Network Model

Dr Sujit Das

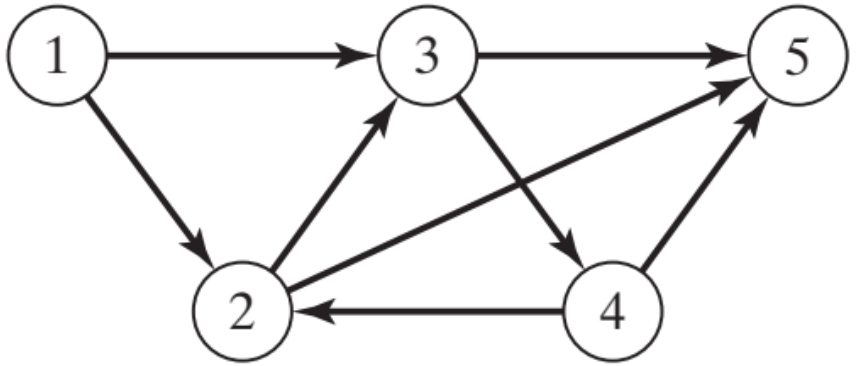
NIT Warangal

Scope of Network Model

- Designing of Gas pipeline connecting various cities with the objective of minimizing the cost of constructing the pipeline.
- Determination of the shortest route between two cities in an existing network of roads
- Determination of the time schedule (start and completion dates) for the activities of a construction project.
- Determination of the minimum-cost flow schedule from oil fields to refineries through a pipeline network
- The solution of these situations is accomplished through a variety of network optimization algorithms.
  - **1.** Minimal spanning tree (situation 1)
  - **2.** Shortest-route algorithm (situation 2)
  - **3.** Maximal-flow algorithm (situation 3)
  - **4.** Critical Path Method (CPM) algorithm (situation 4)

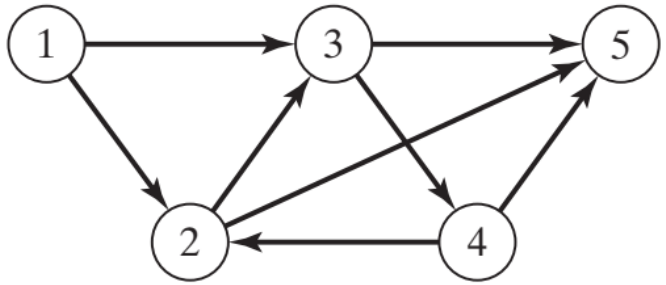
Network Definitions

- A network consists of a set of **nodes** linked by **arcs** (or **branches**).
- The notation for describing a network is (*N, A*), where *N* is the set of nodes, and *A* is the set of arcs.
- As an illustration, the network in given below in Figure is described as
- $N = \{1, 2, 3, 4, 5\}$
- $A = \{(1, 2), (1, 3), (2, 3), (2, 5), (3, 4), (3, 5), (4, 2), (4, 5)\}$



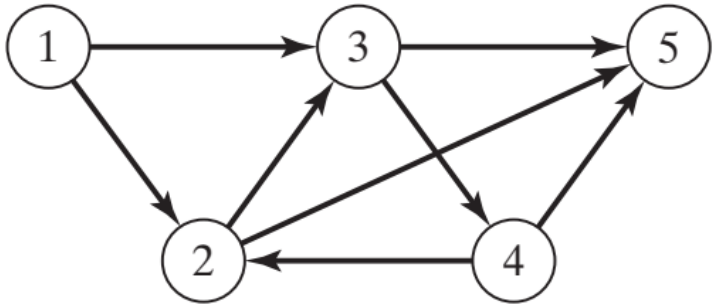
Some ideas on Network Model

- A **flow** (e.g., oil products flow in a pipeline and automobile traffic flow in highways) is associated with each network.
- The maximum flow in a network can be finite or infinite, depending on the capacity of its arcs.
- An arc is said to be **directed** or **oriented** if it allows positive flow in one direction only.
- A **directed network** has all directed arcs.
- A **path** is a set of arcs joining two distinct nodes, passing through other nodes in the network.
- For example, in the following Figure, arcs (1, 2), (2, 3), (3, 4), and (4, 5) form a path between nodes 1 and 5.
- A path forms a **cycle** or a **loop** if it connects a node back to itself through other nodes. In the given Figure, arcs (2, 3), (3, 4), and (4, 2) form a cycle

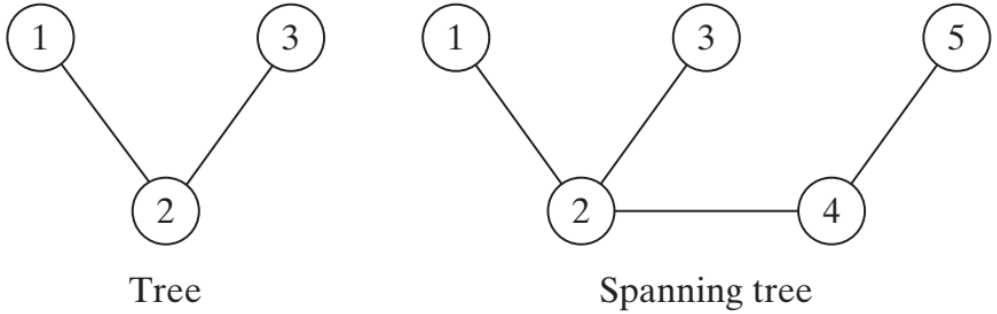


Connected Graph and Spanning Tree

- A network is said to be **connected** if every two distinct nodes are linked by at least one path.
- The network in the given Figure demonstrates this type of network.
- A **tree** is a *cycle-free* connected network comprised of a *subset* of all the nodes, and a **spanning tree** links *all* the nodes of the network.
- Figure (given below) provides examples of a tree and a spanning tree



Example of a Network



Example of Tree and Spanning Tree fro the Network

Minimal Spanning Tree Algorithm

- The minimal spanning tree links the nodes of a network using the smallest total length of connecting branches.
- A typical application occurs in the pavement of roads linking towns, either directly or passing through other towns.
- The minimal spanning tree solution provides the most economical design of the road system.

Let $N = \{1, 2, \ldots, n\}$ be the set of nodes of the network and define

C_k = Set of nodes that have been permanently connected at iteration k

$\overline{C}_k$ = Set of nodes as yet to be connected permanently after iteration k

The following steps describe the minimal spanning tree algorithm:

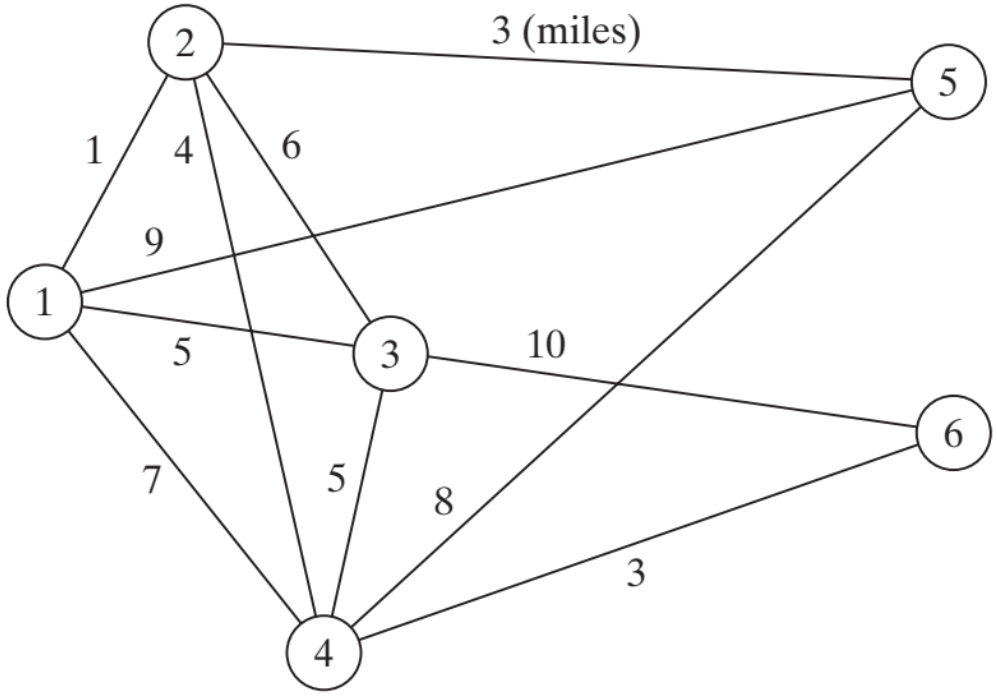
Step 0. Set $C_0 = \varnothing$ and $\overline{C}_0 = N$.

Step 1. Start with *any* node i in the unconnected set $\overline{C}_0$ and set $C_1 = \{i\}$, rendering $\overline{C}_1 = N - \{i\}$. Set $k = 2$.

General step *k*. Select a node, j^*, in the unconnected set $\overline{C}_{k-1}$ that yields the shortest arc to a node in the connected set C_{k-1}. Link j^* permanently to C_{k-1} and remove it from $\overline{C}_{k-1}$ to obtain C_k and $\overline{C}_k$, respectively. Stop if $\overline{C}_k$ is empty; else, set $k = k + 1$ and repeat the step.

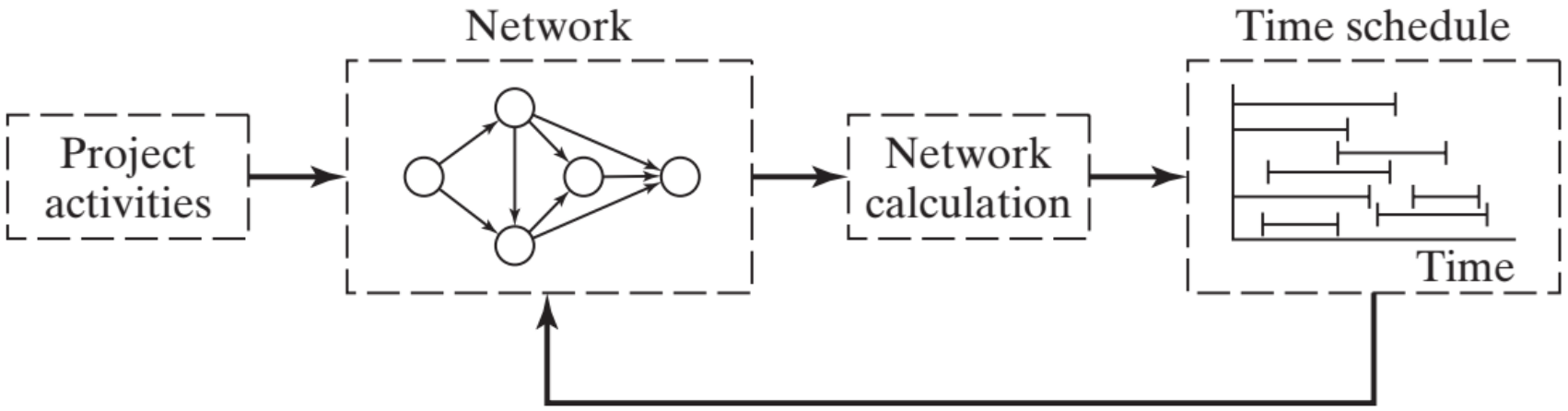
Example: Cable connections for Midwest TV Company

- Midwest TV Cable Company is providing cable service to five new housing developments.
- Following Figure depicts possible TV connections to the five areas, with cable miles affixed on each arc.
- The goal is to determine the most economical cable network.



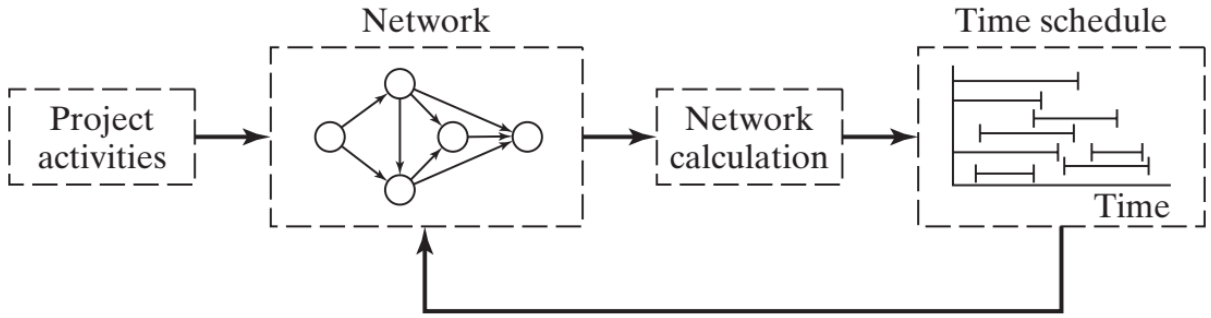
CPM (Critical Path Method) and PERT (Program Evaluation and Review Technique)

- CPM and PERT are network-based methods designed to assist in the planning, scheduling, and control of projects.
- A project is defined as a collection of interrelated activities with each activity consuming time and resources.
- The objective of CPM and PERT is to devise analytic tools for scheduling the activities. Following Figure summarizes the steps of the techniques.
- The two techniques, CPM and PERT, were developed independently. They differ in that CPM assumes deterministic activity durations and PERT assumes probabilistic durations.



CPM and PERT

- First, we define the activities of the project, their precedence relationships, and their time requirements.
- Next, the precedence relationships among the activities are modeled as a network.
- The third step involves specific computations for developing the time schedule.
- During the actual execution phase, execution of the activities may not proceed as planned, in the sense that some of the activities may be expedited or delayed.
- When this happens, the schedule is updated to reflect the realities on the ground. This is the reason for including a feedback loop.

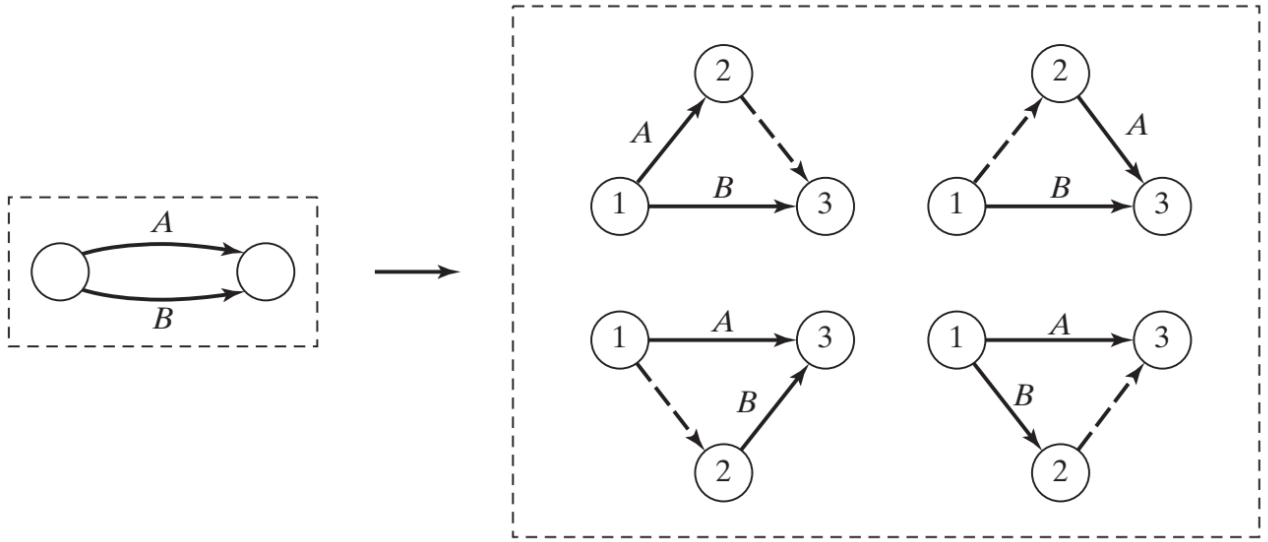


Network Representation

- Each activity is represented by an arc pointing in the direction of progress in the project.
- The nodes of the network establish the precedence relationships among the different activities.
- Three rules are available for constructing the network.
  - **rule 1.** *Each activity is represented by one, and only one, arc.*
  - **rule 2.** *Each activity must be identified by two distinct end nodes.*
  - **rule 3.** *To maintain the correct precedence relationships, the following questions must be answered as each activity is added to the network:*
    - **(a)** *What activities immediately precede the current activity?*
    - **(b)** *What activities immediately follow the current activity?*
    - **(c)** *What activities are concurrent with the current activity?*
- The answers to these questions may require the use of dummy activities to ensure correct precedence among the activities

Dummy activity

- Following Figure shows how a **dummy activity** can be used to provide unique representation of two concurrent activities, *A* and *B*.
- By definition, a (dashed) dummy activity consumes no time or resources.
- Inserting a dummy activity in one of the four ways shown in Figure maintains the concurrence of *A* and *B* and provides unique end nodes for the two activities (to satisfy rule 2).

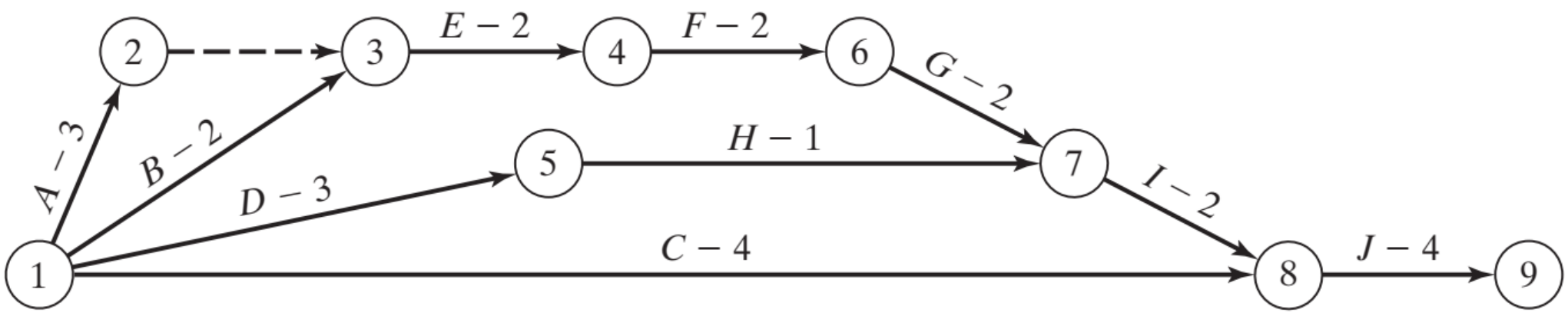


Example

- A publisher has a contract with an author to publish a textbook. The author submits a hard copy and a computer file of the manuscript. The (simplified) activities associated with the production of the textbook are summarized in the following table:

Activity	Predecessor(s)	Duration (weeks)
A: Manuscript proofreading by editor	—	3
B: Sample pages preparation	—	2
C: Book cover design	—	4
D: Artwork preparation	—	3
E: Author's approval of edited manuscript and sample pages	*A*, *B*	2
F: Book formatting	*E*	4
G: Author's review of formatted pages	*F*	2
H: Author's review of artwork	*D*	1
I: Production of printing plates	*G*, *H*	2
J: Book production and binding	*C*, *I*	4

Activity	Predecessor(s)	Duration (weeks)
A: Manuscript proofreading by editor	—	3
B: Sample pages preparation	—	2
C: Book cover design	—	4
D: Artwork preparation	—	3
E: Author's approval of edited manuscript and sample pages	*A*, *B*	2
F: Book formatting	*E*	4
G: Author's review of formatted pages	*F*	2
H: Author's review of artwork	*D*	1
I: Production of printing plates	*G*, *H*	2
J: Book production and binding	*C*, *I*	4



Critical Path Method (CPM) Computations

- The end result in CPM is a time schedule for the project
- To achieve this goal, special computations are carried out to produce the following information:
  - **1.** Total duration needed to complete the project
  - **2.** Classification of the activities of the project as *critical* and *noncritical*
- An activity is **critical** if its start and finish times are predetermined (fixed).
- An activity is **noncritical** if it can be scheduled in a time span greater than its duration, permitting flexible start and finish times (within limits).
- A delay in the start time of a *critical activity* definitely causes a delay in the completion of the entire project, whereas a delay in a *noncritical* activity may not affect the completion date of the project.

Event

- To carry out the necessary computations, we define an **event** as a point in time at which activities are completed and succeeding ones are started.
- In terms of the network, an event corresponds to a node.
- $\square_j$ = Earliest occurrence time of event j
- Δ_j = Latest occurrence time of event j
- D_{ij} = Duration of activity (i, j)
- All event occurrence times are measured from the start time of the project.
- The span $(\square_i, \Delta_j)$ defines the time period during which activity (i, j), of duration D_{ij}, is scheduled.
- If activity (i, j) is critical, then $D_{ij} = \Delta_j - \square_i$.
- Otherwise, $D_{ij} < \Delta_j - \square_i$ for noncritical activity (i, j)

Forward pass and Backward pass

- The critical path calculations involve two passes: **Forward pass and Backward pass**
- The **forward pass** determines the *earliest* occurrence times of the events, and the **backward pass** calculates their *latest* occurrence times.

Forward pass (earliest occurrence times, □)

- The computations start at node 1 and advance recursively to node *n.*
- **Initial Step.**
  - Set $\square_1 = 0$ to indicate that the project starts at time 0.
- **General Step *j*.**
  - Given that nodes *p*, *q*, . . . , and v are linked *directly* to node *j* by incoming activities (*p*, *j*), (*q*, *j*), . . . , and (v, *j*) and that the earliest occurrence times of events (nodes) *p*, *q*, . . . , and v have already been computed, then the earliest occurrence time of event *j* is computed as
  - $\square_j = \max\{\square_p + D_{pj}, \square_q + D_{qj}, \ldots, \square_v + D_{vj}\}$
  - The forward pass is complete when $\square_n$ at node *n* has been computed.
  - By definition, $\square_j$ is the longest path (duration) to node *j*

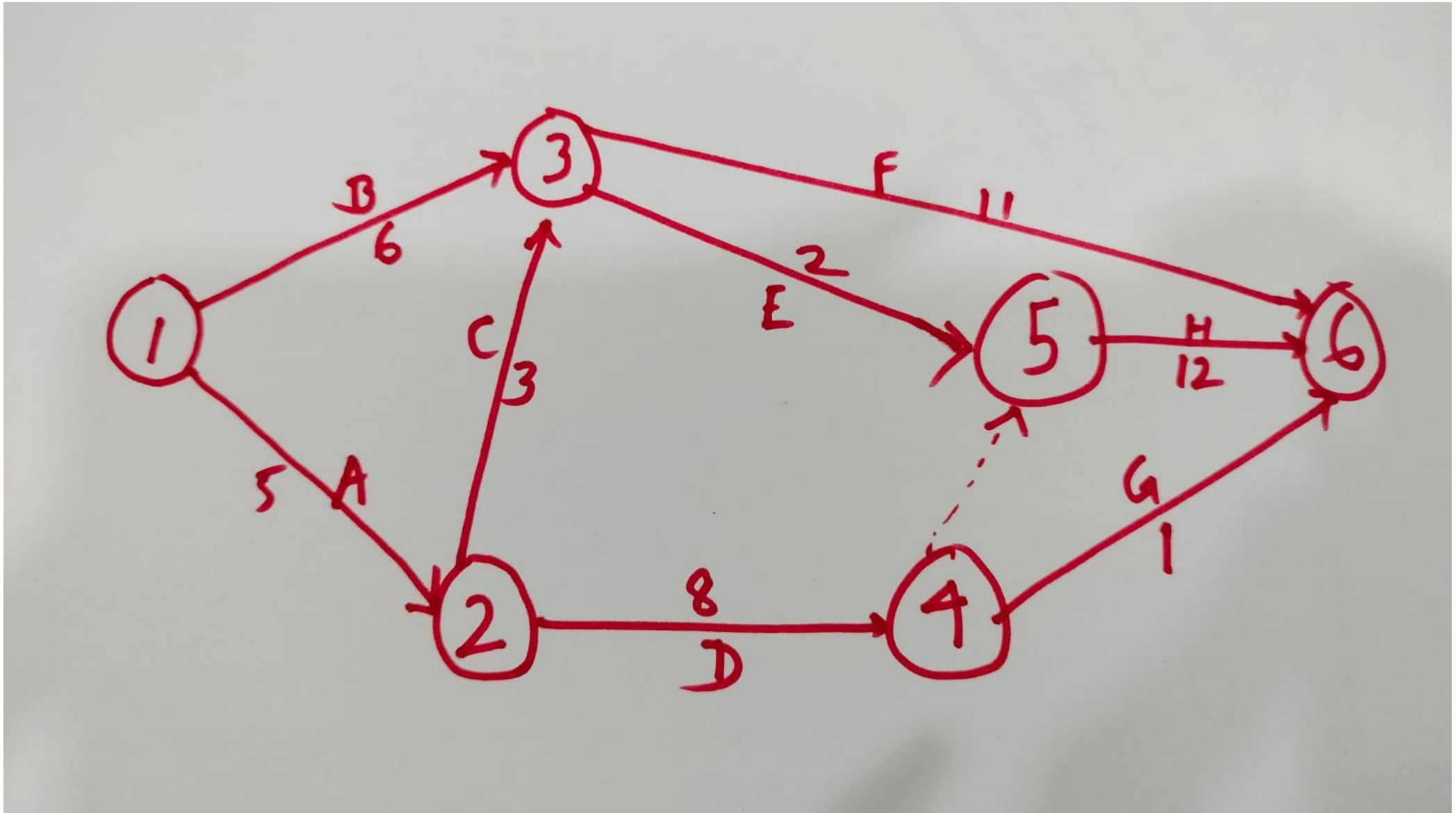
Backward pass (latest occurrence times, Δ)

- The backward pass computations start at node *n* and ends at node 1.
- **Initial Step.**
  - Set $\Delta_n = \square_n$ to indicate that latest occurrences of the last node equals the duration of the project.
- **General Step *j*.**
  - Given that nodes *p*, *q*, . . . , and v are linked *directly* to node *j* by *outgoing* activities (*j*, *p*), (*j*, *q*), . . . , and (*j*, v) and that the latest occurrence times of nodes *p*, *q*, . . . , and v have already been computed, the latest occurrence time of node *j* is *c*.
  - $\Delta_j = \min\{\Delta_p - D_{jp}, \Delta_q - D_{jq}, \ldots, \Delta_v - D_{jv}\}$
  - The backward pass ends with $\Delta_1 = 0$ at node 1.

Critical and Non-critical activity

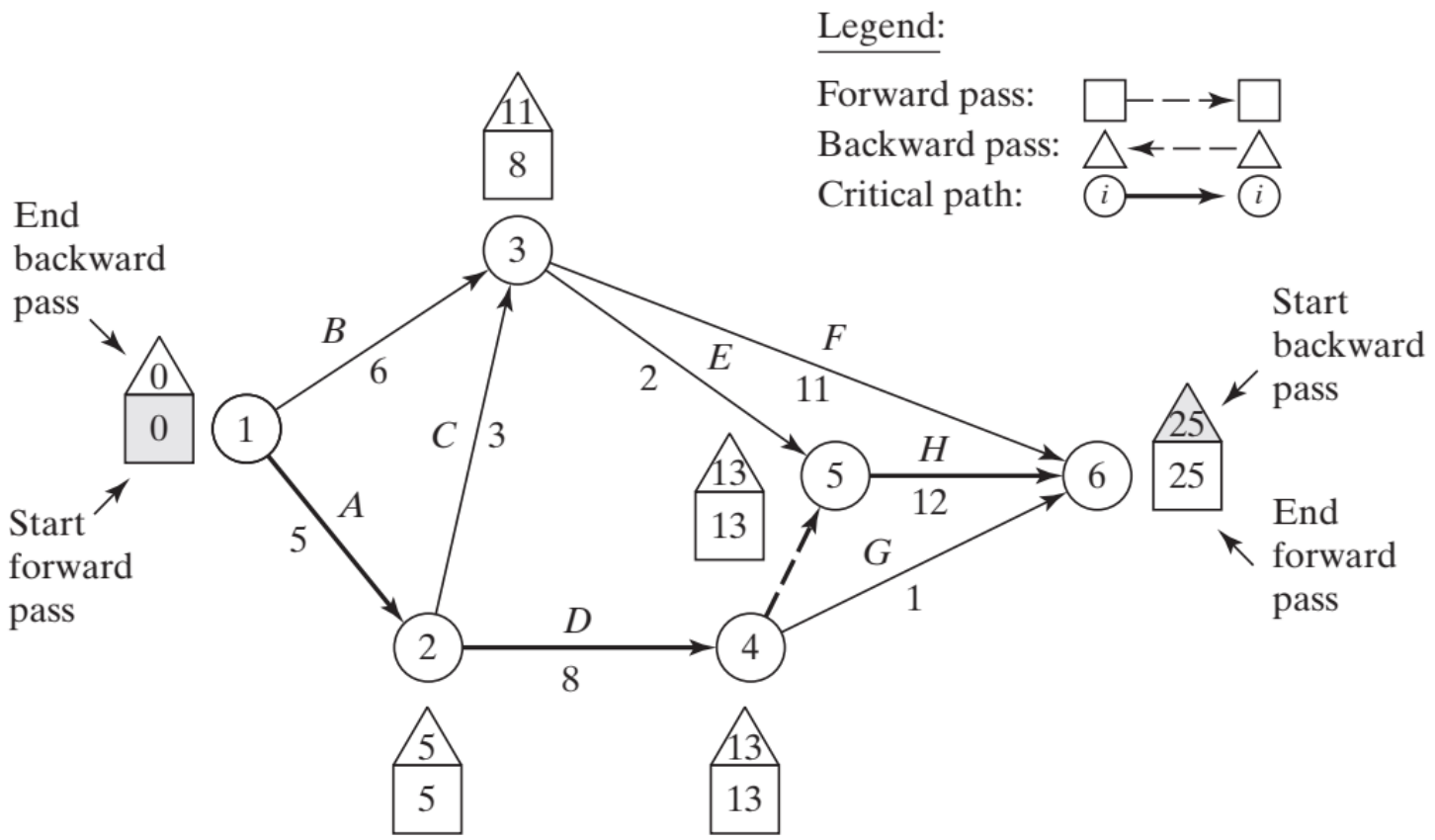
- Based on the preceding calculations, an activity (i, j) will be *critical* if it satisfies three conditions.
  - **1.** $\Delta_i = \square_i$
  - **2.** $\Delta_j = \square_j$
  - **3.** $\Delta_j - \square_i = D_{ij}$
- The three conditions state that the earliest and latest occurrence times of end nodes i and j are equal, and the duration D_{ij} fits "snugly" in the specified time span.
- An activity that does not satisfy all three conditions is *noncritical.*
- By definition, the critical activities of a network constitute the longest path spanning the project network from start to finish.

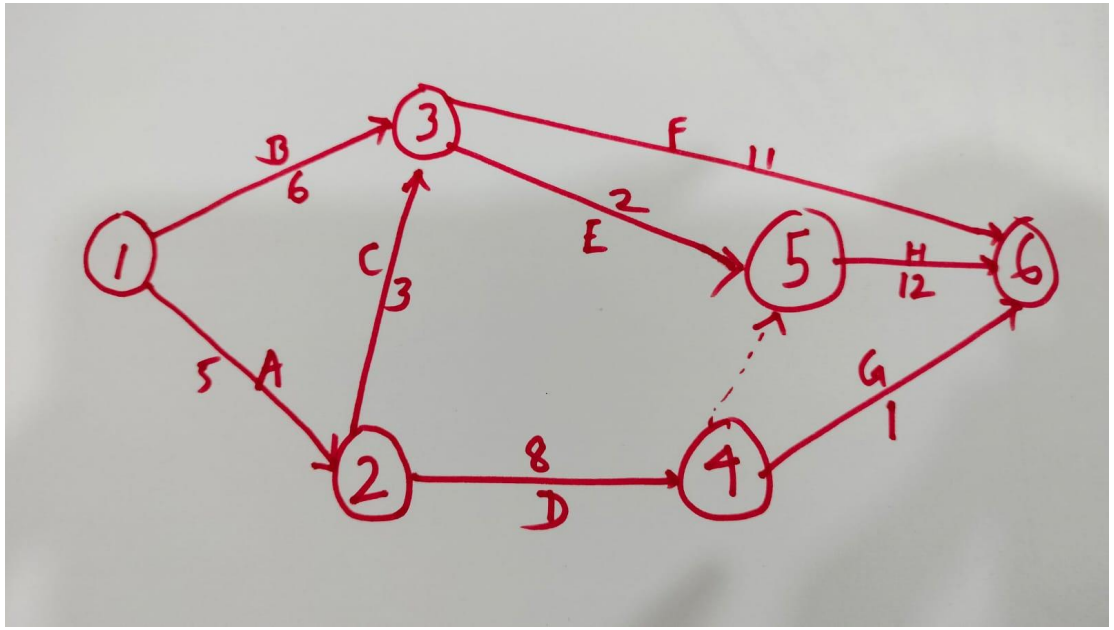
Example



Example

Determine the critical path for the project network in Figure. All the durations are in days.





The computations start at node 1 and advance recursively to node *n.*

Initial Step.

Set $\square_1 = 0$ to indicate that the project starts at time 0.

General Step *j*.

Given that nodes *p*, *q*, . . . , and v are linked *directly* to node *j* by incoming activities (*p*, *j*), (*q*, *j*), . . . , and (v, *j*) and that the earliest occurrence times of events (nodes) *p*, *q*, . . . , and v have already been computed, then the earliest occurrence time of event *j* is computed as

$\square_j = \max\{\square_p + D_{pj}, \square_q + D_{qj}, \ldots, \square_v + D_{vj}\}$

The forward pass is complete when $\square_n$ at node *n* has been computed.

By definition, $\square_j$ is the longest path (duration) to node *j*

Forward Pass

Node 1. Set $\square_1 = 0$

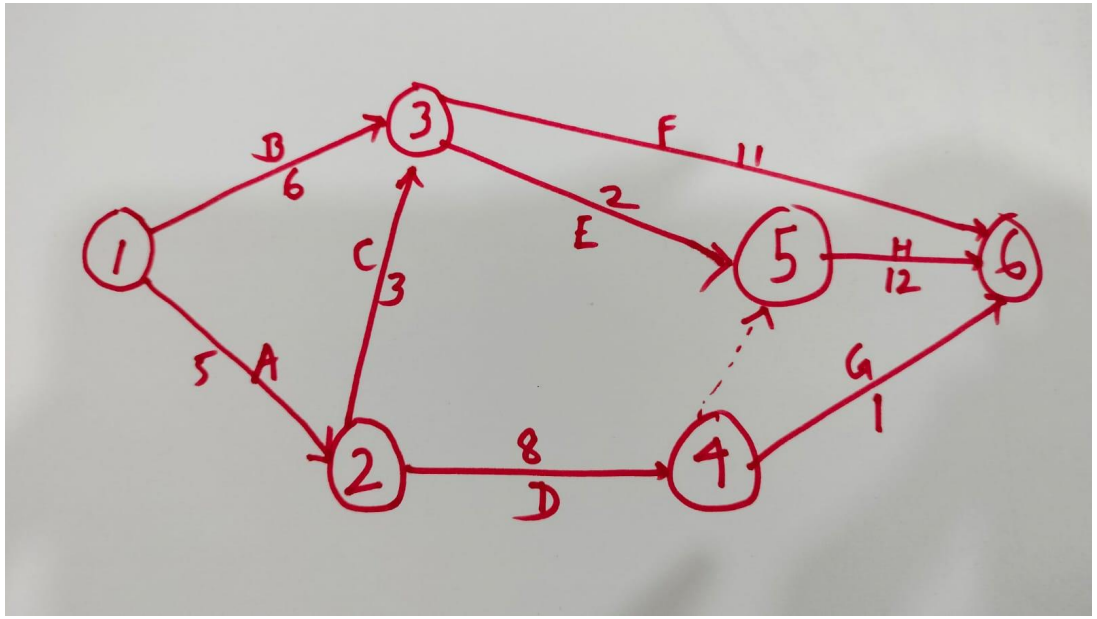
Node 2. $\square_2 = \square_1 + D_{12} = 0 + 5 = 5$

Node 3. $\square_3 = \max\{\square_1 + D_{13}, \square_2 + D_{23}\} = \max\{0 + 6, 5 + 3\} = 8$

Node 4. $\square_4 = \square_2 + D_{24} = 5 + 8 = 13$

Node 5. $\square_5 = \max\{\square_3 + D_{35}, \square_4 + D_{45}\} = \max\{8 + 2, 13 + 0\} = 13$

Node 6. $\square_6 = \max\{\square_3 + D_{36}, \square_4 + D_{46}, \square_5 + D_{56}\}$
$= \max\{8 + 11, 13 + 1, 13 + 12\} = 25$



The backward pass computations start at node *n* and ends at node 1.

Initial Step.

Set $\Delta_n = \square_n$ to indicate that latest occurrences of the last node equals the duration of the project.

General Step *j*.

Given that nodes *p*, *q*, . . . , and v are linked *directly* to node *j* by *outgoing* activities (*j*, *p*), (*j*, *q*), . . . , and (*j*, v) and that the latest occurrence times of nodes *p*, *q*, . . . , and v have already been computed, the latest occurrence time of node *j* is

$\Delta_j = \min\{\Delta_p - D_{jp}, \Delta_q - D_{jq}, \ldots, \Delta_v - D_{jv}\}$

The backward pass ends with $\Delta_1 = 0$ at node 1.

Backward Pass

Node 6. Set $\Delta_6 = \square_6 = 25$

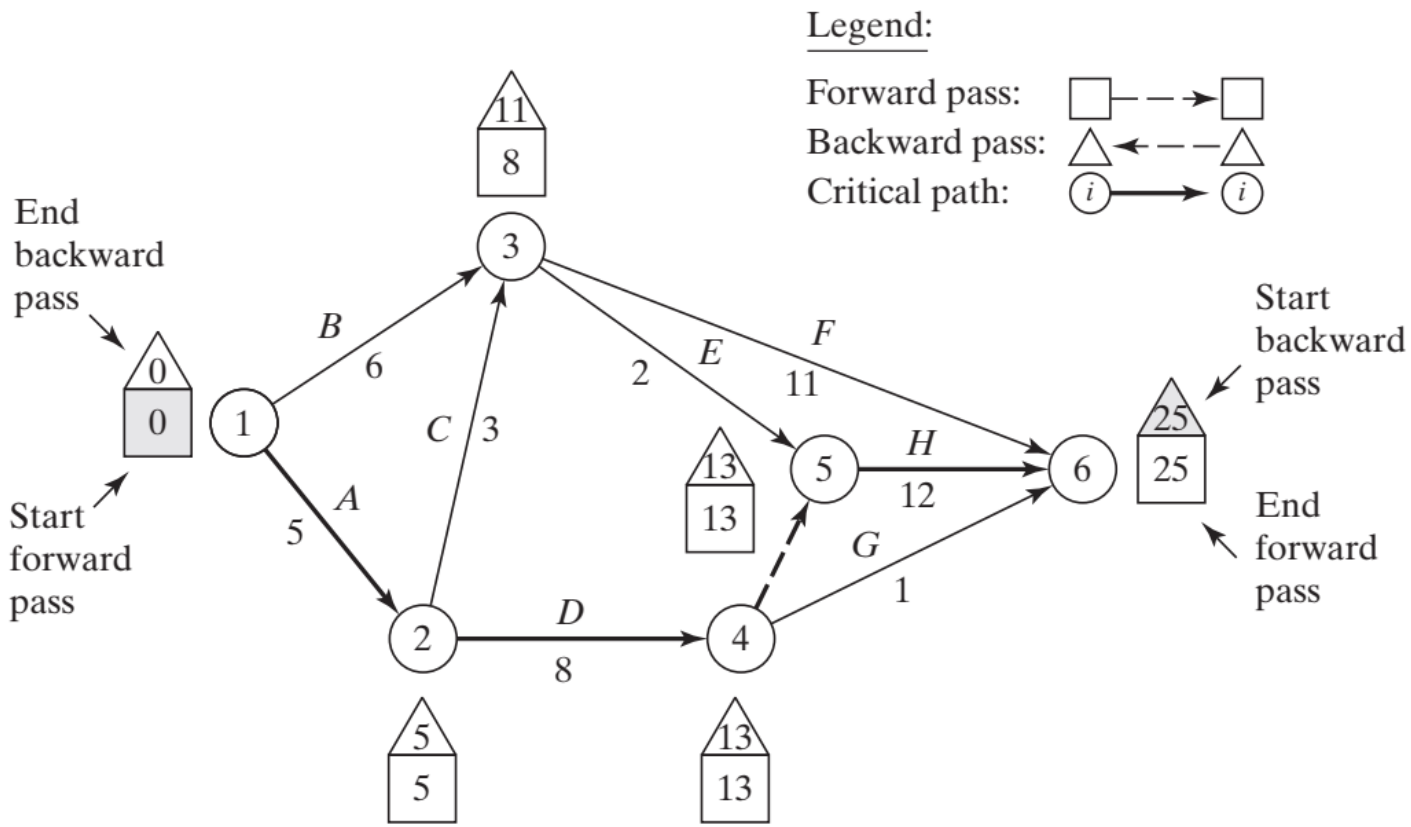
Node 5. $\Delta_5 = \Delta_6 - D_{56} = 25 - 12 = 13$

Node 4. $\Delta_4 = \min\{\Delta_6 - D_{46}, \Delta_5 - D_{45}\} = \min\{25 - 1, 13 - 0\} = 13$

Node 3. $\Delta_3 = \min\{\Delta_6 - D_{36}, \Delta_5 - D_{35}\} = \min\{25 - 11, 13 - 2\} = 11$

Node 2. $\Delta_2 = \min\{\Delta_4 - D_{24}, \Delta_3 - D_{23}\} = \min\{13 - 8, 11 - 3\} = 5$

Node 1. $\Delta_1 = \min\{\Delta_3 - D_{13}, \Delta_2 - D_2\} = \min\{11 - 6, 5 - 5\} = 0$


Legend:
Forward pass:
Backward pass:
Critical path:
i
i
End backward pass
Start forward pass
Start backward pass
End forward pass
0
0
11
8
3
13
13
25
25
5
5
13
13
1
2
4
5
6
A
5
B
6
C
3
D
8
E
2
F
11
G
1
H
12

Observation

- **1.** $\Delta_i = \square_i$
- **2.** $\Delta_j = \square_j$
- **3.** $\Delta_j - \square_i = D_{ij}$

- Correct computations will always end with $\Delta_1 = 0$.
- The computations can be made directly on the network as shown in Figure
- As expected, the critical path 1 ->2 -> 4 -> 5 -> 6 spans the network from start (node 1) to finish (node 6).
- The sum of the durations of the critical activities [(1, 2), (2, 4), (4, 5), and (5, 6)] equals the duration of the project (= 25 days)
- Observe that activity (4, 6) satisfies the first two conditions for a critical activity ($\Delta_4 = \square_4 = 13$ and $\Delta_6 = \square_6 = 25$) but not the third ($\Delta_6 - \square_4 \neq D_{46}$).
- Hence, the activity is noncritical.

PERT Networks

- PERT differs from CPM in that it assumes probabilistic duration times based on three estimates:
  - **1. Optimistic time**, *a*, which occurs when execution goes extremely well.
  - **2. Most likely time**, *m*, which occurs when execution is done under normal conditions.
  - **3. Pessimistic time**, *b*, which occurs when execution goes extremely poorly

- The most likely time, *m*, falls in the range (*a*, *b*).
- Based on the estimates, the average duration time, *D*, and variance, v, are approximated as

$$\overline{D} = \frac{a + 4m + b}{6}$$

$$v = \left(\frac{b - a}{6}\right)^2$$

PERT Networks: Working Principle

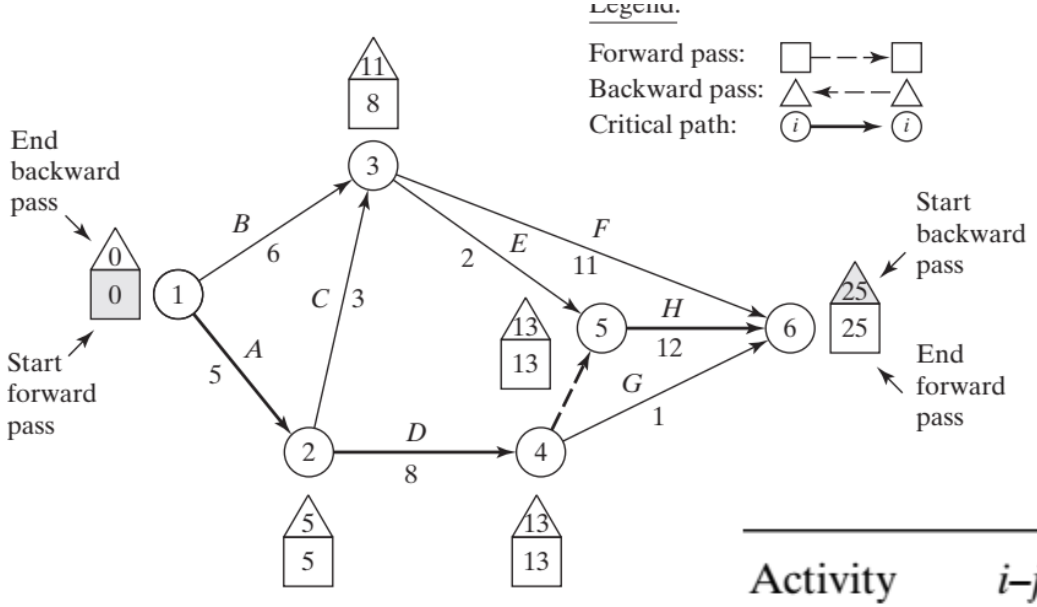
- The earliest occurrence time of node j is given as e_j
- The probability that *j* will occur by a scheduled time, S_j, can be estimated in the following manner:
- Assume that all the activities in the network are statistically independent
- First compute the mean, *E*{*ej*}, and variance, var{*ej*}.
- If there is only one path from the start node to node *j*, then the mean is the sum of expected durations, *D*, for all the activities along this path and the variance is the sum of the variances, v, of the same activities.
- If more than one path leads to node *j*, then it is necessary to determine the statistical distribution of the duration of the longest path,
- A simplifying assumption calls for selecting the path to node *j* having the longest *average* duration.
- If two or more paths have the same mean, the one with the largest variance is selected because it reflects the most uncertainty and, hence, leads to more conservative estimate of probabilities

PERT Networks: Working Principle

- Given the mean and variance of the path to node *j*, *E*{*ej*} and var{*ej*},
- The probability that node *j* occurs by time *Sj* is approximated by the standard normal distribution, *z* that is,

$$P\{e_j \leq S_j\} = P\left\{\frac{e_j - E\{e_j\}}{\sqrt{\text{var}\{e_j\}}} \leq \frac{S_j - E\{e_j\}}{\sqrt{\text{var}\{e_j\}}}\right\} = P\{z \leq K_j\}$$

Example



Activity	i–j	(a, m, b)	Activity	i–j	(a, m, b)
A	1–2	$(3, 5, 7)$	E	3–5	$(1, 2, 3)$
B	1–3	$(4, 6, 8)$	F	3–6	$(9, 11, 13)$
C	2–3	$(1, 3, 5)$	G	4–6	$(1, 1, 1)$
D	2–4	$(5, 8, 11)$	H	5–6	$(10, 12, 14)$

Activity	i–j	(a, m, b)	Activity	i–j	(a, m, b)
A	1–2	(3, 5, 7)	E	3–5	(1, 2, 3)
B	1–3	(4, 6, 8)	F	3–6	(9, 11, 13)
C	2–3	(1, 3, 5)	G	4–6	(1, 1, 1)
D	2–4	(5, 8, 11)	H	5–6	(10, 12, 14)

$$\overline{D} = \frac{a + 4m + b}{6}$$

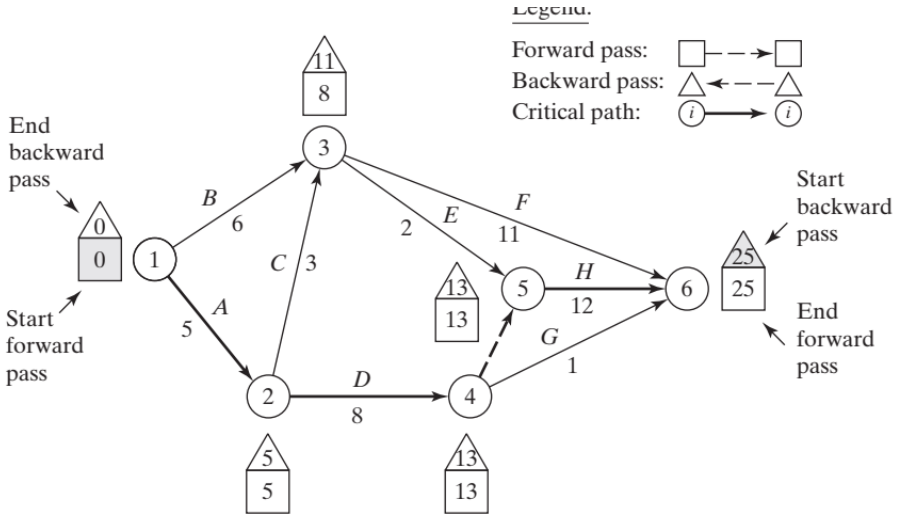
$$v = \left(\frac{b - a}{6}\right)^2$$

- The mean D_{ij} and variance v_{ij} for the different activities are given in the following table.
- Note that a dummy activity with $(a, m, b) = (0, 0, 0)$ has zero mean and variance.

Activity	i–j	$\overline{D_{ij}}$	v_{ij}	Activity	i–j	$\overline{D_{ij}}$	v_{ij}
A	1–2	5	.444	E	3–5	2	.111
B	1–3	6	.444	F	3–6	11	.444
C	2–3	3	.444	G	4–6	1	.000
D	2–4	8	1.000	H	5–6	12	.444

- The following table gives the longest path from node 1 to the different nodes, together with their associated mean and standard deviation.

Node	Longest path based on mean durations	Path mean	Path standard deviation
2	1–2	5.00	0.67
3	1–2–3	8.00	0.94
4	1–2–4	13.00	1.20
5	1–2–4–5	13.00	1.20
6	1–2–4–5–6	25.00	1.37



Activity	i–j	$\overline{D_{ij}}$	v_{ij}	Activity	i–j	$\overline{D_{ij}}$	v_{ij}
A	1–2	5	.444	E	3–5	2	.111
B	1–3	6	.444	F	3–6	11	.444
C	2–3	3	.444	G	4–6	1	.000
D	2–4	8	1.000	H	5–6	12	.444

The following table computes the probability that each node is realized by time S_j

Node	Longest path based on mean durations	Path mean	Path standard deviation
2	1–2	5.00	0.67
3	1–2–3	8.00	0.94
4	1–2–4	13.00	1.20
5	1–2–4–5	13.00	1.20
6	1–2–4–5–6	25.00	1.37

Node j	Longest path	Path mean	Path standard deviation	S_j	K_j	$P\{z \le K_j\}$
2	1–2	5.00	0.67	5.00	0	.5000
3	1–2–3	8.00	0.94	11.00	3.19	.9993
4	1–2–4	13.00	1.20	12.00	−.83	.2033
5	1–2–4–5	13.00	1.20	14.00	.83	.7967
6	1–2–4–5–6	25.00	1.37	26.00	.73	.7673

$$P\{e_j \le S_j\} = P\left\{\frac{e_j - E\{e_j\}}{\sqrt{\text{var}\{e_j\}}} \le \frac{S_j - E\{e_j\}}{\sqrt{\text{var}\{e_j\}}}\right\} = P\{z \le K_j\}$$

Standard Normal Distribution Table

z	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
+0	.50000	.50399	.50798	.51197	.51595	.51994	.52392	.52790	.53188	.53586
+0.1	.53983	.54380	.54776	.55172	.55567	.55966	.56360	.56749	.57142	.57535
+0.2	.57926	.58317	.58706	.59095	.59483	.59871	.60257	.60642	.61026	.61409
+0.3	.61791	.62172	.62552	.62930	.63307	.63683	.64058	.64431	.64803	.65173
+0.4	.65542	.65910	.66276	.66640	.67003	.67364	.67724	.68082	.68439	.68793
+0.5	.69146	.69497	.69847	.70194	.70540	.70884	.71226	.71566	.71904	.72240
+0.6	.72575	.72907	.73237	.73565	.73891	.74215	.74537	.74857	.75175	.75490
+0.7	.75804	.76115	.76424	.76730	.77035	.77337	.77637	.77935	.78230	.78524
+0.8	.78814	.79103	.79389	.79673	.79955	.80234	.80511	.80785	.81057	.81327
+0.9	.81594	.81859	.82121	.82381	.82639	.82894	.83147	.83398	.83646	.83891
+1	.84134	.84375	.84614	.84849	.85083	.85314	.85543	.85769	.85993	.86214
+1.1	.86433	.86650	.86864	.87076	.87286	.87493	.87698	.87900	.88100	.88298
+1.2	.88493	.88686	.88877	.89065	.89251	.89435	.89617	.89796	.89973	.90147
+1.3	.90320	.90490	.90658	.90824	.90988	.91149	.91308	.91466	.91621	.91774
+1.4	.91924	.92073	.92220	.92364	.92507	.92647	.92785	.92922	.93056	.93189
+1.5	.93319	.93448	.93574	.93699	.93822	.93943	.94062	.94179	.94295	.94408
+1.6	.94520	.94630	.94738	.94845	.94950	.95053	.95154	.95254	.95352	.95449
+1.7	.95543	.95637	.95728	.95818	.95907	.95994	.96080	.96164	.96246	.96327
+1.8	.96407	.96485	.96562	.96638	.96712	.96784	.96856	.96926	.96995	.97062
+1.9	.97128	.97193	.97257	.97320	.97381	.97441	.97500	.97558	.97615	.97670
+2	.97725	.97778	.97831	.97882	.97932	.97982	.98030	.98077	.98124	.98169
+2.1	.98214	.98257	.98300	.98341	.98382	.98422	.98461	.98500	.98537	.98574
+2.2	.98610	.98645	.98679	.98713	.98745	.98778	.98809	.98840	.98870	.98899
+2.3	.98928	.98956	.98983	.99010	.99036	.99061	.99086	.99111	.99134	.99158
+2.4	.99180	.99202	.99224	.99245	.99266	.99286	.99305	.99324	.99343	.99361
+2.5	.99379	.99396	.99413	.99430	.99446	.99461	.99477	.99492	.99506	.99520
+2.6	.99534	.99547	.99560	.99573	.99585	.99598	.99609	.99621	.99632	.99643
+2.7	.99653	.99664	.99674	.99683	.99693	.99702	.99711	.99720	.99728	.99736
+2.8	.99744	.99752	.99760	.99767	.99774	.99781	.99788	.99795	.99801	.99807
+2.9	.99813	.99819	.99825	.99831	.99836	.99841	.99846	.99851	.99856	.99861
+3	.99865	.99869	.99874	.99878	.99882	.99886	.99889	.99893	.99896	.99900
+3.1	.99903	.99906	.99910	.99913	.99916	.99918	.99921	.99924	.99926	.99929
+3.2	.99931	.99934	.99936	.99938	.99940	.99942	.99944	.99946	.99948	.99950