comp363 Handout #16
Design and Analysis of Computer Algorithms
Michael Goldwasser
Loyola University Chicago Thursday, 10 April 2003

Homework #6: Shortest Paths
Due Date: Tuesday, 22 April 2003

Guidelines

Please make sure you adhere to the policies on collaboration and academic honesty as outlined in Handout #1.

Reading

Review Ch. 24.1, 24.3, 24.5, 25 of CLRS.

Problems

Problem A (20 points) "Work entirely on your own."
Do CLRS 24.5-1.

Problem B (30 points) "Work entirely on your own."
Do CLRS 24.5-2.

In presenting your solution, give the graph and give an explanation of why this graph satisfies the conditions of a solution.

Hint: One of the challenges will be to show that there is a shortest path tree which does not use the edge (s, v) which is a shortest edge leaving s.

Problem C (50 points) "You may discuss ideas with other students."
Recall that Dijkstra's algorithm for single source shortest path relies on a priority queue, making 1 call to initialize the queue, $O(V)$ calls to Insert, $O(V)$ calls to Extract-Min, and $O(E)$ calls to DecreaseKey. If using a Fibonacci heap as the underlying priority queue implementation, this results in a worst-case running time of $O(E + V \lg V)$.

Suppose edge weights in a graph are integers in the range from 0 to W. Develop a new priority queue implementation for this situation which results in a worst-case bound of $O(E + WV)$ time if used as the underlying priority queue for Dijkstra's algorithm.

Hint: This problem is clearly similar to a previous problem we did involving Prim's algorithm for MST. However they are not truly idential problems. Specifically, the key difference between the two algorithms is in how each vertex's priority is defined within the priority queue. For Prim's algorithm, *a vertex's priority is either infinite or else precisely equal to the weight of a single edge, that which connects the vertex to a member of set S. This is not the case with Dijkstra's. To be successful on this problem you will have to carefully understand the inherent properties of the priority values.*

Problem D **(EXTRA CREDIT – 20 points)**

"You may discuss ideas with other students."

Revisiting Problem C, it is actually possible to do even better in such a situation. For extra credit, redesign the priority queue implementation so that you can achieve an overall running time of $O((V + E) \lg W)$ for Dijkstra's algorithm, where edge weights are integers in the range $[0..W]$.

Hint: For vertices in $V - S$, how many distinct shortest-path estimates (i.e., priorities) can there be at any point in time?