BFS algorithm

Robert Danek

September 14, 2007

1 Algorithm and Correctness

Figure 1 procedure BFS(G,s)

Input:
Graph $G = (V, E)$, vertex $s \in V$

Output:
$\forall u$ reachable from s, $dist(u)$ is set to the distance from s to u.

```
 1: for each u ∈ V do
 2:    dist(u) = ∞;
 3: end for
 4: Q = [s]; /* initialize Q to contain s */
 5: dist(s) = 0;
 6: while Q ≠ ∅ do
 7:    u = dequeue(Q);
 8:    for each (u, v) ∈ E do
 9:       if dist(v) = ∞ then
10:          enqueue(Q, v);
11:          dist(v) = dist(u) + 1;
12:       end if
13:    end for
14: end while
```

Lemma 1: For each $d = 0, 1, 2, ...$ there is a moment at which (1) all nodes at distance $\leq d$ from s have their distances correctly set; (2) all other nodes have their distances set to ∞; and (3) the queue contains exactly the nodes at distance d

Proof: Proof is by induction on d. For the base case, $d = 0$. Here, after line 5 is executed, all the elements of the lemma hold. For the induction hypothesis, assume the lemma is true for $d = k$. We now prove it true for $d = k + 1$. Start by considering the earliest moment when the lemma holds for $d = k$. Either there are no nodes at distance k or there are. In the former

case, the queue is empty, and the lemma automatically holds for $d = k + 1$. In the latter case, the queue is not empty.

Let T_Q be the set of nodes currently in the queue, and T'_Q be the set of nodes that are directly reachable from some node in T_Q. (More formally, $T'_Q = \{v : \exists u : u \in T_Q \wedge (u, v) \in E\}$.) There cannot exist a node $x \in T'_Q$ such that its distance from s is $> k + 1$ since there is a path from s to x via some node in T_Q, which is at distance k. Hence the nodes in T'_Q must be at some distance $\leq k + 1$ from s. If they are at a distance $< k + 1$, then by the induction hypothesis they will already have their distances set correctly. Hence for each node $u \in T_Q$ that is dequeued at line 7, lines 10 and 11 will only execute for those nodes $v \in T'_Q$ which are at a distance $k + 1$ from s. This implies that after the last node in T_Q is dequeued and all of its edges are examined (lines 8..11) the lemma will hold for $d = k + 1$.

2 Time Complexity

The loop at line 1 takes $O(|V|)$ steps. Lines 4 and 5 take $O(1)$ steps. Hence the algorithm up to and including line 5 has time complexity $O(|V|)$. It remains to determine the time complexity of the algorithm after line 5. Observe that a vertex v is added to Q at line 10 only if $dist(v) = \infty$. Further observe that immediately after adding v to Q, that $dist(v)$ is set to a value other than ∞. Hence each vertex can be added to Q at most once. This implies that the check at line 6 and lines 7, 10, and 11 can each execute at most $O(|V|)$ times. Finally we need to determine how many times line 9 executes. Notice that the algorithm examines every edge incident to every vertex dequeued (line 8). Since every vertex in the graph can be enqueued at most once (by the preceding argument), over the course of the algorithm's execution all of the graph's edges can be examined. Hence line 9 executes $O(|E|)$ times. We thus conclude that the algorithm has time complexity $O(|V| + |E|)$.

Note that this argument makes a certain assumption about the data structure used to represent the graph. What is that assumption?