

## Project planning and management: CPM/PERT

We now study the problem of project planning and management. Consider the planning stage of a large scale project (for example, the construction of HK Disneyland in). Successful completion of the project requires completion of hundreds of different activities by several different groups or companies (e.g. MTR builds the rail links, HK Government builds roads, KMB builds bus routes, SHK builds hotels, a roller-coaster company from Japan builds one ride in the park, McDonalds builds some restaurants, etc.) Ideally, we want to complete the project as soon as possible. This could be done if each activity was done in parallel. However, it is often impossible to begin some activities before some other activity has been completed. For example, construction of buildings cannot begin until most of the trees in the Penny Bay area (site of Disneyland) have been cut down and the land is leveled. Thus we shall break a project into a set of well-defined activities. Further, we should know the estimate time it will take to complete each activity. This type of activity-based break-down is typical for most projects.

In different texts, there are two methods: the Critical Path Method (CPM), and the Project Evaluation and Review Technique (PERT), that are used to solve such problems. However, we will combine the two into a single procedure, CPM/PERT. The questions that the CPM/PERT analysis can answer include the following:

1. What is the earliest time that we can complete the entire project?
2. Given an activity, what is the earliest we can begin to do it?
3. Which activities are the bottlenecks (activities that, if their completion is delayed, the entire project will be delayed)?
4. What is the latest time that we can begin an activity, and still not delay the project? For example, it will take six months to hire and train a set of people to wear stuffed animal costumes and dance on the street; Disneyland will open in December 2005. So we don't have to begin the hiring and training until summer of 2005.

## Graphical modeling of Project Activities

In classical modeling of CPM/PERT as graphs, it is convenient to represent each activity as an edge. The weight of the edge indicates the time required to complete it. Each node represents a *milestone*, which is the point in time when a particular activity is completed and/or some activity is begun. The edges are directed, and the head of the edge is incident on the node representing the milestone corresponding to the completion of the activity.

The requirement that some activity cannot begin before another activity has been completed imposes a *precedence relationship*. Therefore, the arrows in our graph will depict the sequence in which the activities can be completed (equally: the paths indicate the sequence in which milestones can be achieved).

The graph also models the precedence relations of the milestones. If activity  $u$  must be completed before starting activity  $v$ , we indicate this by an arrow from the milestone (complete  $u$ ) to the milestone (begin  $v$ ). However, there may be no real activity between these two events – in which case this edge will have a weight = 0; such edges are called *dummy edges*, since they only represent a precedence, but not a real activity.

To successfully model a project in terms of a PERT network, we must first list out all precedence relationships. It is sufficient, for each activity, to identify its immediate predecessors.

### Example

Consider a project to construct a small Hydroelectric Power Plant. The details of the different activities, their precedence constraints, and the estimated times are shown in the table below.

Activity	Description	Immediate Predecessor	Duration
a	Ecological survey	-	6.2
b	File environmental impact report; get approval	a	9.1
c	Economic feasibility study	a	7.3
d	Preliminary design and cost estimation	c	4.2
e	Project approval and funding commitments	b, d	10.2
f	Call quotation for equipment (turbines, generators)	e	4.3
g	Select supplier for equipment	f	3.1
h	Final design of project	e	6.5
i	Select construction contractor	e	2.7
j	Arrange construction materials supply	h, i	5.2
k	Dam building	j	24.8
l	Power station building	j	18.4
m	Power lines erection	g, h	20.3
n	Turbines, generators installation	g, l	6.8
o	Build-up reservoir water level	k	2.1
p	Commission the generators	n, o	1.2
q	Start supplying water	m, p	1.1

Once we have the project plan data in a form such as above, it can be expressed in the form of a directed graph using the following method.

- (i) Construct a node,  $s$ , representing the start of the project.
- (ii) For each activity that has no immediate predecessor, make an edge incident from  $s$ .
- (iii) Other activities are added to extend the graph according to the list of immediate predecessors; when there are two activities such that some of their immediate predecessors are the same, but some are not, we need to introduce a dummy activity to represent the precedence constraint.
- (iv) In the end, all activities that have no successor are connected to a common node that represents the finish of the project.

We now construct the graph for our hydroelectric plant project. In the graph, I denote the edges by the activity names, in lower case letters. Each node represents the end of the activity or activities that are depicted by the edges *incident to* it; likewise, it depicts the point in time when we can begin any activity whose corresponding edge is *incident from* this node. The weight of the edge is the duration of the activity it represents. The dashed lines are the dummy edges: these edges are not given any name, and have weight=0. They are used to depict the precedence constraints.

Figure 1a shows the first few steps in constructing the PERT graph. We begin with node  $s$ , and add the edge  $a$ , for activity  $a$ , that will take 6.2 weeks. Since activity  $a$  is an immediate predecessor for both  $b$  and  $c$ , so the edges for these two activities are incident from the node depicting end of activity  $a$ , namely node  $A$ .

Figure 1b shows the situation where a dummy node is required. Activity  $c$  is an immediate predecessor of activity  $e$ . However, both of activities  $b$  and  $d$  are immediate predecessors of task  $e$ . Therefore we need a dummy edge connecting the events depicted by nodes  $B$  and  $D$ . Since task  $e$  can only begin when all activities depicted by the edges incident to its starting node (namely node  $D$  in the figure), this dummy edge ensures that task  $b$  is also completed before task  $e$  begins.

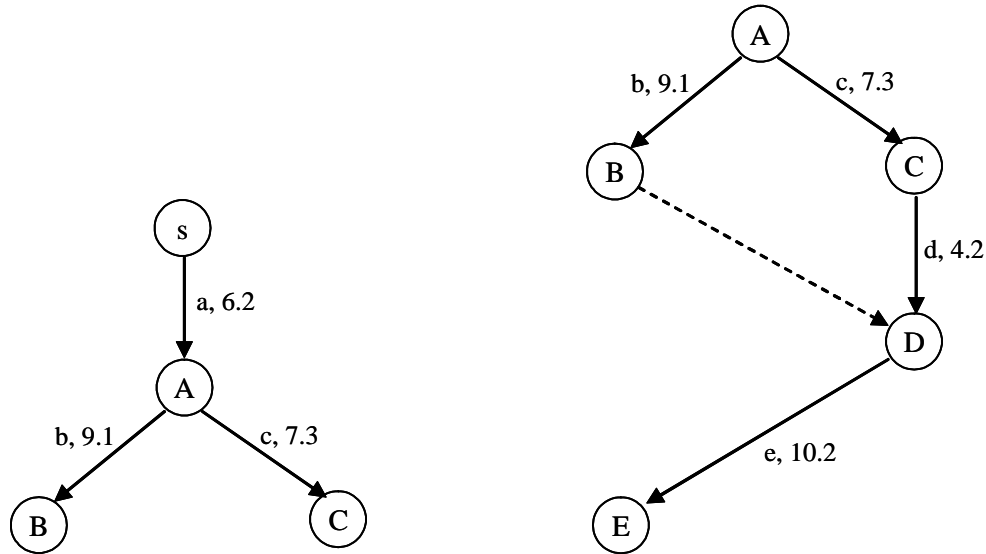


Figure 1

(a) Partial PERT graph for tasks  $a, b, c$       (b) Partial PERT chart for tasks  $b, c, d, e$

In this way, the entire graph is completed, as shown in Figure 2.

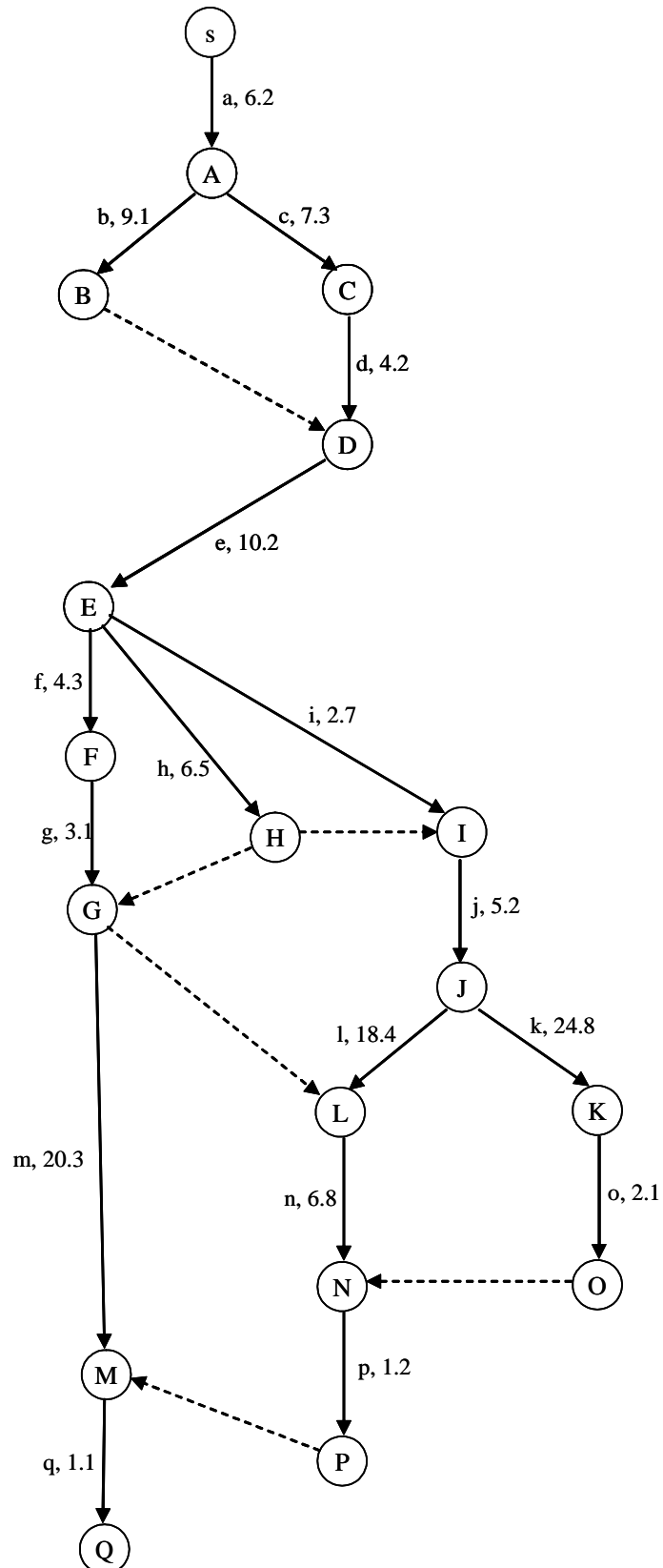


Figure 2. The PERT graph for our example

Once the PERT graph is completely constructed, we process in two stages. In the first stage, called the *Forward pass*, we estimate the *earliest possible start time* for each

activity. In the second stage, called the **Backward pass**, we estimate the *latest allowed start time* for each activity.

### Forward pass

We begin from the start node, and compute the earliest time that each successive activity can begin. Clearly, earliest start time at node  $s = 0$ . Any edge (activity) that is incident from this node can be traversed in time equal to its weight. This gives the completion time of the activity. The maximum completion time due to each edge incident on a node is the earliest start time of any activity that begins from this node. This is the necessary and sufficient condition for all precedence constraints to be satisfied.

Thus in our graph, the earliest end time of activity  $a = 6.2$ ; this is written in square brackets next to the node for “end of activity  $a$ ”, namely node A.

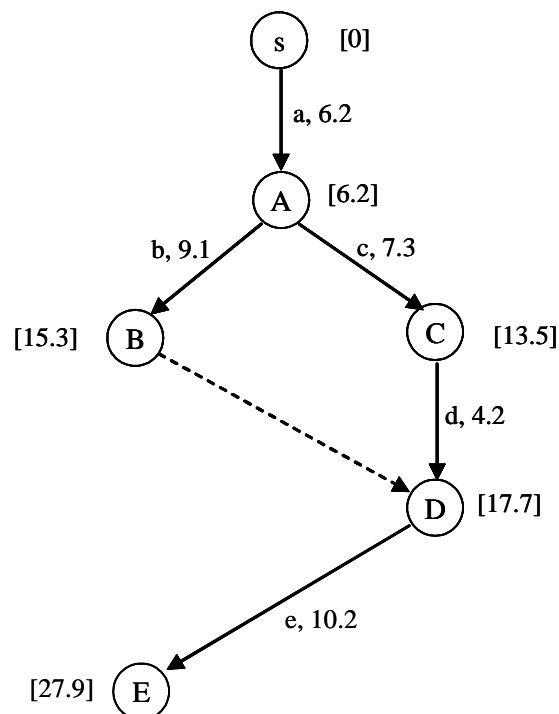


Figure 3. First few steps of the forward pass

Notice that activity  $e$  has two immediate predecessors:  $b$  (which cannot be completed before 15.3), and  $d$  (which cannot be completed before 17.7). Thus, task  $e$ , which begins from node D, cannot begin before the *maximum of these two*, namely 17.7; therefore  $e$  cannot end before  $17.7 + 10.2 = 27.9$ . Following this logic, the forward pass is completed as shown in Figure 4. From the graph, it is clear that the earliest time when we can complete the project is 68.8.

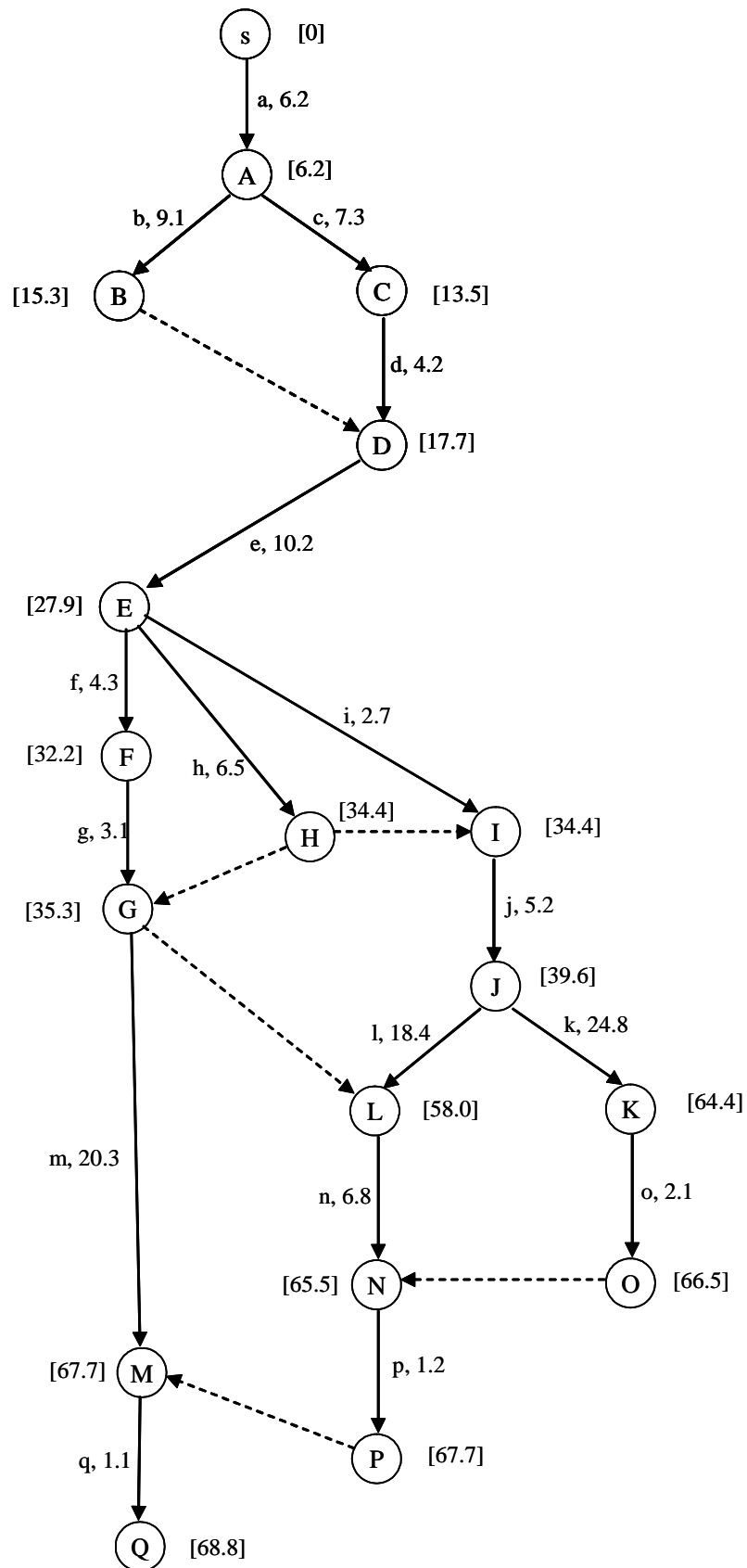


Figure 4. Forward pass for hydroelectric plant PERT example

### Backward pass

For the backward pass, we begin at the finish node (in our case, node Q). The objective is to answer the following question: given the *earliest completion time for the overall project*, what is the latest time we can begin a given activity, without causing any delay to the project?

Certainly we cannot complete the project before 68.8 weeks. The last activity that we perform is  $q$ , which takes 1.1 weeks and cannot begin until all other activities are done. Thus if  $q$  is started at any time after  $68.8 - 1.1 = 67.7$  weeks, then the project will get delayed. Since the start of activity  $q$  is at node M, we write this limit, 67.7, in the square bracket next to node M (see Figure 5a).

We work out the latest allowed start time for each node using this logic. The only case where we need to pay some attention in this process is depicted in Figure 5b. Node L indicates that activity  $l$ , whose duration is 18.4 weeks, must be completed no later than 59.7. Therefore it must begin at a time no later than  $59.7 - 18.4 = 41.3$  weeks. Using the same logic for activity  $k$ , we see that latest start time for activity  $k$  is  $64.4 - 24.8 = 39.6$ . So which of these two numbers, 41.3 and 39.6, is correct for event J? Clearly, we must pick the smaller of the two numbers – since if we denote the latest time at node J by 41.3, then we could only complete activity  $k$  at  $41.3 + 24.4 = 65.7$ , which will delay the project completion time (if we reach node K at any time later than 64.4, we will delay the project).

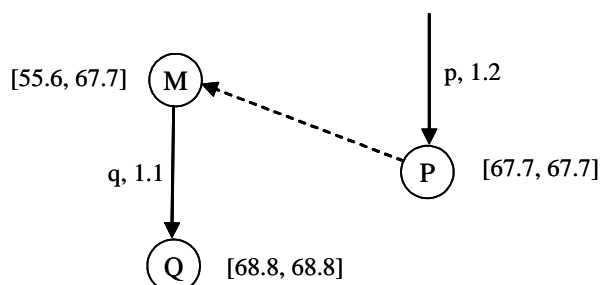


Figure 5a. First few steps of Backward Pass

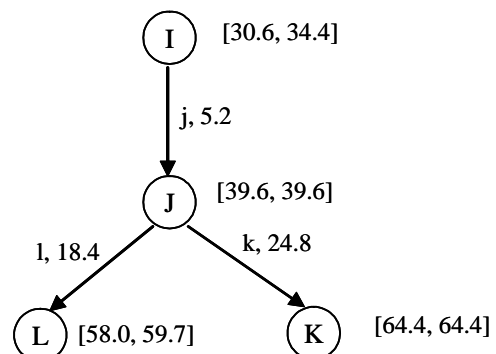


Figure 5b. Logic at branch points

Once more, we follow this logic all the way back to the start node, and complete the latest allowed completion time for each activity. The final graph is after the backward pass is shown in Figure 6. We make the following definition:

**Slack time:** The slack time for *an event* is the difference between its latest and earliest time.

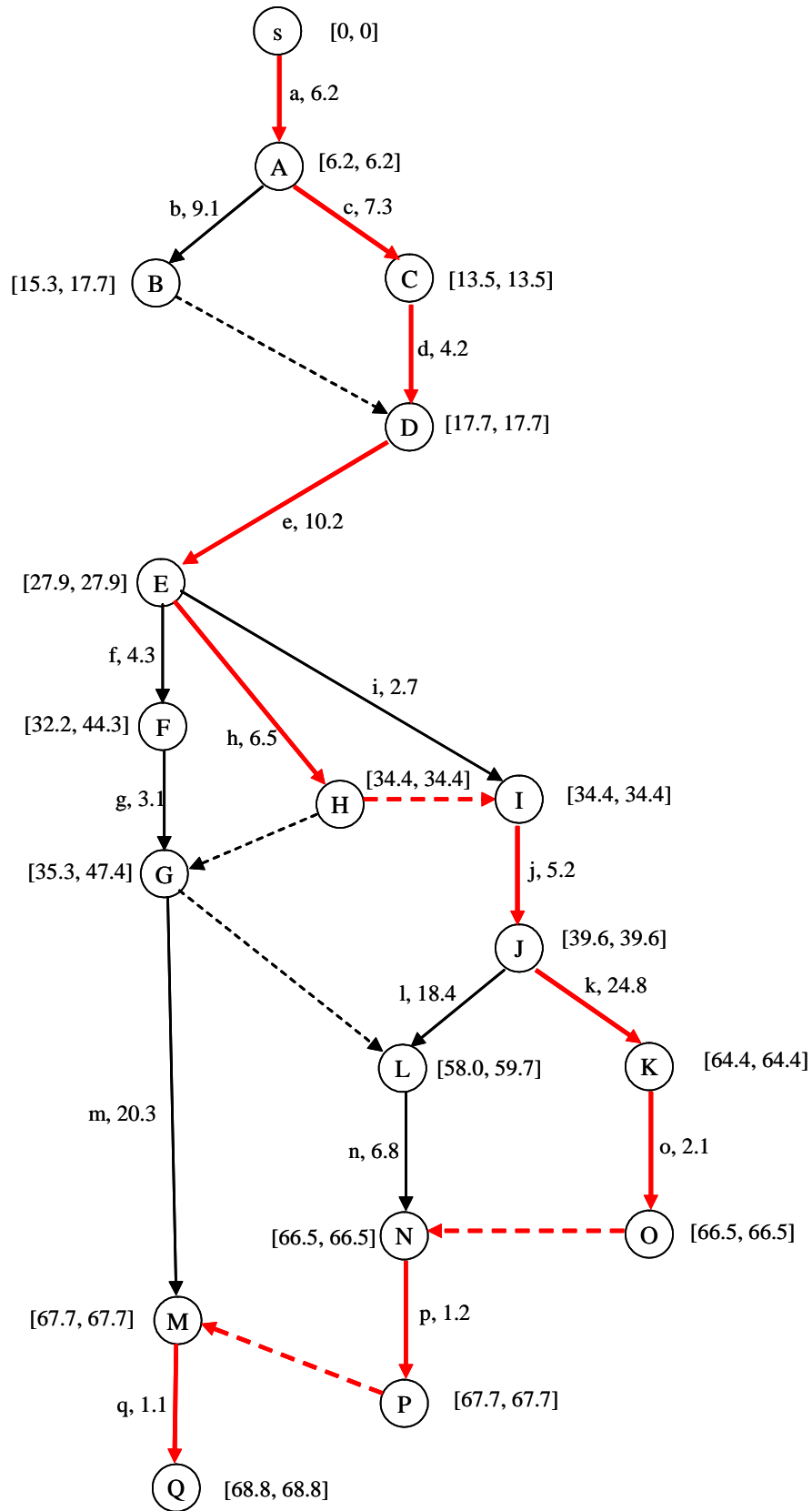


Figure 6. Completed PERT graph for the example problem



Further, we define a **critical path** as any path from the start to finish such that every event on the path has a zero slack time. Note that a given graph may have more than one critical path.

The activities on any critical path are called the **critical activities**. Because of these terms, the procedure is also called the *Critical Path Method*. We can now answer the questions raised in the beginning:

(a) What is the earliest time that we can complete the entire project?

This is the time that we reach the final event (in our example, node Q), as computed by the forward pass.

(b) Given an activity, what is the earliest we can begin to do it?

This is the time indicated by the number at the node corresponding to the start of the activity in the PERT graph, after performing the forward pass.

(c) Which activities are the bottlenecks?

The critical activities are the bottleneck activities.

(d) What is the latest time that we can begin an activity, and still not delay the project?

This is the number computed by the backward pass (the second number in the square brackets next to the event of interest).

We finish this discussion with a few remarks about CPM/PERT methods, especially two major problems that complicate things in real-life projects.

(a) It is often very difficult to estimate exact time for any activity. Hence most modern PERT methods actually associate three time values with each activity: the best case time, the expected time, and the worst case time. Then a probabilistic model is built, using the worst case, best case, and expected duration of each task. This approach is better than deterministic CPM/PERT, but has its own difficulties. The probability distributions themselves are inaccurate; secondly, some activities may not be independent, and it is not easy to model such relationships in probabilistic models.

(b) The second problem is that in constructing PERT graphs, we only consider the precedence relations between tasks. However, in real life projects, several tasks may share some physical resources (e.g. cranes may be used in different tasks during the project). If two or more jobs require the same resource, it may not be possible to do them simultaneously. This type of constraint is not captured by PERT. To consider such cases, we must use a different type of problem model. Such models are studied in a different area of IELM, called resource constrained scheduling. It is sometimes possible to construct algebra-based (e.g. linear programming) formulations of such models.