delayed and that identifies **a-e-h-j** as the critical path and 21 days as the critical time. In Figure 5-10 the critical path is depicted by a bold line—a common practice with PERT/CPM networks. We found the ES and EF for each activity quite easily by beginning at the start node and moving from left to right through the network, calculating as we go from node to node. This is called a "forward pass" (or "left-to-right pass") and makes it simple to find the critical path and time for PERT/CPM networks.

In a similar fashion, we can perform a "backward pass" (or "right-to-left pass") to calculate the LS and LF values for each activity. Referring to Figure 5-10, we begin by assuming that we would like to complete the project within the critical time identified in the forward pass, 21 days in our example. Clearly, activities **i** and **j** must be completed no later than Day 21 in order not to delay the entire project. Therefore, these activities both have LFs of 21.

Given a task time of 4 days, **j** must be started no later than Day 17 in order to be completed by Day 21. Likewise, task **i** can be started as late as Day 15 and still finished by Day 21 given its 6 day task time. Because task **j** cannot be started any later than Day 17, tasks **g** and **h** must be completed by Day 17. In a similar fashion, task **f** must be completed by Day 15 so as not to delay task **i** beyond its LS. Subtracting the task times from the LF for each of the tasks yields LSs of 11, 11, and 12 for tasks **f**, **h**, and **g**, respectively.

Now consider task **d**. Task **d** precedes both tasks **g** (LS = 12) and **h** (LS = 11). The question is, should **d**'s LF be 11 or 12? The correct answer is 11. The reason being, if task **d** were completed on Day 12, task **h** could not start on its LS of Day 11. Therefore we note that in situations where a particular activity precedes more than one task, its LF is equal to the minimum LSs of all activities it precedes.

The LFs and LSs for tasks **b**, **c**, and **e** are easily calculated since each of these tasks precedes a single task. Task **a** precedes tasks **c**, **d**, and **e**. Because task **e** has the lowest ES of five days, the LF for task **a** is calculated to be five days.

Calculating Activity Slack

If activities on the critical path cannot be delayed without causing the entire project to be delayed, does it follow that activities not on the critical path can be delayed without delaying the project? As a matter of fact, it does—within limits. The amount of time a noncritical task can be delayed without delaying the project is called *slack* or *float*. The slack for any activity is easily calculated as LS – ES = LF – EF = slack.

It should be clear that for any task on the critical path, its LF must be the same as its EF. It therefore has zero slack. If it finishes later than its EF, the activity will be late, causing a delay in the project. (Of course, the statement is equally true for its LS and ES.) This rule holds for activities **a**, **e**, **h**, and **j**, all on the critical path. But for activities not on the critical path the LF and EF (or the LS and ES) will differ and this difference is the activity slack. Take activity **i**, for example. It could be completed as early as Day 18 because its ES is Day 12 and it has a six-day duration. It must, however, be completed by Day 21 or the project will be delayed. Because **i** has a duration of six days, it cannot be started later than Day 15 (21−6). Given an LS of 15 and an ES of 12, task **i** could be delayed up to 3 days (LS – ES or LF – EF) without affecting project completion time. Thus, activity **i** has three days of slack.

For another example, consider task **g**. It is a predecessor to task **j** that is on the critical path. Task **j** *must* start on Day 17, which gives task **g** an LF of 17. It has an EF of Day 14 and thus has three days of slack.

Now consider a more complex situation: task **d**. First, task **d** must be completed by Day 12 so as not to make **g** late and thus **j** late and thereby the whole project late! Thus, along this path task **d** has *three* days of slack in the role as a predecessor of task **g**. But **d** is *also*