

"served" event on the Priority Queue. This event has a time, which is the sum of the arrival time + serving time.

(iii) If a "served" event is taken off the Priority Queue, then create a new "leaving" event, as in the last Task, and put it on the Priority Queue. Also the next customer in the service queue should be processed. This means the customer should be served from the service queue, and a new "served" event should be added to the Priority Queue. This event has a time = current time + serving time.

(iv) If a "status" event is taken off the Priority Queue, then print:

- the time,
- the number of people served during the last hour, and
- for the last hour, the maximum number of people that were in the shop at the same time.

Advanced Question *[Bonus 2 marks]*

Object oriented programming can make your code a lot easier to read and extend.

(i) Create a class *Store* which keeps track of the current number of customers, and since the last "status" event, the number of customers served and the maximum number of customers which have been in the store at the same time.

(ii) Create a class *Event* which has the instance variable `time` and the methods `__lt__` , `__gt__` , `__le__`, `__ge__`, and `process`. The method `process` should take as a parameters references to an object of type *Store* and an object of type *Priority Queue*.

(iii) Create subclasses of *Event* for each type of event, i.e., "arrival", "served", "leaving", and "status", with their own version of the method `process`.

(iv) Change your implementation of the Priority Queue so that it stores events, and when it serves an *Event* object it calls the object's process method. In this way the code for each type of event and how to process that event is contained within the class corresponding to that event.

Hall of Fame

Modify the simulation to include some more features, e.g.,

- people tend to arrive at the shop in groups,
- the time it takes to serve depends on the number of coffees ordered,
- people do not like to wait in long lines, and therefore the longer the line the less likely they will join the line,
- people will only stay at the store if there is somewhere to sit, and
- most coffee shops will have one or more people getting the orders, and one or more people making the coffees.