Some Proofs for BFS

Despite how obvious it is what Algorithm BFS is doing, it is surprisingly hard to come up with good proofs of the evident facts. The trouble is that the loop doesn’t correspond to any one simple variable on which to do induction. E.g., the kth iteration doesn’t correspond to finding a vertex at distance k. So here are two sample proofs. The first picks a useful inductive variable, which therefore does not correspond to the number of the iteration. The second picks out a loop invariant (and thus does correspond to a single iteration) but the invariant is rather complicated.

**Theorem 1.** In BFS, all vertices enter the queue (and thus the tree) in order of distance from the start vertex.

Proof: By induction. Consider

\[ P(k): \text{ all vertices distance } k \text{ from the root enter the queue after all closer vertices and before all farther vertices.} \]

To prove the theorem is to prove \( P(k) \) for all \( k = 0, 1, 2, \ldots \)

Basis, \( P(0) \). The root is the only vertex at distance 0. There are no closer vertices (so the closer-than claim is vacuously true) and all farther vertices get put on the queue later because the initialization puts the root on first.

Inductive Step. Strong Induction. Assume \( P(0) \) through \( P(k-1) \) are all true. That means all vertices up to distance \( k-1 \) have been put on the queue in order of distance and nothing else gets put on until all these vertices are on. To prove \( P(k) \) we need to show that all vertices of distance \( k \), and only those vertices, get put on next. Consider what happens as the vertices already known to go on the queue (those of distance 0 to \( k-1 \)) come to the head. As vertices of distance 0 to \( k-2 \) come to the head, only vertices of distance \( k-1 \) or less will be neighbors, and by assumption they come on in order. So we need only look at what happens as vertices of distance \( k-1 \) come to the head. Every vertex of distance \( k \) is adjacent to at least one vertex of distance \( k-1 \); moreover, vertices of distance \( k \) are the only vertices adjacent to vertices at distance \( k-1 \), except for vertices of shorter distance, which have therefore already been visited. Thus, as all the vertices of distance \( k-1 \) come to the head, all the vertices of distance \( k \) will go on the queue, and nothing else will.

**Theorem 2.** In BFS, when the algorithm terminates, all vertices connected to the initial vertex \( v_0 \) have been visited, and no others. (In other words, the final visited set is the connected component of \( v_0 \).)

Proof: Call a vertex finished if it has been visited and taken off the queue. (Note that every vertex gets put on the queue as soon as it is visited, and once visited it cannot be visited and thus put on the queue again. So at all stages, all visited vertices are either on the queue or finished.)

We claim the following is a loop invariant of the BFS algorithm:

March 24, 2002
(1) At the end of each pass through the for-loop, every vertex in the visited set is connected to \(v_0\), and every vertex adjacent to a finished vertex has been visited.

Assuming for the minute that this claim really is an invariant, consider what it says when the algorithm terminates. Since the queue is empty, all visited vertices are finished, and thus the claim says:

(2) At the end of the algorithm,

(2a) every vertex in the finished set is connected to \(v_0\), and

(2b) every vertex adjacent to a finished vertex is finished.

We claim that the theorem follows from (2). Let \(F\) be the final finished set and let \(C\) be the connected component of \(v_0\). (2a) itself says that \(F \subseteq C\), so to conclude that \(C = F\) we must additionally show that \(C \subseteq F\).

To show that \(C \subseteq F\): If \(u \in C\) then there is a path from \(v_0\) to \(u\). Since \(v_0 \in F\), by (2b) and an easy induction (not written out) we conclude that each succeeding vertex on the path is in \(F\). Therefore \(u \in F\).

Now, why is (1) a loop invariant? As usual, we must argue that the invariant is true at each stage using a basis and an inductive step. As usual, the basis is case 0, meaning just after the 0th pass through the loop, meaning just before the first pass.

Basis: Just before the loop is first entered, there are no finished vertices and the only visited vertex is \(v_0\), which is surely adjacent to itself. Thus the invariant is true at this stage.

Inductive step. We assume the invariant was true at the end of the \((k-1)\)st pass and must show it is still true at the end of the \(k\)th pass. In each pass through the loop, one more vertex, call it \(u\), is added to the finished set by being deleted from the queue. Before it is deleted from the queue, all vertices adjacent to it that were not already visited are visited, and nothing else is visited. So, since \(u\) was connected to \(v_0\) at the start of the pass (by the inductive assumption), all newly visited vertices are also connected to \(v_0\), though \(u\) if no other way. Thus it is still true at the end of the pass that all visited vertices are connected to \(v_0\). It is also still true at the end of the pass that all adjacent vertices to finished vertices are visited, because we just visited any previously unvisited neighbors of \(u\), which is the only new finished vertex.

\[\text{Note 1:} \text{ In the first paragraph of the proof of Theorem 2 it is claimed that at all stages all visited vertices are either in the queue or finished. This claim is used to get from (1) to (2). This claim itself is a loop invariant, and therefore must also be proved by induction in a really complete proof. But I thought that this claim was pretty obvious – meaning you should be able to fill in a proof that it is a loop invariant.}\]

\[\text{Note 2: Compare the proof of Theorem 2 with the proof in the solution manual of Problem 1a in the BFS section of the text. The latter is much shorter. How can this be? Which proof do you like better?}\]