MAST20018
Discrete Mathematics

Charl Ras

2015

Topics

1. Scheduling
2. Network optimisation
3. Social choice

Scheduling

§1.1 – Motivating Example

We motivate our first topic of networks by considering how to plan a dinner party, from a mathematical viewpoint.
This application will recur throughout the next two topics.

How to plan a dinner party

Step 1. List the activities associated with the preparation of a dinner party.

Activity
Choose an evening
Plan drinks
Buy drinks
Plan dinner
Buy groceries
Prepare meal
Greet guests
Serve the meal

How to plan a dinner party

Step 2. List the duration of each of the activities associated with the preparation of a dinner party

Activity	Time (mins)
Choose an evening	5
Plan drinks	5
Buy drinks	30
Plan dinner	30
Buy groceries	60
Prepare meal	90
Greet guests	20
Serve the meal	15
	255

How to plan a dinner party

Does this mean it takes 255 minutes (the total of all the times) to prepare for the party?

Or, suppose you had unlimited helpers, would the preparation only take 90 minutes (the length of the longest job) to complete?

Why or why not? What else do we need to consider?

§1.2 – Scheduling

The answers to problems like "how long does it take to prepare for a dinner party?" are found using mathematical scheduling theory.

In this topic we use networks to analyse these types of scheduling problems mathematically. The solutions can very efficient, and when used in planning stages helps the decision making process.

Scheduling with a precedence ordering

Many complex large scale projects, or even simpler projects like planning dinner parties, are made up of many individual tasks, or activities which are to some extent sequential.

For example, the meals can't be prepared before the groceries are bought. One says that buying the groceries has precedence over the meal preparation.

Scheduling with a precedence ordering

One of the earliest uses of this analysis was by the DuPont Corporation, who applied to the problem of shutting down chemical plants for maintenance, and then restarting them again.

Although its real benefit is to complex large scale projects, our illustrations will continue to be in more common place settings, which nevertheless illustrate the essence of the method.

§1.3 – Introduction to Activity Networks

Our first task is to represent the activities, and their precedences and completion times in a diagrammatic form. For this, use will be made of directed graphs, which are collections of points and directed edges. (The term directed graph is often abbreviated digraph.) The particular digraph which results has special properties, and is referred to as an activity network.

We will continue with our example of the preparation of a dinner party.

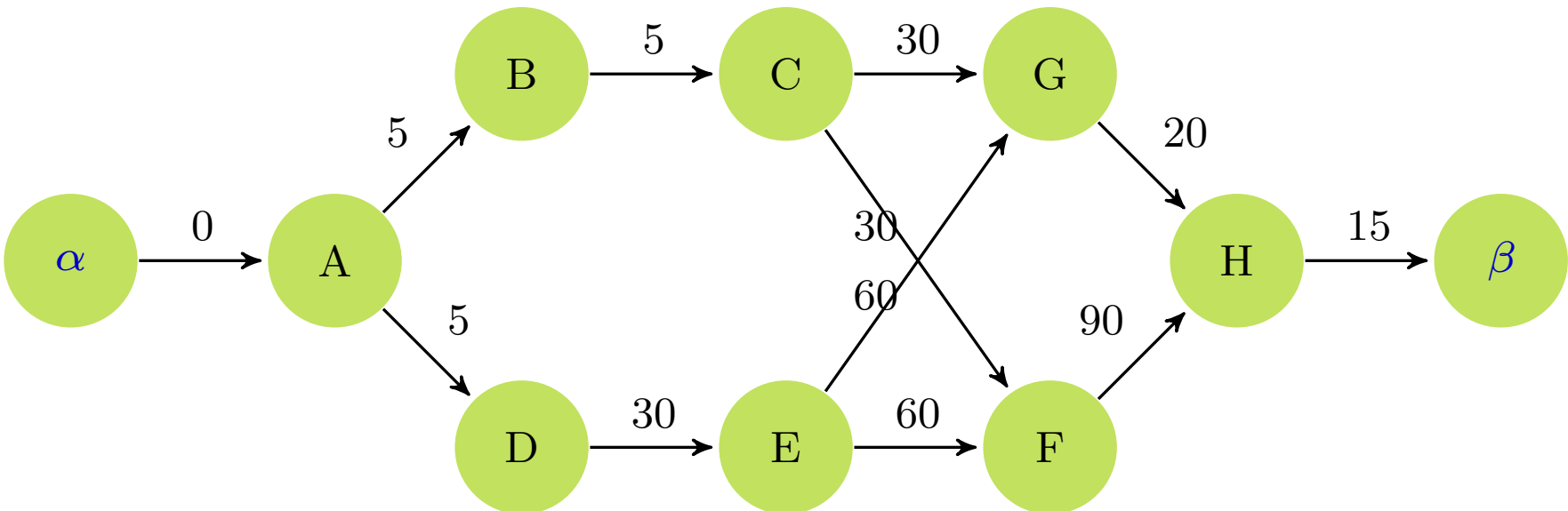
Activity Networks

We take our table from before, and add in precedences and activity labelings.

	Activity	Time	Precedence
A.	Choose an evening	5	
B.	Plan drinks	5	A
C.	Buy drinks	30	B
D.	Plan dinner	30	A
E.	Buy groceries	60	D
F.	Prepare meal	90	C,E
G.	Greet guests	20	C, E
H.	Serve the meal	15	F,G

Activity Networks

We can represent this information graphically:



§2 – Activity on Node

This is called an activity on node or AON activity network. It has the property of having a source node (here α) and a sink node (here β) such that every node is reachable from α, and β is reachable from every node.

The nodes α and β have been added artifically—this can always be done. The node α connects to all nodes with no precedence and thus no existing in-coming arrows, while β connects to all nodes with no existing out-going arrows.

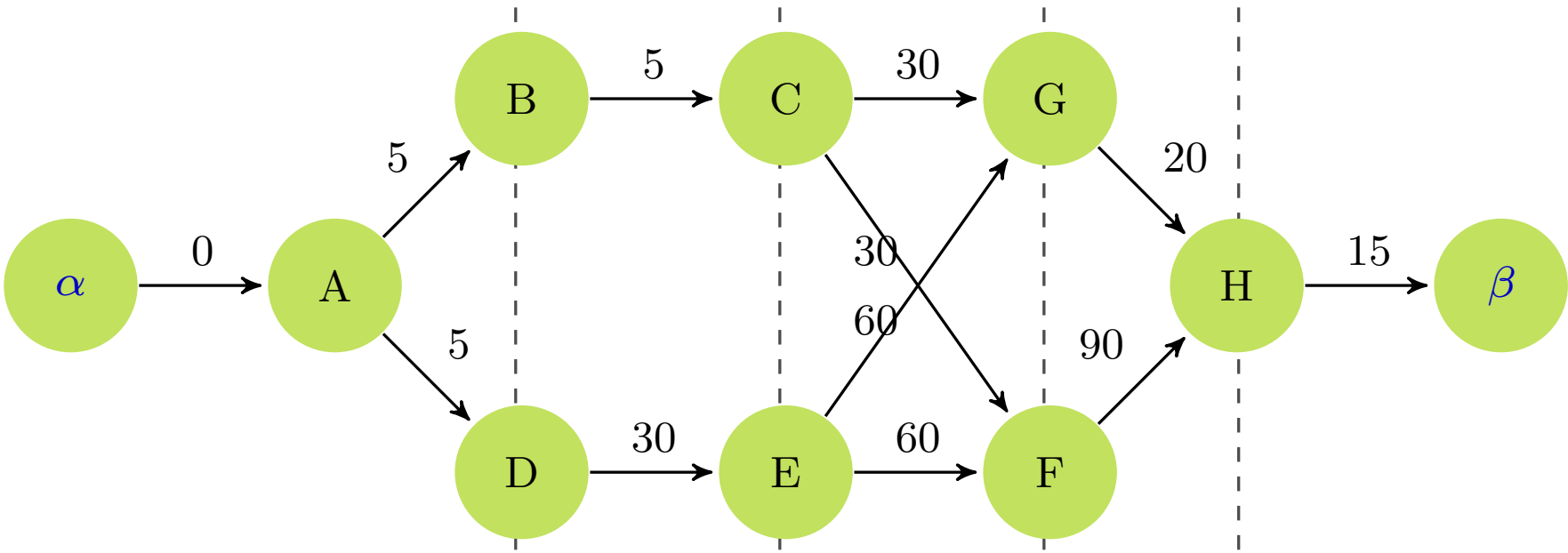
Activity on Node

The arrowed edges, referred to as arcs or more simply as arrows, represent (immediate) precedence. We say immediate precedence because if $X \to Y$ and $Y \to Z$ then $X \to Z$, which technically says that $\to$ is transitive.

The tasks can always be ordered so that the arrows always have a component which points from left to right. So if we draw activity X to the left of activity Y, and X and Y are connected in a chain (i.e. connected by arrows), then $X \to Y$.

§2.1 – Drawing an AON

There is a natural way to order the nodes when reading from left to right according to the number of nodes which are connected to the left:



So as we read left to right, the nodes on each dotted line have one more node in their precedence chain.

A systematic way to draw an AON activity network

To identify the number of nodes in the precedence chain of each activity, the following algorithm can be used.

1. Start the counter k for the number nodes in the precedence chain at $k = 0$.
2. Delete all activities from the list with no precedences, and assign them to having k such nodes.
3. If there are no activities in the list, stop.
4. Otherwise $k = k + 1$ and go to step 2.

In our example, Step 1 isolates activity A as having no precedences. After deleting it, the list reads

A systematic way to draw an AON activity network

Activity	Precedence
B.	—
C.	B
D.	—
E.	D
F.	C, E
G.	C, E
H.	F, G

Now activities B and D have no precedences, so are such that $k = 1$, and we delete them from the list to obtain...

Activity	Precedence
C.	—
E.	—
F.	C, E
G.	C, E
H.	F, G

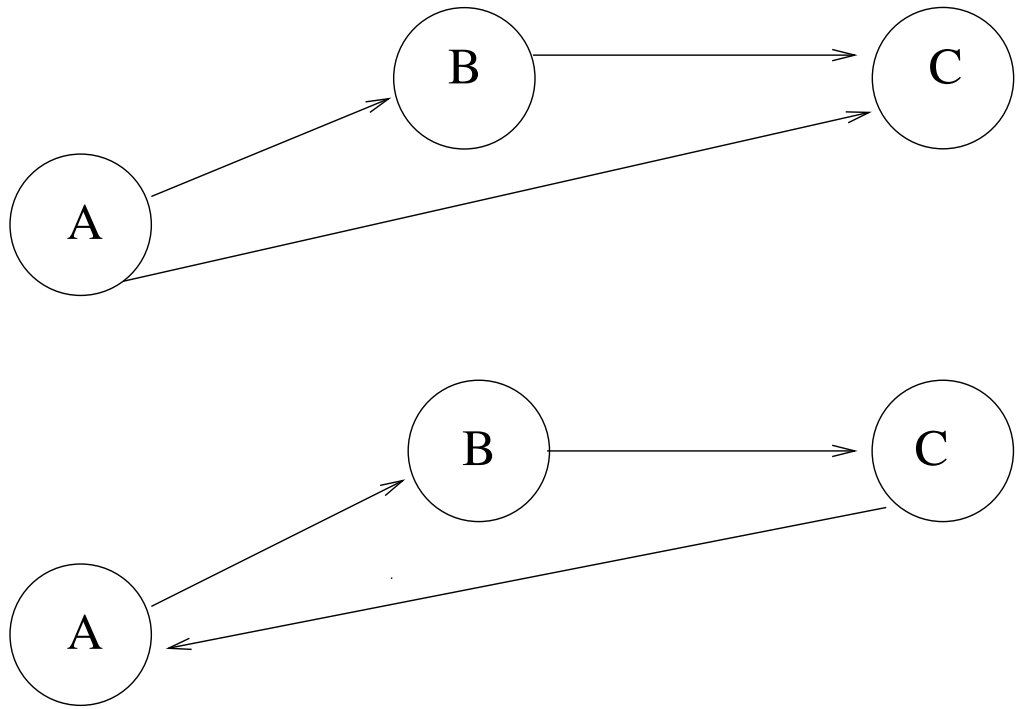
Now activities C and E have no precedences, so are such that $k = 2$, and we delete them from the list to obtain

Activity	Precedence
F.	—
G.	—
H.	F, G

Now activities F and G have no precedences, so are such that $k = 3$, and when we delete them only H remains, which then is such that $k = 4$ and we stop.

True or false?

In an AON activity network, it is not possible to have either of the formations

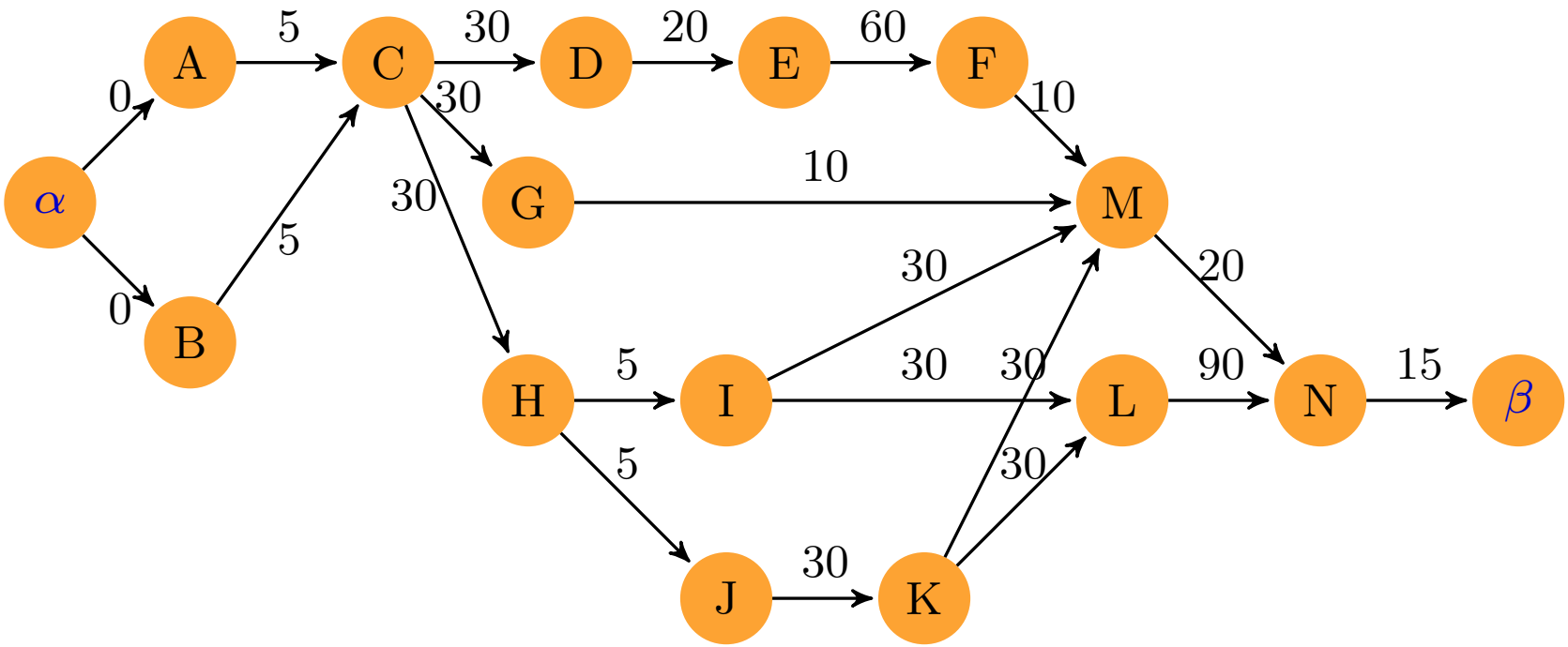


Explain.

Let's now take up the task of constructing an AON activity network for an extension of the task of preparing a dinner party.

	Activity	Time	Precedence
A.	Choose an evening	5	—
B.	Choose friends to invite	5	—
C.	Establish who accepts	30	A, B
D.	Store breakables	20	C
E.	Clean up apartment	60	D
F.	Re-arrange furniture	10	E
G.	Warn neighbours	10	C
H.	Plan drinks	5	C
I.	Buy drinks	30	H
J.	Plan dinner	30	H
K.	Buy groceries	30	J
L.	Prepare meal	90	I, K
M.	Greet guests	20	F, G, I, K
N.	Serve meal	15	L, M

§2.2 – Exercise



Exercise: Use the systematic approach to reproduce the AON activity network above.

§3 – Critical Paths

From knowledge of the times of the individual activities, one would like to proceed to compute the minimum time for the completion of the project. This minimum time only applies if there is no restriction on two or more activities being completed at once, provided that the precedence ordering is still obeyed. In practice, this requires multiple people.

We would also like to identify the sequence of activities in the activity network, starting at α and finishing at β following certain arrows, such that the sum of the completion times equals the minimum time. Such a sequence of activities is called a critical path.

Critical Paths

Nodes on the critical path are termed critical activities. If any critical activity is delayed, the project completion time will be delayed.

On the other hand, activities which are not critical may be delayed a certain amount of time without delaying the project completion time. Features of interest with respect to the minimum time of completion then are:

Critical Paths

- T_i = the time taken to complete activity i
- ES_i = the earliest start time for activity i
- LF_i = the latest finish time for activity i
- LS_i = the latest start time for activity i
- EF_i = the earliest finish time for activity i

For a critical activity, $ES_i = LS_i$ and $EF_i = LF_i$. This follows since the critical activities are on the critical path, which gives the minimum time for the completion of the project.

As an initial condition, $ES_\alpha = 0$.

§3.1 – Calculation of earliest times

- $\mathrm{EF}_i = \mathrm{ES}_i + \mathrm{T}_i$
- $\mathrm{ES}_i = \max\{\mathrm{EF}_{i^*}\}$, where the maximum is taken over all points i^* with (i^*, i) an arc of the network entering the node i, since no activity can begin until all the activities which are its immediate precedences, denoted $\{i^*\}$, are finished.
- EF_β is the minimum time of completion of the project.

Starting with the initial condition $\mathrm{ES}_\alpha = 0$, these earliest times can be computed successively across the network, reading from left to right.

§3.2 – Calculation of latest times

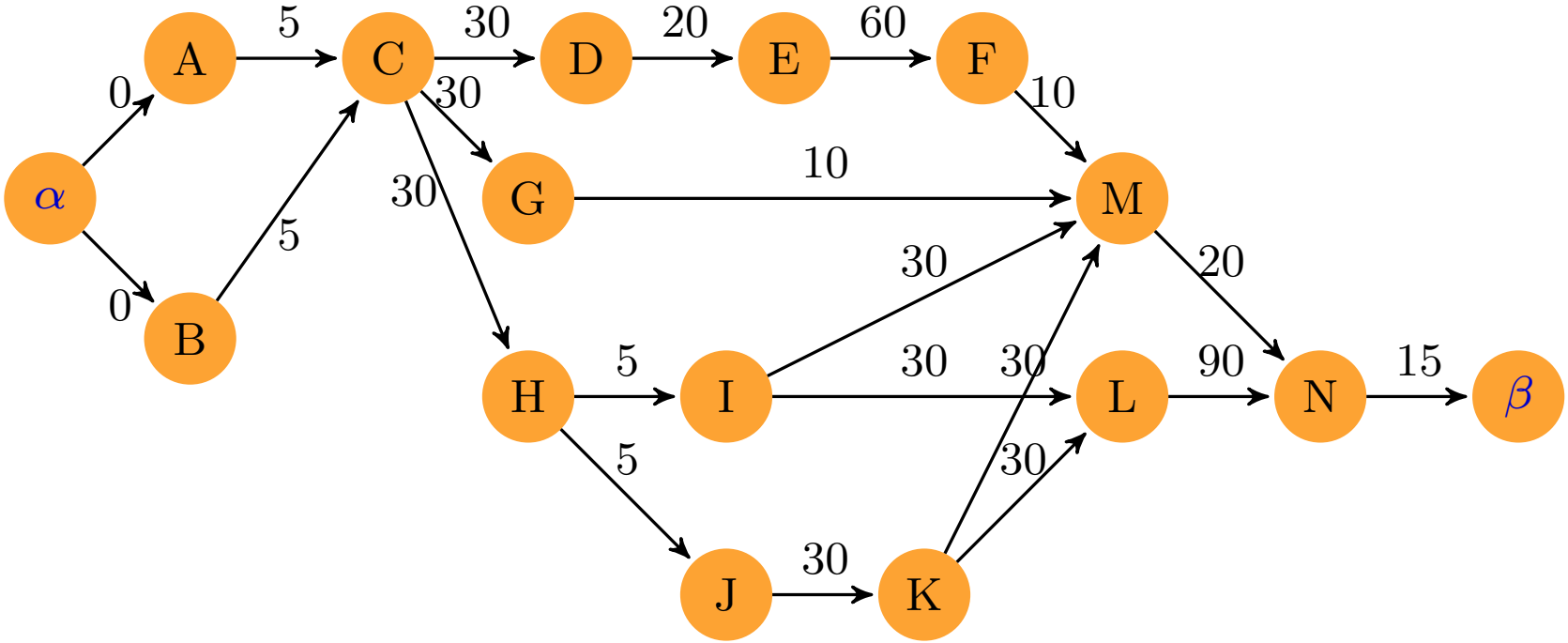
We have $\mathrm{EF}_\beta = \mathrm{LF}_\beta$, giving us an initial condition at the very right of the network.

- $\mathrm{LS}_i = \mathrm{LF}_i - \mathrm{T}_i$, since each activity must start T_i units of time before it ends.
- $\mathrm{LF}_i = \min\{\mathrm{LS}_{i^\#}\}$, where the minimum is taken over all points $i^\#$ with $(i, i^\#)$ an arc of the network leaving the node i

Starting with the initial condition $\mathrm{LF}_\beta = \mathrm{EF}_\beta$, these latest times can be computed successively across the network, reading from right to left.

As a check, the latest start time for α should be computed to equal 0, $\mathrm{LS}_\alpha = 0$.

§3.3 – Example



We first compute the earliest times.

Activity i	ES_i	EF_i
A	$\max\{0\} = 0$	$0 + 5 = 5$
B	$\max\{0\} = 0$	$0 + 5 = 5$
C	$\max\{5,5\} = 5$	$5 + 30 = 35$
D	$\max\{35\} = 35$	$35 + 20 = 55$

Example ...

Activity i	ES_i	EF_i
E	max $\{55\} = 55$	55+60 = 115
F	max $\{115\} = 115$	115 + 10 = 125
G	max $\{35\} = 35$	35 + 10 = 45
H	max $\{35\} = 35$	35 + 5 = 40
I	max $\{40\} = 40$	40 + 30 = 70
J	max $\{40\} = 40$	40 + 30 = 70
K	max $\{70\} = 70$	70 + 30 = 100
L	max $\{70,100\} = 100$	100+90=190
M	max $\{125,45,70,100\} = 125$	125+20 = 145
N	max $\{145,190\} = 190$	190 + 15 = 205
β	max $\{205\} = 205$	205

Example ...

From the requirement that $\mathrm{LF}_\beta = \mathrm{EF}_\beta = 205$, we can proceed to compute the finish times by working backwards through the activity network.

Activity i	LF_i	LS_i
N	$\min\{205\} = 205$	205 - 15= 190
M	$\min\{190\} = 190$	190 - 20 = 170
L	$\min\{190\} = 190$	190 - 90 = 100
K	$\min\{170,100\} = 100$	100 - 30 = 70
F	$\min\{170\} = 170$	170 - 10 = 160
J	$\min\{70\} = 70$	70 - 30 = 40
I	$\min\{170,100\} = 100$	100 - 30 = 70
E	$\min\{160\} = 160$	160 - 60 = 100
H	$\min\{100,40\} = 40$	40 - 5 = 35
G	$\min\{170\} = 170$	170 - 10 = 160
D	$\min\{100\} = 100$	100 - 20 = 80

Example ...

Activity i	LF_i	LS_i
C	$\min\{35,160,80\} = 35$	35 - 30 = 5
B	$\min\{5\} = 5$	5 - 5 = 0
A	$\min\{5\}=5$	5 - 5 = 0
α	$\min\{0,0\} = 0$	0

We have remarked that the critical activities are characterised by the property that $\mathrm{EF}_i = \mathrm{LF}_i$ (it must then be that $\mathrm{LS}_i = \mathrm{ES}_i$). We read off from our table that the critical activities are

A, B, C, H, J, K, L, N

Hence there are two critical paths:

A $\rightarrow$ C $\rightarrow$ H $\rightarrow$ J $\rightarrow$ K $\rightarrow$ L $\rightarrow$ N
B $\rightarrow$ C $\rightarrow$ H $\rightarrow$ J $\rightarrow$ K $\rightarrow$ L $\rightarrow$ N

§3.4 – Total Float

Other useful information can be obtained from our tables.
It is of interest, for example, to compute the total float, denoted TF_i for each activity i. It relates to the accumulated time which is free in the sense that not starting the activity in the time will not lead to a delay. Conversely, not starting an activity before the total float has expired will delay the completion time.
It is defined by

$$\mathrm{TF}_i = \mathrm{LS}_i - \mathrm{ES}_i$$

Note that it can equivalently be written

$$\mathrm{TF}_i = \mathrm{LF}_i - \mathrm{EF}_i \quad \text{or} \quad \mathrm{TF}_i = \mathrm{LF}_i - \mathrm{ES}_i - \mathrm{T}_i$$

So it is equal to the difference between the maximum time available to complete the activity, $\mathrm{LF}_i - \mathrm{ES}_i$, and its duration time T_i.

For our example

Activity i	LS_i	ES_i	TF_i
A	0	0	0
B	0	0	0
C	5	5	0
D	80	35	45
E	100	55	45
F	160	115	45
G	160	35	125
H	35	35	0
I	70	40	30
J	40	40	0
K	70	70	0
L	100	100	0
M	170	125	45
N	190	190	0

Notice that the critical activities can be characterised by having total float equal to 0.

In the example, the activity with the largest total float is G.

For activities with a non-zero total float, there is scope for delaying the start time yet still finishing the job in the smallest amount of time.

§3.5 – Free Float

Another type of float of interest is the free float, denoted FF_i. It is defined as

$$\mathrm{FF}_i = \min\{\mathrm{ES}_{i^\#} - \mathrm{EF}_i\}$$

where $(i, i^\#)$ is an arc of the network leaving node i.

Hence it tells us the minimum amount of time between a job and one of its immediate successors, and thus the amount of spare time in the sense of not delaying any successor activity. It is a local measure, whereas the total float comes about because of an accumulation of free time.

Note that necessary conditions for the free float to be non-zero are that the activity is not critical, and that a successor of the activity has more than one in arrow.

Note too that $\mathrm{FF}_i \leq \mathrm{TF}_i$.

Exercises Compute the free float for activities E, M and G.

Do these results support the claim that $\mathrm{FF}_i \leq \mathrm{TF}_i$?

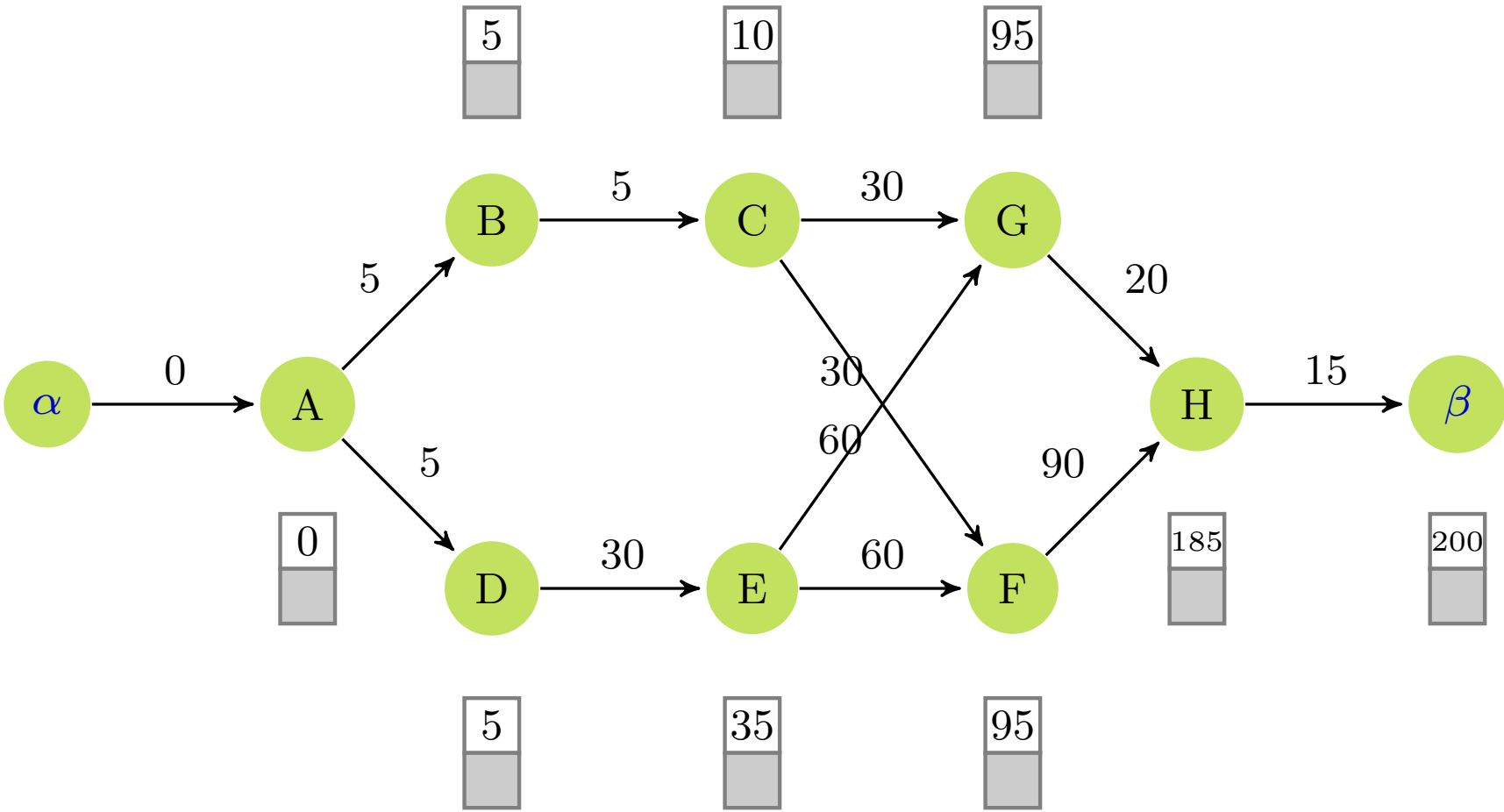
§3.6 – A table free approach to critical paths

The critical path of an activity network can be more efficiently found using the following method.

Here we'll use our first example of the simpler dinner party, and finally determine how long the preparation is going to take (as well as the path of critical activities).

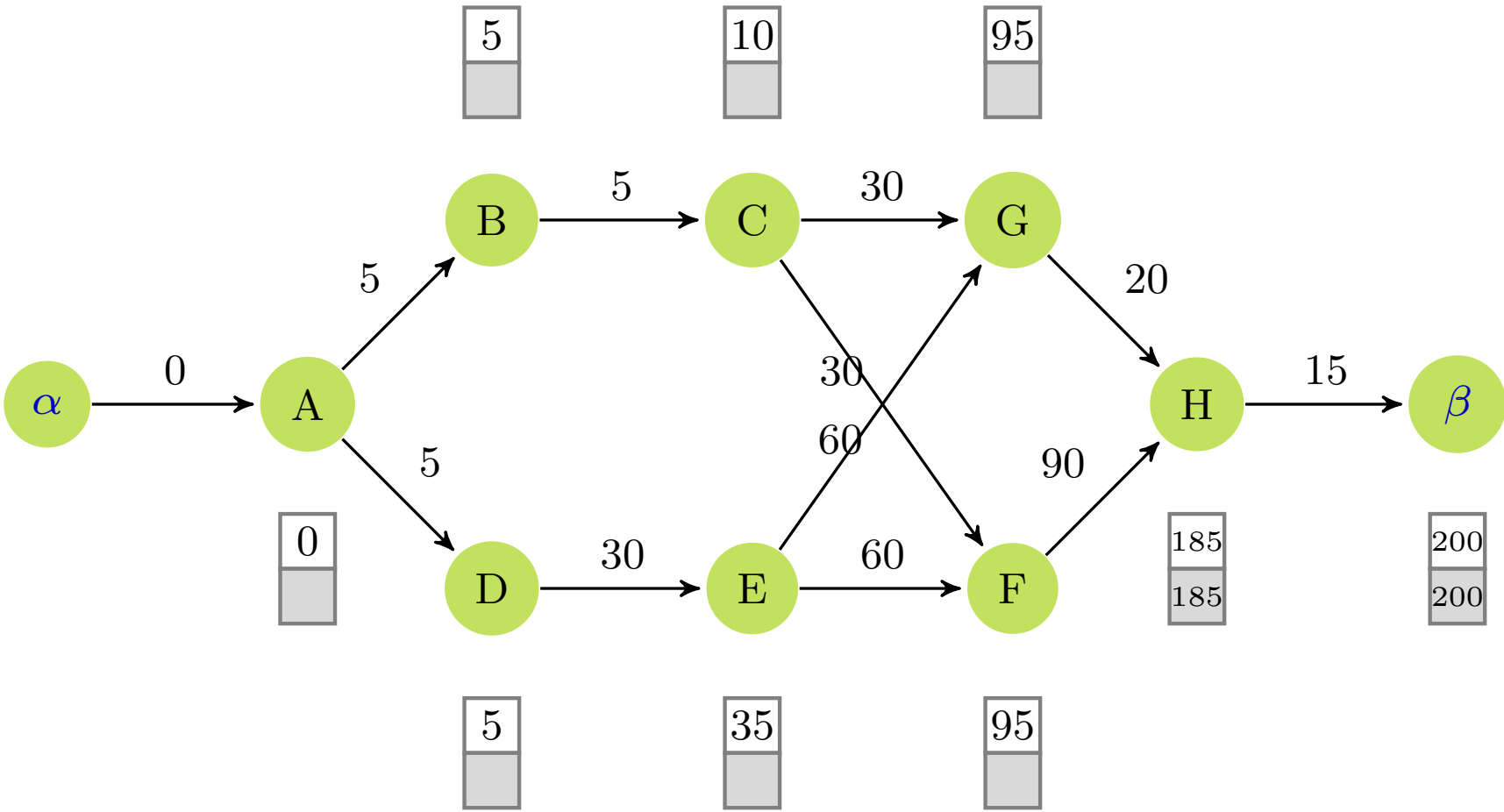
Step 1. Starting from α determine the earliest start time (white blocks) for each activity.

As when using the table, care must be taken when more than one arrow is entering a node, and the maximum of the earliest finish times must be taken.

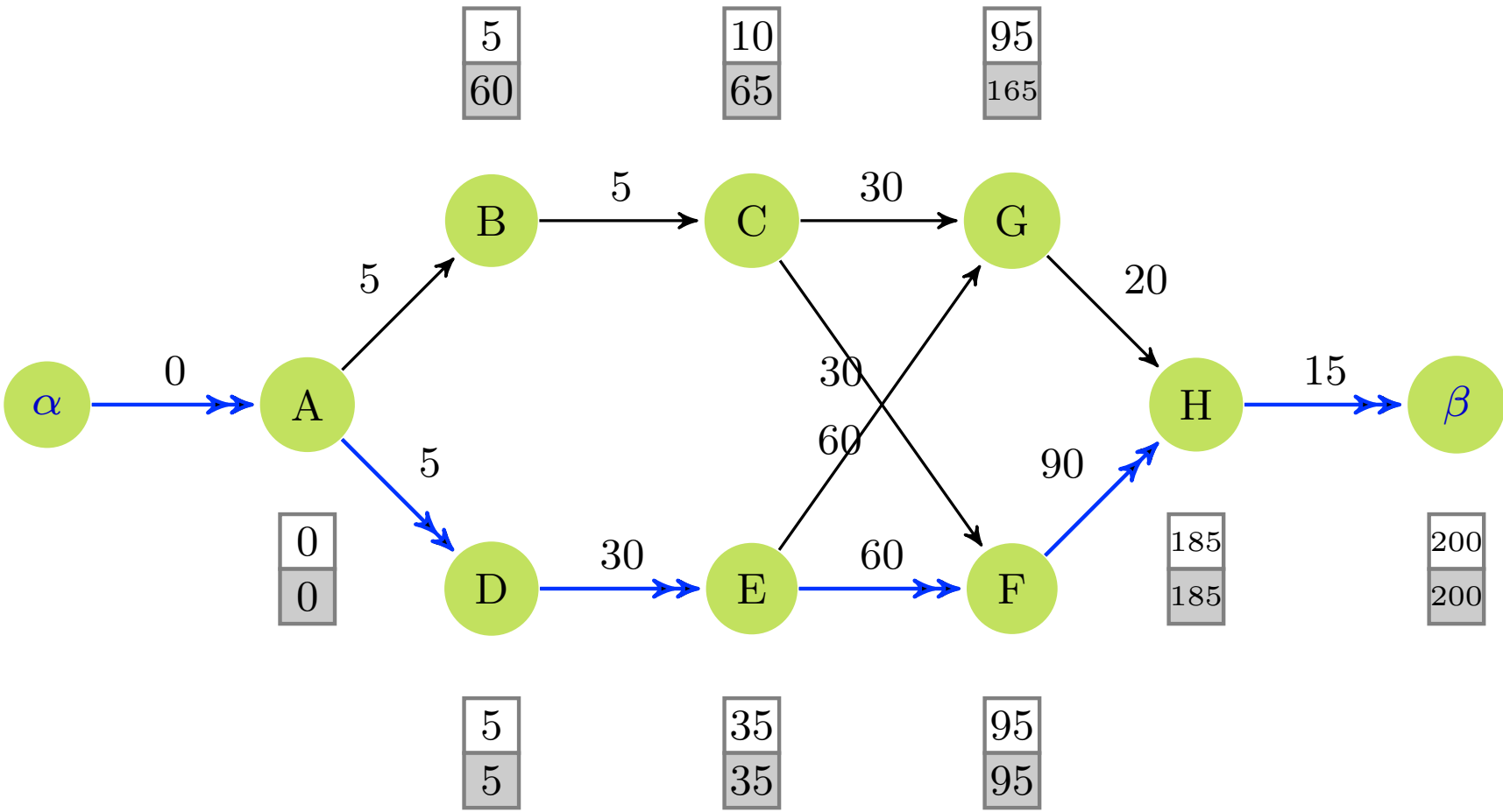


Step 2. Working backwards from β determine the latest start time (grey blocks) for each activity.

As when using the table, care must be taken when the activity being considered has more than one outgoing arrow, and again, the minimum of these latest start times must be taken.



Step 3. In each box you now have the earliest and latest start times. It follows that any box with the same number repeated must have float zero and therefore be on the critical path.



A table free approach to critical paths

And so, the minimum time required to prepare for the dinner party is 3 hours and 20 minutes. Job done!

Note: As this method does not display the earliest finish time of each activity, we can't immediately calculate the free float. To find the free float using the diagram we must first recall the formula $\mathrm{EF}_i = \mathrm{ES}_i + \mathrm{T}_i$.

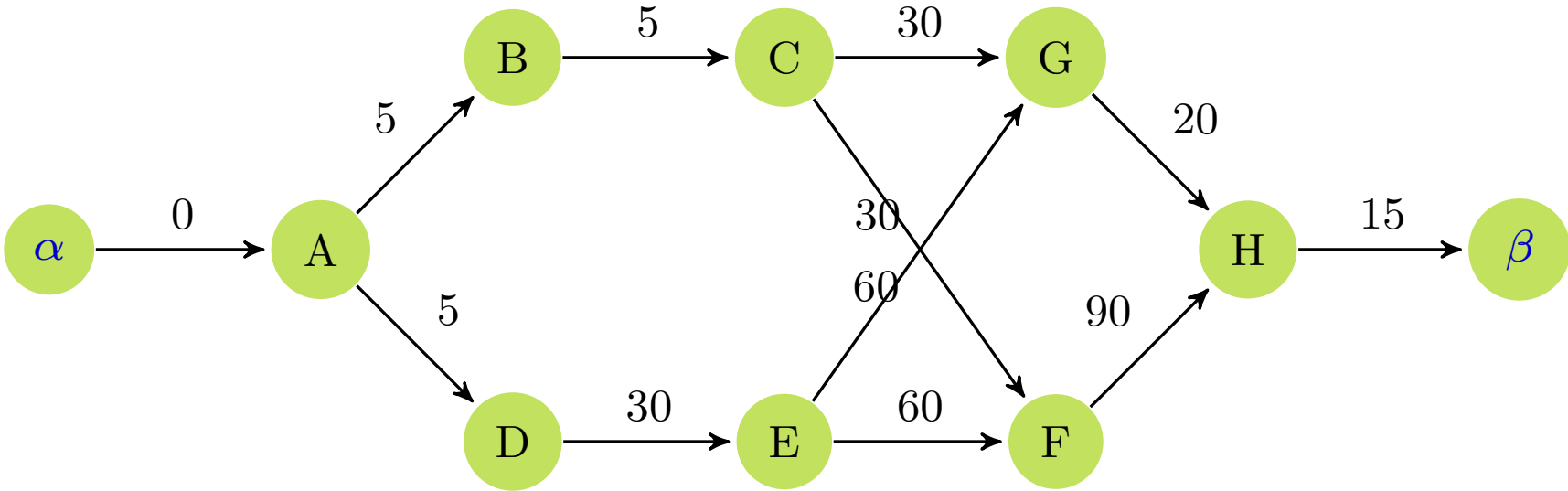
§4 – Activity-on-Arc Networks

AOA stands for activity on arc, or activity on arrow, in contrast to AON, which is activity on node. An AOA network contains the same information as an AON network, but represented differently.

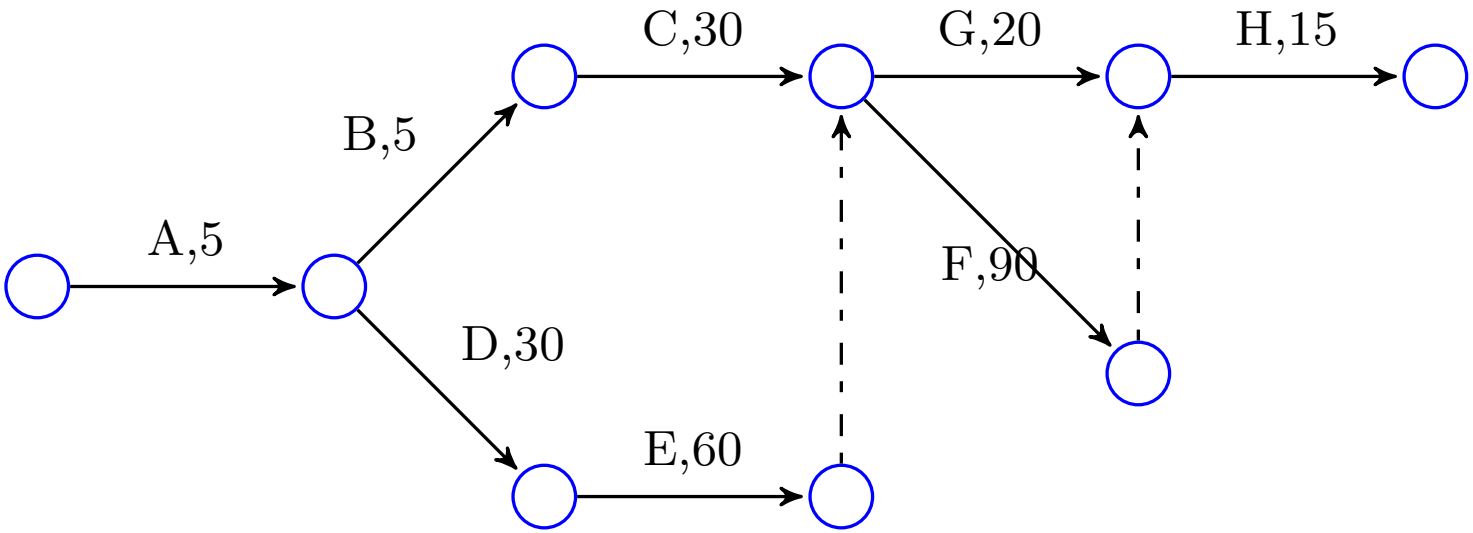
Whereas the activities in an AON network are the nodes, the activities in an AOA network are the arrows themselves. The node at the end of each arrow represents the end of the activity associated with the arrow.

Since there will typically be more precedence relations than activities, it is necessary to add in dummy arrows, which are dotted and not labelled, to indicate those precedences not otherwise indicated.

For example, consider the AON network



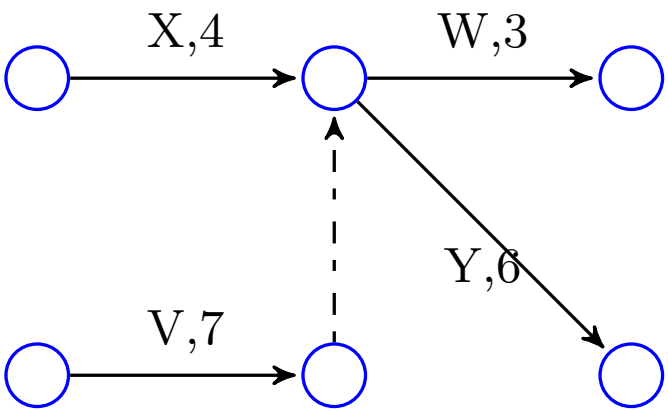
From this we construct an AOA network

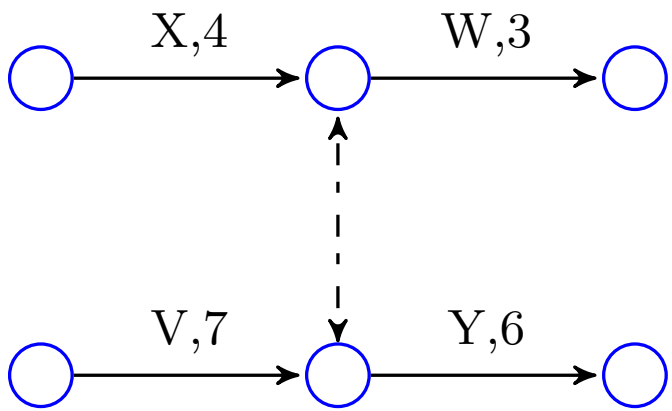


This is of no more, or less, use to us for purposes of the calculation of earliest and latest start and finish times, as both the AON and AOA networks convey the same information.

Some points to note relating to AOA networks:

- The nodes representing the ends of the activities can again be ordered according to the number of nodes connected to the left.
- No two arrows can end at the same node. This is because it would cause ambiguity about the finish time associated with that node.
- There can be situations where the direction of the dummy arrow is not immediately determined, and so can be written with two arrows:





§5 – Max-plus Matrix Algebra

The differences between AON and AOA networks reveal themselves when using techniques which make explicit use of the underlying digraph (i.e. the nodes and arrows).

One such technique is to use matrix structures, together with what is referred to as max plus algebra. As an introduction to the use of matrix algebra in studying digraphs, form the incidence matrix $M = [a_{ij}]$ with entries

$$a_{ij} = \begin{cases} 1, & (i,j) \in A \\ 0, & \text{otherwise} \end{cases}$$

where A is the set of arrows.

Notice that a_{ij} tells the number of ways to get from vertex i to vertex j using exactly 1 arrow.

From the rules of matrix multiplication

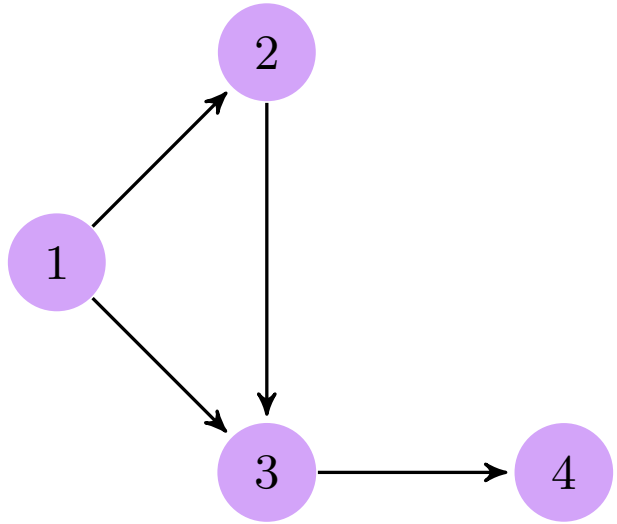
$$M^2 = [b_{ij}], \qquad b_{ij} = \sum_l a_{il} a_{lj}.$$

Observe

$$\sum_l a_{il} a_{lj} = \#\text{paths between } i \text{ and } j \text{ using 2 arrows.}$$

Similarly, the general entry (ij) of M^p is equal to # paths between i and j using exactly p arrows.

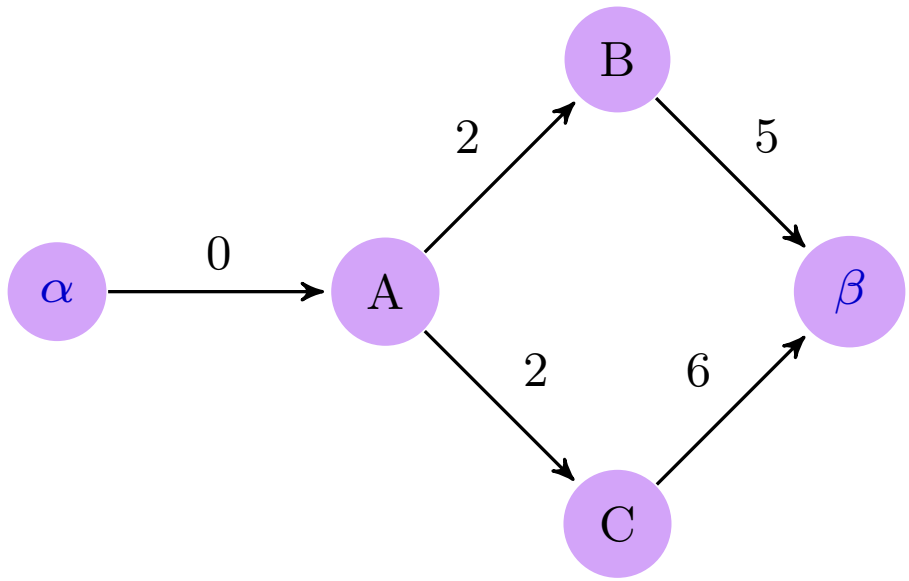
§5.1 – Example



$$M = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad M^2 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M^3 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad M^4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

In scheduling problems, we are not interested in the number of paths of a specific length (i.e. consisting of a specific number of edges), but the earliest time of completion, starting with activity i and finishing with activity j.

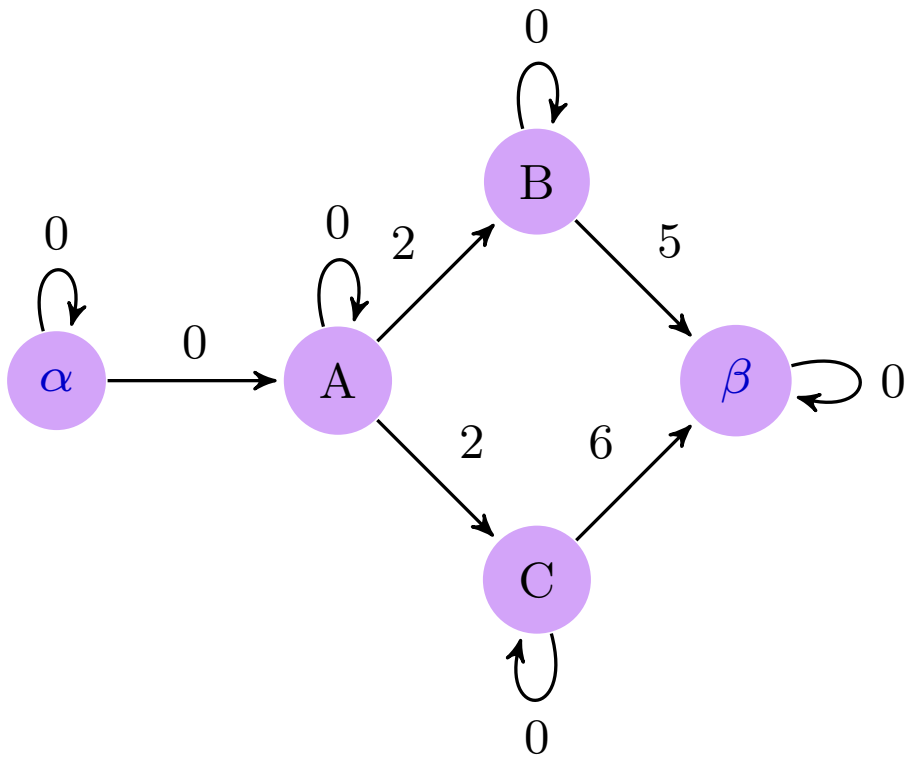


For this define a matrix Q labelled by the nodes with entries

$$q_{ij} = \begin{cases} T_{ij}, & (i,j) \in A \\ 0, & i = j \\ -\infty, & otherwise \end{cases}$$

where A is the set of arrows.

The choice $q_{ii} = 0$ means we are effectively considering the network



For this network the explicit form of the matrix Q is

$$Q = \begin{bmatrix} 0 & 0 & -\infty & -\infty & -\infty \\ -\infty & 0 & 2 & 2 & -\infty \\ -\infty & -\infty & 0 & -\infty & 5 \\ -\infty & -\infty & -\infty & 0 & 6 \\ -\infty & -\infty & -\infty & -\infty & 0 \end{bmatrix}$$

Each entry ij gives the earliest time it takes to complete activity j starting with activity i, following exactly 1 arrow. Note that if there is no such path we write $-\infty$.

Let $A = [a_{ij}]_{i,j=1,\dots,N}$ and $B = [b_{ij}]_{i,j=1,\dots,N}$. Introduce the operation, referred to as max plus

$$A \otimes B = [\max\{a_{il} + b_{lj}\}_{l=1,\dots,N}]_{i,j=1,\dots,N}.$$

For example, when $N = 2$,

$$A \otimes B = \begin{bmatrix} \max(a_{11} + b_{11}, a_{12} + b_{21}) & \max(a_{11} + b_{12}, a_{12} + b_{22}) \\ \max(a_{21} + b_{11}, a_{22} + b_{21}) & \max(a_{21} + b_{12}, a_{22} + b_{22}) \end{bmatrix}.$$

With usual matrix multiplication, and $A = [\vec{a}_j]_{j=1,\dots,N}$ and $B = [\vec{b}_j]_{j=1,\dots,N}$ the entry ij of AB is equal to

$$\vec{a}_i \cdot \vec{b}_j = \sum_{l=1}^{N} a_{il} b_{lj}.$$

Similarly, for the operation $\otimes$, the entry ij of AB is equal to

$$\vec{a}_i \otimes \vec{b}_j = \max \{a_{il} + b_{lj}\}_{l=1,\dots,N}.$$

In comparison to $\vec{a}_i \cdot \vec{b}_j$, multiplication of the elements has been replaced by $+$, and addition of the elements replaced by max.

We see that the ij element of $Q \otimes Q$ (which is written Q^2 when the use of max plus is implied) gives the earliest time to start activity j, beginning with activity i, using two arrows or less (the "or less" is accounted for by the loops

$$Q^2 = \begin{bmatrix} 0 & 0 & 2 & 2 & -\infty \\ -\infty & 0 & 2 & 2 & 8 \\ -\infty & -\infty & 0 & -\infty & 5 \\ -\infty & -\infty & -\infty & 0 & 6 \\ -\infty & -\infty & -\infty & -\infty & 0 \end{bmatrix}$$

And the elements of $Q \otimes Q \otimes Q = Q^3$ gives the earliest time to start activity j, beginning with activity i, using 3 arrows or less.

$$Q^3 = \begin{bmatrix} 0 & 0 & 2 & 2 & 8 \\ -\infty & 0 & 2 & 2 & 8 \\ -\infty & -\infty & 0 & -\infty & 5 \\ -\infty & -\infty & -\infty & 0 & 6 \\ -\infty & -\infty & -\infty & -\infty & 0 \end{bmatrix}$$

After calculating Q^4 (try this as an exercise) we notice that $Q^3 = Q^4$. This tells us that the earliest start times of all the paths occurred using 3 arrows. In particular, the earliest start time, from the beginning of the network (node α) to the finish (node β) is 8.

All the information relating to the critical path is contained in the first row of the successive powers of Q. Thus, in the example we have the sequence of first rows

$$(0\ 0\ -\infty\ -\infty\ -\infty) \qquad \text{first row of } Q$$
$$(0\ 0\ 2\ 2\ -\infty) \qquad \text{first row of } Q^2$$

The final first row $(0\,0\,2\,2\,8)$ tells us the earliest start times for each of the nodes in succession, starting from node α. From this, by working backwards in the network, we can determine which of the nodes are critical.

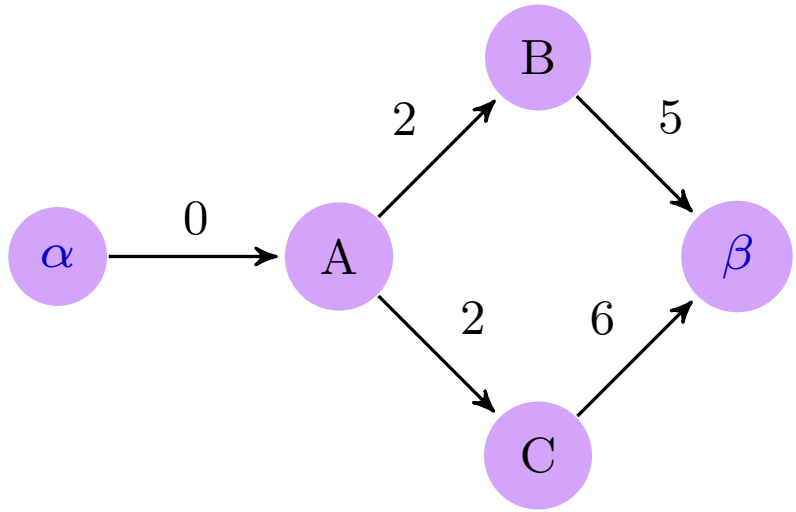
The question we have to ask is,

Does the earliest start time at vertex i' equal the earliest start time at vertex i plus the time it takes to go from i' to i, where (i', i) is an arrow?

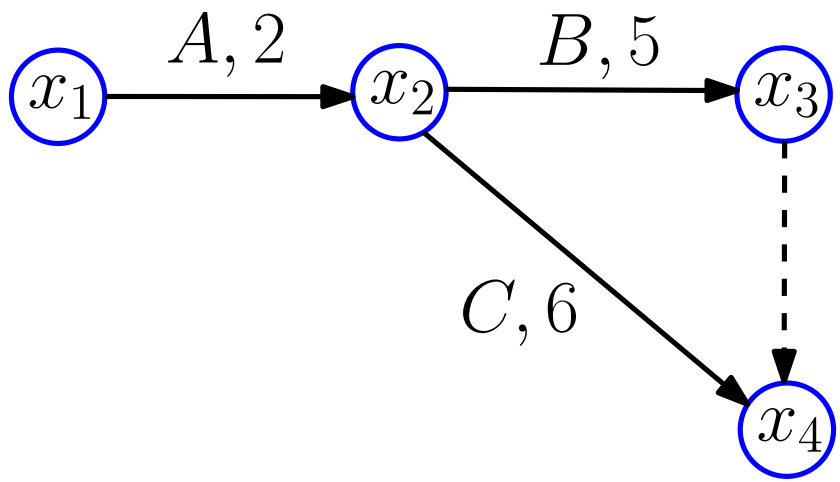
If so i' is a critical node, and we repeat. This gives the critical path $\alpha AC\beta$.

Since only the first row is of interest in the study of critical paths, we may just as well calculate $\vec{q}_0^T \otimes Q$, $\vec{q}_0^T \otimes Q^2$, $\vec{q}_0^T \otimes Q^3$, ... where $\vec{q}_0^T$ is the row vector with first entry 0 and all other entries equal to $-\infty$. This gives the successive first rows.

Let's now express the AON activity network



as an AOA activity network



Here the labels on the nodes are just for convenience; the activities are on the arrows.

The appropriate form of the matrix Q is

$$Q = \begin{bmatrix} 0 & 2 & -\infty & -\infty \\ -\infty & 0 & 5 & 6 \\ -\infty & -\infty & 0 & 0 \\ -\infty & -\infty & -\infty & 0 \end{bmatrix}$$

Notice that the dotted (dummy) arrow x_3 to x_4 is given a time of completion 0.

$$\begin{bmatrix} 0 & -\infty & -\infty & -\infty \end{bmatrix} \otimes Q = \begin{bmatrix} 0 & 2 & -\infty & -\infty \end{bmatrix}$$

$$\begin{aligned} \begin{bmatrix} 0 & -\infty & -\infty & -\infty \end{bmatrix} \otimes Q^2 &= \begin{bmatrix} 0 & 2 & -\infty & -\infty \end{bmatrix} \otimes Q \\ &= \begin{bmatrix} 0 & 2 & 7 & 8 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \begin{bmatrix} 0 & -\infty & -\infty & -\infty \end{bmatrix} \otimes Q^3 &= \begin{bmatrix} 0 & 2 & 7 & 8 \end{bmatrix} \otimes Q \\ &= \begin{bmatrix} 0 & 2 & 7 & 8 \end{bmatrix} \end{aligned}$$

Here the labels on the nodes correspond to the finish of particular activities. So the earliest finish times when beginning at vertex x_1 and arriving at the other vertices is given by $[0\ 2\ 7\ 8]$.

Backtracking through the vertices with this knowledge reveals that the critical path consists of the vertices x_1, x_2, x_4, which in turn corresponds to the activities A and C. Recall: in AOA networks it is not necessary to include α, β.