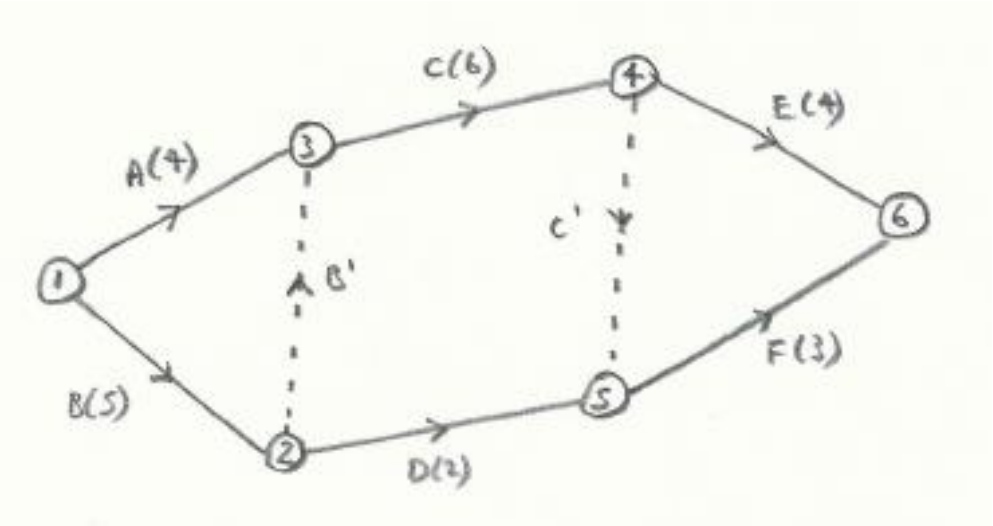
Critical Path Analysis - Part 1 (12 pages; 21/1/20)

(1) The starting point is the **precedence table** (aka dependence table)

Example

Activity	Duration	Immediately preceding activities
A	4	-
B	5	-
C	6	A,B
D	2	B
E	4	C
F	3	C,D

(2) The **activity network** can then be constructed from this:



[The dashed arcs (the 'dummy' activities) are explained below. The labels B' and C' are intended to show that they relate to B and C, respectively. They would normally be omitted]

With the networks involved in Dijkstra's Algorithm or the Route Inspection problem, we are only concerned with one route (the shortest one, and the one that covers all the arcs in the minimum distance, respectively). For Critical Path Analysis, however, the various branches of the network are in action at the same time.

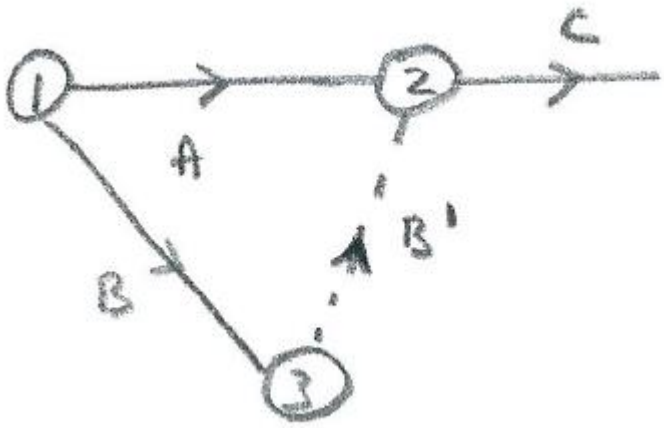
This can be described as an **activity-on-arc** network. The nodes represent the starting points of activities. The first node is sometimes called the **source node**, and the final node is sometimes called the **sink node**.

[An alternative method uses an **activity-on-node** network. An example of this is given later. The AQA exams are based on this method.]

(3) Dummy Activities (activity on arc only)

The dummy activity is a device used to deal with a couple of complications. It has zero duration.

(a) To avoid two activities having the same start and end nodes (and instead have unique labellings) [Note: the B' label wouldn't normally be included.] See the diagram below.



(b) A' in Fig.1 below is needed to reflect the fact that, whereas E depends on A & C, D only depends on A (as opposed to the situation in Fig. 2, where both D & E depend on A & C).

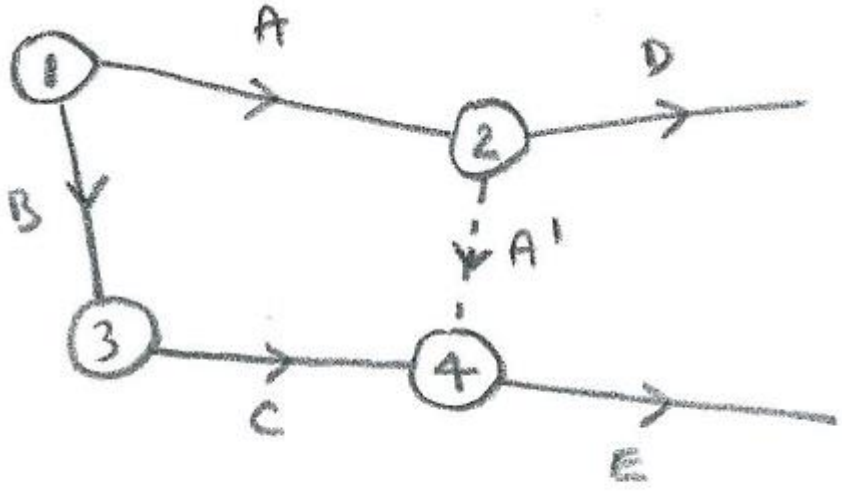


Fig. 1

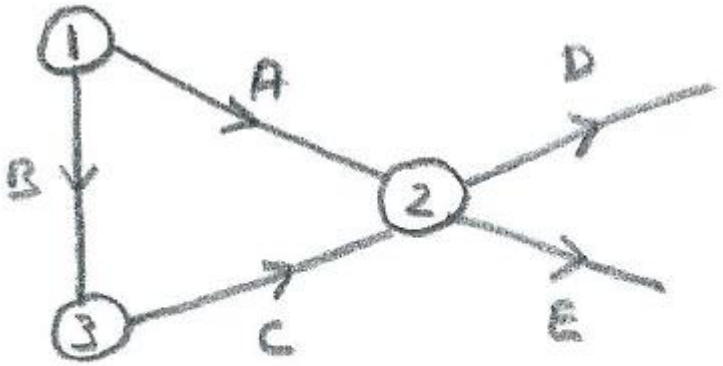


Fig. 2

(c) More complicated versions of (b) are possible:

Exercise: Construct an activity network for the following part of a precedence table:

	depends on
A	–
B	–
C	–
D	A, B
E	A, C

Solution

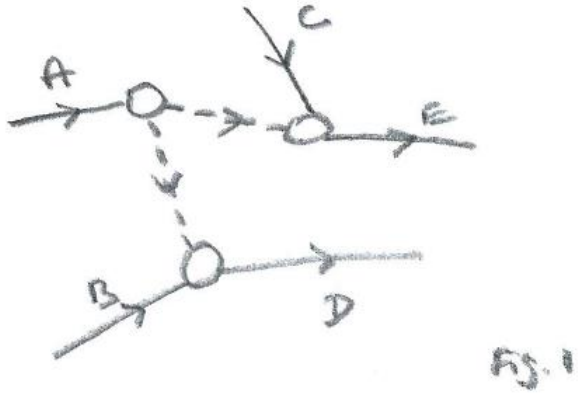


Fig. 1

Note: Fig. 2 (below) doesn't work, because it implies that E depends on B. Also, Fig. 3 doesn't work, because it implies that D depends on C.

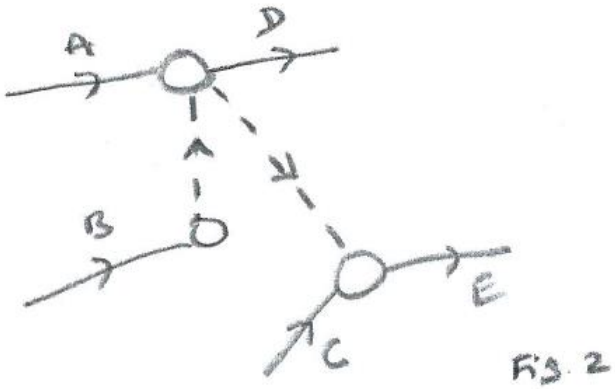


Fig. 2

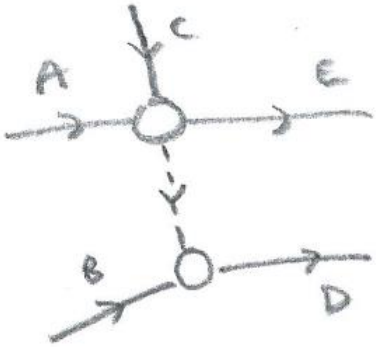
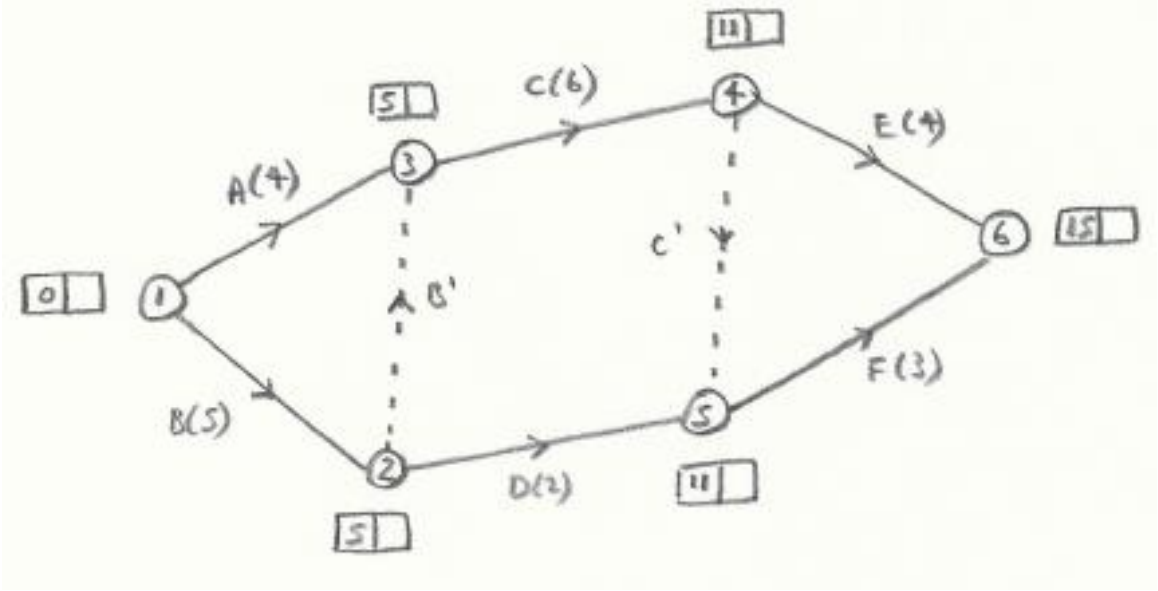


Fig. 3

(4) Earliest/latest event times

Each node has associated with it an **earliest event time** [EET](aka early event time), which is the earliest time that activities leading from it can start, based on the timings of the earlier activities. The EET of the final node is the **critical time** or **minimum completion time** for the whole operation. This is shown below for the earlier example.



For example, the EET for node 3 is determined as follows:

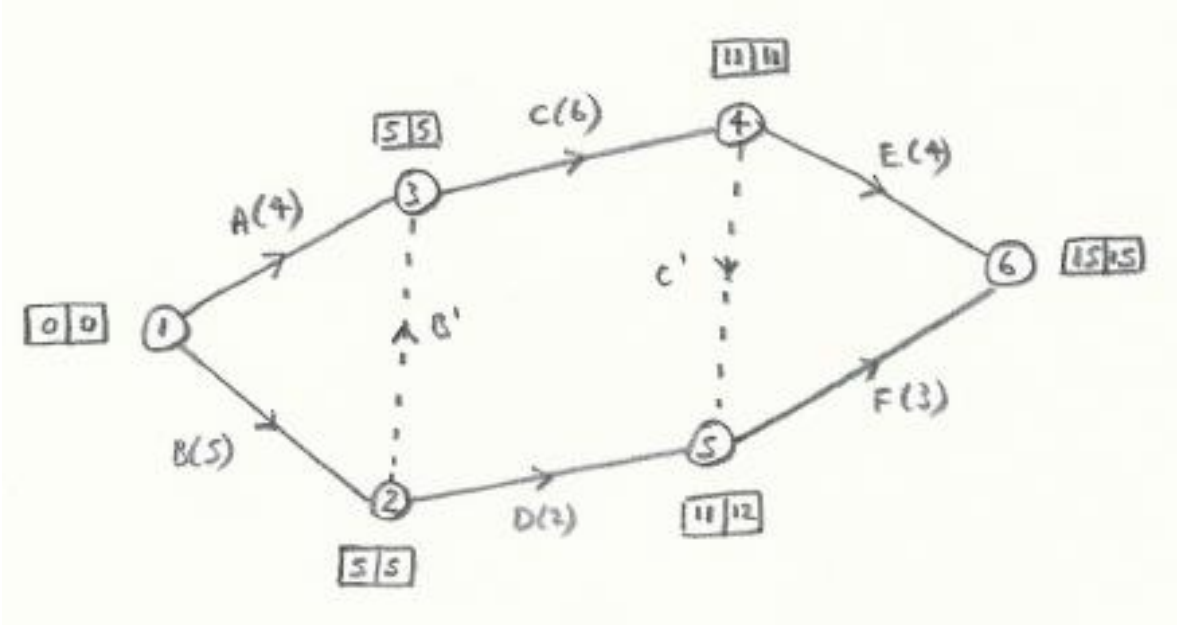
Activity C has to wait until activities A and B' (and hence B) have been completed; ie EET for node $3 = \max(4,5) = 5$.

The process of establishing the EETs is known as a **forward pass**.

To clarify: these earliest event times indicated at the nodes are elapsed times from the start, whereas the times shown for the activities are the durations of those activities.

A **latest event time** [LET] (aka late event time) is also established for each node: this is the latest time at which the activities leading from that node can start, in order that the critical time for the whole operation can still be achieved.

The LETs are determined by setting the LET for the final node equal to its EET, and working back from the final node, as shown below:



For example, the LET for node 2 is determined as follows:

Activity D could start as late as time 10, in order for activity F to have started by time 12. However, activity C has to start by time 5 and, because of the dummy activity B^1, the activities leading from node 2 cannot therefore be delayed beyond time 5.

The process of establishing the LETs is known as a **backward pass.**

(5) Earliest start and latest finish times

Suppose activity (i, j) starts at node i and finishes at node j. Then the **earliest start time** for this activity is $EET(i)$, and the **latest finish time** is $LET(j)$. (These terms sound very similar to "earliest event time" and "latest event time", but we are now considering an activity, whereas the events relate to the points where activities start and end.)

These expressions can be used to define the total float (as seen below), and are also used in the activity-on-node method for constructing the activity network, as described next.

(6) Activity-on-node method

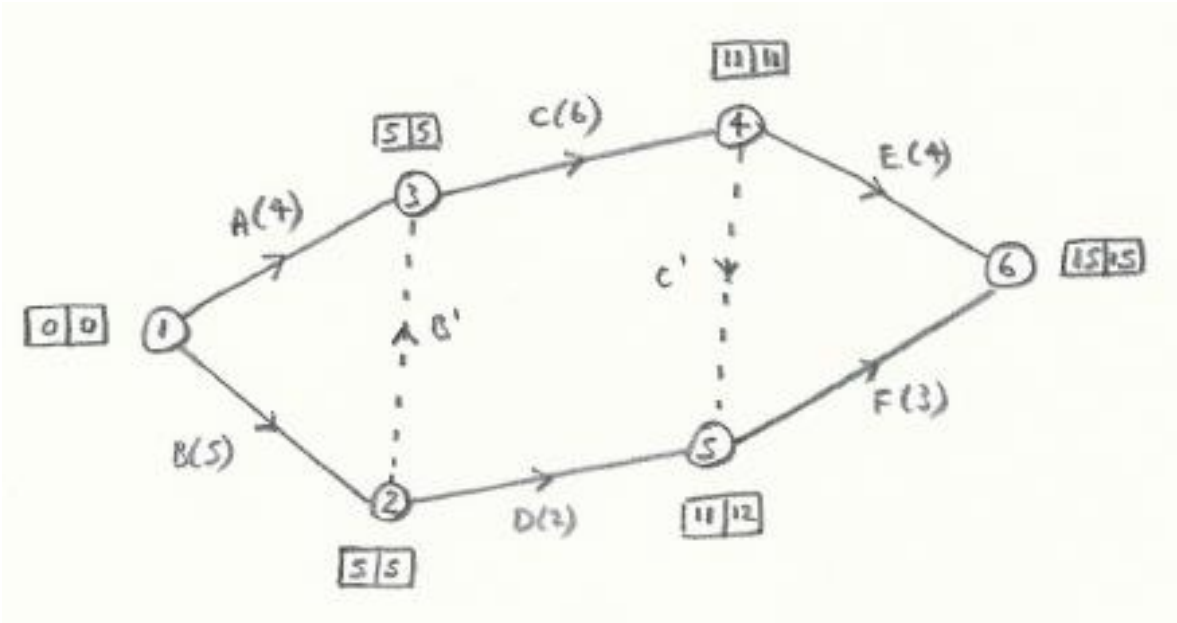
As the title suggests, each activity is represented by a node, which shows the following information for the activity:

- the earliest start time

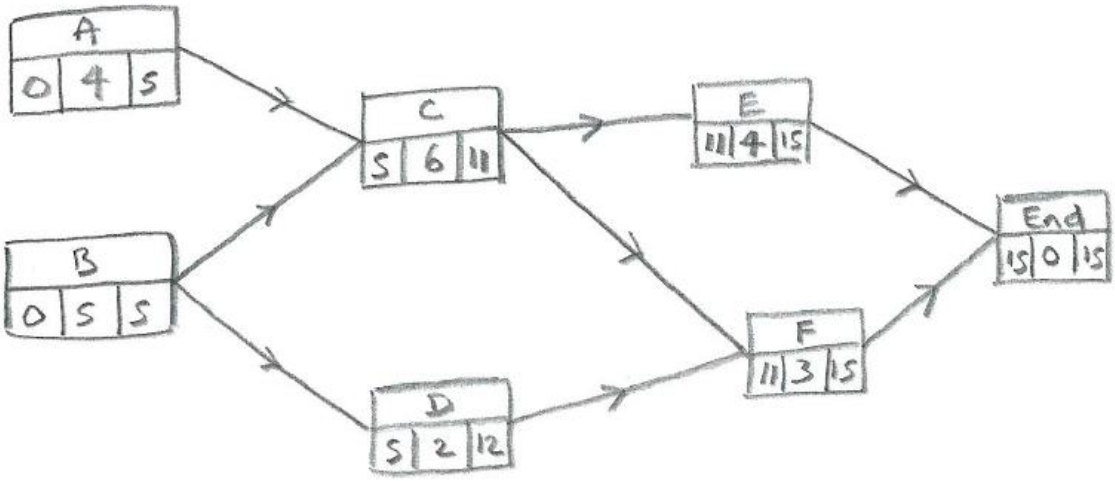
- the duration

- the latest finish time

The network for the example above, using the activity-on-arc method, is reproduced below, and the corresponding activity-on-node network is then shown.



Activity-on-arc (above)



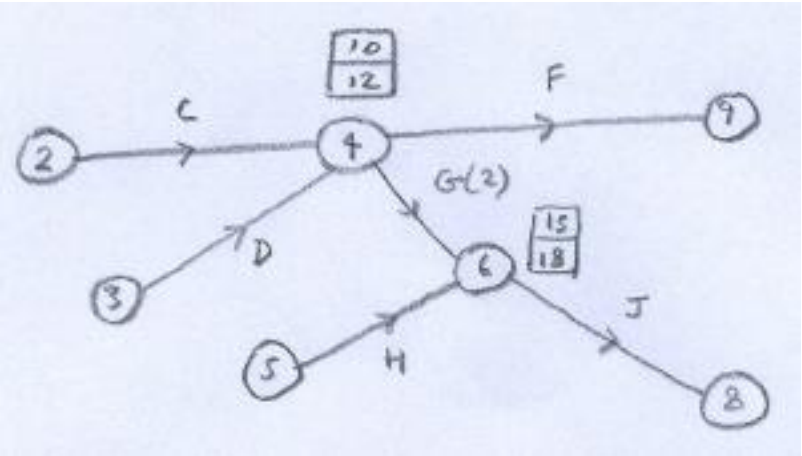
Activity-on-node (above)

Notes

(i) A forward pass is carried out to determine the earliest start times, and a backward pass to determine the latest finish times.

(ii) No dummy activities are needed for this method.

(7) Floats



Referring to the diagram above:

10 is the earliest time by which C and D can both be completed (and hence the earliest time by which F or G could start).

12 is the latest time by which both C and D need to have been completed, in order that F or G (as necessary) can start early enough for the critical time to be achieved.

$18 - 10 = 8$ is the maximum time available for G, if given priority over other activities for any spare time.

2 is the (minimum) duration needed for G.

The **total float** for G $= 8 - 2 = 6$ (ie the maximum spare time available).

The **independent float** for G $= (15 - 12) - 2 = 1$ (the float that G can be assured of, without relying on the other activities).

The independent float is set equal to zero, if it would otherwise be negative.

The **interfering float** is $6 - 1 = 5$ (ie total float − independent float).

In general, the total float is

$LET(j) - EET(i) -$ duration of activity,

where i represents the 1st node and j represents the 2nd

(or latest finish time − earliest start time −duration of activity)

and the independent float is:

$Max\{EET(j) - LET(i) -$ duration of activity, $0\}$

(8) Critical Activities

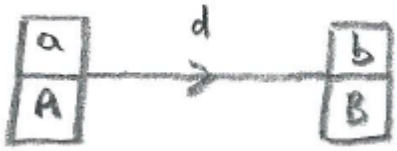
A **critical activity** is where there is no margin for delay, and so where the total float is zero.

Thus $LET(j) - EET(i) =$ duration of activity

Also, it can be shown that

$LET(i) = EET(i)$ and $LET(j) = EET(j)$.

Proof



Referring to the diagram above,

Critical activity $\Rightarrow B - a = d$ (1)

Also $b \geq a + d$ (2) & $A \leq B - d$ (3)

and of course $a \leq A$ (4) & $b \leq B$ (5)

Then , from (1) & (3), $A \leq (d + a) - d = a$,

so that $A \leq a$ & $a \leq A$, from (4), and hence $a = A$

Also, from (1) & (2), $b \geq B$ and since $b \leq B$ from (5), it follows that $b = B$.

So a critical activity can always be represented as shown below.

(9) Critical Paths

A **critical path** is one made up entirely of critical activities, and its length is the critical time. (It is sometimes noted that this length is the longest length of a path in the network: paths involving non-critical activities will be shorter.)

As we have seen, critical paths can be identified by looking for the nodes where $LET = EET$, and joining the ones where

$LET(j) - EET(i) =$ duration of activity

(Note that there may be activities for which $LET(i) = EET(i)$ and $LET(j) = EET(j)$, but where $LET(j) - EET(i) >$ duration of activity, and these are obviously not critical activities.)

There may be more than one critical path. If so, they will all have the same length.

For the network in (4), the only critical path is $BB'CE$.