Termination

The algorithm terminates after all the rounds are executed. It is straightforward to modify the algorithm so that a process exits after the round in which it sets its *parent* variable (see Exercise 5.1).

Complexity

- The local space complexity at a node is of the order of the degree of edge incidence.
- The local time complexity at a node is of the order of (diameter + degree of edge incidence).
- The global space complexity is the sum of the local space complexities.
- This algorithm sends at least one message per edge, and at most two messages per edge. Thus the number of messages is between l and $2l$.
- The message time complexity is d rounds or message hops.

The spanning tree obtained is a breadth-first tree (BFS). Although the code is the same for all processes, the predesignated root executes a different logic to being with. Hence, in the strictest sense, the algorithm is asymmetric.

5.5.2 Asynchronous single-initiator spanning tree algorithm using flooding

This algorithm assumes a designated root node which initiates the algorithm. The pseudo-code for each process P_i is shown in Algorithm 5.2. The root initiates a flooding of QUERY messages in the graph to identify tree edges. The parent of a node is that node from which a QUERY is first received; an ACCEPT message is sent in response to such a QUERY. Other QUERY messages received are replied to by a REJECT message. Each node terminates its algorithm when it has received from all its non-parent neighbors a response to the QUERY sent to them. Procedures 1, 2, 3, and 4 are each executed atomically.

In this asynchronous system, there is no bound on the time it takes to propagate a message, and hence no notion of a message round. Unlike in the synchronous algorithm, each node here needs to track its neighbors to determine which nodes are its children and which nodes are not. This tracking is necessary in order to know when to terminate. After sending QUERY messages on the outgoing links, the sender needs to know how long to keep waiting. This is accomplished by requiring each node to return an "acknowledgement" for each QUERY it receives. The acknowledgement message has to be of a different type than the QUERY type. The algorithm in the figure uses two messages types – called as ACCEPT (+ ack) and REJECT (- ack) – besides the QUERY to distinguish between the child nodes and non-child nodes.

```
(local variables)
int parent ⟵ ⊥
set of int Children, Unrelated ⟵ ∅
set of int Neighbors ⟵ set of neighbors
(message types)
QUERY, ACCEPT, REJECT

(1)  When the predesignated root node wants to initiate the algorithm:
(1a) if (i = root and parent = ⊥) then
(1b)     send QUERY to all neighbors;
(1c)     parent ⟵ i.

(2)  When QUERY arrives from j:
(2a) if parent = ⊥ then
(2b)     parent ⟵ j;
(2c)     send ACCEPT to j;
(2d)     send QUERY to all neighbors except j;
(2e)     if (Children ∪ Unrelated) = (Neighbors/{parent}) then
(2f)         terminate.
(2g) else send REJECT to j.

(3)  When ACCEPT arrives from j:
(3a) Children ⟵ Children ∪ {j};
(3b) if (Children ∪ Unrelated) = (Neighbors/{parent}) then
(3c)     terminate.

(4)  When REJECT arrives from j:
(4a) Unrelated ⟵ Unrelated ∪ {j};
(4b) if (Children ∪ Unrelated) = (Neighbors/{parent}) then
(4c)     terminate.
```

Algorithm 5.2 Spanning tree algorithm: the asynchronous algorithm assuming a designated root that initiates a flooding. The code shown is for processor P_i, $1 \leq i \leq n$.

Termination

The termination condition is given above. Some notes on distributed algorithms are in place. In some algorithms such as this algorithm, it is possible to locally determine the termination condition; however, for some algorithms, the termination condition is not locally determinable and an explicit termination detection algorithm needs to be executed.

Complexity

- The local space complexity at a node is of the order of the degree of edge incidence.

- The local time complexity at a node is also of the order of the degree of edge incidence.
- The global space complexity is the sum of the local space complexities.
- This algorithm sends at least two messages (QUERY and its response) per edge, and at most four messages per edge (when two QUERIES are sent concurrently, each will have a REJECT response). Thus the number of messages is between $2l$ and $4l$.
- The message time complexity is $(d+1)$ message hops, assuming synchronous communication. In an asynchronous system, we cannot make any claim about the tree obtained, and its depth may be equal to the length of the longest path from the root to any other node, which is bounded only by $n-1$ corresponding to a depth-first tree.

Example Figure 5.3 shows an example execution of the asynchronous algorithm (i.e., in an asynchronous system). The resulting spanning tree rooted at A is shown in boldface. The numbers next to the QUERY messages indicate the approximate chronological order in which messages get sent. Recall that each procedure is executed atomically; hence the sending of a message sent at a particular time is triggered by the receipt of a corresponding message at the same time. The same numbering used for messages sent by different nodes implies that those actions occur concurrently and independently. ACCEPT and REJECT messages are not shown to keep the figure simple. It does not matter when the ACCEPT and REJECT messages are delivered.

1. A sends a QUERY to B and F.
2. F receives QUERY from A and determines that AF is a *tree edge*. F forwards the QUERY to E and C.
3. E receives a QUERY from F and determines that FE is a *tree edge*. E forwards the QUERY to B and D. C receives a QUERY from F and determines that FC is a *tree edge*. C forwards the QUERY to B and D.
4. B receives a QUERY from E and determines that EB is a *tree edge*. B forwards the QUERY to A, C, and D.
5. D receives a QUERY from E and determines that ED is a *tree edge*. D forwards the QUERY to B and C.

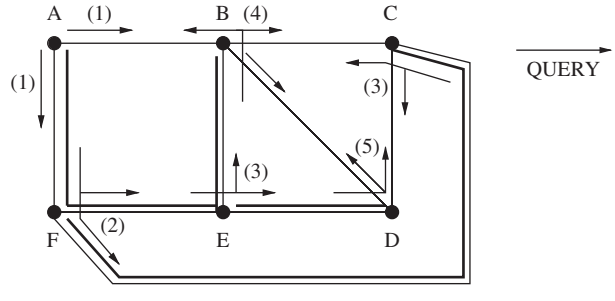

Figure 5.3 Example execution of the asynchronous flooding-based single initiator spanning tree algorithm (Algorithm 5.2).

Each node sends an ACCEPT message (not shown in Figure 5.3 for simplicity) back to the parent node from which it received its first QUERY. This is to enable the parent, i.e., the sender of the QUERY, to recognize that the edge is a tree edge, and to identify its child. All other QUERY messages are negatively acknowledged by a REJECT (also not shown for simplicity). Thus, a REJECT gets sent on each back edge (such as BA) and each cross edge (such as BD, BC, and CD) to enable the sender of the QUERY on that edge to recognize that that edge does not lead to a child node. We can also observe that on each tree edge, two messages (a QUERY and an ACCEPT) get sent. On each cross-edge and each back-edge, four messages (two QUERY and two REJECT) get sent.

Note that this algorithm does not guarantee a breadth-first tree. Exercise 5.3 asks you to modify this algorithm to obtain a BFS tree.

5.5.3 Asynchronous concurrent-initiator spanning tree algorithm using flooding

We modify Algorithm 5.2 by assuming that any node may spontaneously initiate the spanning tree algorithm provided it has not already been invoked locally due to the receipt of a QUERY message. The resulting algorithm is shown in Algorithm 5.3. The crucial problem to handle is that of dealing with concurrent initiations, where two or more processes that are not yet participating in the algorithm initiate the algorithm concurrently. As the objective is to construct a single spanning tree, two options seem available when concurrent initiations are detected. Note that even though there can be multiple concurrent initiations, along any single edge, only two concurrent initiations will be detected.

Design 1

When two concurrent initiations are detected by two adjacent nodes that have sent a QUERY from different initiations to each other, the two partially computed spanning trees can be merged. However, this merging cannot be done based only on local knowledge or there might be cycles.

Example In Figure 5.4, consider that the algorithm is initiated concurrently by A, G, and J. The dotted lines show the portions of the graphs covered by the three algorithms. At this time, the initiations by A and G are detected along edge BD, the initiations by A and J are detected along edge CF, the initiations by G and J are detected along edge HI. If the three partially computed spanning trees are merged along BD, CF, and HI, there is no longer a spanning tree.