Class Note #08

Date: 02/06/2006

[Overall Information]

In today’s class, we proved the correctness of Dijkstra's Shortest Path Algorithm, using induction and the Lemma from last class.

[During the Lecture]

1. The second quiz of CSCI303@Spring 2006 will take place in the coming Wednesday (02/06/2006) at the beginning of that day’s class. Most of the quiz questions will be similar to homework ones and there will still be one Big-O notation related question. Please be sure to show up on time at the quiz day. The solution of homework#2 was distributed at the end of today’s class.
2. After the lecture began, the pseudo code of shortest path algorithm was rewritten onto the whiteboard. We reviewed the computation procedure of the algorithm as well as the lemma we talked about in last class.

3. The output that the algorithm produces are $d[]$ and $\text{pred}[]$. The correctness of the algorithm can then be stated as:
(1) pred[v] is the last hop node on a shortest s-v path;

(2) d[v] stores the shortest distance from s to v.

After that, a specific example was given and we were able to see that for each finalized (or explored) node, its pred[] and d[] are correct. This may not necessarily be true for a node that hasn't been finalized, and we need to be careful about how exactly the algorithm makes progress.

4. We’d like to claim the algorithm makes progress through the
following three aspects:

(1) For all nodes with finite \( d[v] \), \( \text{pred}[v] \) is the last hop on an s-v path of length \( d[v] \);

(2) For all explored nodes, \( d[v] \) holds the shortest path length from s to v;

(3) For all enqueued nodes, \( d[v] \) is the length of a shortest s-v path using only explored nodes.

The above three statements were proved in today’s class using induction.

\[\text{variant}^1\]

\[\begin{align*}
\text{(1) For all nodes with } d[v] < \infty, \text{ pred}[v] & \text{ is the last hop on an s-v path of length } d[v]; \\
\text{(2) For all explored nodes, } d[v] & = \text{dist}(s,v); \\
\text{(3) For all enqueued nodes, } d[v] & \text{ is the length of a shortest s-v path using only explored nodes.}
\end{align*}\]

5. The base case, when only node s is enqueued, and no node is explored, is obviously correct. For the inductive step, we need to show that all three invariants still hold after one more node has been explored. We start by proving invariant (2). First, since all previously explored nodes
have correct $d[v]$ values (by Induction Hypothesis), and we do not change them at all, they still have those correct values. So we only need to prove something about the newly explored node $v$.

Before finalizing, $d[v]$ is the length of shortest path using only explored nodes. After finalizing, we want to claim that $d[v]$ is actually the shortest path length from $s$ to $v$ (using any nodes).

To prove this, let $S$ be the set of nodes explored before $v$. For contradiction, assume that there is a shorter path $P$, and let $w$ be the first node on $P$ not in $S$, and $u$ the node immediately before $w$. 

We know that before finalizing, $d[v]$ is the shortest path from $s$ to $v$ using only explored nodes $S$.

We claim that $d[v] = \text{shortest distance from } s \text{ to } v \text{ using only explored nodes } S$.

Let $P$ be a shortest $s$-$v$ path. Let $w$ be the first node on $P$ not in $S$. Then.

Reason: this subpath $P(s, u)$ is a shortest $s$-$w$ path by Lemma from last class, so $d(w) = \text{length of } P(s, u)$. But by (iii), $d(w) \geq \text{length of } P(s, u)$, because $P(s, u)$ is a path.
6. P can be divided into two parts. The first part (from s to w) has length $d[u] + l(u,w)$, because by the Lemma from last lecture, the part of P from s to u must be a shortest path, and $d[u]$ is the correct shortest path distance by induction hypothesis. But then, because the algorithm chose v to the node with the smallest value $d[u'] + l(u',v)$ over all nodes u' in S, we know that the newly finalized $d[v]$ value is at most $d[u]+l(u,w)$. Also, because all edge lengths are non-negative, the total length of the second part of P (from w to v) is non-negative. But then, P has length at least $d[v]$, which is a contradiction. So $d[v]$ is actually a shortest path length. This establishes invariant (2).

7. Next, we need to establish invariants (3) and (1), by talking about newly enqueued or updated nodes. If u is a newly enqueued node, then $d[u]=d[v]+l(v,u)$. Because u has not enqueued before, the only path from s to u (using only explored nodes) must pass through v, so the assigned value is correct (because $d[v]$ is correct). Also, pred[u] is correctly set to v.

8. Suppose $d[u]$ needs to be changed, then the new shortest s-u path using $S \cup \{v\}$ would have to include v. We can also tell that v has to be the last hop on the shortest s-u path, otherwise, if there is a node w immediately after v and before u (w is in S), $d[w]$ will be incorrect according to the lemma (contradicting the induction hypothesis). So the length of this shortest path is $d[v]+l(v,u)$, which the algorithm
assigns as $d[u]$. Again, $\text{pred}[u]=v$ is also correct.

9. Finally, by the previous statement: all the updates are caused by exploring $v$, as a result, enqueued nodes which have no connection from $v$ will not be affected at all. (If they did, we would have the same contradiction about an incorrect $d[w]$ value for some previously explored node $w$.)
Now, we have proved all the three statements. If we apply them to the final state of the algorithm, when all nodes reachable from s are explored, we obtain that all d[] and pred[] values are explored.

Notice that we should really also prove that all nodes reachable from s are actually explored, which is an easy induction.

Next class, we will analyze the running time of shortest path algorithm.
By the previous argument, if any configured node needs updating, the node replacing $v$, then the last hop is from $v$.
If an node $w$ with edge $(v, w)$ missing needs to be updated, and invariant (b) is also held for non-neighbors, so inductive step is proved.

$\text{dist}(s, v)$ is last node before $v$ with $s$ as point of interest.

$\text{dist}(s, v)$ is the distance from $s$ to $v$. $\text{dist}(s, v)$. 