

MAST20018

Discrete Maths and Operations Research: Discrete Mathematics

Semester 2

Topics

1. Scheduling
2. Network optimisation
3. Social choice

Scheduling

Motivating example

We motivate our first topic of networks by considering **how to plan a dinner party**, from a mathematical viewpoint.

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?

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 be very efficient, and when used in planning stages helps the decision making process.

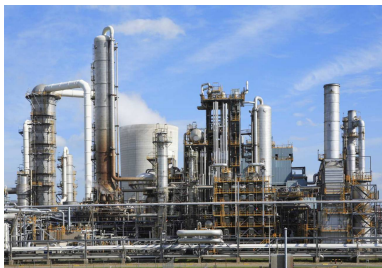
Scheduling with a precedence ordering

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.

Introduction to activity networks

Our first task is to represent the activities, their precedences and their completion times in a diagrammatic form.

For this, use will be made of **directed graphs**, which are collections of nodes and directed edges (called arcs). The term directed graph is often abbreviated **digraph**.

The digraphs used for activity scheduling have special properties and are referred to as a **activity networks**.

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

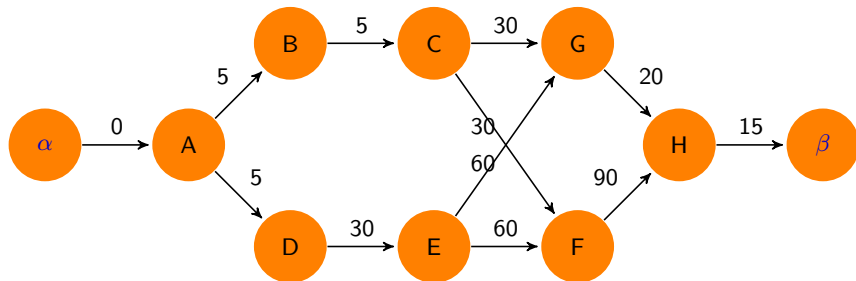
Activity networks

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

	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:



Activity on node networks

The above graph is called an **activity on node network** (or AON network).

It has 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 are called **artificial nodes**. The node α connects to all nodes with no precedence and thus has no in-coming arcs, while β connects to all nodes with no out-going arcs.

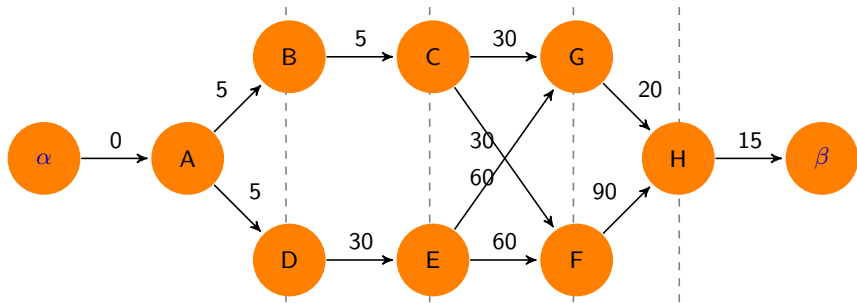
Activity on node networks

The arced edges, referred to as **arcs**, represent (immediate) precedence. We say immediate precedence because if $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$, which technically says that \rightarrow is **transitive**.

The tasks can always be ordered so that the arcs 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 arcs), then $X \rightarrow Y$.

Drawing an AONNetwork

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**.

An algorithm for drawing AON networks

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 let the number of precedences for each of these activities be k .
3. If there are no activities in the list, stop.
4. Otherwise $k = k + 1$ and go to step 2.

Example

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

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 in the list and therefore have $k = 1$ precedences in total. We delete them from the list to obtain. . .

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

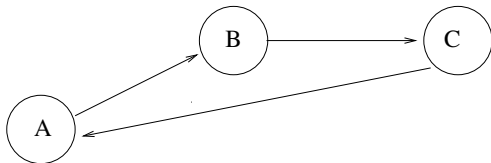
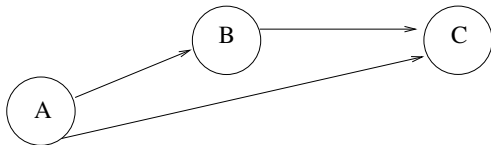
Now Activities C and E have no precedences in the list and therefore have $k = 2$ precedences in total. We delete them from the list to obtain

Activity	Precedence
F.	—
G.	—
H.	F, G

Now Activities F and G have no precedences in the list and therefore have $k = 3$ precedences in total. When we delete them only H remains and therefore Activity H has $k = 4$ precedences and we stop.

True or false?

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

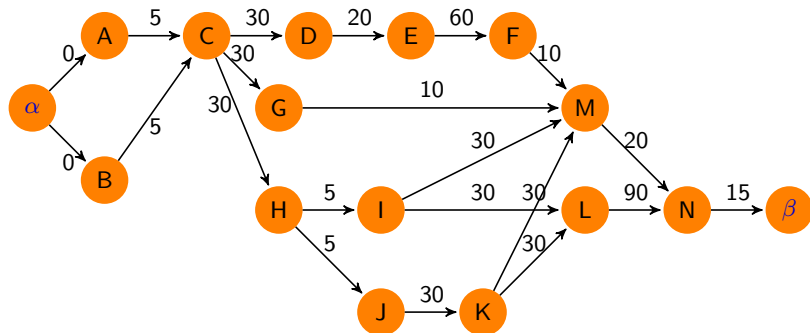


Explain.

An extended example

	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

Exercise



Exercise: Use the algorithm to reproduce the AON activity network above.

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 the same time**, 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 arcs, 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.

Critical paths

Features of interest with respect to the minimum time of completion are:

- 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, $ES_\alpha = 0$.

Calculation of earliest times

- $EF_i = ES_i + T_i$
- $ES_i = \max \{EF_{i^*}\}$, 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 $ES_\alpha = 0$, these earliest times can be computed successively across the network, reading from left to right.

Calculation of latest times

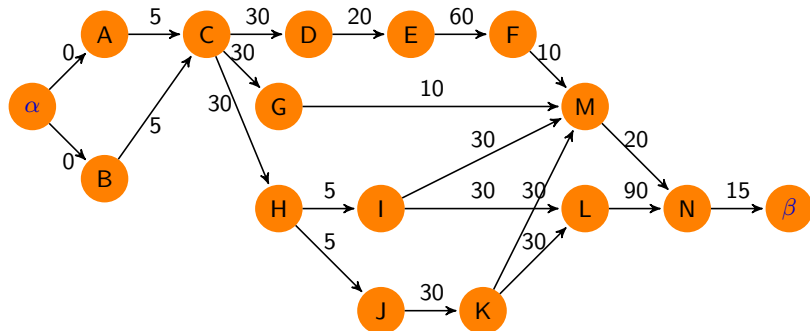
We have $EF_\beta = LF_\beta$, giving us an initial condition at the very right of the network.

- $LS_i = LF_i - T_i$, since each activity must start T_i units of time before it ends.
- $LF_i = \min \{LS_{i^\#}\}$, 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 $LF_\beta = EF_\beta$, 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, $LS_\alpha = 0$.

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 $LF_\beta = EF_\beta = 205$, 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	$\min \{70, 40\} = 40$	$40 - 5 = 35$
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 $EF_i = LF_i$ (it must then be that $LS_i = ES_i$). We read off from our table that the critical activities are

α , A, B, C, H, J, K, L, N, β

Hence there are two critical paths:

$\alpha \rightarrow A \rightarrow C \rightarrow H \rightarrow J \rightarrow K \rightarrow L \rightarrow N \rightarrow \beta$

$\alpha \rightarrow B \rightarrow C \rightarrow H \rightarrow J \rightarrow K \rightarrow L \rightarrow N \rightarrow \beta$

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

$$TF_i = LS_i - ES_i$$

Note that it can equivalently be written

$$TF_i = LF_i - EF_i \quad \text{or} \quad TF_i = LF_i - ES_i - T_i$$

So it is equal to the difference between the maximum time available to complete the activity, $LF_i - ES_i$, 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 least amount of time.

Free float

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

$$FF_i = \min \{ES_{i^\#} - EF_i\}$$

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-arc.

Note too that $FF_i \leq TF_i$.

Exercises

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

Do these results support the claim that $FF_i \leq TF_i$?

Max-plus matrix algebra

As an introduction to the use of **matrix algebra** in studying digraphs, we 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 arc.

The adjacency matrix can give us insight into the **structure** of a graph, and can act as a **data structure** for graph algorithms.

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

From the rules of matrix multiplication

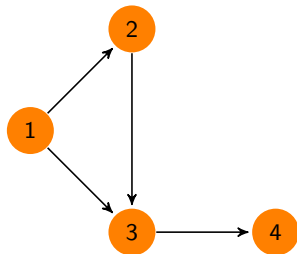
$$M^2 = [b_{ij}], \quad b_{ij} = \sum_l a_{il}a_{lj}.$$

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 arcs.

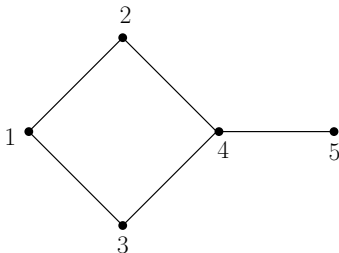
Example



$$M = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad M^2 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
$$M^3 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad M^4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The adjacency matrix

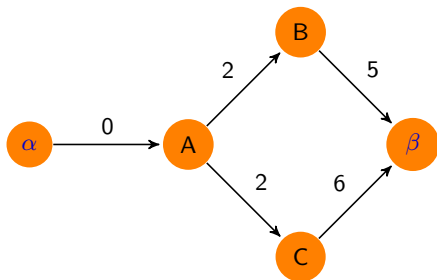
Incidence matrices can also be calculated for **undirected graphs**. In the example below, explain the significance of the entries on the main diagonal of M^2 .



$$M = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$M^2 = \begin{bmatrix} 2 & 0 & 0 & 2 & 0 \\ 0 & 2 & 2 & 0 & 1 \\ 0 & 2 & 2 & 0 & 1 \\ 2 & 0 & 0 & 3 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

In scheduling problems, we are not interested in the number of paths of a specific length (i.e. consisting of a specific number of arcs), 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

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

where A is the set of arcs.

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 arc. 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 arcs 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 arcs 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 arcs.

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

$$\begin{array}{ll} (0 \ 0 \ -\infty \ -\infty \ -\infty) & \text{first row of } Q \\ (0 \ 0 \ 2 \ 2 \ -\infty) & \text{first row of } Q^2 \end{array}$$

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 ask is: does the 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 arc? If so then 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 $\vec{q}_0^T \otimes Q, \vec{q}_0^T \otimes Q^2, \vec{q}_0^T \otimes Q^3, \dots$ 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.