1.204 Quiz 1 Solutions

Spring 2008

Name _____

Exam guidelines:

1) 80 minutes are allowed to complete the quiz.
2) Open notes; open book.
3) There are 4 questions (100 points) and 7 pages (including this one) in the exam booklet.
4) No laptop computers, calculators, cell phones or messaging devices are allowed. Please turn off any that you have brought.
5) Please write legibly – you are welcome to use both sides of the paper; we can provide additional paper if necessary.

Question 1. Data modeling (25 points)

There is a set of traffic sensors that count cars in an urban area. Some are at the approaches to intersections, one sensor per lane of traffic entering the intersection. Other sensors are between intersections, one sensor per lane of traffic on the road. An intersection has several lanes entering it. Each lane is associated with a road. Each sensor has a status (working or not) and a count of the number of vehicles it has detected, in 5 second intervals, for the previous 3 minutes. Each road and each intersection has a name and a unique identifier.

Draw a data model that corresponds to this set of system rules. You only need to create one drawing that includes all the elements listed in steps a-e.

a. **Draw a box for each entity: give each an appropriate name**

b. **List the attributes in the box for each entity**

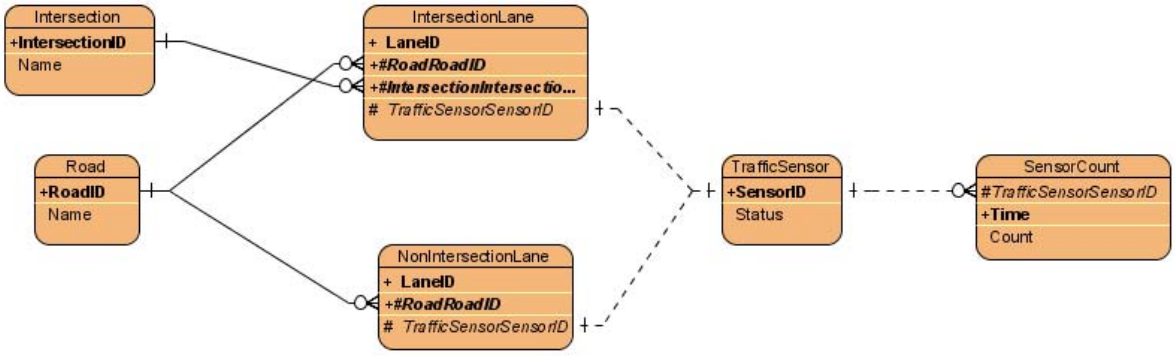
c. **Indicate the primary key for each entity by placing the phrase (PK) next to its name.**

d. **Draw all relationships between the entities in the model. Indicate foreign keys by placing the phrase (FK) next to attributes that are foreign keys.**

e. **Indicate the cardinality of the relationship: many-many, many-one or one-one. Use crows-foot notation; if you use another notation, define it.**

To repeat: You only need to create one drawing that includes all the elements listed in steps a-e above.

Please draw your data model on this page.



2. SQL (15 points)

Assume that the data model you built in the previous question has been implemented exactly as you drew it in a relational database management system such as SQL Server. Each entity is stored as a table, and attributes, keys and relationships are as you specified them. Write the SQL query to return the list of intersection IDs that have more than 10 cars in a 5 second interval in an approaching lane.

SELECT DISTINCT IntersectionIntersectionID FROM IntersectionLane, TrafficSensor, SensorCount WHERE IntersectionLane.TrafficSensorTrafficSensorID= TrafficSensor.TrafficSensorID AND TrafficSensor.TrafficSensorID= SensorCount.TrafficSensorSensorID AND Count > 10

With less name duplication:

SELECT DISTINCT IntersectionID FROM IntersectionLane, TrafficSensor, SensorCount WHERE IntersectionLane.TrafficSensorID= TrafficSensor.TrafficSensorID AND TrafficSensor.TrafficSensorID= SensorCount.TrafficSensorID AND Count > 10

3. Analysis of algorithms. (25 points)

a. What is the best case running time for heapify? Describe the case, and state what its running time as a function of its size n. Use the heapify code from lecture as the implementation for which you derive the bound.

The best case for heapify is O(n). If the heap is in order, the top half of the elements will be looked at, as always, and each will just make one comparison to its parent. The best case must look at n/2 elements, so it's still O(n), the same as the worst case.

b. What is the best case running time for heapsort? Describe the case, and state what its running time as a function of its size n. Use the heapsort code from lecture as the implementation for which you derive the bound.

The best case for heapsort is the same as the worst case, which is O(n lg n). If it's a max heap, the top element is put at the end, and the last one at the top, from where it must bubble down. The last element is, in general, smaller than the elements at the levels above it, so it will go lg n levels. (If it's a min heap, the same behavior occurs.) Thus n elements are placed at the top of the heap and bubbled down lg n levels. There is no ordering in a heap that makes heapsort go any faster than its worst, or average case. The proof requires some detail; this answer is just a sketch of the analysis.

There is one odd case in which heapsort is O(n). If all elements are equal (same value) in the heap, then putting the last element at the top will result in only 2 comparisons (to each child) and the element is not moved. Thus, only 3n operations are done: one to move the last element to the top and two comparisons.

4. Algorithm design (35 points)

At a grain elevator there are n trucks in its parking lot waiting to unload. The grain elevator owns the trucks and pays the drivers by the hour. It wishes to find an efficient order in which to unload the trucks. (After a truck has been unloaded the driver can make another trip.) There is one unloading facility at the elevator.

The time required to unload each truck is known; it depends on its capacity and the type and size of its unloading hatches. The trucks can be unloaded in any order the elevator chooses.

The grain elevator chooses to minimize

$$T = \sum_{i=1}^{n} t_i$$

where T = total time in system for all trucks

t_i = time that truck i waits until it is unloaded

n = number of trucks

Design an algorithm to let the grain elevator optimize its chosen objective function.

a. Write pseudocode (please define any non-Java-like symbols or conventions you use) that defines how the algorithm works.

To minimize total waiting time, unload the trucks in order of unloading time. The truck that's quickest to unload is done first, then the next quickest, and so on. Each truck waits for the n-1 trucks before it to be unloaded, and this is minimized by having the quickest ones done first.

Each truck imposes its unloading time on all trucks that follow it. Put the trucks that take a long time to unload at the back of the queue.

```
public static int minimize(int[] time) {   // Returns total time T
      Arrays.sort(time);                    // In ascending order
      int totalTime= 0;
      int timeThisTruckWaits= 0;
      for (int i= 0; i < time.length; i++) {
            timeThisTruckWaits += time[i];
            totalTime += timeThisTruckWaits;
      }
      return totalTime;
}
```

b. What kind of algorithm is it: divide and conquer, greedy, or ad hoc? Say why, in one sentence.

Greedy algorithm: sort and then make a local decision.

c. What is the running time of the algorithm? Use O(), Ω() or Θ() notation as appropriate. Briefly support your answer. No derivations are needed; justify your answer informally.

The algorithm requires a sort, which is O(n lg n), which is the bottleneck step. The algorithm itself is O(n), since we just loop through all the data once. (Just count the operations in the pseudocode above.)

1.204 Computer Algorithms in Systems Engineering

Spring 2010