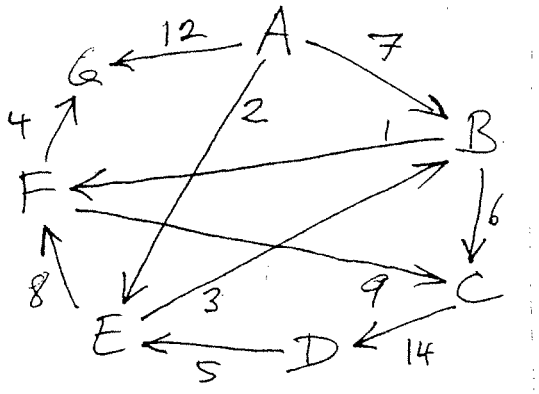
CSE 326: Data Structures
Assignment #7
December 4, 2009

You do not have to turn in this assignment. Its purpose is to give you practice for the final exam on recent topics.

1. Page 331, problem 1. Show the partially ordered tree after each Insert and after each DeleteMin. Also show the heap in table form after the last Insert.

2. Page 331, problem 2.

3. (a) Simulate Dijkstra's algorithm on the following graph to find the least cost path (the path itself, not just its cost) from vertex A to every other vertex. Each time a vertex is removed from U, do the following: redraw the vertices (but not the edges) of the graph, circle the vertices that still remain in U, show the current values of Distance(v) next to each vertex v, and draw an edge from v to the current value of Parent(v) for each vertex v for which $0 < \text{Distance}(v) < \infty$. (As in Assignment 6, Parent(v) is the immediate predecessor of v on the least cost path from A to v found by the algorithm so far.) For each of these steps, also show the partially ordered tree corresponding to U, labeling each node with both its Key and Info values.



   (b) From your output in part (a), how would you find the least cost path from A to D? Find it and its cost.

4. There is a way of implementing Dijkstra's algorithm without the need for the curious operation DecreasePriority and the auxiliary table of heap indices from Assignment 6. The changes proposed in this exercise make the Priority Queue interface much cleaner.

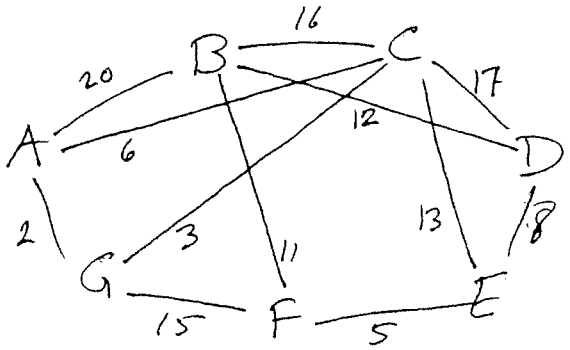
   Delete the last line of Algorithm 12.6 and instead insert a new entry in the priority queue with Info = w and Key = Distance(v) + $c(v, w)$. That is, at any time there may be many different entries in the priority queue with the same Info field.

(a) What other changes need to be made to Algorithm 12.6 so that it still works correctly?

(b) How big can the priority queue become with these changes? Notice that this affects both the running time of each priority queue operation, and also the number of times the main iteration in Algorithm 12.6 must be executed, since you have to iterate until the priority queue is empty to be sure you have processed every vertex. Despite these problems, show that this implementation still runs in the same time $O((n + e) \log n)$.

(c) What additional changes need to be made to find the least cost paths themselves rather than their costs?

5. (a) Simulate Kruskal's algorithm to find a minimum cost spanning tree T of the following graph. Each time an edge e is added to T, redraw the current forest T and the full collection of up-trees resulting from adding e to T. Each time an edge e is considered but not added to T, list the Find operations that explain why e was not included in T. Use Weighted Unions, but without path compression.



(b) What is the cost of the minimum spanning tree that you found?