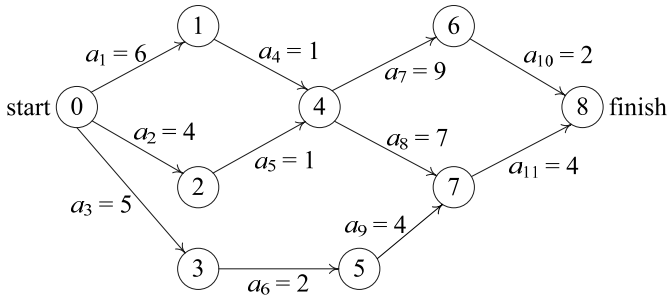
Other Factors (2/3)

- Latest time of activity
  - The latest time the activity may start without increasing the project duration
  - $e(6) = 5$ and $l(6) = 8$, $e(8) = 7$ and $l(8) = 7$



Other Factors (3/3)

- Critical activity
  - An activity for which $e(i) = l(i)$
  - $e(8) = l(8)$
- $l(i) - e(i)$
  - A measure of the criticality of an activity
  - $l(6) - e(6) = 8 - 5 = 3$

Critical-Path Analysis

- **Purpose:** To identify critical activities so that resources may be concentrated on these activities to reduce project finish time.

➡ Speeding a critical activity will not result in a reduced project length unless that activity is on all critical paths.

(1) Calculate $e(i)$ and $l(i)$ for all activities in an AOE network to identify the critical activities.

(2) Delete all noncritical activities.

(3) Generate all the paths from the start to finish vertex.

Earliest and Latest Event Times

- When computing the early and late activity times, it is easiest first to obtain the earliest event time, $ee[j]$, and latest event time, $le[j]$, for all events, j, in the network.
- The times $ee[j]$ and $le[j]$ are computed in two stages: a forward stage and a backward stage.

Calculation of the Earliest Event Time (1/2)

- During the forward stage, we start with $ee[0] = 0$ and compute the remaining early start times, using the formula

$$ee[j] = \max_{i \in P(j)} \{ee[i] + \text{duration of} < i, j >\}$$

where $P(j)$ is the set of all vertices adjacent to vertex j.

➡ We can modify topSort to return the vertices in topological order.

➡ We must have easy access to the vertex set $P(j)$

Calculation of the Earliest Event Time (2/2)

- We begin with the *ee* array initialized to zero and insert the code

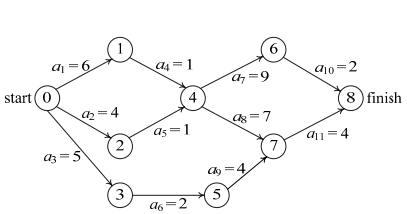
```
if (ee[k] < ee[j] + ptr->duration)
    ee[k] = ee[j] + ptr->duration;
```

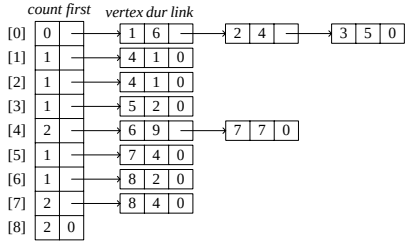
just after the line (topSort)

```
k = ptr->vertex;
```

➡ ee(k) is updated each time the ee() of one of its predecessors is known (i.e., when j is ready for output).

Computing ee Using Modified topSort





ee	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	Stack
initial	0	0	0	0	0	0	0	0	0	[0]

Calculation of the Latest Event Time (1/2)

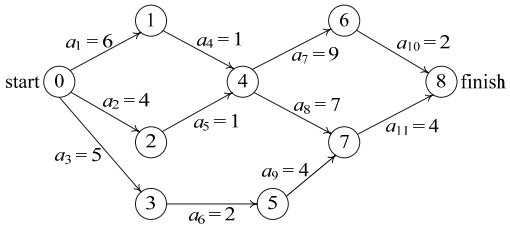
- In the backward stage, the values of $le[i]$ are computed by starting with $le[n–1] = ee[n–1]$ and using the equation

$$le[j] = \min_{i \in S(j)} \{le[i] - \text{duration of} < j, i >\}$$

where $S(j)$ is the set of vertices adjacent from vertex j.

➡ If $<j, i>$ is an activity and the latest start time for event i is $le[i]$, then event j must occur no later than $le[i]$ – duration of $<j, i>$.

Calculation of the Latest Event Time (2/2)



$le[8] = ee[8] = 18$
$le[6] = \min\{le[8] - 2\} = 16$
$le[7] = \min\{le[8] - 4\} = 14$
$le[4] = \min\{le[6] - 9;\ le[7] - 7\} = 7$
$le[1] = \min\{le[4] - 1\} = 6$
$le[2] = \min\{le[4] - 1\} = 6$
$le[5] = \min\{le[7] - 4\} = 10$
$le[3] = \min\{le[5] - 2\} = 8$
$le[0] = \min\{le[1] - 6;\ le[2] - 4;\ le[3] - 5\} = 0$

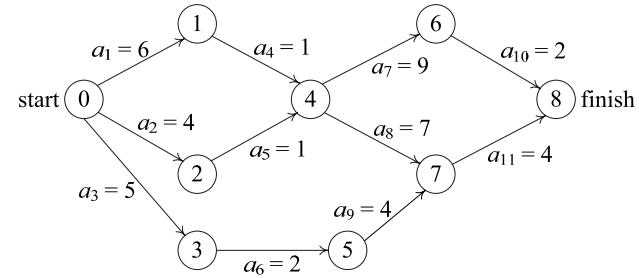
Calculation of Early/Late Activity Times

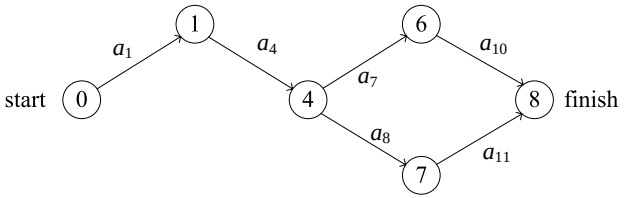
- Using the values of ee and of le, we may compute the early and late time $e(i)$ and $l(i)$ and the degree of criticality (also called slack) from:
  - $e(i) = ee[k]$
  - $l(i) = le[l]$ – duration of activity a_i

Early, Late, and Criticality Values

activity	early time e	late time l	slack $l - e$	critical $l - e = 0$
a_1	0	0	0	Yes
a_2	0	2	2	No
a_3	0	3	3	No
a_4	6	6	0	Yes
a_5	4	6	2	No
a_6	5	8	3	No
a_7	7	7	0	Yes
a_8	7	7	0	Yes
a_9	7	10	3	No
a_{10}	16	16	0	Yes
a_{11}	14	14	0	Yes

Critical Network





Graph obtained after deleting all noncritical activities

AOE Network with Unreachable Activities

- When a critical-path analysis is carried out on such networks, there will be several vertices with $ee[i] = 0$.

➡ Since only the start vertex can have $ee[i] = 0$, critical-path analysis can be used to detect unreachable activities in project planning.