

V3.0

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ  
*ΕΝΙΣΧΥΤΙΚΗ ΔΙΔΑΣΚΑΛΙΑ 2024-2025*

ΧΟΝΔΡΟΜΑΤΙΔΗΣ ΕΛΕΥΘΕΡΙΟΣ

Ο Αλγόριθμος του Τραπεζίτη



TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
—	—	—

Ελέγξτε αν η  
κατάσταση είναι  
ασφαλής και  
συμπληρώστε τα κενά.

MAX

	A	B	C
P0	0	1	1
P1	6	5	6
P2	4	0	2
P3	0	4	6
P4	5	2	1

ALLOCATED

	A	B	C
P0	0	0	1
P1	2	0	4
P2	2	0	2
P3	0	4	1
P4	1	2	0

NEED

	A	B	C
P0	—	—	—
P1	—	—	—
P2	—	—	—
P3	—	—	—
P4	—	—	—



ΛΥΣΗ



TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
—	—	—

Για να υπολογίσω τον πίνακα  
**NEED** κάνω: **MAX - ALLOCATED**  
κελί προς κελί.

MAX

	A	B	C
P0	0	1	1
P1	6	5	6
P2	4	0	2
P3	0	4	6
P4	5	2	1

ALLOCATED

	A	B	C
P0	0	0	1
P1	2	0	4
P2	2	0	2
P3	0	4	1
P4	1	2	0

NEED

	A	B	C
P0	0	1	0
P1	4	5	2
P2	2	0	0
P3	0	0	5
P4	4	0	1



TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
6	2	2

Για να υπολογίσω αρχικά το AVAILABLE, κάνω:

TOTAL - (Άθροισμα) ALLOCATED

(κατα στήλη, κελί προς κελί)

Π.χ στον πόρο A έκανα:  $11 - (0+2+2+0+1) = 6$

MAX

	A	B	C
P0	0	1	1
P1	6	5	6
P2	4	0	2
P3	0	4	6
P4	5	2	1

ALLOCATED

	A	B	C
P0	0	0	1
P1	2	0	4
P2	2	0	2
P3	0	4	1
P4	1	2	0

NEED

	A	B	C
P0	0	1	0
P1	4	5	2
P2	2	0	0
P3	0	0	5
P4	4	0	1



## TOTAL

A	B	C
11	8	10

## AVAILABLE

A	B	C
6	2	2

Τώρα, θα πρέπει να κοιτάω τον πίνακα NEED, να περάσω από όλες τις διεργασίες μια προς μια και να ελέγχω για το αν το τρέχων AVAILABLE αρκεί για την κάλυψη των πόρων. Δηλαδή, αν ΚΑΘΕ πόρος του Available (A,B,C) έχει τιμή  $\geq$  του αντίστοιχου πόρου στο NEED.

Αν αρκεί, εκτελώ τη διεργασία, διαφορετικά, την αγνοώ και πάω στην επόμενη.

Μόλις εκτελεστεί μια διεργασία, πάω ξανά στη πρώτη γραμμή του πίνακα και κατεβαίνω ξανά.

Αυτό το κάνω κάθε φορά που μια διεργασία εκτελείται.

## MAX

	A	B	C
P0	0	1	1
P1	6	5	6
P2	4	0	2
P3	0	4	6
P4	5	2	1

## ALLOCATED

	A	B	C
P0	0	0	1
P1	2	0	4
P2	2	0	2
P3	0	4	1
P4	1	2	0

## NEED

	A	B	C
P0	0	1	0
P1	4	5	2
P2	2	0	0
P3	0	0	5
P4	4	0	1



TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
6	2	2

Για να ΑΝΑΝΕΩΝΩ το AVAILABLE  
KANΩ:  
ALLOCATED + (τρέχων) AVAILABLE

MAX

	A	B	C
P0	0	1	1
P1	6	5	6
P2	4	0	2
P3	0	4	6
P4	5	2	1

ALLOCATED

	A	B	C
P0	0	0	1
P1	2	0	4
P2	2	0	2
P3	0	4	1
P4	1	2	0

NEED

	A	B	C
P0	0	1	0
P1	4	5	2
P2	2	0	0
P3	0	0	5
P4	4	0	1



# ΕΚΤΕΛΕΣΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ ΤΟΥ ΤΡΑΠΕΖΙΤΗ





TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
6	2	2

Στον Αλγόριθμο του  
Τραπεζίτη, πάμε ΠΑΝΤΑ με  
τη σειρά.

Δηλαδή από τη διεργασία  
P0 στην P4.

MAX

	A	B	C
P0	0	1	1
P1	6	5	6
P2	4	0	2
P3	0	4	6
P4	5	2	1

ALLOCATED

	A	B	C
P0	0	0	1
P1	2	0	4
P2	2	0	2
P3	0	4	1
P4	1	2	0

NEED

	A	B	C
P0	0	1	0
P1	4	5	2
P2	2	0	0
P3	0	0	5
P4	4	0	1



TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
6	2	<u>2</u>

Ξεκινώντας με το να ελέγξουμε τη πρώτη γραμμή, τη διεργασία P0 δηλαδή: (6, 2, 2) το AVAILABLE και (0, 1, 0) το NEED.

$6 \geq 0$ ,  $2 \geq 1$  και  $2 \geq 0$ .

Η P0 μπορεί να εκτελεστεί!

Άρα, προσθέτουμε το ALLOCATED της P0 στο τρέχων AVAILABLE.  $1+2=3$

MAX

	A	B	C
P0	0	1	1
P1	6	5	6
P2	4	0	2
P3	0	4	6
P4	5	2	1

ALLOCATED

	A	B	C
P0	0	0	<u>1</u>
P1	2	0	4
P2	2	0	2
P3	0	4	1
P4	1	2	0

NEED

	A	B	C
<u>P0</u>	<u>0</u>	<u>1</u>	<u>0</u>
P1	4	5	2
P2	2	0	0
P3	0	0	5
P4	4	0	1



TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
6	2	3

Το AVAILABLE ανανεώθηκε!  
Από (6, 2, 2) έγινε (6, 2, 3).

Επιπλέον, στο MAX, στο ALLOCATED  
και στο NEED μηδενίζονται όλες οι  
τιμές στη γραμμή της P0.

MAX

	A	B	C
P0	0	0	0
P1	6	5	6
P2	4	0	2
P3	0	4	6
P4	5	2	1

ALLOCATED

	A	B	C
P0	0	0	0
P1	2	0	4
P2	2	0	2
P3	0	4	1
P4	1	2	0

NEED

	A	B	C
<u>P0</u>	<u>0</u>	<u>0</u>	<u>0</u>
P1	4	5	2
P2	2	0	0
P3	0	0	5
P4	4	0	1



Λίστα διεργασιών που εκτελέσαμε: P0



TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
6	2	3

Η διεργασία P1, δεν εκτελείται ακόμα!

Αυτό συμβαίνει, επειδή έχοντας (6, 2, 3) στο τρέχων AVAILABLE και (4, 5, 2) στο NEED ισχύει:  
 $6 \geq 4$ , αλλά όχι  $2 \geq 5$

Άρα, αγνοώ την P1 για τώρα, και πάω να ελέγξω τη διεργασία P2.

Ο αλγόριθμος βρίσκεται εδώ

MAX

ALLOCATED

NEED

	A	B	C
P0	0	0	0
P1	6	5	6
P2	4	0	2
P3	0	4	6
P4	5	2	1

	A	B	C
P0	0	0	0
P1	2	0	4
P2	2	0	2
P3	0	4	1
P4	1	2	0

	A	B	C
P0	0	0	0
P1	4	5	2
P2	2	0	0
P3	0	0	5
P4	4	0	1



Λίστα διεργασιών που εκτελέσαμε: P0



TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
<u>6</u>	2	<u>3</u>

Ας ελέγξουμε τη P2:

(6, 2, 3) το AVAILABLE, (2, 0, 0) το NEED.

$6 \geq 2$ ,  $2 \geq 0$  και  $3 \geq 0$ .

Άρα, η P2 μπορεί να εκτελεστεί!

Υπενθύμιση:

νέο AVAILABLE = τρέχων AVAILABLE + ALLOCATED

MAX

	A	B	C
P0	0	0	0
P1	6	5	6
P2	4	0	2
P3	0	4	6
P4	5	2	1

ALLOCATED

	A	B	C
P0	0	0	0
P1	2	0	4
P2	<u>2</u>	0	<u>2</u>
P3	0	4	1
P4	1	2	0

NEED

	A	B	C
P0	0	0	0
P1	4	5	2
<u>P2</u>	<u>2</u>	<u>0</u>	<u>0</u>
P3	0	0	5
P4	4	0	1



Λίστα διεργασιών που εκτελέσαμε: P0



TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
8	2	5

Το AVAILABLE ανανεώθηκε, και όλες οι τιμές στη γραμμή της P2 στο MAX, το ALLOCATED και το NEED μηδενίστηκαν!

MAX

	A	B	C
P0	0	0	0
P1	6	5	6
P2	0	0	0
P3	0	4	6
P4	5	2	1

ALLOCATED

	A	B	C
P0	0	0	0
P1	2	0	4
P2	0	0	0
P3	0	4	1
P4	1	2	0

NEED

	A	B	C
P0	0	0	0
P1	4	5	2
<u>P2</u>	<u>0</u>	<u>0</u>	<u>0</u>
P3	0	0	5
P4	4	0	1



Λίστα διεργασιών που εκτελέσαμε: P0, P2

TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
8	2	5

Αφού η P3 εκτελέστηκε, ο αλγόριθμος θα κάνει “reset”, δηλαδή θα πάει ξανά στη πρώτη γραμμή και θα ελέγξει τις διεργασίες μια προς μια.

Ο αλγόριθμος βρίσκεται εδώ

MAX

ALLOCATED

NEED

	A	B	C
P0	0	0	0
P1	6	5	6
P2	0	0	0
P3	0	4	6
P4	5	2	1

	A	B	C
P0	0	0	0
P1	2	0	4
P2	0	0	0
P3	0	4	1
P4	1	2	0

	A	B	C
P0	0	0	0
P1	4	5	2
P2	0	0	0
P3	0	0	5
P4	4	0	1



Λίστα διεργασιών που εκτελέσαμε: P0, P2



TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
8	2	5


Η P1 δεν μπορεί να εκτελεστεί  
ακόμα, διότι έχουμε (8, 2, 5) στο  
AVAILABLE και (4, 5, 2) στο NEED.

8>=2, αλλά όχι 5>2.

Άρα, πάμε στη διεργασία P2 και εκ  
των υστέρων στη P3.

Ο αλγόριθμος βρίσκεται εδώ

MAX



	A	B	C
P0	0	0	0
P1	6	5	6
P2	0	0	0
P3	0	4	6
P4	5	2	1

ALLOCATED

	A	B	C
P0	0	0	0
P1	2	0	4
P2	0	0	0
P3	0	4	1
P4	1	2	0

NEED

	A	B	C
P0	0	0	0
P1	4	5	2
P2	0	0	0
P3	0	0	5
P4	4	0	1



Λίστα διεργασιών που εκτελέσαμε: P0, P2





TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
8	2	5

Η P2 έχει ήδη ολοκληρώσει την εκτέλεση της...

Ο αλγόριθμος βρίσκεται εδώ

MAX

	A	B	C
P0	0	0	0
P1	6	5	6
P2	0	0	0
P3	0	4	6
P4	5	2	1

ALLOCATED

	A	B	C
P0	0	0	0
P1	2	0	4
P2	0	0	0
P3	0	4	1
P4	1	2	0

NEED

	A	B	C
P0	0	0	0
P1	4	5	2
P2	0	0	0
P3	0	0	5
P4	4	0	1



Λίστα διεργασιών που εκτελέσαμε: P0, P2



TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
8	<u>2</u>	<u>5</u>

Φτάσαμε στη P3!

Ας ελέγξουμε αν μπορεί να εκτελεστεί:  
 AVAILABLE (8, 2, 5) και NEED (0, 0, 5).

$8 \geq 0$ ,  $2 \geq 0$  και  $5 \geq 5$ .

Όλες οι συνθήκες πληρούνται, άρα η  
 P3 μπορεί να εκτελεστεί.

MAX

	A	B	C
P0	0	0	0
P1	6	5	6
P2	0	0	0
P3	0	4	6
P4	5	2	1

ALLOCATED

	A	B	C
P0	0	0	0
P1	2	0	4
P2	0	0	0
P3	0	<u>4</u>	<u>1</u>
P4	1	2	0

NEED

	A	B	C
P0	0	0	0
P1	4	5	2
P2	0	0	0
<u>P3</u>	<u>0</u>	<u>0</u>	<u>5</u>
P4	4	0	1



Λίστα διεργασιών που εκτελέσαμε: P0, P2

TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
8	6	6

Φτάσαμε στη P3!

Ας ελέγξουμε αν μπορεί να εκτελεστεί:  
 AVAILABLE (8, 2, 5) και NEED (0, 0, 5).

$8 \geq 0$ ,  $2 \geq 0$  και  $5 \geq 5$ .

Όλες οι συνθήκες πληρούνται, άρα η  
 P3 μπορεί να εκτελεστεί.

MAX

	A	B	C
P0	0	0	0
P1	6	5	6
P2	0	0	0
P3	0	0	0
P4	5	2	1

ALLOCATED

	A	B	C
P0	0	0	0
P1	2	0	4
P2	0	0	0
P3	0	0	0
P4	1	2	0

NEED

	A	B	C
P0	0	0	0
P1	4	5	2
P2	0	0	0
<u>P3</u>	<u>0</u>	<u>0</u>	<u>0</u>
P4	4	0	1



Λίστα διεργασιών που εκτελέσαμε: P0, P2, P3

TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
8	6	6

Αφού κάποια διεργασία εκτελέστηκε, ο αλγόριθμος θα πάει πάλι στη πρώτη γραμμή.

Η P0 έχει ήδη εκτελεστεί, οπότε πάμε να εξετάσουμε την P1.

Ο αλγόριθμος βρίσκεται εδώ

MAX

	A	B	C
P0	0	0	0
P1	6	5	6
P2	0	0	0
P3	0	0	0
P4	5	2	1

ALLOCATED

	A	B	C
P0	0	0	0
P1	2	0	4
P2	0	0	0
P3	0	0	0
P4	1	2	0

NEED

	A	B	C
P0	0	0	0
P1	4	5	2
P2	0	0	0
P3	0	0	0
P4	4	0	1



Λίστα διεργασιών που εκτελέσαμε: P0, P2, P3



## TOTAL

A	B	C
11	8	10

## AVAILABLE

A	B	C
<u>8</u>	6	<u>6</u>

Είμαστε ξανά στη P1.

Πάμε να την ελέγξουμε:

(8, 6, 6) το AVAILABLE και (4, 5, 2) το NEED.

$8 \geq 4$ ,  $6 \geq 5$  και  $6 \geq 2$ .

Όλες οι συνθήκες είναι αληθείς, η P1 μπορεί να εκτελεστεί.

## MAX

	A	B	C
P0	0	0	0
P1	6	5	6
P2	0	0	0
P3	0	0	0
P4	5	2	1

## ALLOCATED

	A	B	C
P0	0	0	0
P1	<u>2</u>	0	<u>4</u>
P2	0	0	0
P3	0	0	0
P4	1	2	0

## NEED

	A	B	C
P0	0	0	0
<u>P1</u>	<u>4</u>	<u>5</u>	<u>2</u>
P2	0	0	0
P3	0	0	0
P4	4	0	1



Λίστα διεργασιών που εκτελέσαμε: P0, P2, P3

## TOTAL

A	B	C
11	8	10

## AVAILABLE

A	B	C
10	6	10

Είμαστε ξανά στη P1.

Πάμε να την ελέγξουμε:

(8, 6, 6) το AVAILABLE και (4, 5, 2) το NEED.

$8 \geq 4$ ,  $6 \geq 5$  και  $6 \geq 2$ .

Όλες οι συνθήκες είναι αληθείς, η P1 μπορεί να εκτελεστεί.

## MAX

	A	B	C
P0	0	0	0
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	5	2	1

## ALLOCATED

	A	B	C
P0	0	0	0
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	1	2	0

## NEED

	A	B	C
P0	0	0	0
<u>P1</u>	<u>0</u>	<u>0</u>	<u>0</u>
P2	0	0	0
P3	0	0	0
P4	4	0	1



Λίστα διεργασιών που εκτελέσαμε: P0, P2, P3, P1

## TOTAL

A	B	C
11	8	10

## AVAILABLE

A	B	C
10	6	10

Ο αλγόριθμος του τραπεζίτη όπως και τις άλλες φορές, αφού εκτελέστηκε μια διεργασία θα πάει ξανά από την αρχή.

Οι διεργασίες P0, P1, P2, P3 έχουν σημειωθεί ως “ήδη εκτελεσμένες”, άρα προχωράμε μέχρι να βρούμε διεργασία που δεν έχει εκτελεστεί, δηλαδή τη P4.

## MAX

	A	B	C
P0	0	0	0
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	5	2	1

## ALLOCATED

	A	B	C
P0	0	0	0
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	1	2	0

## NEED

	A	B	C
P0	0	0	0
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	4	0	1



Λίστα διεργασιών που εκτελέσαμε: P0, P2, P3, P1

## TOTAL

A	B	C
11	8	10

## AVAILABLE

A	B	C
<u>10</u>	<u>6</u>	10

Και τέλος, φτάνουμε στη διεργασία P4.

Ας ελέγξουμε αν μπορεί να εκτελεστεί:  
(10, 6, 10) το AVAILABLE, (4, 0, 1) το NEED  
 $10 \geq 4$ ,  $6 \geq 0$  και  $10 \geq 1$

Πληρούνται όλες οι συνθήκες, άρα μπορεί  
και η P4 να εκτελεστεί!

## MAX

	A	B	C
P0	0	0	0
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	5	2	1

## ALLOCATED

	A	B	C
P0	0	0	0
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	<u>1</u>	<u>2</u>	0

## NEED

	A	B	C
P0	0	0	0
P1	0	0	0
P2	0	0	0
P3	0	0	0
<u>P4</u>	<u>4</u>	<u>0</u>	<u>1</u>



Λίστα διεργασιών που εκτελέσαμε: P0, P2, P3, P1



## TOTAL

A	B	C
11	8	10

## AVAILABLE

A	B	C
11	8	10

Και τέλος, φτάνουμε στη διεργασία P4.

Ας ελέγξουμε αν μπορεί να εκτελεστεί:  
 (10, 6, 10) το AVAILABLE, (4, 0, 1) το NEED  
 $10 \geq 4$ ,  $6 \geq 0$  και  $10 \geq 1$

Πληρούνται όλες οι συνθήκες, άρα μπορεί  
και η P4 να εκτελεστεί!

## MAX

	A	B	C
P0	0	0	0
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	0	0	0

## ALLOCATED

	A	B	C
P0	0	0	0
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	0	0	0

## NEED

	A	B	C
P0	0	0	0
P1	0	0	0
P2	0	0	0
P3	0	0	0
<u>P4</u>	<u>0</u>	<u>0</u>	<u>0</u>



Λίστα διεργασιών που εκτελέσαμε: P0, P2, P3, P1, P4



TOTAL

A	B	C
11	8	10

AVAILABLE

A	B	C
11	8	10

Tip

Για να ελέγξετε πως όντως τα  
κάνετε όλα τα βήματα σωστά, να  
κοιτάτε το εξής:

AVAILABLE == TOTAL.

MAX

	A	B	C
P0	0	0	0
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	0	0	0

ALLOCATED

	A	B	C
P0	0	0	0
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	0	0	0

NEED

	A	B	C
P0	0	0	0
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	0	0	0



Σειρά Εκτέλεσης Διεργασιών: P0 -> P2 -> P3 -> P1 -> P4



# Ο ΑΛΓΟΡΙΘΜΟΣ ΤΟΥ ΤΡΑΠΕΖΙΤΗ ΟΛΟΚΛΗΡΩΘΗΚΕ ΜΕ ΕΠΙΤΥΧΙΑ!

Σειρά Εκτέλεσης Διεργασιών: P0 -> P2 -> P3 -> P1 -> P4

Αφού όλες οι διεργασίες (P0, P1, P2, P3, P4) εκτελέστηκαν, καταλήγουμε στο συμπέρασμα πως ΔΕΝ ΕΧΟΥΜΕ αδιέξοδο!

Αν όμως έστω και μια διεργασία δεν ήταν δυνατόν ΠΟΤΕ να εκτελεστεί, θα βρισκόμασταν σε κατάσταση ΑΔΙΕΞΟΔΟΥ!



## Άλλο παράδειγμα! (Θα λυθεί στο μάθημα)

⌋

Available  
A B C D  
1 5 2 0

	Max	Allocation	Need
	A B C D	A B C D	A B C D
P0	0 0 1 2	0 0 1 2	P0
P1	1 7 5 0	1 0 0 0	P1
P2	2 3 5 6	1 3 5 4	P2
P3	0 6 5 2	0 6 3 2	P3
P4	0 6 5 6	0 0 1 4	P4



**ΤΕΛΟΣ**  
**ΕΥΧΑΡΙΣΤΩ ΠΟΛΥ!**

