SIENAcollege
Computer Science

Computer Science 385
Design and Analysis of Algorithms
Siena College
Spring 2017

Lab 8: Greedy Algorithms

Due: Start of your next lab session

You will be assigned groups of size 2 or 3 for this lab. Only one submission per group is needed. Please have your instructor initial in each box as you complete each task.

Names: ____

Question 1:
Hashing Practice: Complete Levitin Exercises 7.3.1 and 7.3.2 on p. 274-275. (10 points)

☐

If you have not studied or do not recall the details of heaps and heapsort, please review Levitin Section 6.4 and/or the topic notes on Heaps linked from the schedule page.

Question 2:
Consider the construction of a min-heap by starting with an empty heap, then inserting values in a given order. Show the min-heap after each of the values 2, 8, 97, 31, 12, 3, and 39 are added to an initially empty heap. (6 points)

Question 3:

Consider the construction of a min-heap as done in the first phase of a heapsort. Start with an array containing the values 2, 8, 97, 31, 12, 3, and 39. Show the contents of the array (in array or tree form, your choice) after each non-trivial iteration of the first phase of the heapsort (which goes from an unsorted array to a min-heap). (7 points)

? Question 4:

Finally, show the transformation from the min-heap you just constructed to a sorted array, completing the heapsort procedure. (7 points)

? Question 5:

Consider a binary min-heap like the ones in Levitin Section 6.4.

a. Which locations in a binary min-heap of n elements could possibly contain the third-smallest element? (3 points)

b. Which locations in a binary min-heap of n elements could possibly contain the largest element? (2 points)

The heaps we have seen here, where the heap is represented by a binary tree stored in an array, are one specific case of a more general structure called a *d-heap*. In a d-heap, each node has up to d children. So the binary heaps you would have seen before would be 2-heaps. For the questions below, assume that the minimum value is stored at the root node (*i.e.*, that it is a min-heap).

? Question 6:

Show the construction of a 2-heap that results from the following values being inserted: 18, 9, 23, 17, 1, 43, 65, 12. How many comparisons are needed? (8 points)

? Question 7:

Show the construction of a 3-heap that results from the following values being inserted: 18, 9, 23, 17, 1, 43, 65, 12. How many comparisons are needed? (8 points)

Question 8:
For the heap element at position i in the underlying array of a 3-heap, what are the positions of its immediate chidren and its parent? (Give formulas in terms of i.) (2 points)

Question 9:
For the heap element at position i in the underlying array of a d-heap, what are the positions of its immediate chidren and its parent? (Give formulas in terms of i and d.) (2 points)

Question 10:
Show the construction of a 1-heap that results from the following values being inserted: 18, 9, 23, 17, 1, 43, 65, 12. How many comparisons are needed? (5 points)

? Question 11:

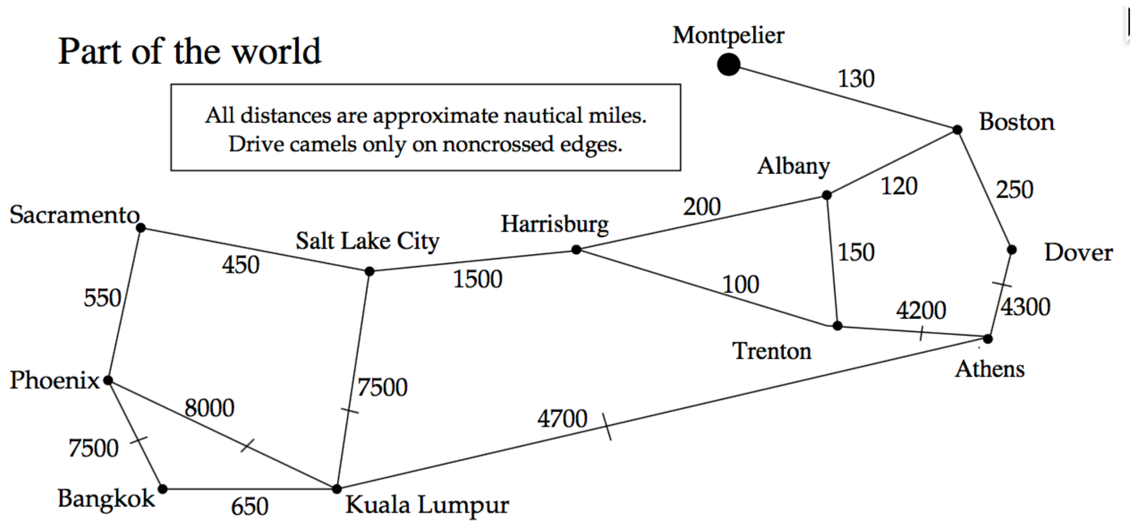
Show the construction of a 7-heap that results from the following values being inserted: 18, 9, 23, 17, 1, 43, 65, 12. How many comparisons are needed? (5 points)

Read Levitin Section 9.2 about Kruskal's Algorithm, and then complete the problem below.

? Question 12:

Using the style as shown in Levitin Figure 9.5, p. 326, demonstrate the steps of Kruskal's Algorithm on the graphs in Levitin Exercise 9.2.1, p. 331-332. (15 points, 6 for part a, 9 for part b)

Question 13:
Using the graph below, (credit: Bailey Problem 16.7, p. 436), use Dijkstra's Algorithm to compute the shortest distance from Dover to Phoenix by filling in the tables below, using the algorithm and notation as shown in the example from class. (25 points)



Fill in the table in Table 1 version of this document or create an equivalent table of your own. which is a map of cities as keys to the shortest distance from Dover to the last edge traversed on that shortest route as values.

It is easiest to specify edges by the labels of their endpoints rather than the edge label itself, which might not be unique.

Also, use the table in Table 2 to keep track of your priority queue. Remember, don't erase entries when you remove them from the queue, just cross them out and mark them with a number in the "Seq" column of the table entry to indicate the sequence in which the values were removed from the queue.

City	(distance,last-edge)
Dover	(0, `null`)

Table 1: Map of Dijkstra's algorithm results for last edge traversed to find each city for the first time.

(distance,last-edge)	Seq

Table 2: Table showing the progress of the priority queue in Dijkstra's algorithm.