CMPSCI 250 Discussion #10: Uniform-Cost Search
Individual Handout

David Mix Barrington
8 April 2013

In last week's discussion we used matrices to solve the **all-pairs shortest path problem** for a directed graph labeled with distances on each edge. Today, we'll use a different approach to solve the **single-source shortest path problem** on the same graph. This is essentially the same thing as **Dijkstra's Algorithm** presented in section 10.6 of Rosen, but we'll describe the algorithm differently to emphasize its relationship to depth-first and breadth-first search.

In both those algorithms we find all those vertices reachable from some **start point**. Beginning with the start point, we **explore** each vertex we find. To explore a vertex means to take all of its neighbors and put a record for each onto a data structure called the **open list**. For each neighbor, more precisely, that is not already on another data structure called the **closed list**. When we are done exploring the current vertex, we take the next vertex off of the open list and explore that, continuing in this way until the open list is empty or until we find a particular **goal node**.

In DFS the open list is a **stack**, and in BFS it is a **queue**. In those two searches we put a node onto the closed list once we have started exploring it, since there is no point in visiting it again. In today's algorithm, called **uniform-cost search**, the open list will be a **priority queue**. The records in it will contain both a node x and a **priority**, which will be the distance from the start node to x along the path we have just found. When we take a node off the priority queue, we get the one with the lowest priority, that is, the one that is closest to the start node as best we know.

How do we get the distance to put in the record for a new node? Suppose the start node is s, and in the course of exploring x we find an edge to a node y that is not on the closed list. We have a shortest-path distance d_x from s to x, which was the label on the record for x that we took off the queue. We also have the label d_{xy} for the edge from x to y. The new label will be $d_x + d_{xy}$, reflecting the fact that we can go from s to y in this distance by taking the shortest path from s to x and then the edge to y.

The key step in showing that we actually get the shortest path from the start node to each other node is that when a node x comes off the open list, with distance d, we have checked all possible paths of length less than d from the start node. Thus no path that we haven't yet checked could be shorter than the one that got this record put onto the open list.

Note that we do not put a node on the closed list until it has come *off* of the open list. We might have a path already, but later discover a better path.

(questions on other side)

Today's assignment deals with the same labeled directed graph from last week, which has vertices named a, b, c, d, and e, and the following single-source distance matrix:

$$\begin{pmatrix} 0 & 1 & 3 & \infty & 7 \\ \infty & 0 & 1 & \infty & \infty \\ \infty & \infty & 0 & 1 & 3 \\ 4 & 3 & \infty & 0 & 1 \\ \infty & \infty & 4 & 0 & 0 \end{pmatrix}$$

Question 1: Carry out a uniform-cost search of this graph starting with vertex a. Indicate which records come on and off the priority queue, in which order. Neighbors of the same node go onto the queue in alphabetical order, and if two nodes are tied for having the lowest priority, the one that went on first comes off first.

Question 2: There is a path of length 6 from a through c to e. Did your algorithm look at this path before finding a better one? Why or why not?

Question 3: Suppose we had a graph with n vertices where each node had only $O(1)$ neighbors. What would be the big-O running time needed to solve the all-pairs shortest path problem by solving the single-source shortest path problem for each vertex? (Assume we keep our priority queue as a heap, so that each enqueueing or dequeueing operation takes $O(\log n)$ time.) How does this compare with the matrix multiplication method from last week?