



SAPIENZA
UNIVERSITÀ DI ROMA

Network Optimization

Volume 5 — Project management

Project management with the critical path method: project networks, earliest and latest times, slacks and critical paths.

Fabio Furini

fabiofurini.github.io

Department of Computer, Control and Management Engineering — Sapienza University of Rome

Online version and downloadable material:
fabiofurini.github.io/network-optimization/
github.com/fabiofurini/network-optimization

Academic year 2026–2027

Contents

9	Project management	2
9.1	Projects and project networks	3
9.2	Paths and critical paths	7
9.3	The forward pass: earliest times	8
9.4	The backward pass: latest times	12
9.5	Slacks and critical activities	15
9.6	The critical path method	19
9.7	The linear programming model	21
9.8	Two critical paths and changes of durations	23
9.9	Typical mistakes	26
9.10	In summary	26



Project management

Prerequisites

- From [Chapter 1 «Graphs»](#): directed graphs, directed paths and cycles.
- From [Chapter 3 «Shortest paths»](#): the topological order of an acyclic graph, the algorithm for acyclic graphs and longest paths, potentials and reduced costs, the LP model of a shortest path and its dual.
- From the module Linear Programming, [Chapter 9 «Duality theory»](#): the dual of an LP model and the weak duality theorem.

Objectives

At the end of this chapter you will be able to:

- describe a project by its activities, durations and precedences, and draw its project network with the START and FINISH nodes;
- compute the earliest and the latest start and finish times with the forward and the backward pass, and prove that the minimum project duration is the length of a critical path;
- compute the total and the free slacks, and find the critical activities and all the critical paths;
- draw the Gantt chart of a schedule;
- write the LP model of the project duration and its dual, a longest path model;
- predict the effect of a change of one duration on the project duration.

9.1 Projects and project networks

Definition 9.1: project

A **project** is a set of n **activities**, numbered $1, 2, \dots, n$, with the following data.

- Every activity j has a **duration** $d_j \in \mathbb{Q}_{>0}$.
- Some pairs (i, j) of activities are **precedences**: activity i is an **immediate predecessor** of activity j , and j can start only when i is finished. An activity with several immediate predecessors can start only when all of them are finished.

- To describe a project we need three types of information: the **activities**, at the desired level of detail; the **precedences**, that is, the immediate predecessors of each activity; the **durations**, that is, an estimate of the time each activity takes.
- In the whole chapter we make two assumptions:
 - (H1) the durations are **deterministic**: every activity takes exactly its estimated duration d_j , without interruptions;
 - (H2) the resources are **unlimited**: activities that are not linked by precedences can be carried out at the same time.
- The method of this chapter is the **critical path method** (CPM). It was developed in the late 1950s, independently of the program evaluation and review technique (PERT), and the two names are often combined into PERT/CPM.

Definition 9.2: project network

The **project network** of a project with n activities is the directed graph $G = (N, A)$ with $N = \{0, 1, \dots, n+1\}$ and the following nodes and arcs.

- Node $j \in \{1, 2, \dots, n\}$ is activity j . Node 0 is the **START** node and node $n+1$ is the **FINISH** node, with durations $d_0 = d_{n+1} = 0$.
- A contains the arc (i, j) for every precedence, the arc $(0, j)$ for every activity j without immediate predecessors, and the arc $(j, n+1)$ for every activity j that is not an immediate predecessor of any activity.

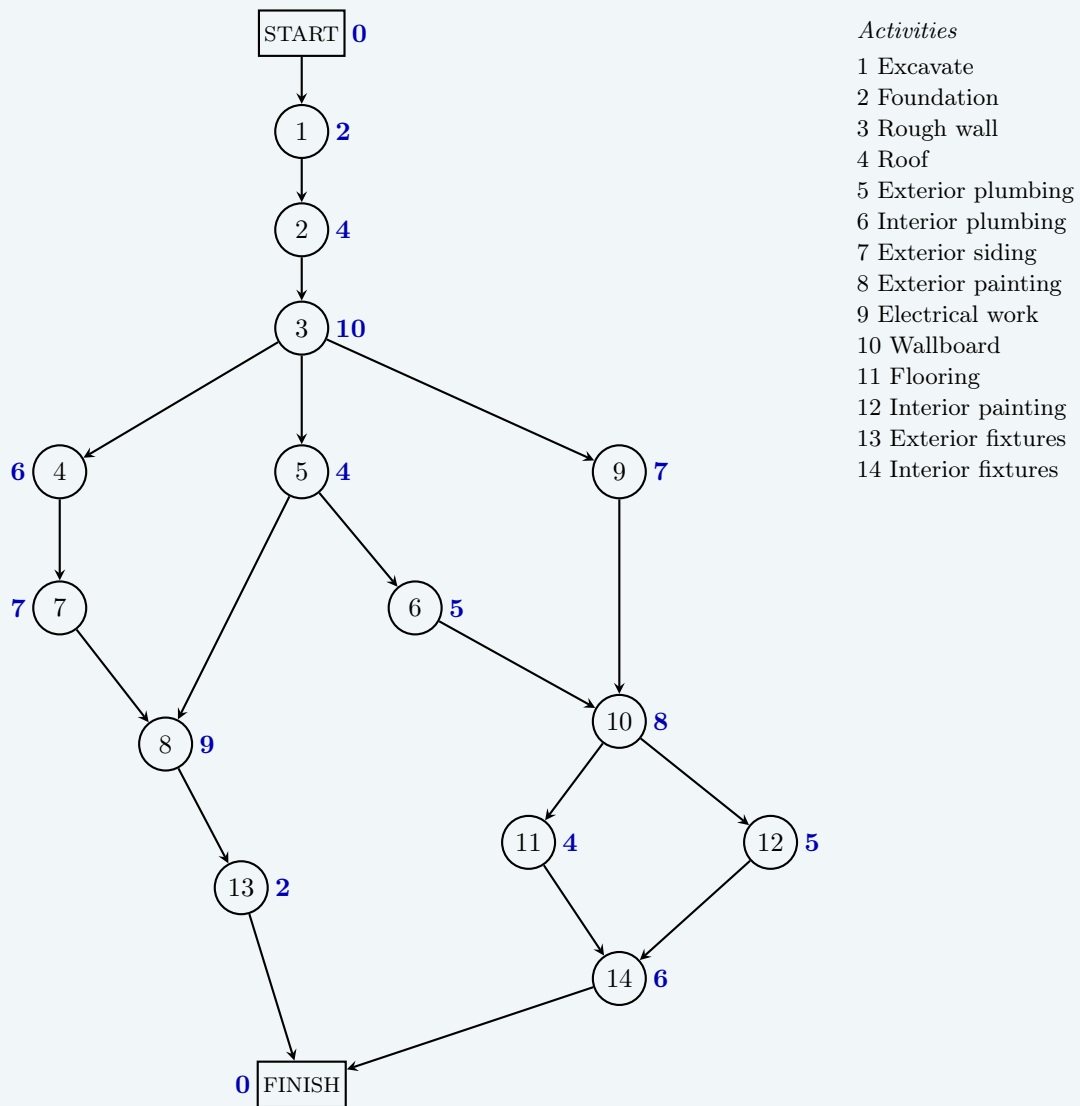
- This is the **activity-on-node** (AON) network: the activities are the nodes, and the arcs only show the precedences. In the **activity-on-arc** (AOA) network each activity is an arc, and a node separates an activity from its immediate predecessors.
- The AON network is easier to build, easier to read for users without experience, and easier to revise when the project changes: we always use it, and call it simply the project network.
- In practice, the project network is built by project management software from the list of activities.

Example 9.1: the case study

A construction company has to schedule the crews that carry out the 14 activities of a building project. The table gives, for every activity, its immediate predecessors and its estimated duration.

activity j	description	immediate predecessors	d_j
1	Excavate	—	2 weeks
2	Lay the foundation	1	4 weeks
3	Put up the rough wall	2	10 weeks
4	Put up the roof	3	6 weeks
5	Install the exterior plumbing	3	4 weeks
6	Install the interior plumbing	5	5 weeks
7	Put up the exterior siding	4	7 weeks
8	Do the exterior painting	5, 7	9 weeks
9	Do the electrical work	3	7 weeks
10	Put up the wallboard	6, 9	8 weeks
11	Install the flooring	10	4 weeks
12	Do the interior painting	10	5 weeks
13	Install the exterior fixtures	8	2 weeks
14	Install the interior fixtures	11, 12	6 weeks

Adding up the durations gives a total of 79 weeks. Several activities can be carried out at the same time, which can reduce the project duration substantially. The project network has the 16 nodes $0, 1, \dots, 15$ and 19 arcs; next to each node, in blue, its duration.



Activity 1 is the only activity without immediate predecessors, so START has the single arc (0, 1); activities 13 and 14 are not immediate predecessors of any activity, so they enter FINISH.

9.1.1 Schedules and acyclic networks

Definition 9.3: schedule

A **schedule** of a project is a vector $\mathbf{t} = (t_0, t_1, \dots, t_{n+1})$ of **start times** with $t_0 = 0$ and

$$t_j \geq t_i + d_i, \quad \forall (i, j) \in A.$$

Node j starts at t_j and finishes at $t_j + d_j$. The **duration** of the schedule is t_{n+1} , and the **minimum project duration** T is the smallest duration of a schedule.

- The constraint of the arc (i, j) says that j starts after i is finished. By (H1) and (H2), the precedences are the only constraints of a schedule.
- The arcs $(j, n+1)$ make FINISH start after every activity is finished: t_{n+1} is the completion time of the project.

Proposition 9.4: a schedule needs an acyclic network

A project has a schedule if and only if its project network is acyclic.

Proof. No arc enters node 0 and no arc leaves node $n+1$, so every directed cycle of G is made of activities.

1. Let $i_0, i_1, \dots, i_k = i_0$ be a directed cycle and $\mathbf{t}$ a schedule. Adding up the constraints of its arcs, the start times telescope:

$$0 = t_{i_k} - t_{i_0} \geq \sum_{s=0}^{k-1} d_{i_s} > 0,$$

a contradiction, since the durations of the activities are positive.

2. Let G be acyclic, and let $v_0, v_1, \dots, v_{n+1}$ be a topological order ([Chapter 3, Proposition «topological orders»](#)) with $v_0 = 0$: node 0 has no entering arc and can be removed first. Let D be the largest duration and $t_{v_a} = a D$ for every a .
3. Then $t_0 = 0$, and every arc (v_a, v_b) goes forward, $a < b$, so $t_{v_b} - t_{v_a} = (b - a) D \geq D \geq d_{v_a}$: $\mathbf{t}$ is a schedule. $\square$

- From now on the project network is acyclic. Then every activity lies on a path from START to FINISH: going backwards along the arcs we reach a node without immediate predecessors, which is linked to START, and going forwards we reach a node without successors, which is linked to FINISH.
- A cycle in the precedences is an error of the plan: an activity would have to finish before it starts.

A project is a set of activities with durations and precedences. Its project network has a node for every activity, plus START and FINISH, and an arc for every precedence; a schedule exists if and only if the network is acyclic.

9.2 Paths and critical paths

- Under (H1) and (H2), the critical path method answers the following questions.
 1. What is the minimum time required to complete the project?
 2. When can each activity start and finish at the earliest?
 3. When must each activity start and finish at the latest, so as not to delay the completion of the project?
 4. Which activities are critical, so that any delay on them delays the project?
 5. For the other activities, how much delay can be tolerated?

Definition 9.5: length of a path and critical path

Every arc $(i, j) \in A$ has the **length** d_i , the duration of its tail. The **length** of a directed path $P = i_0, i_1, \dots, i_k$ is

$$L(P) = \sum_{s=0}^{k-1} d_{i_s},$$

the total duration of its nodes except the last one. A **critical path** is a path from START to FINISH of maximum length. An activity is **critical** if it lies on a critical path.

- On a path from START to FINISH the durations of the first and the last node are zero: $L(P)$ is the total duration of the activities of the path.
- A project can have several critical paths: when several paths tie for the maximum length, they are all critical.

Lemma 9.6: every path bounds the duration

For every schedule $\mathbf{t}$ and every path P from START to FINISH, $t_{n+1} \geq L(P)$.

Proof. Let $P = i_0, i_1, \dots, i_k$ with $i_0 = 0$ and $i_k = n + 1$. Adding up the constraints $t_{i_{s+1}} \geq t_{i_s} + d_{i_s}$ of the arcs of P , the start times telescope:

$$t_{n+1} - t_0 \geq \sum_{s=0}^{k-1} d_{i_s} = L(P),$$

and $t_0 = 0$. □

- The activities of a path must be done one after the other, without overlap: the project cannot be shorter than any path.

Example 9.2: the paths of the case study

The network of [Example 9.1](#) has six paths from START to FINISH; their lengths, in weeks, are:

path from START to FINISH	length (weeks)
START, 1, 2, 3, 4, 7, 8, 13, FINISH	$2 + 4 + 10 + 6 + 7 + 9 + 2 = 40$
START, 1, 2, 3, 5, 6, 10, 11, 14, FINISH	$2 + 4 + 10 + 4 + 5 + 8 + 4 + 6 = 43$
START, 1, 2, 3, 5, 6, 10, 12, 14, FINISH	$2 + 4 + 10 + 4 + 5 + 8 + 5 + 6 = 44$
START, 1, 2, 3, 5, 8, 13, FINISH	$2 + 4 + 10 + 4 + 9 + 2 = 31$
START, 1, 2, 3, 9, 10, 11, 14, FINISH	$2 + 4 + 10 + 7 + 8 + 4 + 6 = 41$
START, 1, 2, 3, 9, 10, 12, 14, FINISH	$2 + 4 + 10 + 7 + 8 + 5 + 6 = 42$

- The longest path is START, 1, 2, 3, 5, 6, 10, 12, 14, FINISH, of length 44: by Lemma 9.6, every schedule lasts at least 44 weeks.
- The project can last more than a given path. On the path START, 1, 2, 3, 5, 8, 13, FINISH, of length 31, activity 8 waits for activity 5, which can finish at week $2 + 4 + 10 + 4 = 20$, but also for activity 7, which is not on the path and cannot finish before week $2 + 4 + 10 + 6 + 7 = 29$.

The number of paths

Listing the paths does not work on large projects: their number can grow exponentially with n . A sequence of k pairs of parallel activities, each pair after the previous one, has 2^k paths from START to FINISH. The critical path method finds the length of a longest path without listing the paths.

The length of a path from START to FINISH is the total duration of its activities. Every schedule lasts at least as long as every such path; the critical paths are the longest ones.

9.3 The forward pass: earliest times

- We first assume that no delay occurs: every activity takes its estimated duration and starts as soon as all its immediate predecessors are finished.

Definition 9.7: earliest start and finish times

Let $v_0, v_1, \dots, v_{n+1}$ be a topological order of the project network. The **earliest start time** ES_j and the **earliest finish time** EF_j of the nodes are computed in this order by the **forward pass**:

$$ES_0 = 0, \quad ES_j = \max_{(i,j) \in \delta^-(j)} EF_i, \quad \forall j \in N \setminus \{0\}, \quad EF_j = ES_j + d_j, \quad \forall j \in N.$$

- This is the **earliest start time rule**: the earliest start time of an activity is the largest earliest finish time of its immediate predecessors. When node j is processed, its predecessors come before it in the topological order, so their earliest finish times are known.
- As in Chapter 3, the topological order removes, at every step, the node without entering arcs of **smallest index**. Node 0 is removed first.

Theorem 9.8: earliest times

For every node $j \in N$, ES_j is the length of a longest path from START to j . Moreover, ES is a schedule, and every schedule t satisfies $t_j \geq ES_j$ for every $j \in N$.

Proof. We prove the first and the last claim by induction along the topological order.

1. *Node 0.* No arc enters START: the only path from START to 0 has no arcs and length $0 = ES_0$, and every schedule has $t_0 = 0 = ES_0$.
2. *Longest paths.* Let $j \neq 0$. Every path from START to j is a path from START to some i followed by an arc $(i, j) \in \delta^-(j)$ of length d_i , and i comes before j . By induction its maximum length is

$$\max_{(i,j) \in \delta^-(j)} (ES_i + d_i) = \max_{(i,j) \in \delta^-(j)} EF_i = ES_j.$$

3. *Lower bound.* For every arc $(i, j) \in \delta^-(j)$, by induction $t_j \geq t_i + d_i \geq ES_i + d_i = EF_i$; hence $t_j \geq ES_j$.
4. *ES is a schedule.* $ES_0 = 0$, and $ES_j \geq EF_i = ES_i + d_i$ for every arc $(i, j) \in A$, by the maximum in the definition. $\square$

Theorem 9.9: minimum project duration

Under (H1) and (H2), the minimum project duration is $T = ES_{n+1}$, the length of a critical path, and the schedule ES attains it.

Proof. By Theorem 9.8, ES is a schedule of duration ES_{n+1} , and every schedule has $t_{n+1} \geq ES_{n+1}$: hence $T = ES_{n+1}$. By the same theorem, ES_{n+1} is the length of a longest path from START to FINISH, that is, of a critical path. $\square$

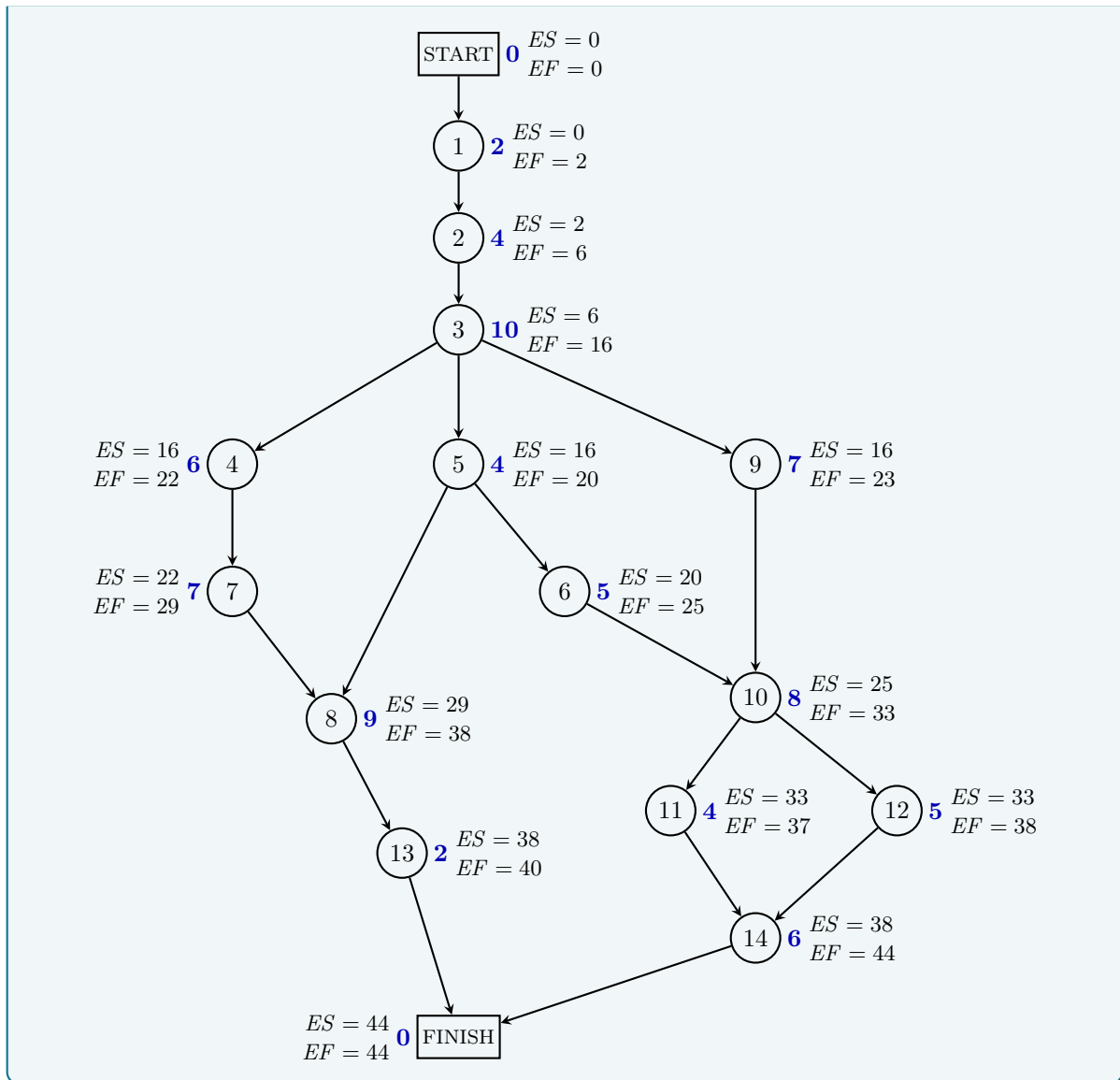
- Both assumptions are used: (H1) makes the lengths of the paths fixed numbers, and (H2) makes the precedences the only constraints of a schedule.
- The forward pass is the algorithm for acyclic graphs of Chapter 3 «Shortest paths», with the maximum in place of the minimum: with the arc lengths $-d_i$, that algorithm computes the labels $d(j) = -ES_j$. A longest path is easy to find because the network is acyclic.

Example 9.3: the forward pass on the case study

In the case study every activity has immediate predecessors of smaller index, so the topological order is $0, 1, 2, \dots, 15$. Each row processes one node: the earliest finish times of its immediate predecessors (the maximum in bold), then ES_j and EF_j .

node j	d_j	EF_i of the predecessors (i)	ES_j	EF_j
0	0	—	0	0
1	2	0 (0)	0	2
2	4	2 (1)	2	6
3	10	6 (2)	6	16
4	6	16 (3)	16	22
5	4	16 (3)	16	20
6	5	20 (5)	20	25
7	7	22 (4)	22	29
8	9	20 (5), 29 (7)	29	38
9	7	16 (3)	16	23
10	8	25 (6), 23 (9)	25	33
11	4	33 (10)	33	37
12	5	33 (10)	33	38
13	2	38 (8)	38	40
14	6	37 (11), 38 (12)	38	44
15	0	40 (13), 44 (14)	44	44

The FINISH node gives $T = ES_{15} = 44$: if everything stays on schedule, the project is completed in 44 weeks. In the figure, next to each node, its duration (blue) and its earliest times.



The forward pass computes, in topological order, $ES_j = \max EF_i$ over the immediate predecessors and $EF_j = ES_j + d_j$. ES_j is the length of a longest path from START to j , and $T = ES_{n+1}$ is the length of a critical path.

9.4 The backward pass: latest times

- How much later than ES_j can an activity start without delaying the project? The latest times answer, assuming that no other delay occurs.

Definition 9.10: latest start and finish times

Let $T = ES_{n+1}$. The **latest finish time** LF_j and the **latest start time** LS_j of the nodes are computed in the reverse topological order by the **backward pass**:

$$LF_{n+1} = T, \quad LF_j = \min_{(j,k) \in \delta^+(j)} LS_k, \quad \forall j \in N \setminus \{n+1\}, \quad LS_j = LF_j - d_j, \quad \forall j \in N.$$

- This is the **latest finish time rule**: the latest finish time of an activity is the smallest latest start time of its immediate successors. The immediate successors cannot start before the activity is finished, so the activity must finish in time to let all of them start by their latest start times.

Theorem 9.11: latest times

For every node $j \in N$, $T - LS_j$ is the length of a longest path from j to FINISH. Moreover, LS is a schedule of duration T , and every schedule t with $t_{n+1} \leq T$ satisfies $t_j \leq LS_j$ for every $j \in N$.

Proof. We prove the first and the last claim by induction along the reverse topological order.

1. *Node $n+1$.* The only path from $n+1$ to FINISH has length $0 = T - LS_{n+1}$, and $t_{n+1} \leq T = LS_{n+1}$.
2. *Longest paths.* Let $j \neq n+1$. Every path from j to FINISH is an arc $(j, k) \in \delta^+(j)$, of length d_j , followed by a path from k to FINISH. By induction its maximum length is

$$d_j + \max_{(j,k) \in \delta^+(j)} (T - LS_k) = d_j + T - LF_j = T - LS_j.$$
3. *Upper bound.* For every arc $(j, k) \in \delta^+(j)$, by induction $t_j \leq t_k - d_j \leq LS_k - d_j$; hence $t_j \leq LF_j - d_j = LS_j$.
4. *LS is a schedule.* $LS_k \geq LF_j = LS_j + d_j$ for every arc $(j, k) \in A$, by the minimum in the definition. Moreover $LS_{n+1} = T$, and $LS_0 = T - T = 0$, since a longest path from START to FINISH has length T . $\square$

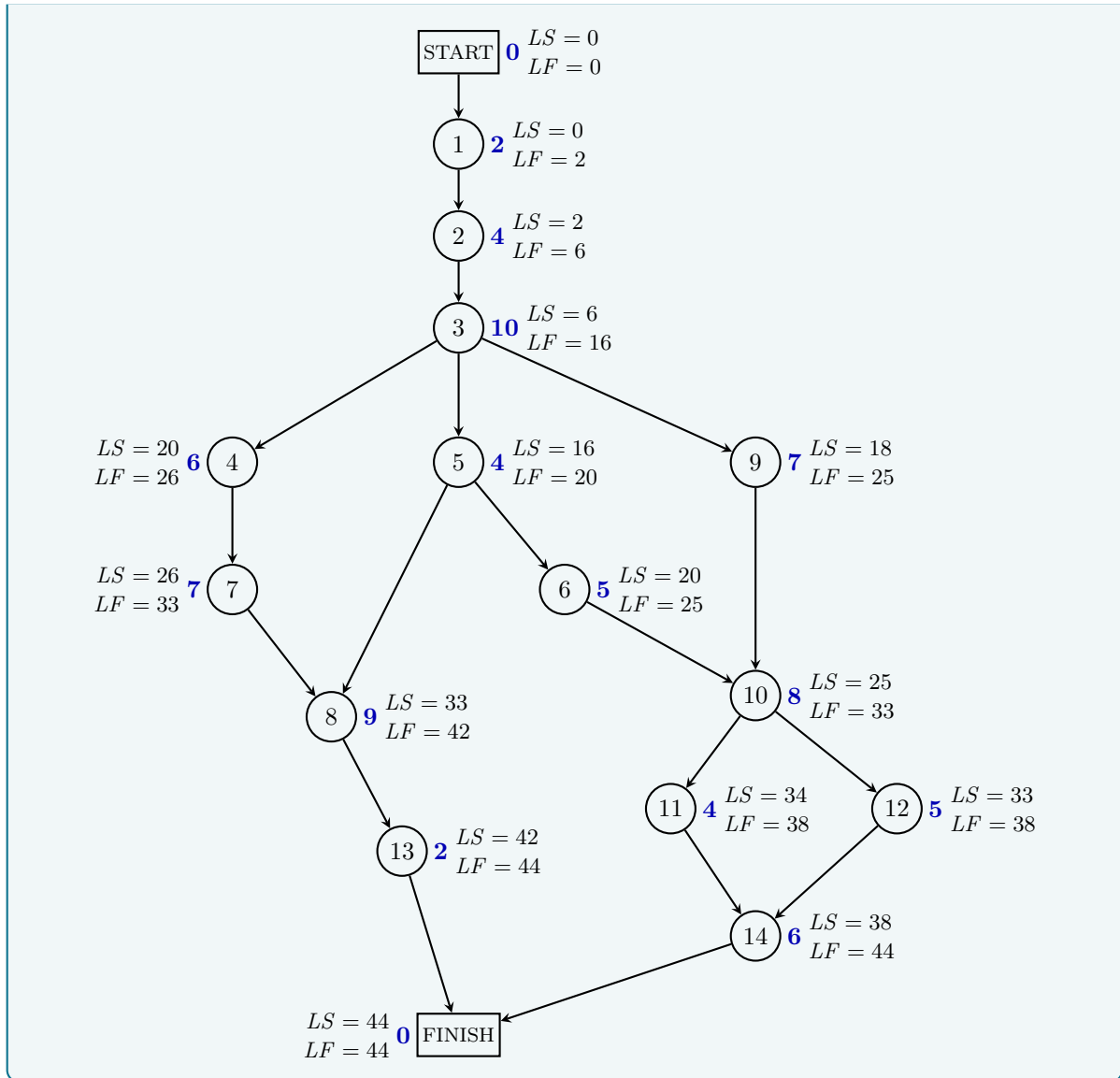
- In every schedule of duration T , node j starts between ES_j and LS_j , and both extremes are attained, by the schedules ES and LS .

Example 9.4: the backward pass on the case study

The backward pass processes the nodes in the order $15, 14, \dots, 0$. Each row gives the latest start times of the immediate successors (the minimum in bold), then LF_j and LS_j .

node j	d_j	LS_k of the successors (k)	LF_j	LS_j
15	0	—	44	44
14	6	44 (15)	44	38
13	2	44 (15)	44	42
12	5	38 (14)	38	33
11	4	38 (14)	38	34
10	8	34 (11), 33 (12)	33	25
9	7	25 (10)	25	18
8	9	42 (13)	42	33
7	7	33 (8)	33	26
6	5	25 (10)	25	20
5	4	20 (6), 33 (8)	20	16
4	6	26 (7)	26	20
3	10	20 (4), 16 (5), 18 (9)	16	6
2	4	6 (3)	6	2
1	2	2 (2)	2	0
0	0	0 (1)	0	0

The pass ends with $LS_0 = 0$, as [Theorem 9.11](#) predicts. For example, activity 9 must finish by $LF_9 = LS_{10} = 25$, so it must start by $LS_9 = 18$, two weeks after its earliest start time $ES_9 = 16$. In the figure, next to each node, its duration (blue) and its latest times.



The backward pass computes, in reverse topological order, $LF_j = \min LS_k$ over the immediate successors and $LS_j = LF_j - d_j$, starting from $LF_{n+1} = T$. $T - LS_j$ is the length of a longest path from j to FINISH, and in every schedule of duration T node j starts between ES_j and LS_j .

9.5 Slacks and critical activities

Definition 9.12: slacks and tight arcs

For every activity $j \in \{1, 2, \dots, n\}$:

- the **total slack** is $TS_j = LS_j - ES_j = LF_j - EF_j$;
- the **free slack** is $FS_j = \min_{(j,k) \in \delta^+(j)} ES_k - EF_j$.

An arc $(i, j) \in A$ is **tight** if $ES_j = EF_i$.

- The two differences in TS_j coincide, since both pairs of times differ by d_j .
- The total slack is the delay of activity j that does not delay the project; the free slack is the delay that does not move the earliest start time of any other activity. The next two results make this precise.

Theorem 9.13: total slack and critical activities

For every activity j , let L_j be the maximum length of a path from START to FINISH through j . Then:

- $TS_j = T - L_j \geq 0$, so $TS_j = 0$ if and only if j is critical;
- for every δ with $0 \leq \delta \leq TS_j$ there is a schedule of duration T with $t_j = ES_j + \delta$, and no schedule of duration T has $t_j > ES_j + TS_j$.

Proof. 1. A path from START to FINISH through j is a path from START to j followed by a path from j to FINISH, and its length is the sum of the two lengths. Taking the longest ones, by Theorem 9.8 and Theorem 9.11,

$$L_j = ES_j + (T - LS_j) = T - TS_j.$$

- Every path from START to FINISH has length at most T , so $TS_j \geq 0$, with equality if and only if a path through j has length T , that is, j is critical.
- If $TS_j > 0$, let $\lambda = \delta / TS_j \in [0, 1]$. ES and LS are schedules of duration T , and the constraints of a schedule are linear, so $(1 - \lambda)ES + \lambda LS$ is a schedule of duration T , with $t_j = ES_j + \lambda TS_j = ES_j + \delta$. If $TS_j = 0$, then $\delta = 0$ and ES works.
- By Theorem 9.11, every schedule of duration T has $t_j \leq LS_j = ES_j + TS_j$. $\square$

Proposition 9.14: free slack

For every activity j , $0 \leq FS_j \leq TS_j$. Moreover, for every δ with $0 \leq \delta \leq FS_j$, the start times ES with $ES_j + \delta$ in place of ES_j form a schedule of duration T : activity j is delayed by δ , and every other node keeps its earliest start time.

Proof. 1. By the forward pass, $ES_k \geq EF_j$ for every arc (j, k) , so $FS_j \geq 0$. Moreover $ES_k \leq LS_k$ for every node k , by [Theorem 9.13](#) for the activities and since $ES_{n+1} = LS_{n+1} = T$. Hence

$$FS_j = \min_{(j,k) \in \delta^+(j)} ES_k - EF_j \leq \min_{(j,k) \in \delta^+(j)} LS_k - EF_j = LF_j - EF_j = TS_j.$$

2. Only the constraints of the arcs of j change. For $(i, j) \in \delta^-(j)$: $ES_j + \delta \geq ES_j \geq EF_i$. For $(j, k) \in \delta^+(j)$: $ES_k \geq EF_j + FS_j \geq (ES_j + \delta) + d_j$.

3. The duration is still $ES_{n+1} = T$, since $j \neq n + 1$. □

Theorem 9.15: critical paths are paths of tight arcs

A path P from START to FINISH is critical if and only if all its arcs are tight.

Proof. Let $P = i_0, i_1, \dots, i_k$ with $i_0 = 0$ and $i_k = n + 1$. Every arc has $ES_{i_{s+1}} - EF_{i_s} \geq 0$. Adding up over the arcs of P , the earliest start times telescope:

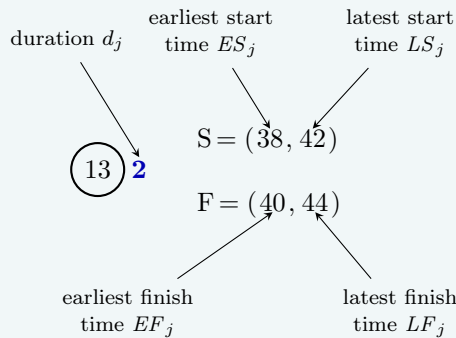
$$\sum_{s=0}^{k-1} (ES_{i_{s+1}} - EF_{i_s}) = ES_{n+1} - ES_0 - \sum_{s=0}^{k-1} d_{i_s} = T - L(P).$$

A sum of non-negative terms is zero if and only if every term is zero: $L(P) = T$ if and only if every arc of P is tight. □

- The critical paths are the paths from START to FINISH that use only tight arcs: to list all of them, follow the tight arcs from START.
- Every activity of a critical path has total slack zero. A path made of critical activities, instead, is critical only if its arcs are tight.

Example 9.5: slacks and critical path of the case study

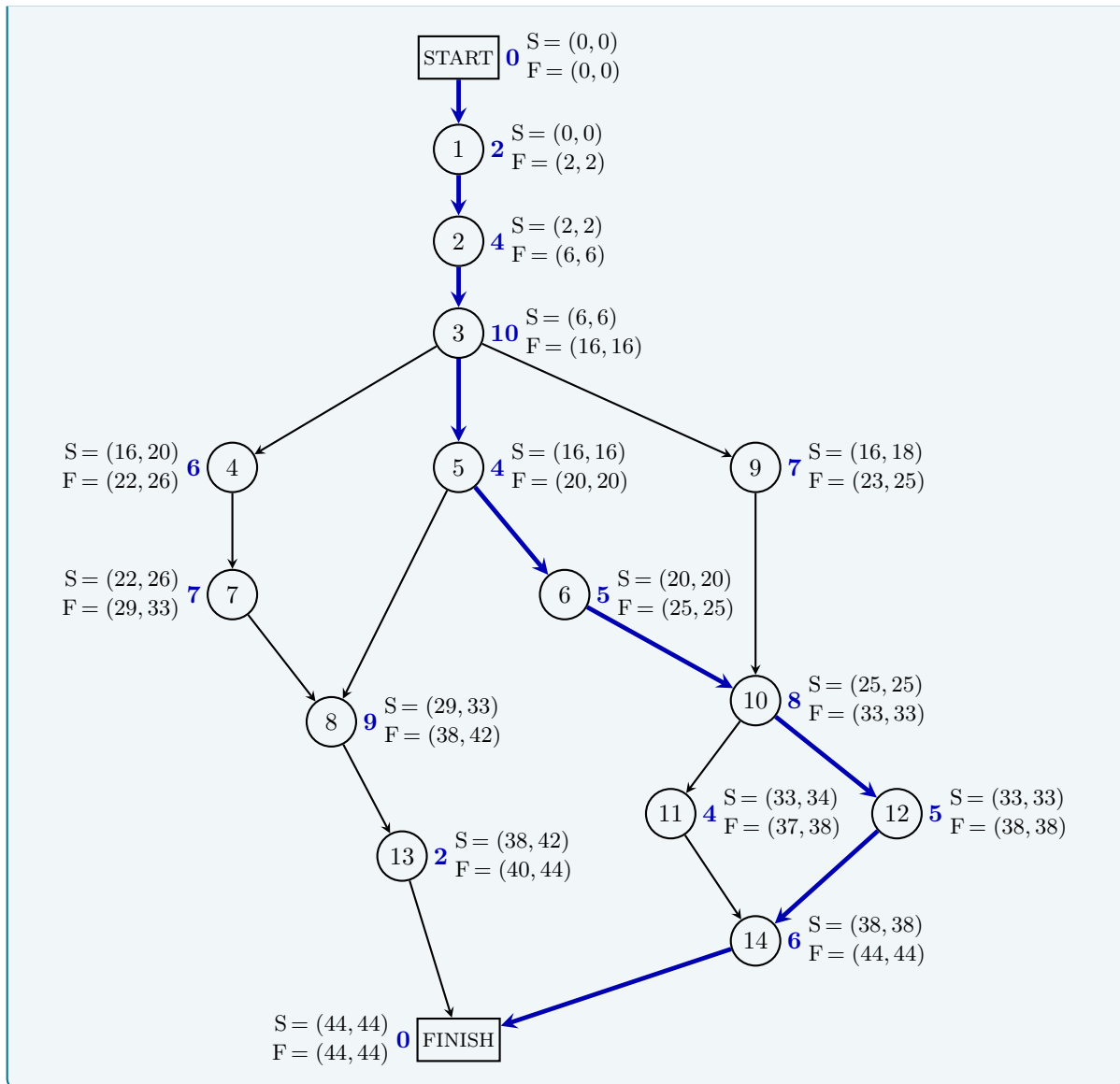
To show all the times at once, we write next to each node the pair $S = (ES_j, LS_j)$ of its start times and the pair $F = (EF_j, LF_j)$ of its finish times, as in the figure for activity 13:



The table collects the times and the slacks of the activities of [Example 9.3](#) and [Example 9.4](#).

j	d_j	ES_j	EF_j	LS_j	LF_j	TS_j	FS_j	critical
1	2	0	2	0	2	0	0	yes
2	4	2	6	2	6	0	0	yes
3	10	6	16	6	16	0	0	yes
4	6	16	22	20	26	4	0	
5	4	16	20	16	20	0	0	yes
6	5	20	25	20	25	0	0	yes
7	7	22	29	26	33	4	0	
8	9	29	38	33	42	4	0	
9	7	16	23	18	25	2	2	
10	8	25	33	25	33	0	0	yes
11	4	33	37	34	38	1	1	
12	5	33	38	33	38	0	0	yes
13	2	38	40	42	44	4	4	
14	6	38	44	38	44	0	0	yes

- The critical activities are 1, 2, 3, 5, 6, 10, 12, 14. The tight arcs from START reach FINISH only along START, 1, 2, 3, 5, 6, 10, 12, 14, FINISH: the critical path is unique, and its length is $T = 44$ (blue in the figure).
- The activities 4, 7, 8, 13 form a chain with $TS_4 = TS_7 = TS_8 = TS_{13} = 4$, but only the last one has free slack, $FS_{13} = 4$. If activity 4 starts 4 weeks late, activities 7, 8, 13 must start 4 weeks late too, and the project still ends at week 44.
- Activities 9 and 11 have the same total and free slack, 2 and 1 weeks.



Total slacks are shared

The total slack of an activity is shared with the other activities of its paths: the four total slacks of the chain 4, 7, 8, 13 are the same 4 weeks. If activity 4 takes 3 weeks more and activity 8 takes 2 weeks more, the path through them grows by $5 > 4$ weeks, and the project ends at week 45.

The total slack $TS_j = LS_j - ES_j$ is the delay of j that does not delay the project; the free slack $FS_j \leq TS_j$ is the delay that does not move any other activity. The critical activities have $TS_j = 0$, and the critical paths are the paths from START to FINISH made of tight arcs.

9.6 The critical path method

Algorithm

Algorithm: Critical path method (CPM)

Input: a project with activities $1, 2, \dots, n$, durations $d_j > 0$ and precedences

Output: the times ES_j, EF_j, LS_j, LF_j , the slacks, the minimum project duration T and the tight arcs

- 1 **Step 0 (network).** Build the project network $G = (N, A)$ and a topological order: repeatedly remove a node without entering arcs (the smallest index, if more than one).
- 2 If nodes are left and all of them have entering arcs, **STOP**: the precedences contain a cycle.
- 3 **Step 1 (forward pass).** Set $ES_0 := 0$. For each node $j \neq 0$ in topological order, set $ES_j := \max_{(i,j) \in \delta^-(j)} EF_i$; set $EF_j := ES_j + d_j$ for every j , and $T := ES_{n+1}$.
- 4 **Step 2 (backward pass).** Set $LF_{n+1} := T$. For each node $j \neq n+1$ in reverse topological order, set $LF_j := \min_{(j,k) \in \delta^+(j)} LS_k$; set $LS_j := LF_j - d_j$ for every j .
- 5 **Step 3 (slacks).** For every activity j , set $TS_j := LS_j - ES_j$ and $FS_j := \min_{(j,k) \in \delta^+(j)} ES_k - EF_j$; mark the tight arcs, $ES_j = EF_i$. The critical paths are the paths of tight arcs from START to FINISH.
- 6 **STOP.**

Theorem 9.16: running time of the critical path method

The critical path method runs in $O(n + m)$ time, where $m = |A|$.

Proof. 1. Step 0 builds the arcs from the lists of immediate predecessors, plus at most $2n$ arcs of START and FINISH. The topological order takes $O(n + m)$ time, with a counter of the entering arcs of every node, as in [Chapter 3](#).

2. Step 1 scans every arc once, as an entering arc of its head; Step 2 scans every arc once, as a leaving arc of its tail.

3. Step 3 scans the leaving arcs of every activity once more, and marks the tight arcs with one comparison per arc. $\square$

- The output gives the subgraph of the tight arcs, and its paths from START to FINISH are the critical paths ([Theorem 9.15](#)). Listing them one by one can take longer, when they are many.

9.6.1 The Gantt chart

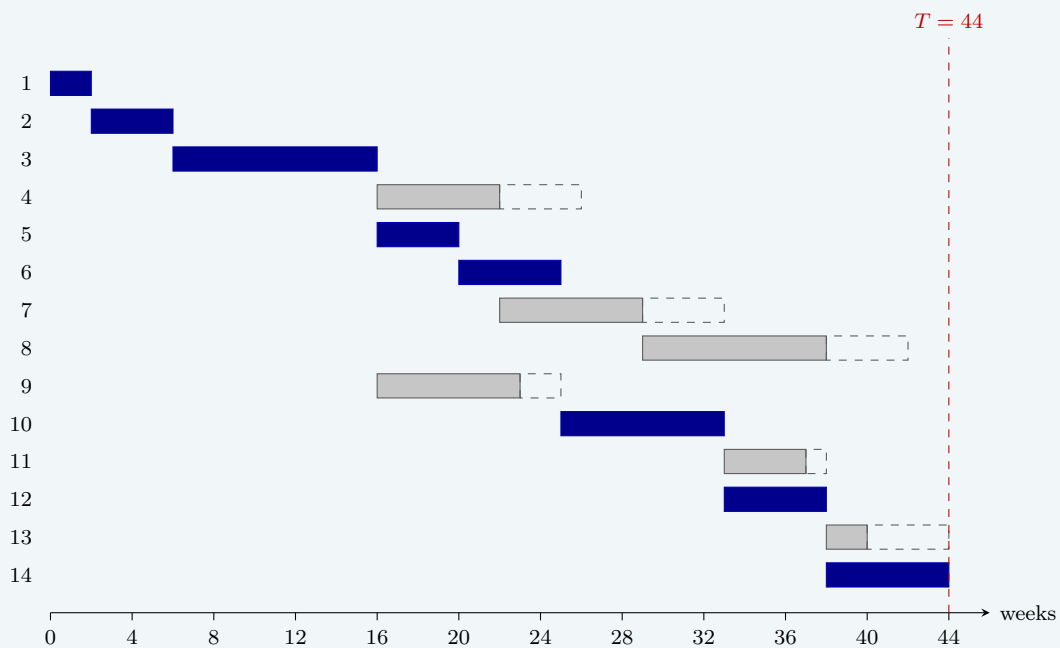
Definition 9.17: Gantt chart

The **Gantt chart** of a schedule t has a time axis and one row for every activity: the row of activity j has a bar from t_j to $t_j + d_j$.

- The chart shows at a glance which activities are in progress at every time. We draw the earliest schedule ES , with the bars of the critical activities in blue, and a dashed box from EF_j to LF_j that shows the total slack of each non-critical activity.

Example 9.6: the Gantt chart of the case study

The Gantt chart of the earliest schedule of the case study, in weeks:



- The blue bars of the critical activities follow each other without gaps from week 0 to week 44: a delay on any of them delays the project.
- At week 16 the activities 4, 5 and 9 start together: by (H2), three crews work in parallel.
- The dashed boxes of the chain 4, 7, 8, 13 are all 4 weeks long, but the bars of the chain follow each other: the 4 weeks of slack are shared.

The critical path method builds the network and a topological order, runs the forward and the backward pass and computes the slacks, in $O(n + m)$ time. The Gantt chart of the earliest schedule shows the activities in progress at every time, the critical path and the slacks.

9.7 The linear programming model

- The minimum project duration is the optimal value of an LP model whose variables are the start times. We use $n + 2$ free variables, one for every node:

$$t_j = \text{start time of node } j, \quad \forall j \in N.$$

Using these variables, an LP model for the minimum project duration reads as follows:

$$\min \quad t_{n+1} - t_0 \quad (9.1a)$$

$$\text{subject to} \quad t_j - t_i \geq d_i, \quad \forall (i, j) \in A, \quad (9.1b)$$

$$t_i \in \mathbb{R}, \quad \forall i \in N. \quad (9.1c)$$

This LP model is referred to as the PD model.

- Description of the objective function and constraints of the PD model:
 - the objective function (9.1a) minimizes the time from the start of the project to FINISH;
 - constraints (9.1b) make every node start after each of its immediate predecessors is finished;
 - finally, constraints (9.1c) define the variables of the LP model, which are free.
- Adding the same constant to all the start times keeps the constraints and the value: the schedules are the feasible solutions with $t_0 = 0$, and $z(\text{PD}) = T$.

9.7.1 The dual model: a longest path

- Following the rules of [Chapter 9 «Duality theory»](#) of the module Linear Programming, we use a dual variable $x_{ij} \geq 0$ for every constraint (9.1b), a $\geq$ constraint of a minimization model. Every free variable t_i gives a dual equation, whose right-hand side is the coefficient of t_i in the objective function.
- The variable t_i has coefficient $+1$ in the constraints of the arcs entering i and -1 in those of the arcs leaving i . With the balances

$$b_0 = 1, \quad b_{n+1} = -1, \quad b_i = 0, \quad \forall i \in N \setminus \{0, n+1\},$$

the equation of t_i , multiplied by -1 , is a flow conservation constraint.

Using a non-negative dual variable x_{ij} for each constraint (9.1b), the dual model of the PD model is:

$$\max \quad \sum_{(i,j) \in A} d_i x_{ij} \quad (9.2a)$$

$$\text{subject to} \quad \sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} = b_i, \quad \forall i \in N, \quad (9.2b)$$

$$x_{ij} \geq 0, \quad \forall (i, j) \in A. \quad (9.2c)$$

This LP model is referred to as the $D(\text{PD})$ model.

- Description of the dual model:
 - constraints (9.2b) send one unit of flow from START to FINISH;
 - the objective function (9.2a) maximizes the length of the flow, with length d_i on the arc (i, j) : the incidence vector of a path P from START to FINISH is feasible, with value $L(P)$.
- With the lengths $-d_i$, the $D(\text{PD})$ model is the model $\text{SP}(0, n+1)$ of Chapter 3 with the objective multiplied by -1 , and the PD model is its dual: the start times are the potentials, $\pi_i = t_i$.
- The reduced cost of the arc (i, j) is then $\bar{c}_{ij} = -d_i - t_i + t_j$. With $\mathbf{t} = ES$ it is $\bar{c}_{ij} = ES_j - EF_i \geq 0$: the tight arcs are the arcs with reduced cost zero, and Theorem 9.15 is the certificate of Chapter 3, feasible potentials and a path of arcs with reduced cost zero.

Theorem 9.18: LP models and critical paths

$z(\text{PD}) = z(D(\text{PD})) = T$. The schedules ES and LS are optimal for the PD model, and the incidence vector of every critical path is optimal for the $D(\text{PD})$ model. Moreover, every optimal solution of the PD model satisfies $t_j = t_i + d_i$ on every arc (i, j) of every critical path.

Proof. 1. ES and LS are feasible for the PD model, with value $T - 0 = T$, by Theorem 9.8 and Theorem 9.11.

2. Let P be a critical path and $\mathbf{x}^P$ its incidence vector, with $x_{ij}^P = 1$ on the arcs of P and 0 elsewhere. One unit leaves START, one unit enters FINISH, and every other node of P has one entering and one leaving arc of P : $\mathbf{x}^P$ is feasible for the $D(\text{PD})$ model, with value $L(P) = T$.

3. A feasible solution of the PD model and a feasible solution of its dual with the same value are both optimal, by weak duality.

4. Let $\mathbf{t}$ be optimal for the PD model, and $P = i_0, i_1, \dots, i_k$ a critical path. Adding up the constraints of its arcs,

$$T = t_{n+1} - t_0 = \sum_{s=0}^{k-1} (t_{i_{s+1}} - t_{i_s}) \geq \sum_{s=0}^{k-1} d_{i_s} = L(P) = T,$$

so every constraint of an arc of P holds with equality. $\square$

- In every schedule of minimum duration the critical activities start at their earliest start times, which equal their latest start times.

Example 9.7: the LP models of the case study

For the case study the PD model has 16 variables and 19 constraints; for example, the constraint of the arc (5, 8) is $t_8 - t_5 \geq 4$. Solved with Gurobi:

- $z(\text{PD}) = z(D(\text{PD})) = 44$, and the optimal dual solution returned by Gurobi is the incidence vector of the critical path START, 1, 2, 3, 5, 6, 10, 12, 14, FINISH;
- the optimal solution of the PD model is not unique: ES and LS are both optimal, and they differ, for example, on activity 4, with $ES_4 = 16$ and $LS_4 = 20$;
- with $t = ES$ the reduced costs $\bar{c}_{ij} = ES_j - EF_i$ are zero on every arc except $\bar{c}_{58} = 9$, $\bar{c}_{9,10} = 2$, $\bar{c}_{11,14} = 1$ and $\bar{c}_{13,15} = 4$. The arcs (3, 4), (4, 7), (7, 8), (8, 13) are tight, but the tight arcs from activity 13 do not reach FINISH, so the chain 4, 7, 8, 13 is not on a critical path.

The PD model minimizes $t_{n+1} - t_0$ subject to $t_j - t_i \geq d_i$ on every arc; its dual sends one unit of flow from START to FINISH along a longest path. Both optimal values are T : ES and LS are optimal schedules, and every critical path is an optimal dual solution.

9.8 Two critical paths and changes of durations

- A planner often asks what happens to the project if one activity takes longer, or if more crews make it shorter. The slacks and the critical paths answer without running the method again.

Proposition 9.19: changing one duration

Let j be an activity, $\delta > 0$, and T' the minimum project duration after the change.

- If d_j increases by δ , then $T' = T + \max\{0, \delta - TS_j\}$.
- If d_j decreases by $\delta < d_j$, then $T - \delta \leq T' \leq T$, and $T' = T$ if some critical path does not contain j .

Proof. By [Theorem 9.9](#), the minimum project duration is the maximum length of a path from START to FINISH. The change modifies by $\pm\delta$ the length of the paths through j , and only of them.

- By [Theorem 9.13](#), the longest paths through j grow to $L_j + \delta = T - TS_j + \delta$. If $TS_j > 0$, the critical paths avoid j and keep the length T ; if $TS_j = 0$, the paths avoiding j have length at most $T < T + \delta$. In both cases $T' = \max\{T, T - TS_j + \delta\}$.
- No path grows and every path loses at most δ , so $T - \delta \leq T' \leq T$. A critical path that avoids j keeps the length T . $\square$

Example 9.8: a project with two critical paths

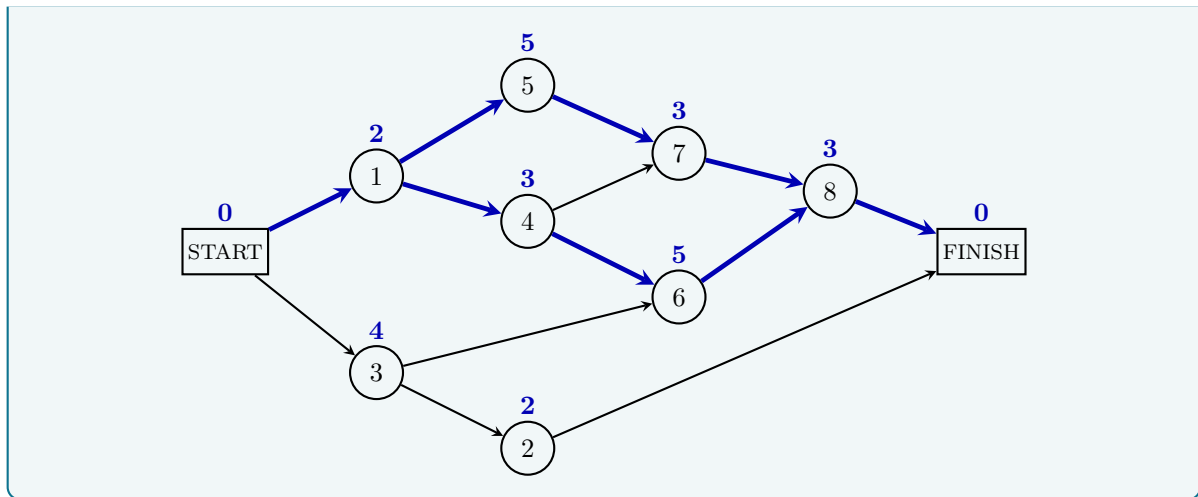
A project has 8 activities with the immediate predecessors and the durations of the table.

activity j	immediate predecessors	d_j
1	—	2 days
2	3	2 days
3	—	4 days
4	1	3 days
5	1	5 days
6	3, 4	5 days
7	4, 5	3 days
8	6, 7	3 days

Activity 2 follows activity 3, so the topological order is not the natural one. START frees 1 and 3; node 1 is removed first and frees 4 and 5; then 3 is the free node of smallest index and frees 2, which comes before 4. The order is 0, 1, 3, 2, 4, 5, 6, 7, 8, 9, and the forward and the backward pass give:

j	d_j	ES_j	EF_j	LS_j	LF_j	TS_j	FS_j	critical
1	2	0	2	0	2	0	0	yes
2	2	4	6	11	13	7	7	
3	4	0	4	1	5	1	0	
4	3	2	5	2	5	0	0	yes
5	5	2	7	2	7	0	0	yes
6	5	5	10	5	10	0	0	yes
7	3	7	10	7	10	0	0	yes
8	3	10	13	10	13	0	0	yes

- $T = 13$, and two paths of tight arcs go from START to FINISH: START, 1, 4, 6, 8, FINISH and START, 1, 5, 7, 8, FINISH, both of length 13 (blue in the figure, with the durations above the nodes).
- Activity 3 has $TS_3 = 1$ and $FS_3 = 0$: one day of delay moves the earliest start time of activity 2, not the end of the project. Activity 2 has $TS_2 = FS_2 = 7$.
- The path START, 1, 4, 7, 8, FINISH uses only critical activities, but its arc (4, 7) is not tight, $EF_4 = 5 < 7 = ES_7$: its length is $2 + 3 + 3 + 3 = 11$, and it is not critical.



Example 9.9: changes of one duration

We change one duration at a time in the project of [Example 9.8](#); the new duration and the critical paths are computed by the critical path method, and agree with [Proposition 9.19](#).

change	TS_j	new T	critical paths after the change
$d_5 = 5 \rightarrow 4$	0	13	START, 1, 4, 6, 8, FINISH
$d_8 = 3 \rightarrow 2$	0	12	START, 1, 4, 6, 8, FINISH; START, 1, 5, 7, 8, FINISH
$d_3 = 4 \rightarrow 6$	1	14	START, 3, 6, 8, FINISH

- Shortening activity 5 by one day does not shorten the project: the critical path START, 1, 4, 6, 8, FINISH avoids activity 5, and $T' = T = 13$ by (b). Only one critical path is left.
- Activity 8 lies on both critical paths: shortening it by one day gives $T' = T - 1 = 12$, and both paths stay critical.
- Activity 3 takes 2 days more, more than its total slack $TS_3 = 1$: by (a), $T' = 13 + (2 - 1) = 14$, and START, 3, 6, 8, FINISH becomes the critical path.

Shortening a project

To shorten a project, every critical path must become shorter. When there are several critical paths, an activity that lies on only one of them does not help: shorten an activity common to all of them, or one activity on each.

If d_j grows by δ , the project lasts $\max\{0, \delta - TS_j\}$ more. If d_j decreases, the project gains at most the same amount, and nothing if a critical path avoids j : with several critical paths, all of them must be shortened.

9.9 Typical mistakes

Warning

- **Adding up all the durations.** The case study lasts $T = 44$ weeks, not 79: activities that are not linked by precedences run in parallel.
- **Taking the minimum in the forward pass.** An activity starts when *all* its immediate predecessors are finished: ES_j is the maximum of their earliest finish times. Symmetrically, LF_j is the minimum of the latest start times of the successors.
- **Forgetting START, FINISH and their arcs.** Every activity without immediate predecessors needs the arc from START, and every activity without successors the arc to FINISH.
- **Adding up total slacks along a path.** The chain 4, 7, 8, 13 of the case study has 4 weeks of slack in all, not 16.
- **Confusing total and free slack.** A delay larger than FS_j moves other activities, even when it does not delay the project.
- **Reading a critical path from the critical activities only.** In [Example 9.8](#) the path START, 1, 4, 7, 8, FINISH has only critical activities, and it is not critical: check that the arcs are tight.
- **Shortening an activity on one of several critical paths.** The project duration does not change.
- **Accepting precedences with a cycle.** No schedule exists, and the topological order stops before ordering all the nodes.

9.10 In summary

Summary

- A project has activities with deterministic durations and precedences; its project network has a node for every activity, plus START and FINISH, and must be acyclic.
- The length of a path from START to FINISH is the total duration of its activities; every schedule lasts at least as long as every path, and the critical paths are the longest ones.
- The forward pass gives $ES_j = \max EF_i$ and $T = ES_{n+1}$, the length of a critical path; the backward pass gives $LF_j = \min LS_k$ starting from $LF_{n+1} = T$.
- $TS_j = LS_j - ES_j$ is the delay that does not delay the project, $FS_j \leq TS_j$ the delay that does not move other activities; the critical paths are the paths of tight arcs.
- The PD model $\min\{t_{n+1} - t_0 : t_j - t_i \geq d_i\}$ has the longest path model as its dual, and both have optimal value T .

- Increasing d_j by δ delays the project by $\max\{0, \delta - TS_j\}$; shortening helps only if every critical path becomes shorter.

The two projects at a glance.

case study	14 activities, 79 weeks in all; $T = 44$ weeks
critical path	START, 1, 2, 3, 5, 6, 10, 12, 14, FINISH
non-critical j (TS_j, FS_j)	4 (4, 0), 7 (4, 0), 8 (4, 0), 9 (2, 2), 11 (1, 1), 13 (4, 4)
LP models	$z(\text{PD}) = z(D(\text{PD})) = 44$; ES and LS optimal
second example	$T = 13$, 2 critical paths: START, 1, 4, 6, 8, FINISH and START, 1, 5, 7, 8, FINISH
changes	$d_5 = 5 \rightarrow 4$: $T = 13$; $d_8 = 3 \rightarrow 2$: $T = 12$; $d_3 = 4 \rightarrow 6$: $T = 14$