Αναδρομικός υπολογισμός (1)

- **Α**κολουθία a_1 , a_2 , a_3 , …., a_n

$a_n = 2 \times \max \{a_{n/2} , a_{n/2+1} \} + a_{n/2-1}$

με τιμές: $a_1 = 1$, $a_2 = 1$ και $a_3 = 1$.

(Θεωρήστε ότι *i/2* = το κάτω ακέραιο μέρος του ρητού *i/2*)

Πολυπλοκότητα ?

Αναδρομικός υπολογισμός (2)

- Ακολουθία b_1 , b_2 , b_3 , …., b_n που προκύπτει από τον αναδρομικό τύπο

$b_n = 2 \times \max \{b_{n-1}, b_{n-2}\} + b_{n-3}$

με αρχικές τιμές: $b_1 = 1$, $b_2 = 1$ και $b_3 = 1$.

Πολυπλοκότητα ?

Δυναμικός Προγραμματισμός (DP)

- Ποιος αλγόριθμος είναι ταχύτερος;
- Να υπολογιστούν με δυναμικό προγραμματισμό. Μπορεί να γίνει αυτό σε γραμμικό χρόνο και χώρο;
- Μπορεί για κάποιον αλγόριθμο δυναμικού προγραμματισμού από τους παραπάνω δύο να γίνει σταθερός ο χώρος;

Αναδρομικός υπολογισμός (1)

- **Α**κολουθία a_1 , a_2 , a_3 , …., a_n

$a_n = 2 \times \max \{a_{n/2} , a_{n/2+1} \} + a_{n/2-1}$

με τιμές: $a_1 = 1$, $a_2 = 1$ και $a_3 = 1$.

RA:

return 2×max{RA(n/2), RA(n/2+1)} + RA(n/2-1)

Αναδρομικός υπολογισμός (1)

RA:

return 2×max{RA(n/2), RA(n/2+1)} + RA(n/2-1)

- Χρόνος πολυωνυμικός: $O(n^{1.585})$

Αναδρομικός υπολογισμός (2)

- Ακολουθία b_1 , b_2 , b_3 , …., b_n που προκύπτει από τον αναδρομικό τύπο

$b_n = 2 \times \max \{b_{n-1}, b_{n-2}\} + b_{n-3}$

με αρχικές τιμές: $b_1 = 1$, $b_2 = 1$ και $b_3 = 1$.

RB:

return 2×max{RB(n-1), RB(n-2)} + RB(n-3)

Αναδρομικός υπολογισμός (2)

RB:

return 2×max{RB(n-1), RB(n-2)} + RB(n-3)

$\Omega(1.44^n) = \omega(n^{1.585})$

Δυναμικός Προγραμματισμός (1)

- θα κατασκευάσουμε τον πίνακα Α, όπου στις 3 πρώτες θέσεις 0, 1, 2 θα αποθηκεύσουμε τις αρχικές τιμές

A[0]= $\mathbf{a_1 = 1}$, A[1]= $\mathbf{a_2 = 1}$, A[2]= $\mathbf{a_3 = 1}$

A[i]= 2×max{A[i/2], A[i/2+1]}+A[i/2-1]

γραμμικός χρόνος και γραμμικός χώρος.

Δυναμικός Προγραμματισμός (2)

- θα κατασκευάσουμε τον πίνακα Β, όπου στις 3 πρώτες θέσεις 0, 1, 2 θα αποθηκεύσουμε τις αρχικές τιμές

Β[0]= $\mathbf{b_1 = 1}$, Β[1]= $\mathbf{b_2 = 1}$, Β[2]= $\mathbf{b_3 = 1}$

- **B[i]= 2×max{B[i-1], B[i-2]}+B[i-3]**

γραμμικός χρόνος και γραμμικός χώρος.

Δυναμικός Προγραμματισμός (2)

- Ο χώρος για τον πίνακα Β εύκολα μειώνεται σε 3 μόνο κελιά. Αρχικά, ας παρατηρήσουμε ότι χρειαζόμαστε 3 κελιά για τις αρχικές τιμές, δλδ στο κελί 0 γράφει την αρχική τιμή $\mathbf{b_1 = 1,}$ στο κελί 1 γράφει την $\mathbf{b_2}$ **,** στο κελί 2 γράφει την $\mathbf{b_3 = 1}$.
- Για κάθε i> 3, κελί επανεγγραφής της $\mathbf{b_i}$ προσδιορίζεται ως το **B[mod(i-1, 3)]**.

Δυναμικός Προγραμματισμός

Αποτελεί μία υπολογιστική μέθοδο η οποία εφαρμόζεται σε προβλήματα που δεν είναι δυνατόν να λυθούν με "άπληστες μεθόδους" (Greedy Algorithm) ή τη μέθοδο "διαίρει και βασίλευε". Θεμέλιο του δυναμικού προγραμματισμού αποτελεί η αρχή βελτιστοποίησης, δηλαδή ότι οι βέλτιστες λύσεις των υποπροβλημάτων αποτελούν το κλειδί για την βέλτιστη λύση του γενικότερου προβλήματος.

Θεμελιώθηκε το 1953 από τον Richard Bellman (1920 – 1984) με στόχο να περιγράψει τη διαδικασία επίλυσης προβλημάτων που διασπώνται σε μία αλληλουχία διαδοχικών αποφάσεων. Η λέξη "προγραμματισμός" χρησιμοποιήθηκε για να δηλώσει την κατάστρωση ενός σχεδίου και δεν έχει καμία σχέση με τον προγραμματισμό υπολογιστών. Χρησιμοποιείται ως συνώνυμο της βελτιστοποίησης και προέρχεται από τον όρο μαθηματικός προγραμματισμός. Η λέξη "δυναμικός", υποδηλώνει την χρονικά μεταβαλλόμενη φύση της διαδικασίας του δυναμικού προγραμματισμού, καθώς συμβαίνει σε πολλαπλά διαδοχικά στάδια.

Δυναμικός Προγραμματισμός

Πλεονεκτήματα:

- Έχει εφαρμογές σε πάρα πολλά προβλήματα βελτιστοποίησης, ακόμα και σε προβλήματα στοχαστικού βέλτιστου ελέγχου.
- Είναι ιδανική για προγραμματισμό με υπολογιστή.
- Εγγυάται για το ολικό ελάχιστο.
- Σε σχέση με την μέθοδο της άμεσης απαρίθμησης έχει πιο μειωμένο υπολογιστικό κόστος.

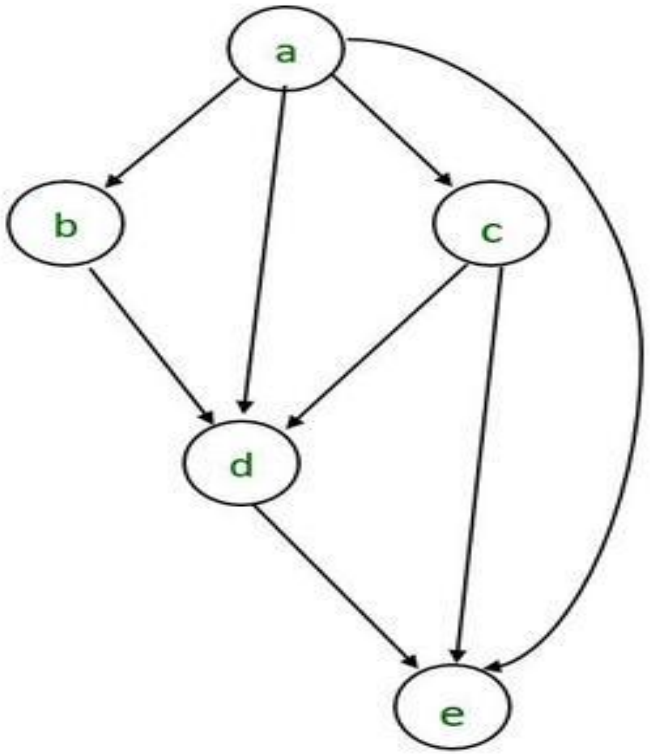
Μειονεκτήματα:

- Ο παράγοντας που περιορίζει την εξάπλωση αυτής της μεθόδου είναι η "κατάρα της διαστατικότητας" καθώς υπάρχουν υπερβολικές απαιτήσεις ακόμα και για μικρής τάξης συστήματα.
- Δεν είναι πάντα δυνατόν να βρεθεί μία αναλυτική λύση.

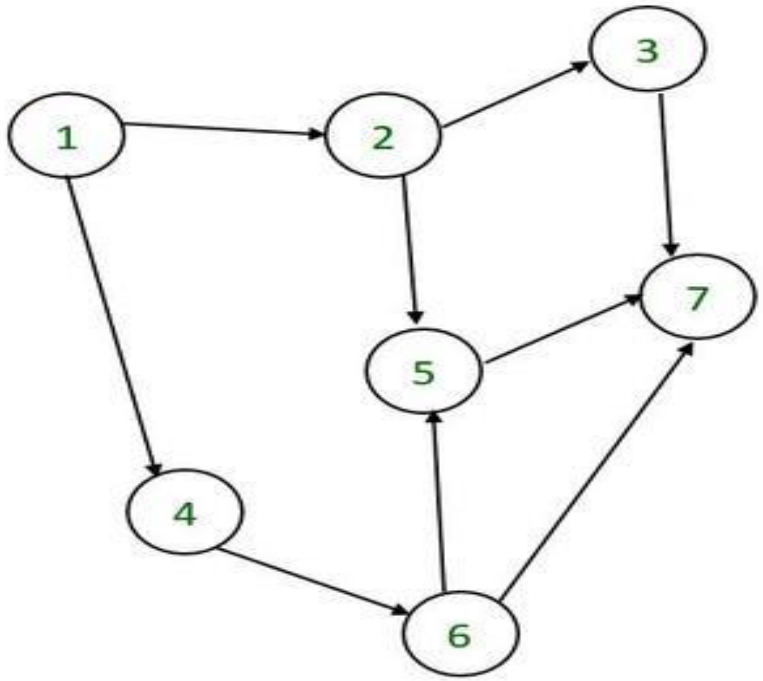
Directed Acyclic Graph (DAG)

Ένας γράφος G=(V, E), αποτελείται από ένα σύνολο πλευρών V και από ένα σύνολο πλευρών Ε που συνδέουν ζευγάρια κόμβων (υποσύνολο Ε του καρτεσιανού γινομένου VXV), όπου οι κόμβοι μπορούν να είναι οποιοδήποτε είδος αντικειμένου που συνδέονται σε ζεύγη κατά άκρα. Στην περίπτωση ενός κατευθυνόμενου γράφου, κάθε κόμβος έχει προσανατολισμό, από τον έναν στον άλλο, μέσα από μία πλευρά. Μια διαδρομή σε έναν κατευθυνόμενο γράφο είναι μια ακολουθία κόμβων που έχουν την ιδιότητα ότι ο τελικός κόμβος στην ακολουθία είναι ίδιος με τον αρχικό κόμβο στην ακολουθία. Μια διαδρομή σχηματίζει έναν κύκλο εάν ο αρχικός κόμβος της ισούται με την τελικό της κόμβο. Με πιο απλά λόγια ένας κατευθυνόμενος κυκλικός γράφος (Directed Acyclic Graph-DAG) είναι ένας κατευθυνόμενος γράφος που δεν έχει κύκλους.

Examples of directed acyclic graph



(A)



(B)

Δυναμικός Προγραμματισμός

Πότε εφαρμόζεται;

Σε προβλήματα με:

- **Βέλτιστη υπο-δομή**: η βέλτιστη λύση του προβλήματος συνίσταται από βέλτιστες λύσεις υπο-προβλημάτων

- **Επικαλυπτόμενα υποπροβλήματα**: λίγα συνολικά υπο-προβλήματα, πολλά αναδρομικά στιγμιότυπα του καθενός

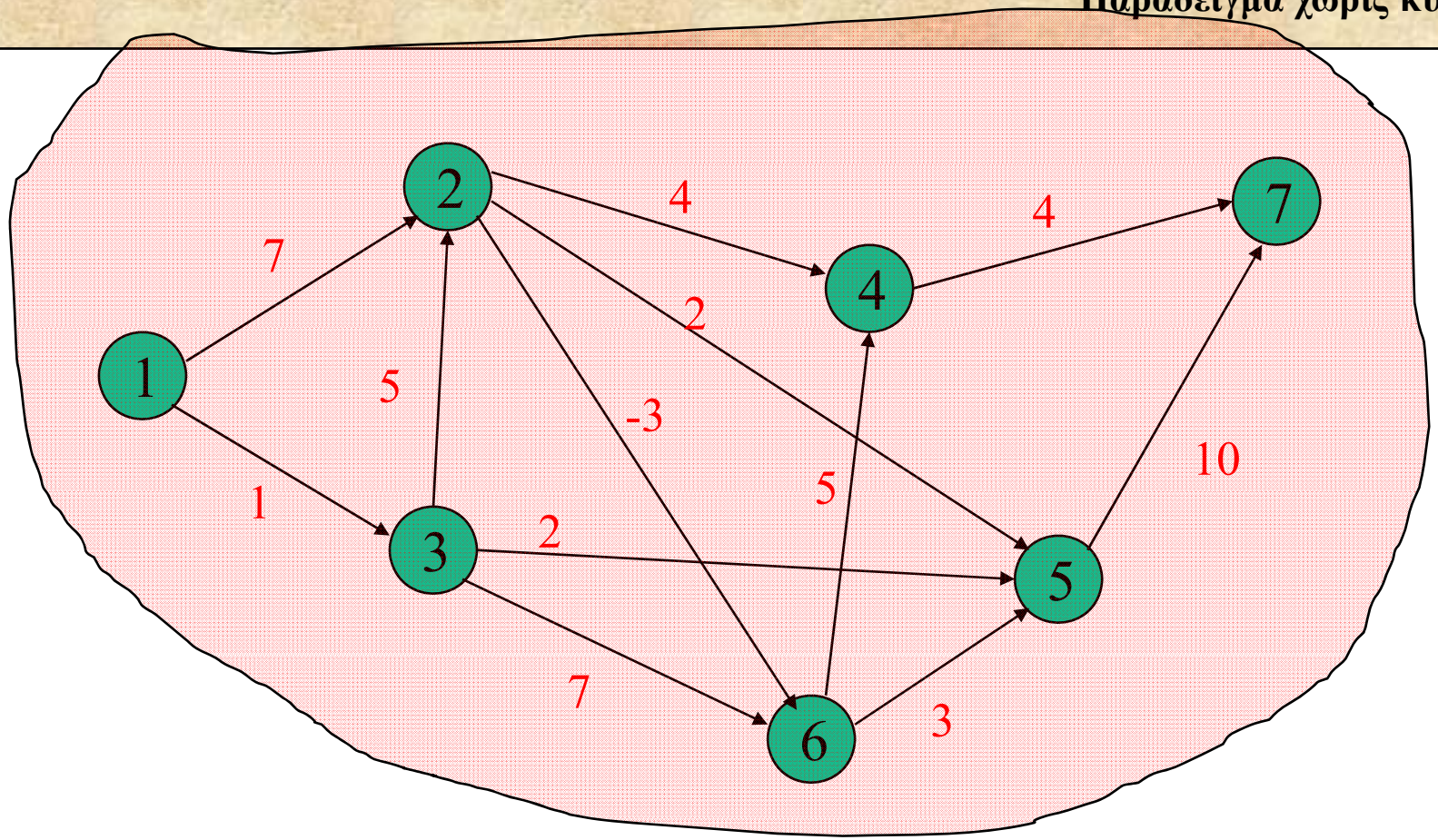
Βασική προσέγγιση

Πως;

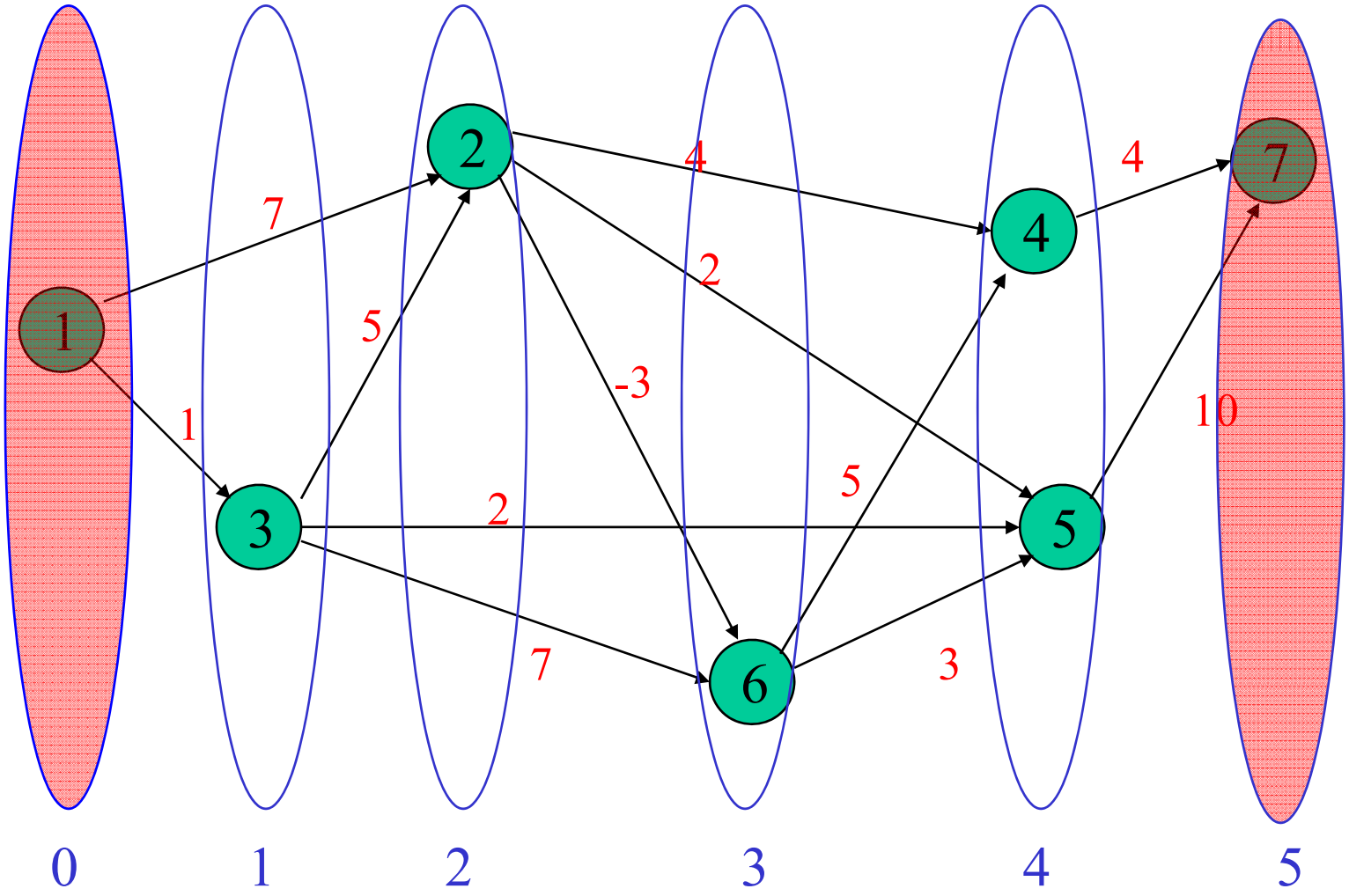
- Ορισμός υπο-προβλημάτων
- Σύνδεση της βέλτιστης λύσης με τις βέλτιστες λύσεις μέσω μιας αναδρομικής σχέσης
- Κατασκευή ενός πίνακα από λυμένα υπο-προβλήματα που χρησιμοποιούνται για την επίλυση των μεγαλύτερων
- Εύρεση της τιμής της λύσης από τα κάτω προς τα πάνω
- Κατασκευή της δομής της λύσης από πάνω προς τα κάτω

Συντομότερα μονοπάτια

Παράδειγμα χωρίς κύκλο - dag

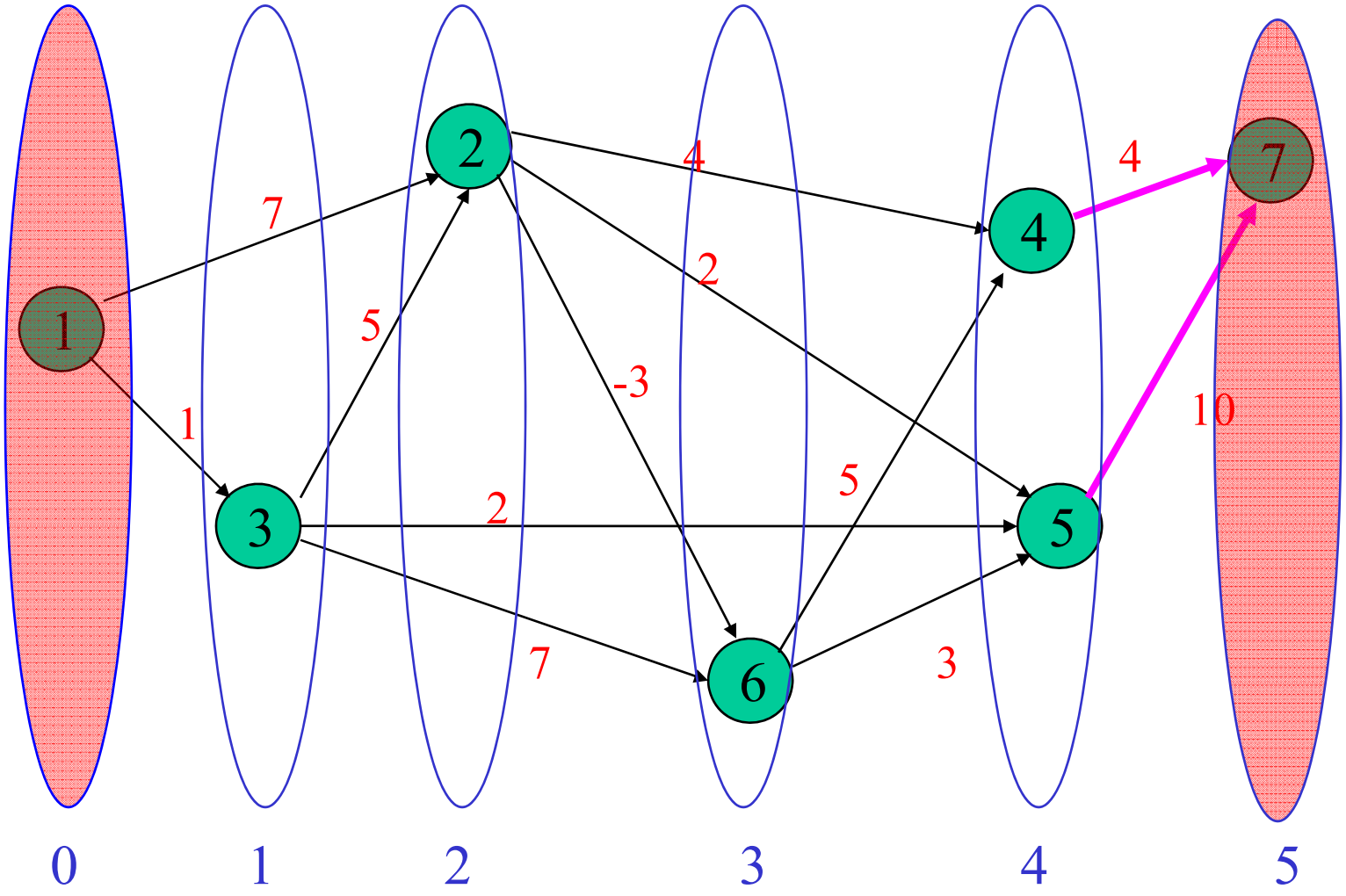


Συντομότερο μονοπάτι από τον κόμβο 1 στον 7



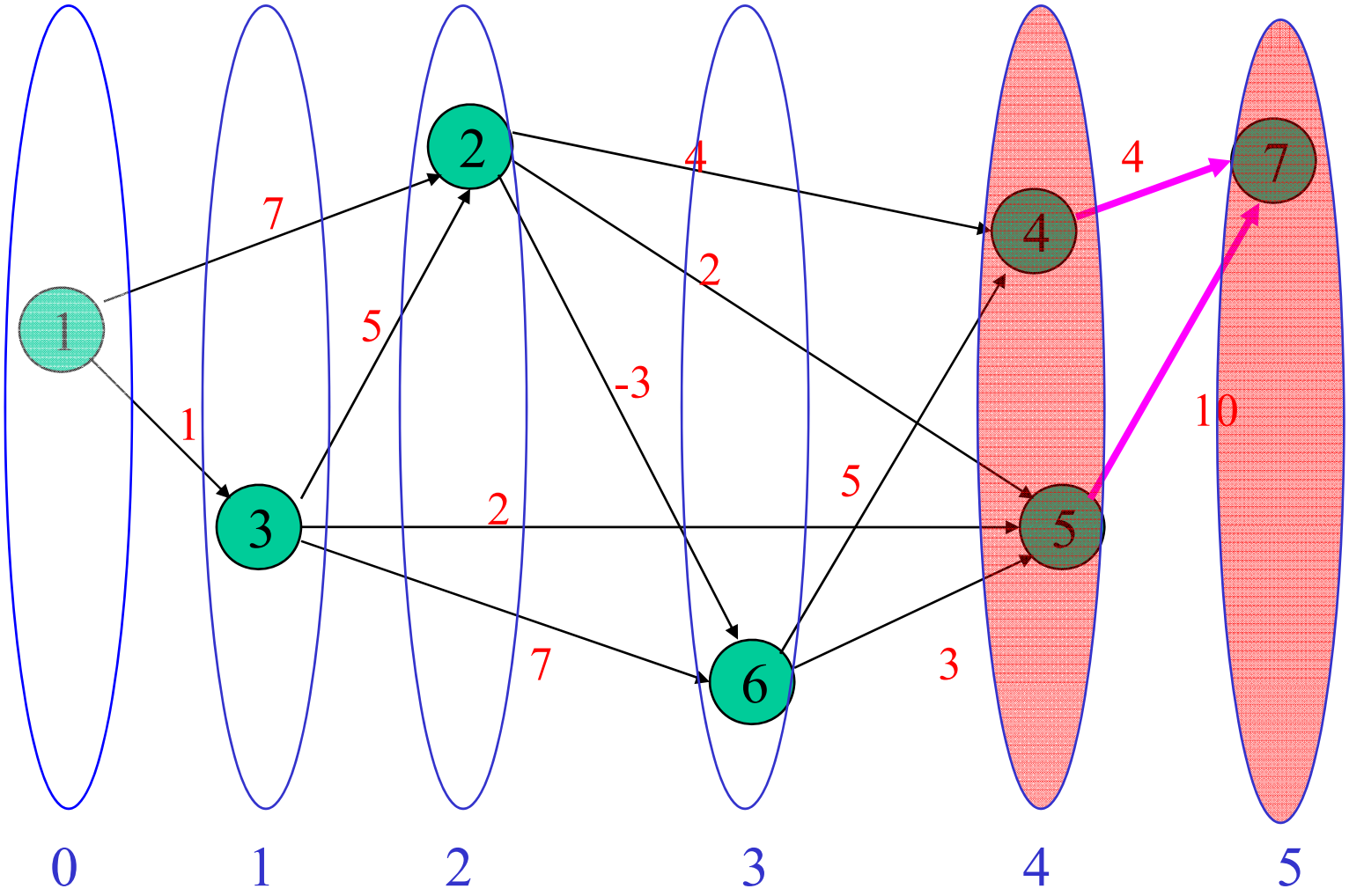
Παράδειγμα χωρίς κύκλο/Levels

Παράδειγμα χωρίς κύκλο



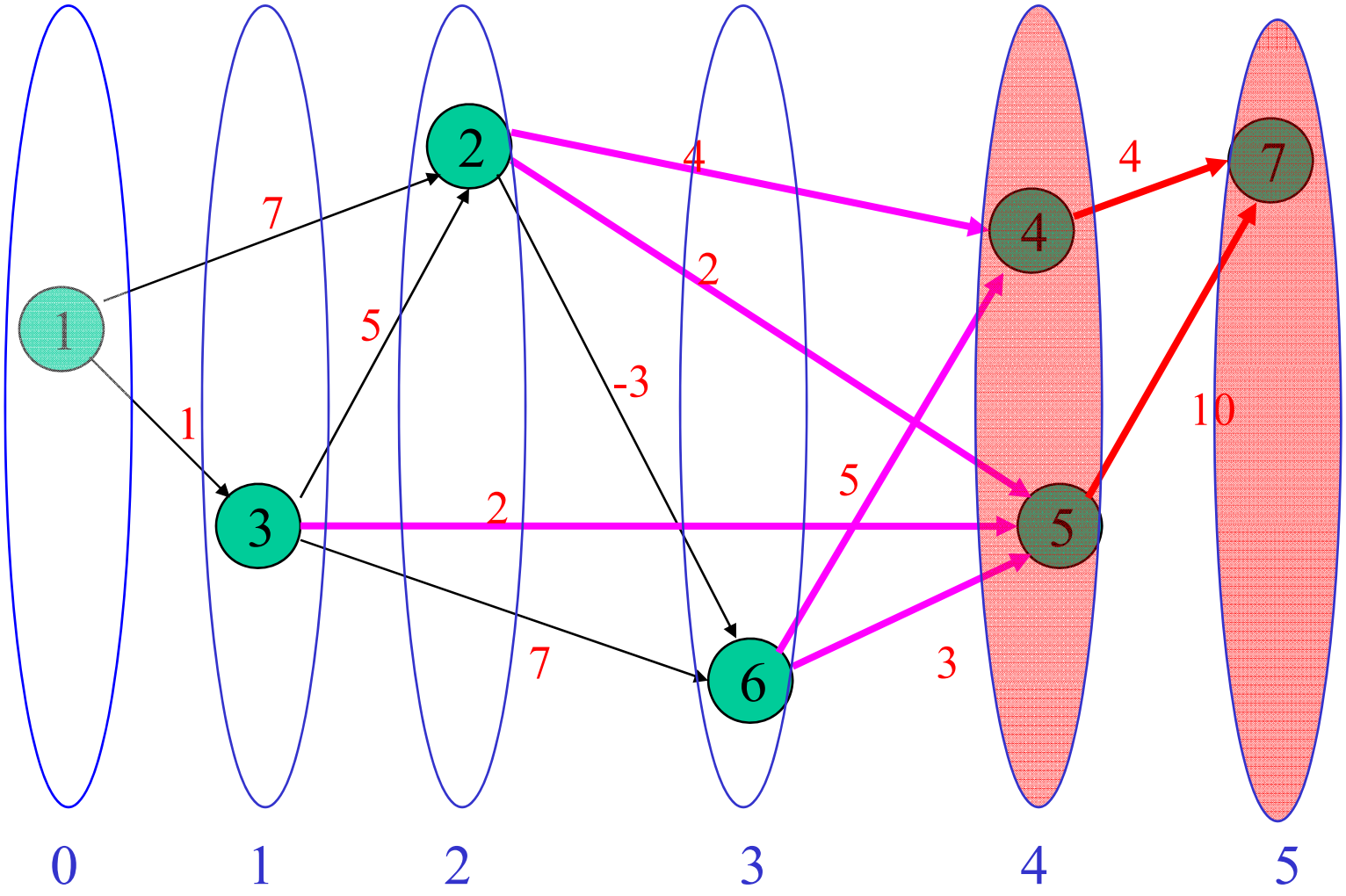
Συντομότερα μονοπάτια από τον κόμβο 1 στους κόμβους 4 και 5

Παράδειγμα χωρίς κύκλο



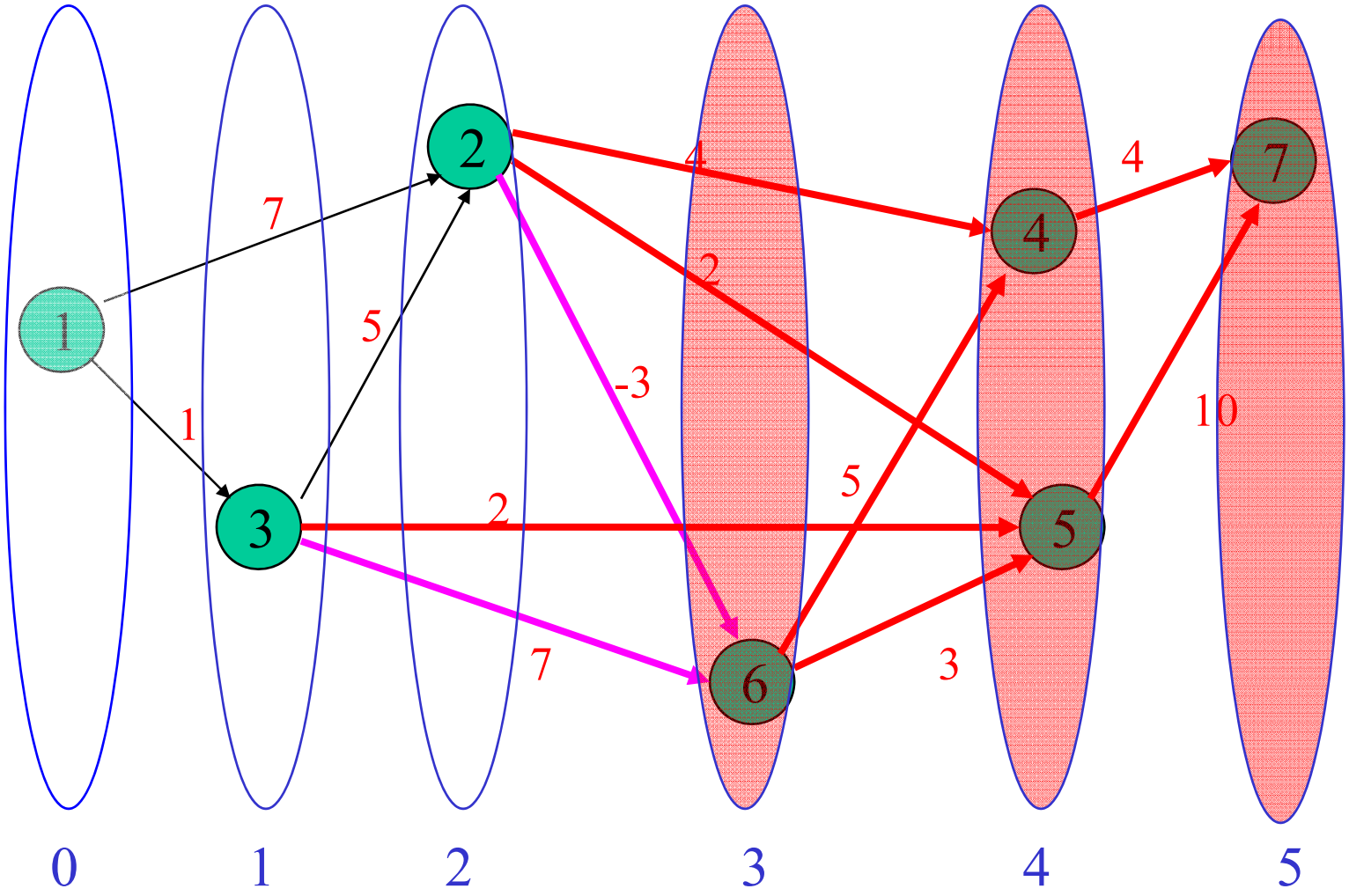
Συντομότερα μονοπάτια από το κόμβο 1 στους 4 και 5

Παράδειγμα χωρίς κύκλο



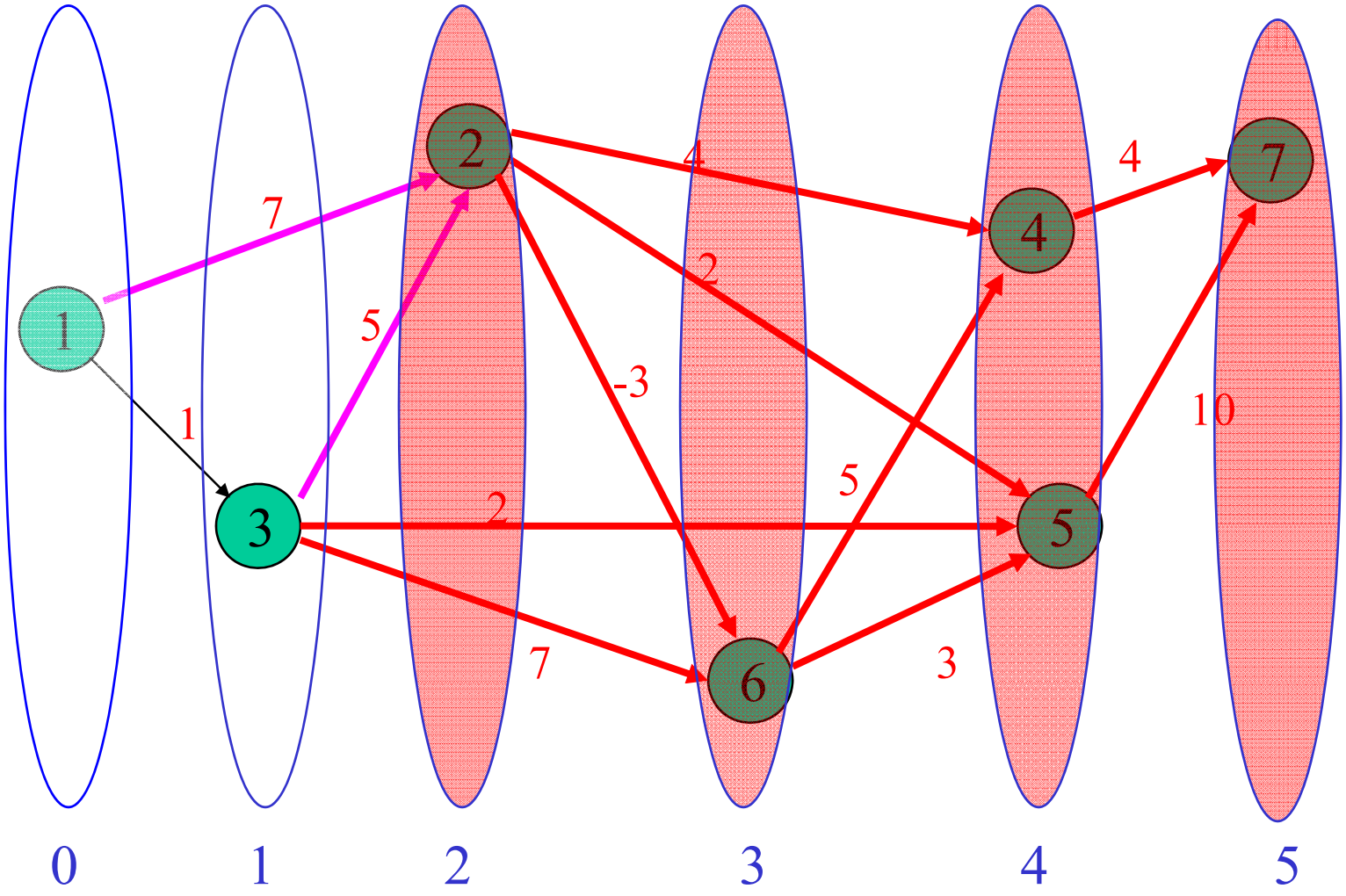
Συντομότερα μονοπάτια από το κόμβο 1 στους 2, 3 και 6

Παράδειγμα χωρίς κύκλο



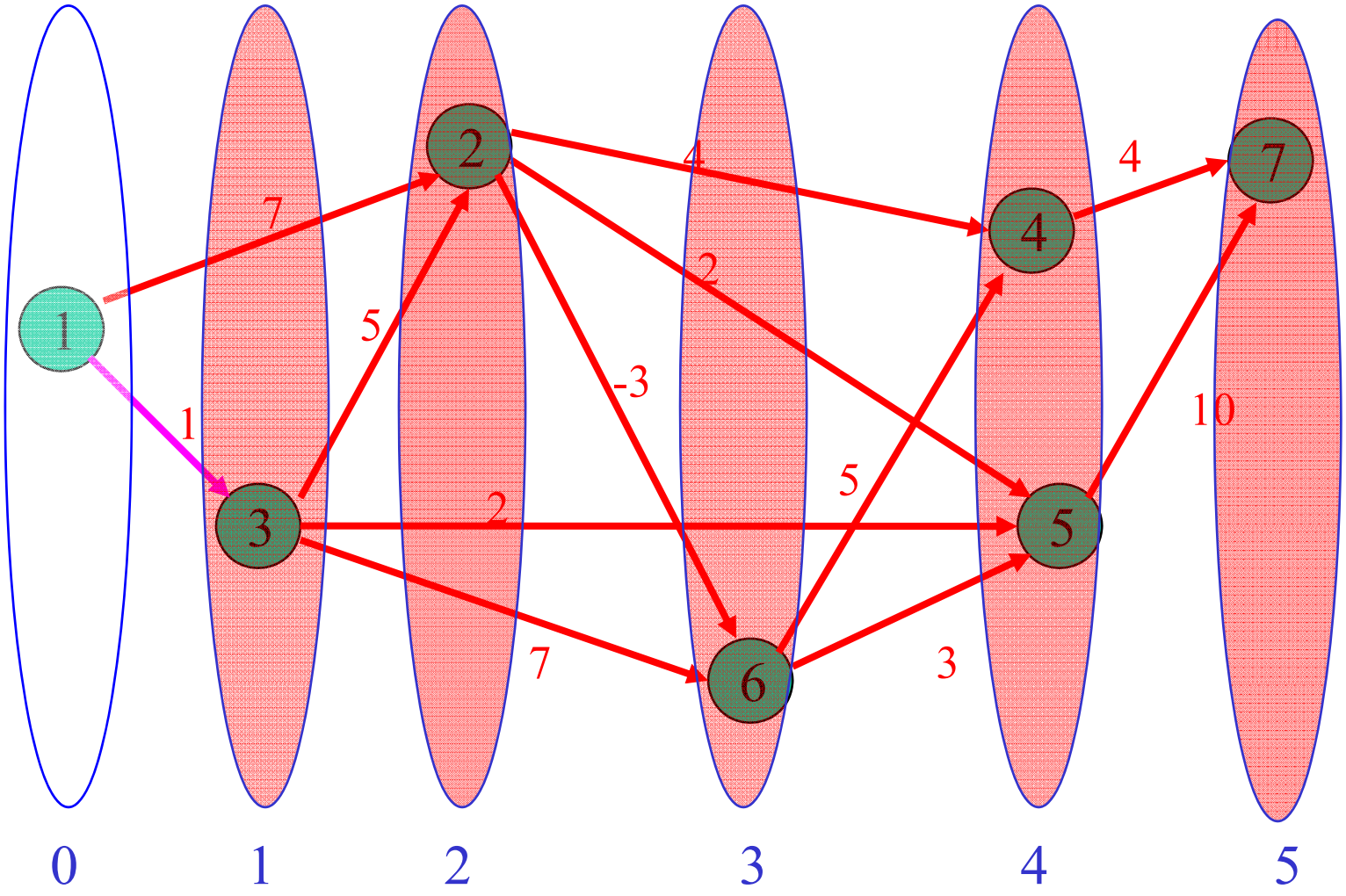
Συντομότερα μονοπάτια από το κόμβο 1 στους κόμβους 2 και 3

Παράδειγμα χωρίς κύκλο



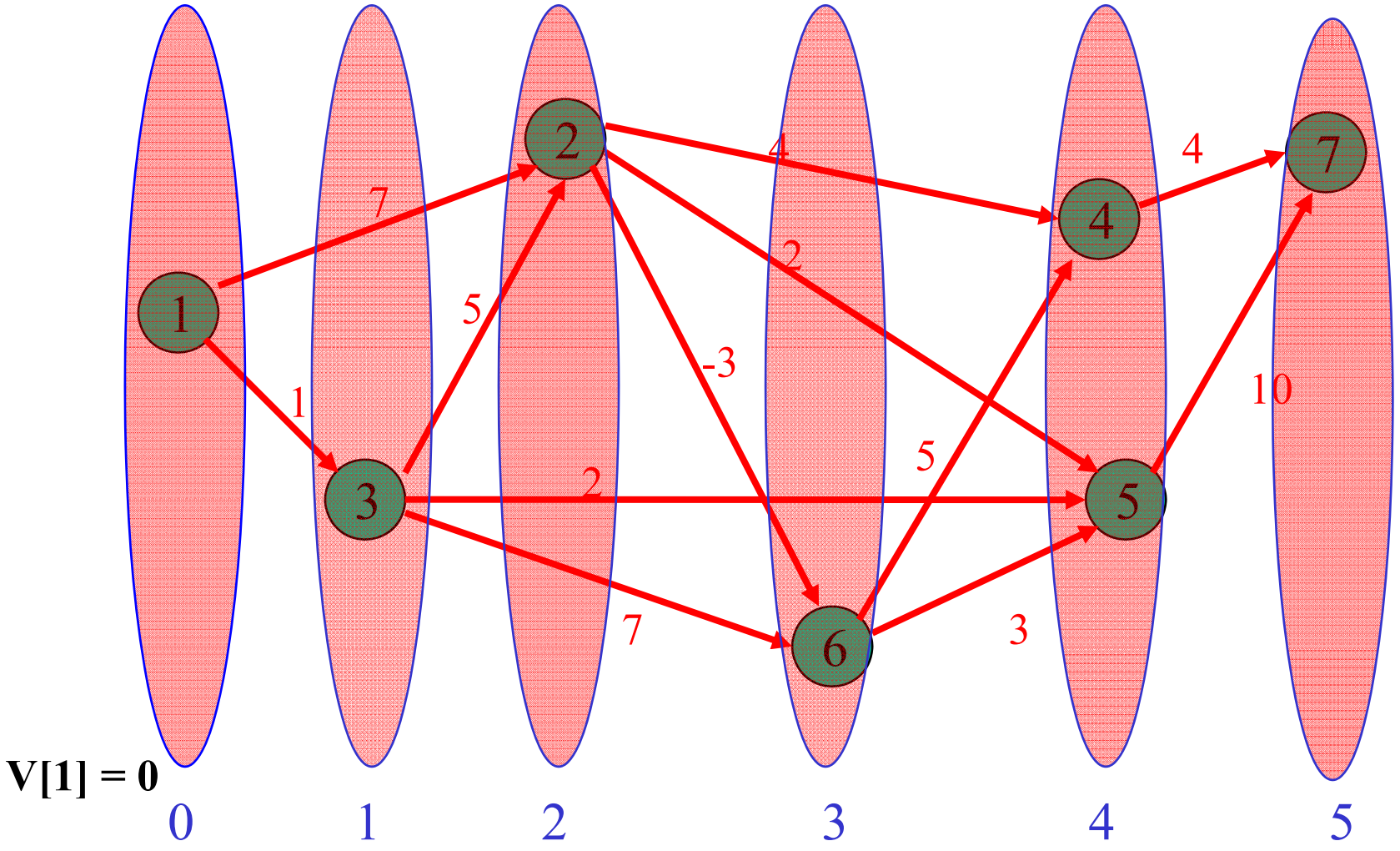
Συντομότερα μονοπάτια από το κόμβο 1 στους κόμβους 1 και 3

Παράδειγμα χωρίς κύκλο



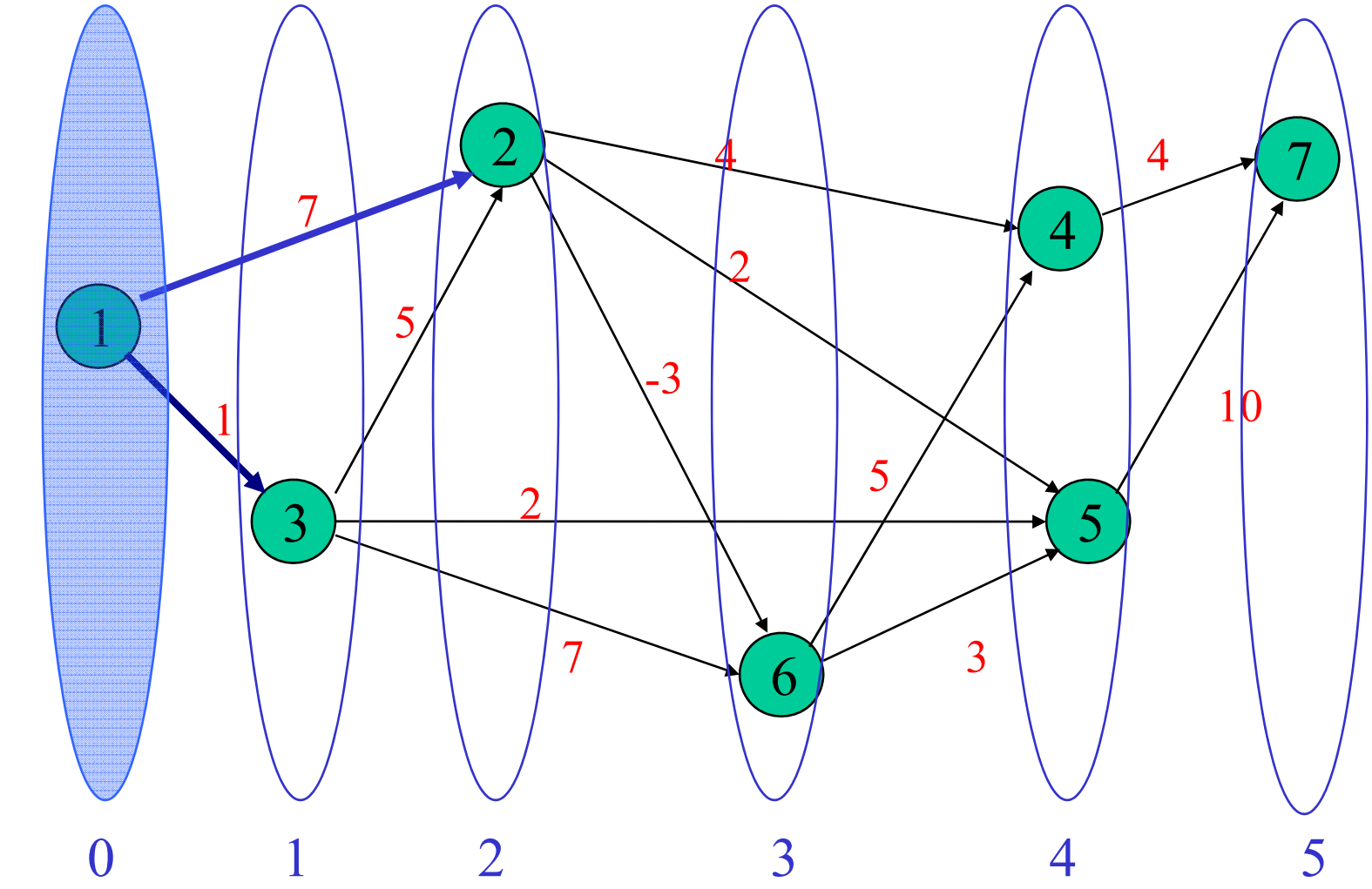
Συντομότερο μονοπάτι από το κόμβο 1 στον κόμβο 1

Παράδειγμα χωρίς κύκλο



Συντομότερο μονοπάτι από το κόμβο 1 στον κόμβο 1

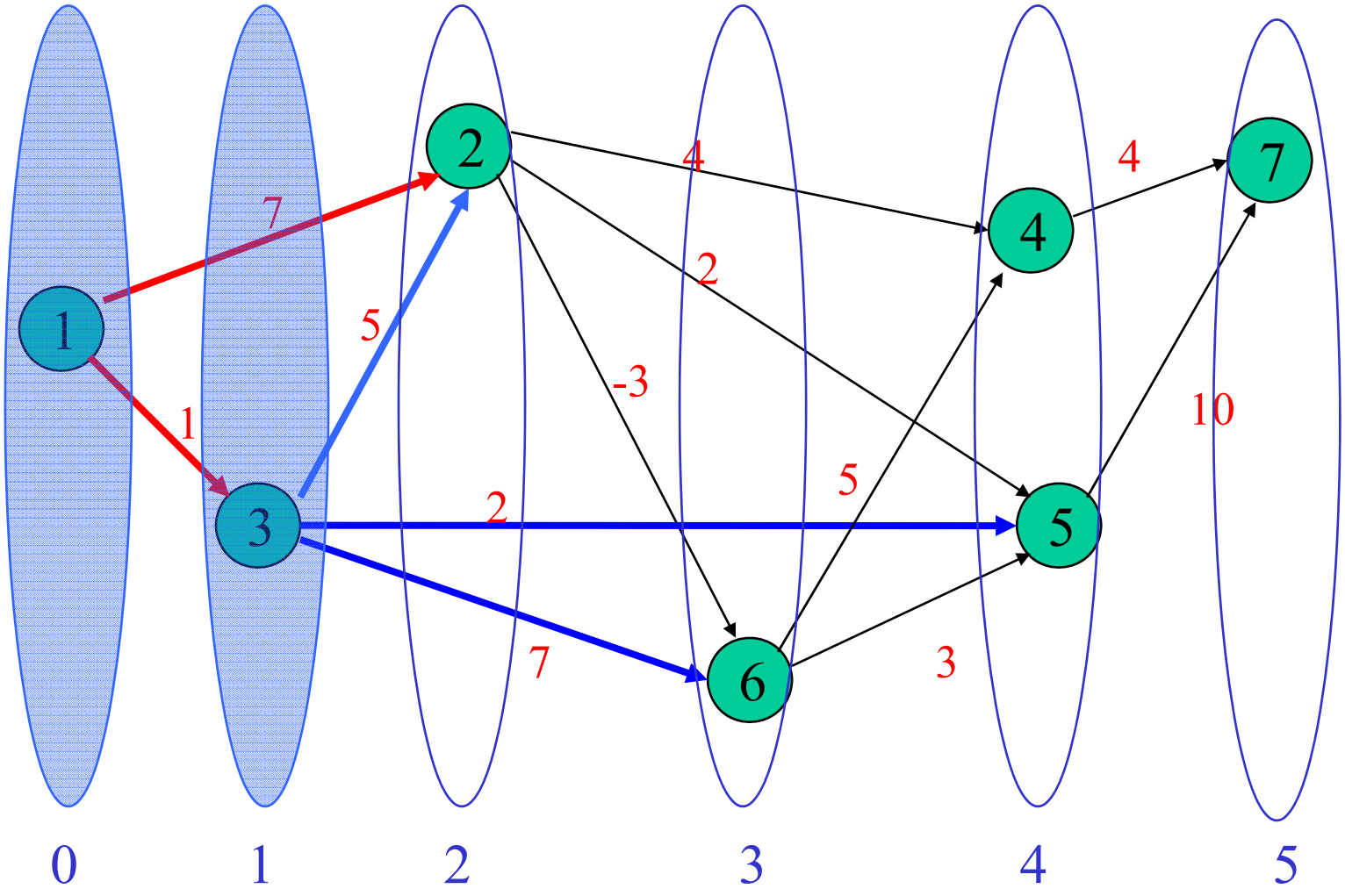
Παράδειγμα χωρίς κύκλο



V[1] → V[3]

Επίλυση υπο-προβλήματος 2

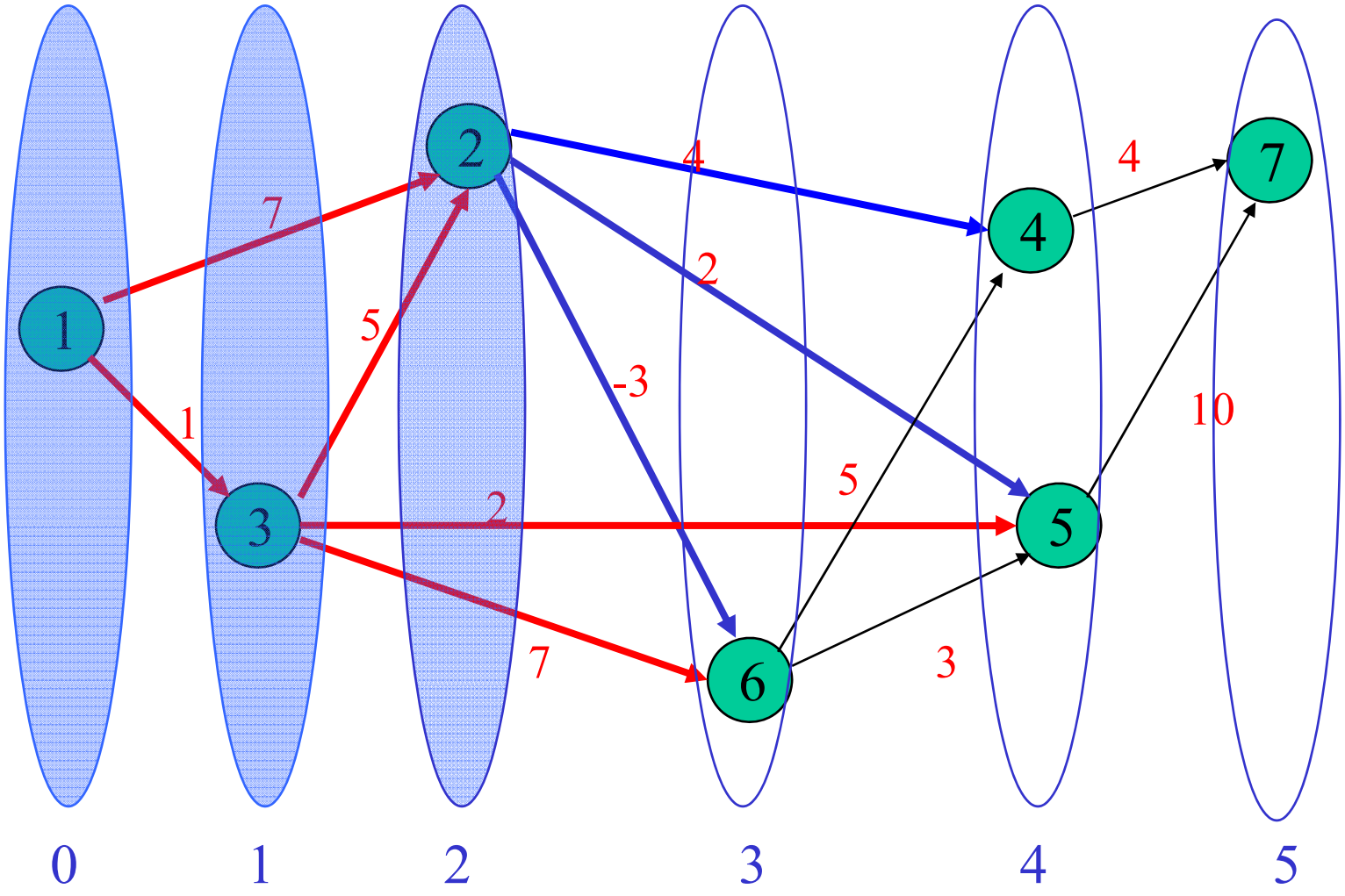
Παράδειγμα χωρίς κύκλο



V[1] → V[3] →

Επίλυση υπο-προβλήματος 3

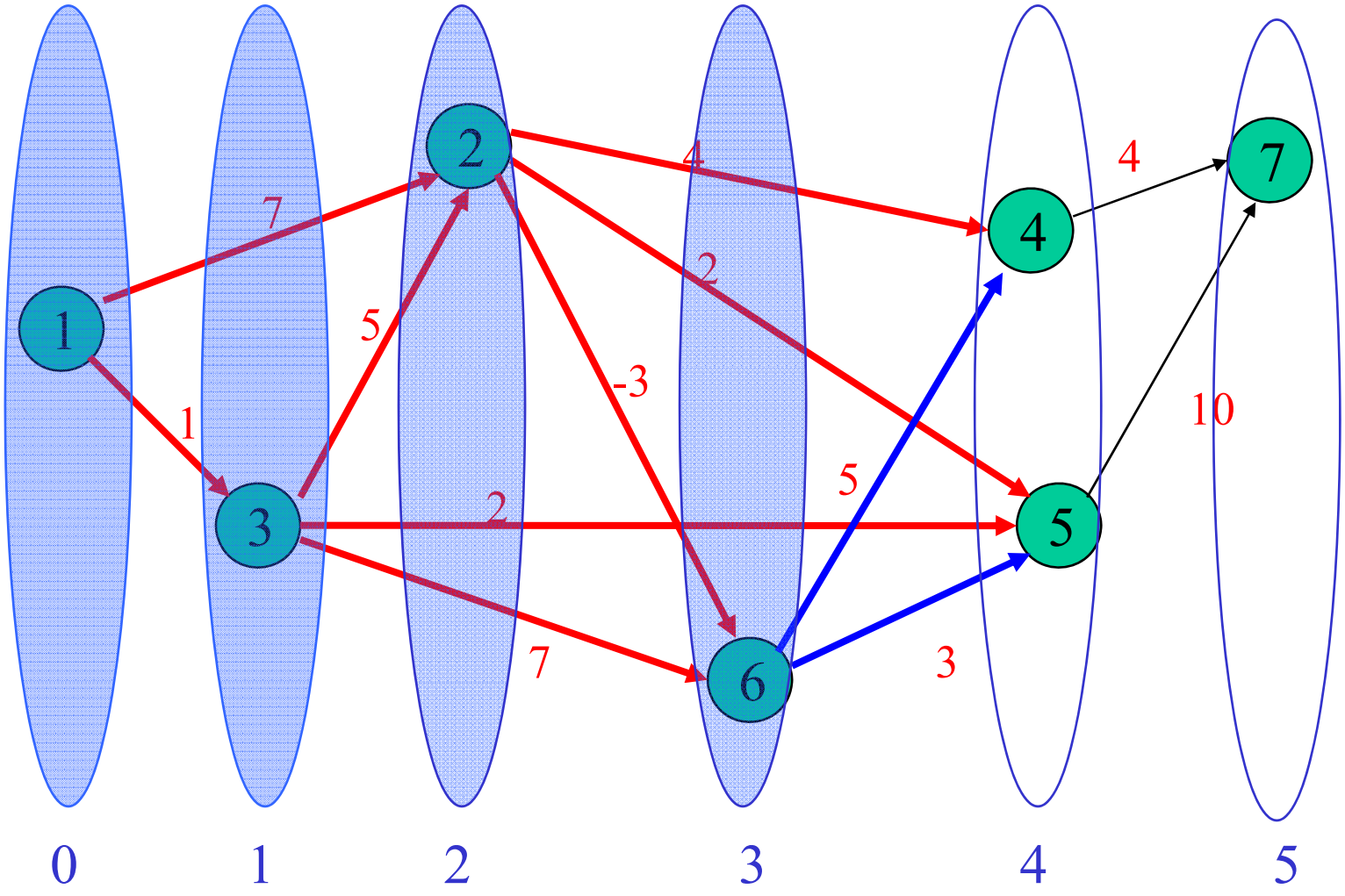
Παράδειγμα χωρίς κύκλο



V[1] → V[3] → V[2] →

Επίλυση υπο-προβλήματος 4

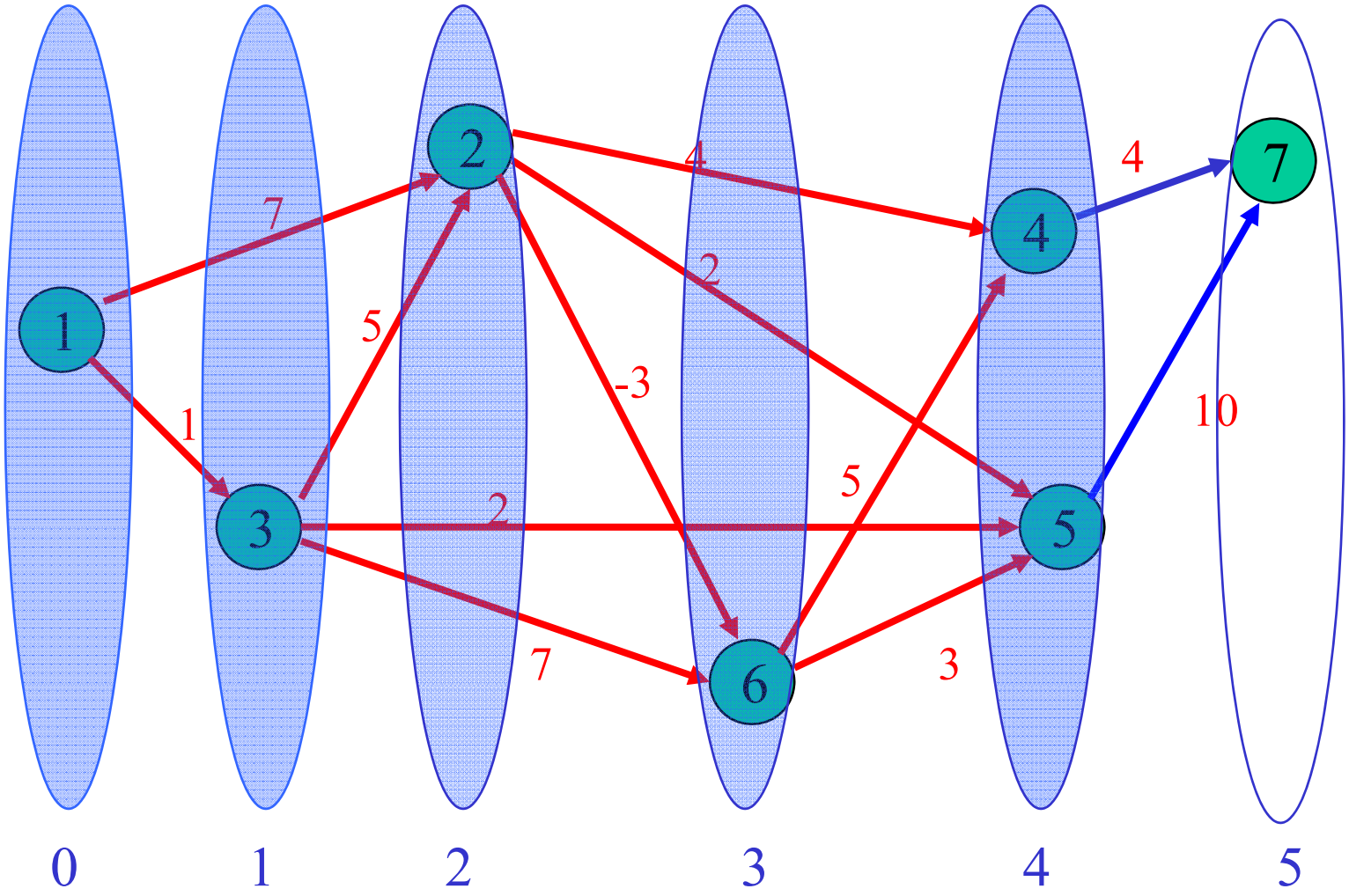
Παράδειγμα χωρίς κύκλο



V[1] → V[3] → V[2] → V[6]

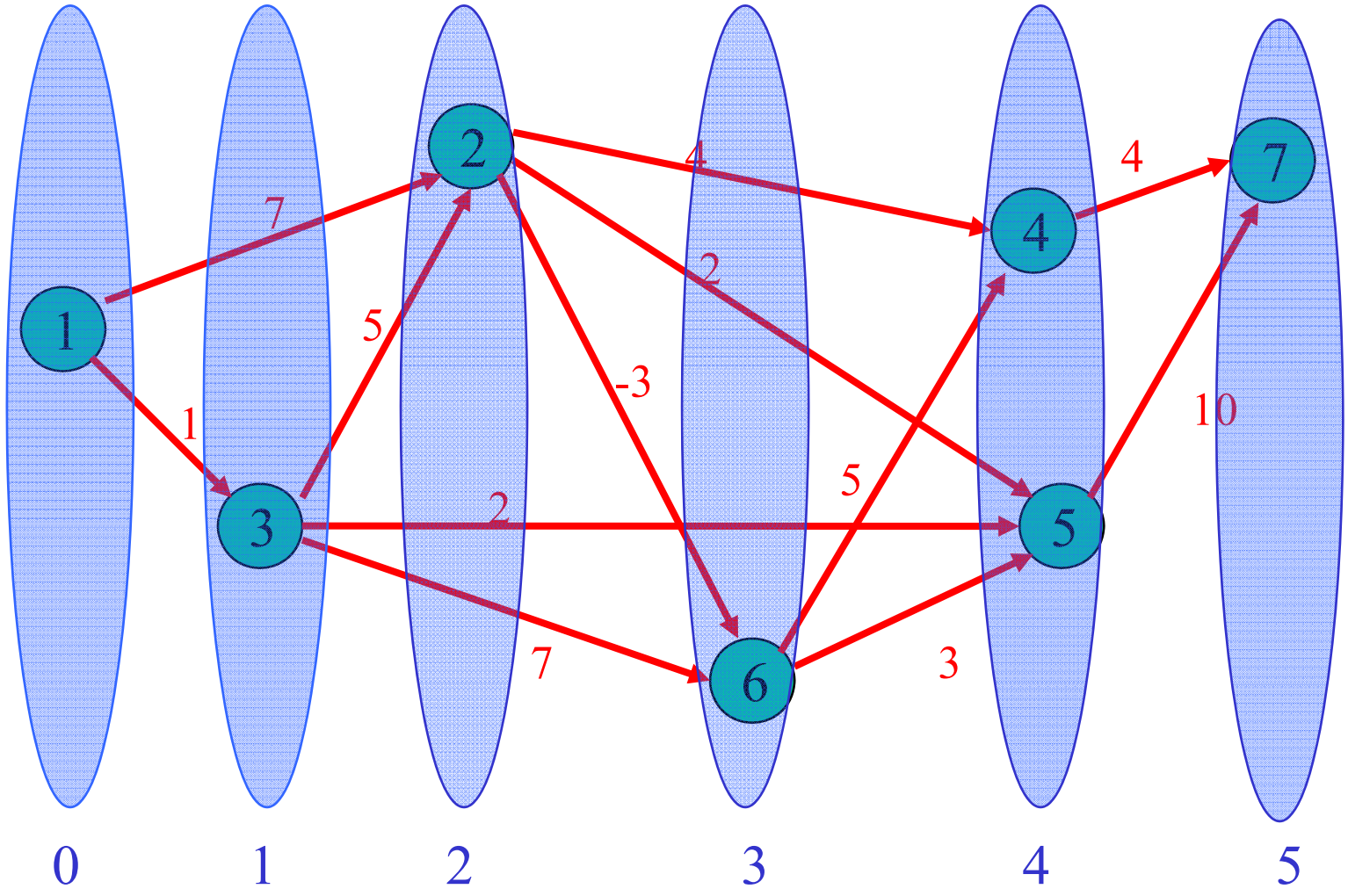
Επίλυση υπο-προβλήματος 4

Παράδειγμα χωρίς κύκλο



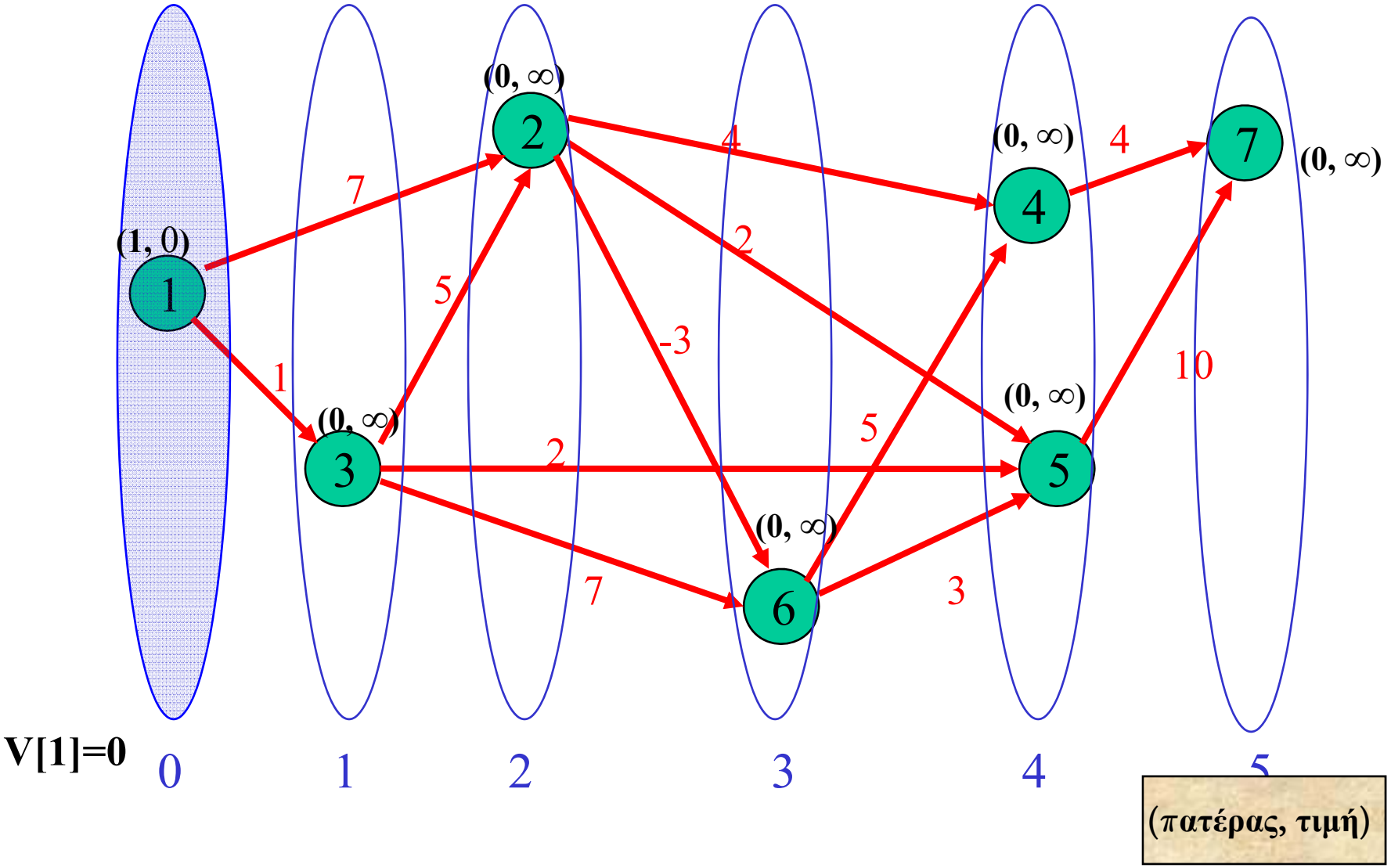
Επίλυση υπο-προβλημάτων 5 και 6

Παράδειγμα χωρίς κύκλο



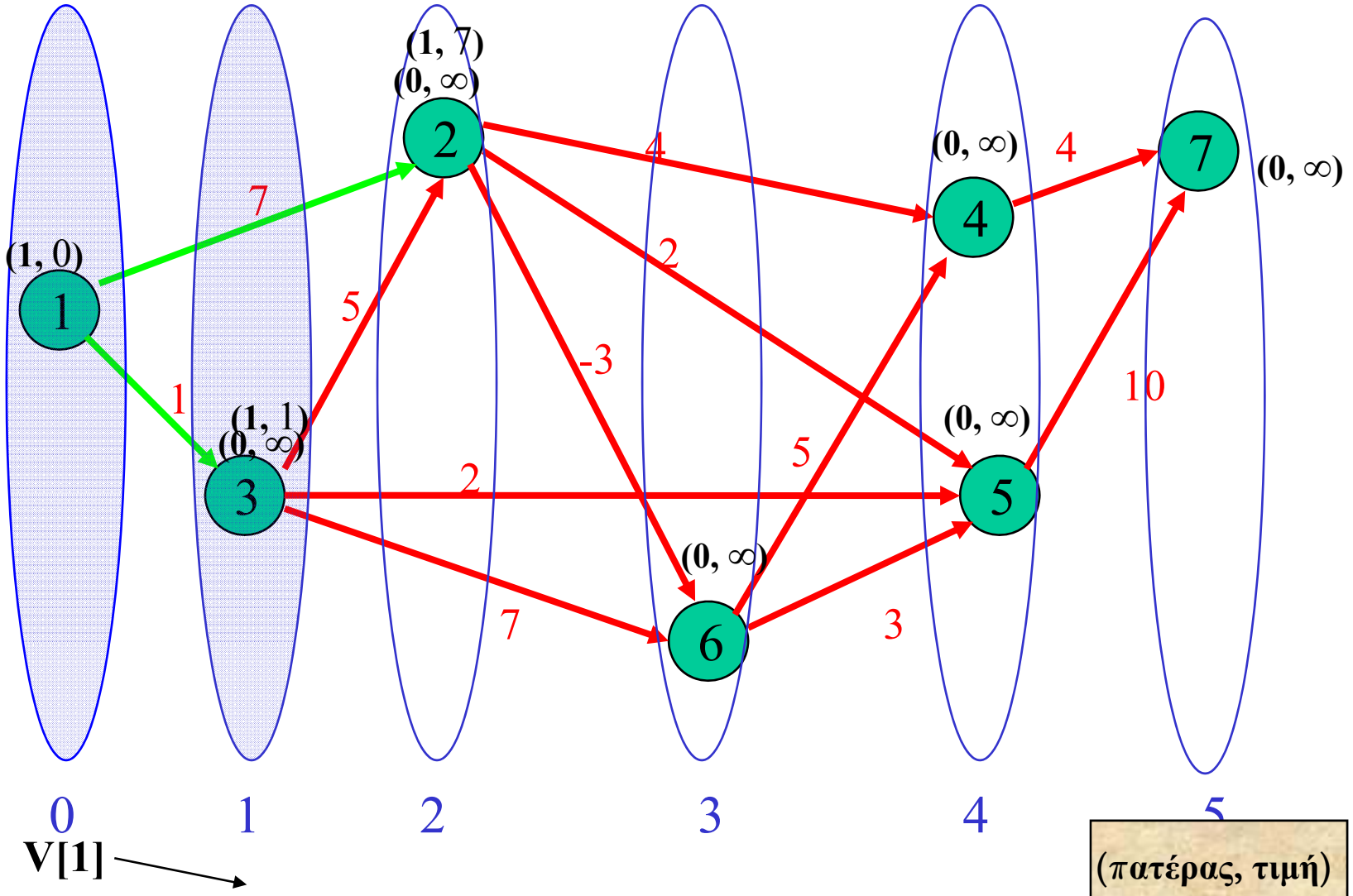
Επίλυση υπο-προβλήματος *7*

Αλγόριθμος για dag



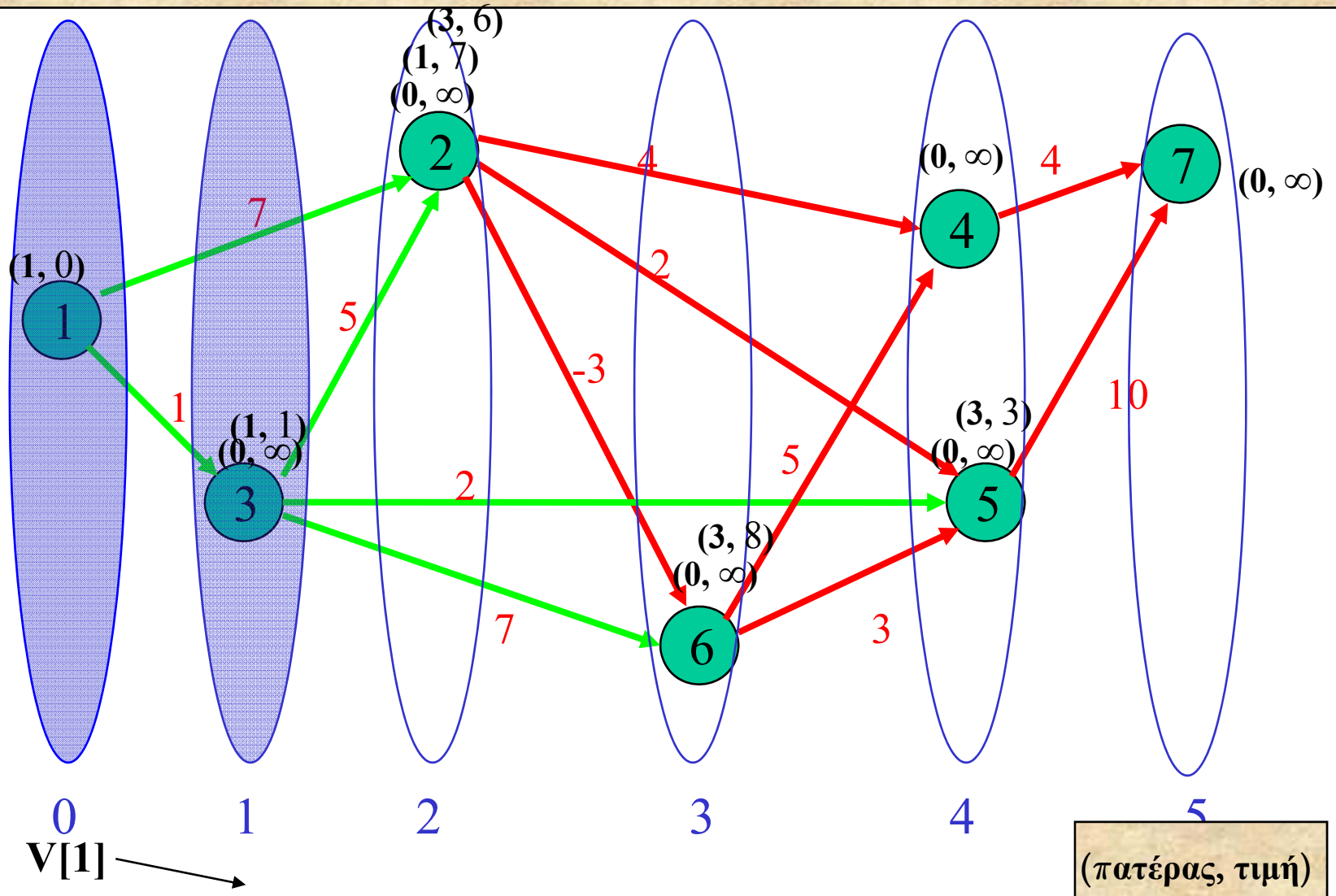
Συντομότερο μονοπάτι από το κόμβο 1 στον κόμβο 1

Αλγόριθμος για dag



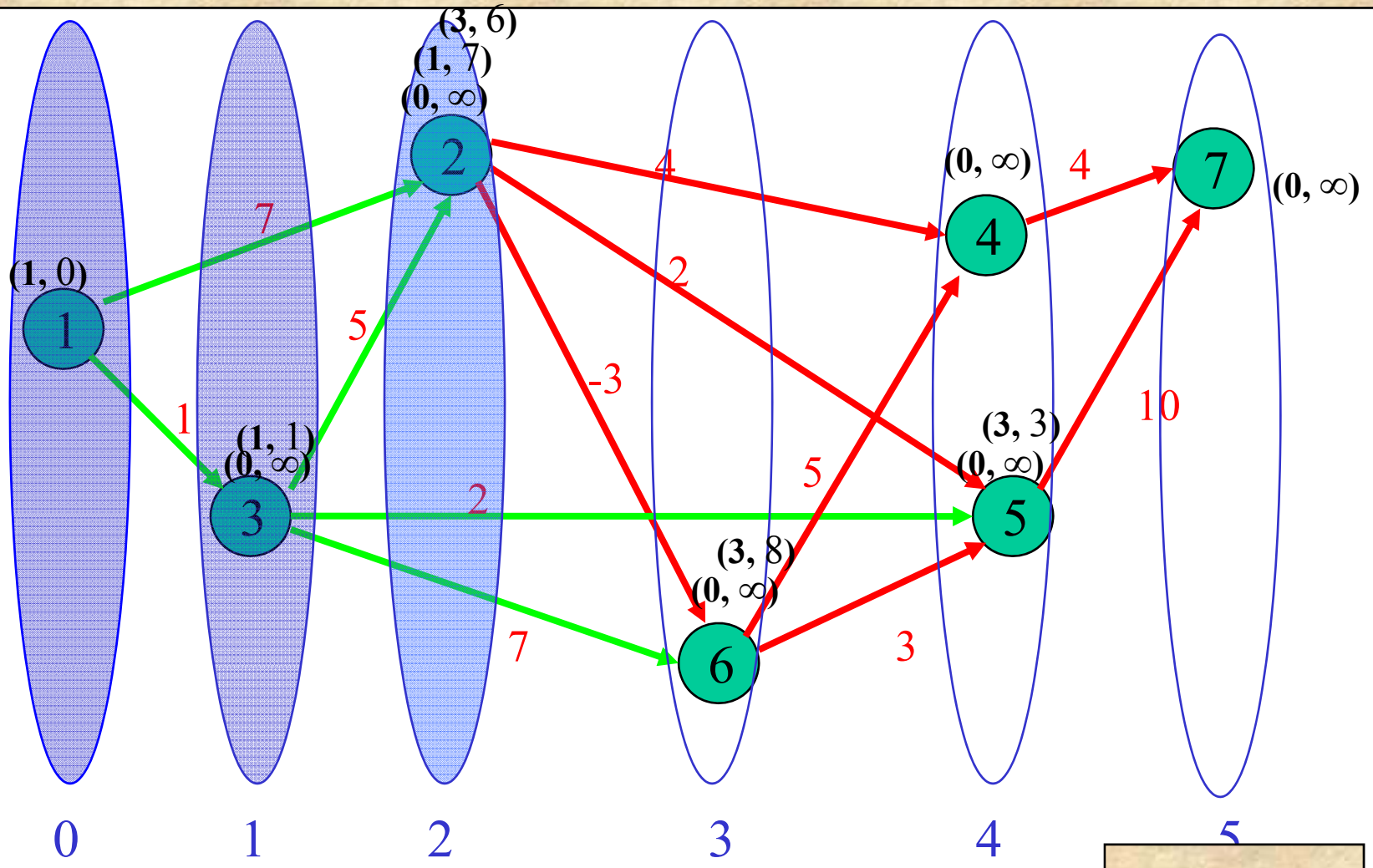
Συντομότερο μονοπάτι από το κόμβο 1 στον κόμβο 3

Αλγόριθμος για dag



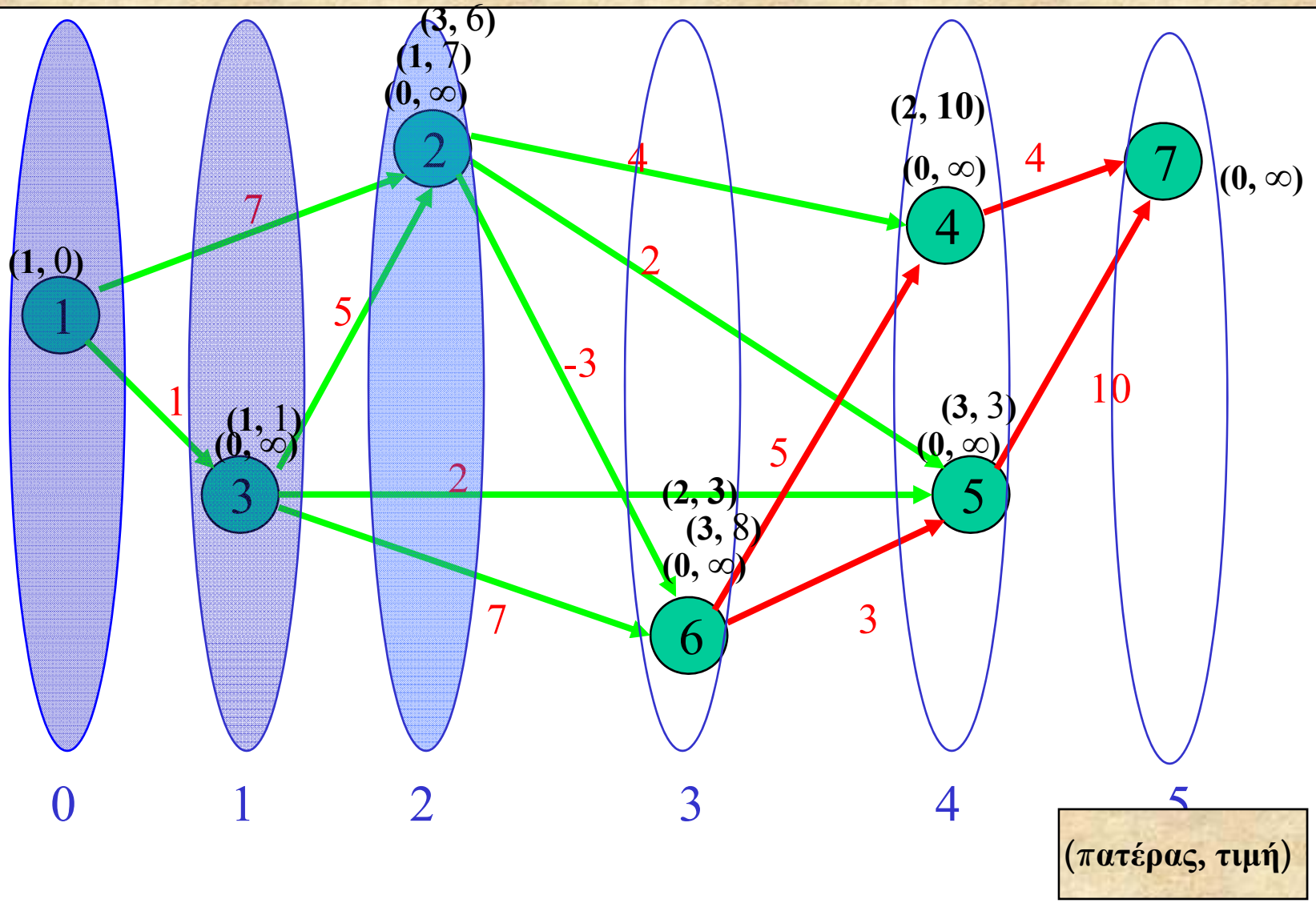
Συντομότερο μονοπάτι από το κόμβο 1 στον κόμβο 2

Αλγόριθμος για dag



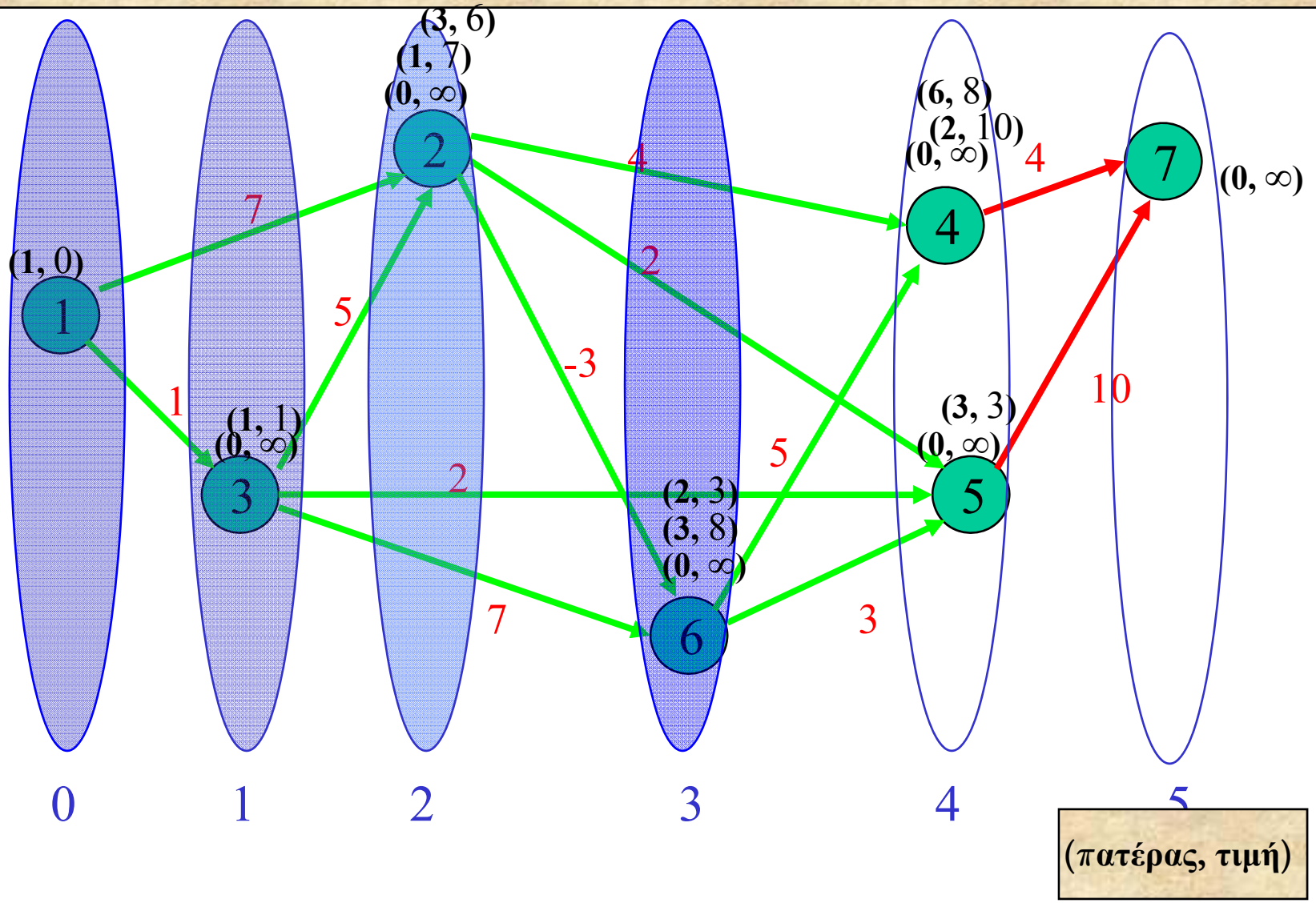
Συντομότερο μονοπάτι από το κόμβο 1 στον κόμβο 2

Αλγόριθμος για dag



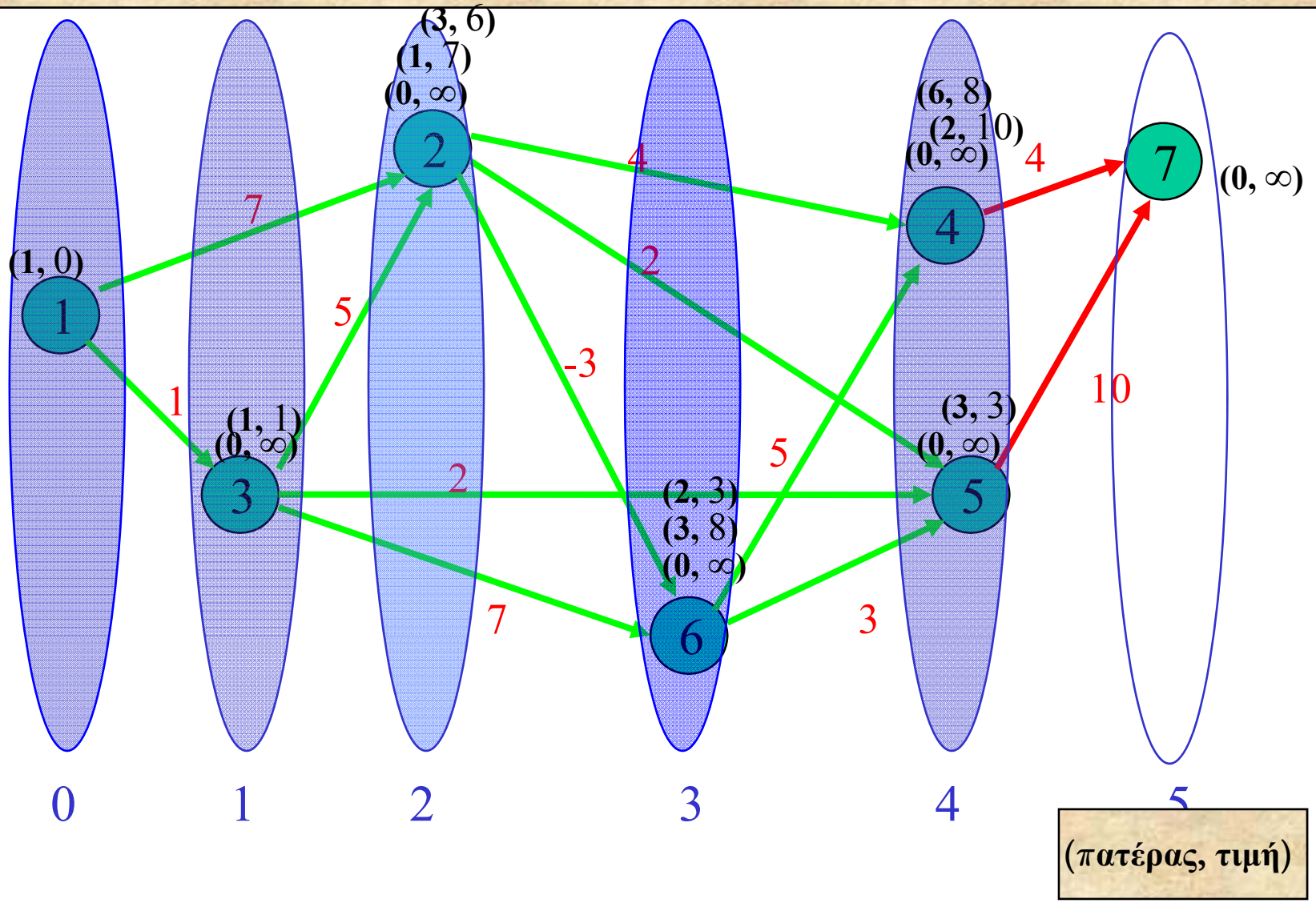
Συντομότερο μονοπάτι από το κόμβο 1 στον κόμβο 6

Αλγόριθμος για dag



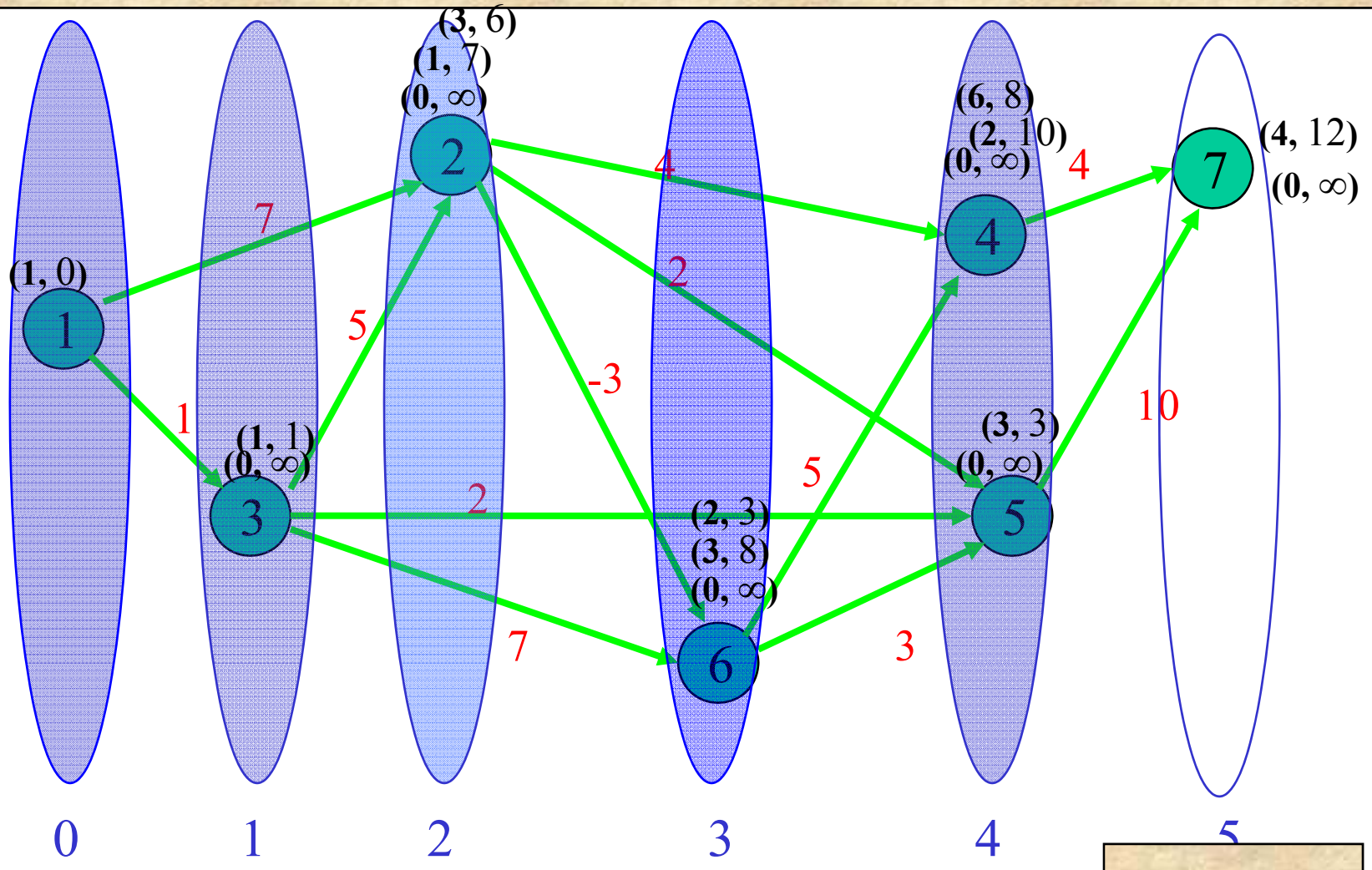
Συντομότερα μονοπάτια από το κόμβο 1 στους κόμβους 4 και 5

Αλγόριθμος για dag



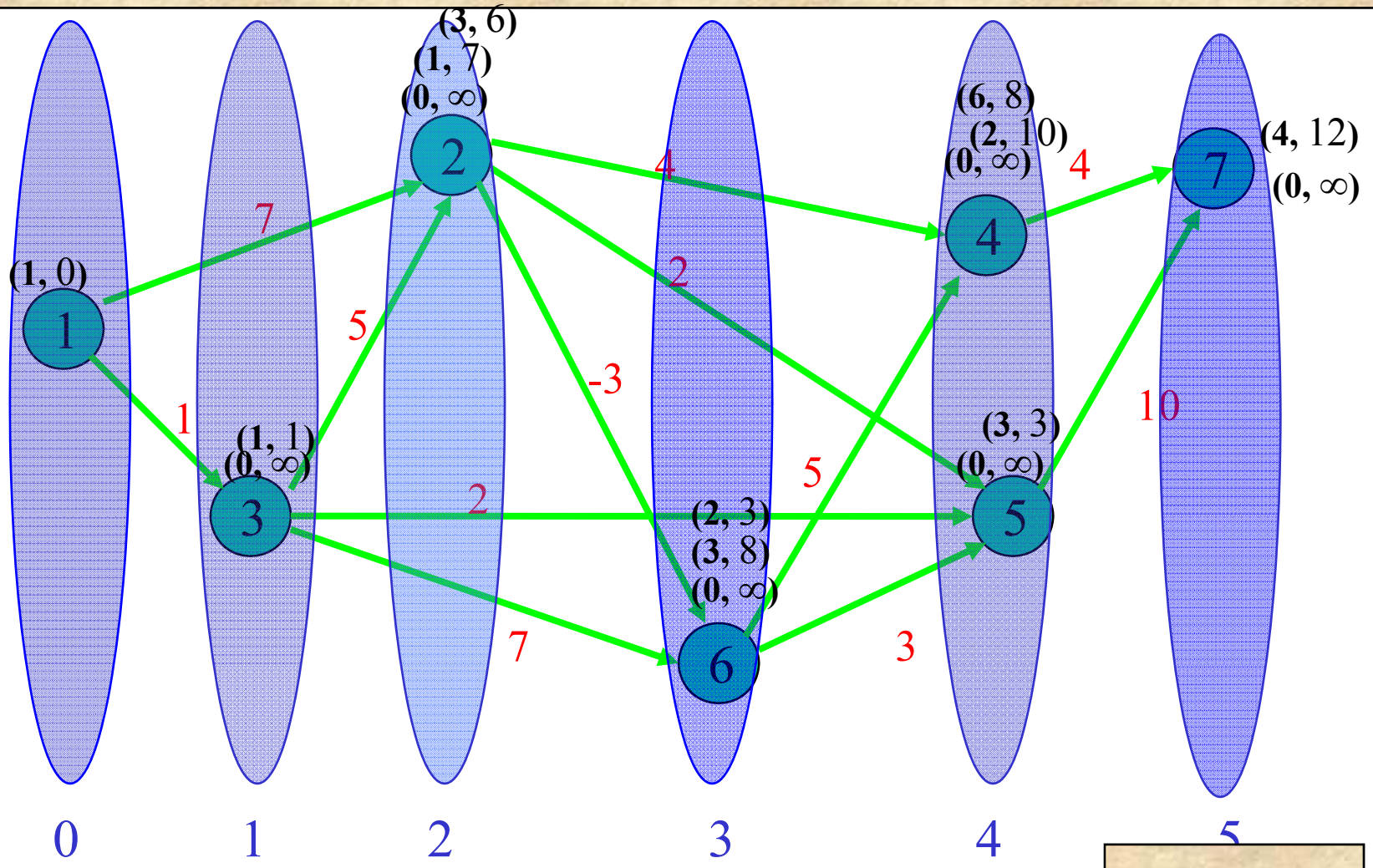
Συντομότερα μονοπάτια από το κόμβο 1 στους κόμβους 4 και 5

Αλγόριθμος για dag



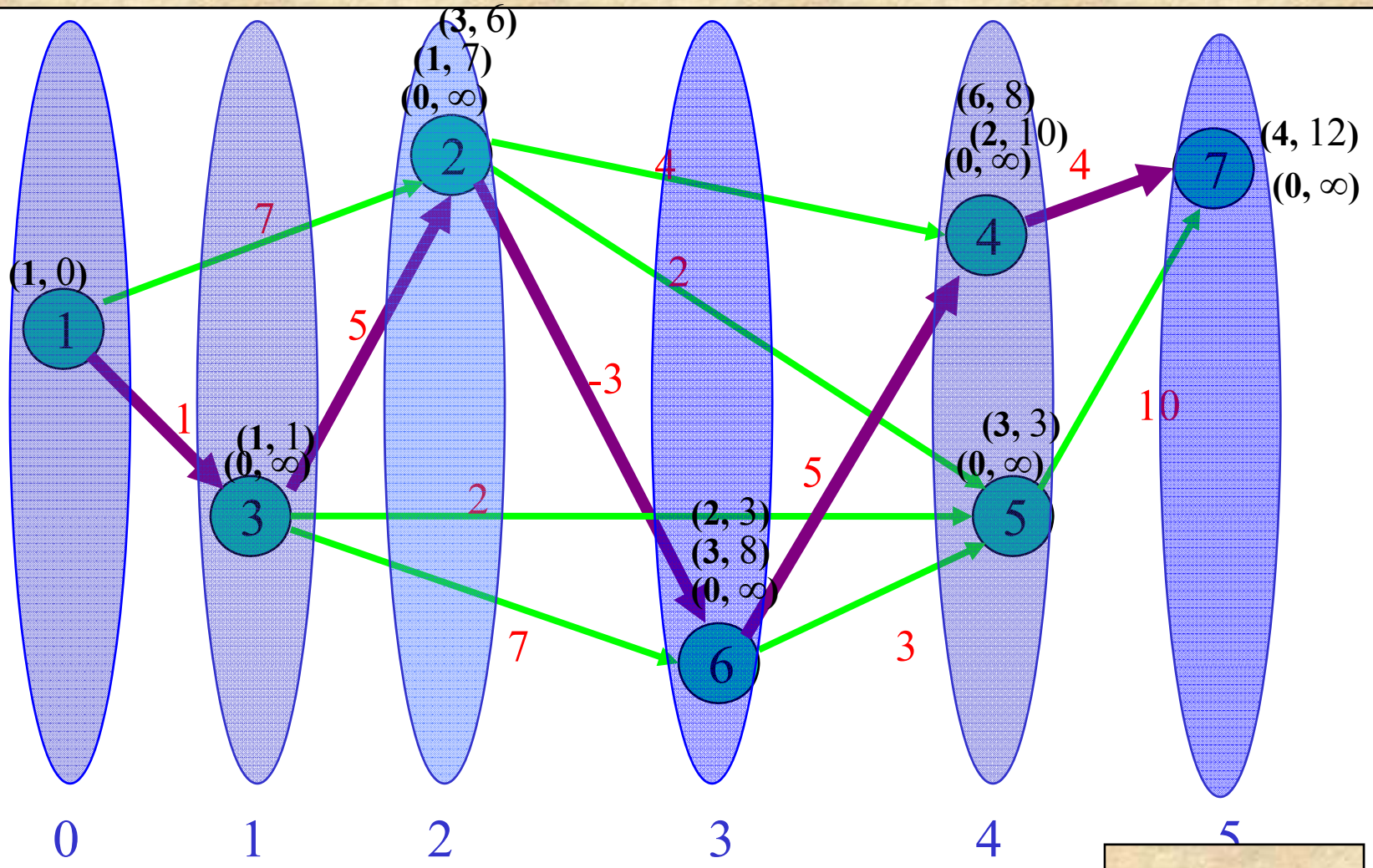
Συντομότερο μονοπάτι από το κόμβο 1 στον κόμβο 7

Αλγόριθμος για dag



Συντομότερο μονοπάτι από το κόμβο 1 στον κόμβο 7

Αλγόριθμος για dag



Συντομότερο μονοπάτι από το κόμβο 1 στον κόμβο 7

Συντομότερα μονοπάτια σε DAGs - υποπροβλήματα

Στο γράφο $G = (X, U)$ $\quad |X| = n$

ή $\quad (G = (X, \Gamma))$

n υπο-προβλήματα

V[y]: βέλτιστη τιμή του υπο-προβλήματος από τον 1 στον y

P(1): 1 → 1 V[1] = 0
P(2): 1 → 2 V[2]
……
P(y): 1 → y V[y] = min {V[x]+W(x, y), x προηγούμενος του y}
……
P(n): 1 → n V[n]

Αλγόριθμος Bellman για DAGs

```
Τοπολογική ταξινόμηση: Sorted[i],Layer[i]
```

```
Initialization V to ∞; Initialization P to 0;
V[s] := 0; P[s] := s; V[y] := W(s,y); P[y] := s;
Y successor of s;
```

```
for i := 1 to N

      ---

      ---


endfor
```

Αλγόριθμος Bellman για DAGs

```
for i := 1 to N
    x := Sorted[i]
    if Layer[x] > Layer[s] then
        for each successor y of x such that
                                V[x]+W(x,y)<V[y]
            V[y] := V[x] + W(x,y)
            P[y] := x
        endfor
    endif
endfor
```

Πολυπλοκότητα Bellman για DAGs

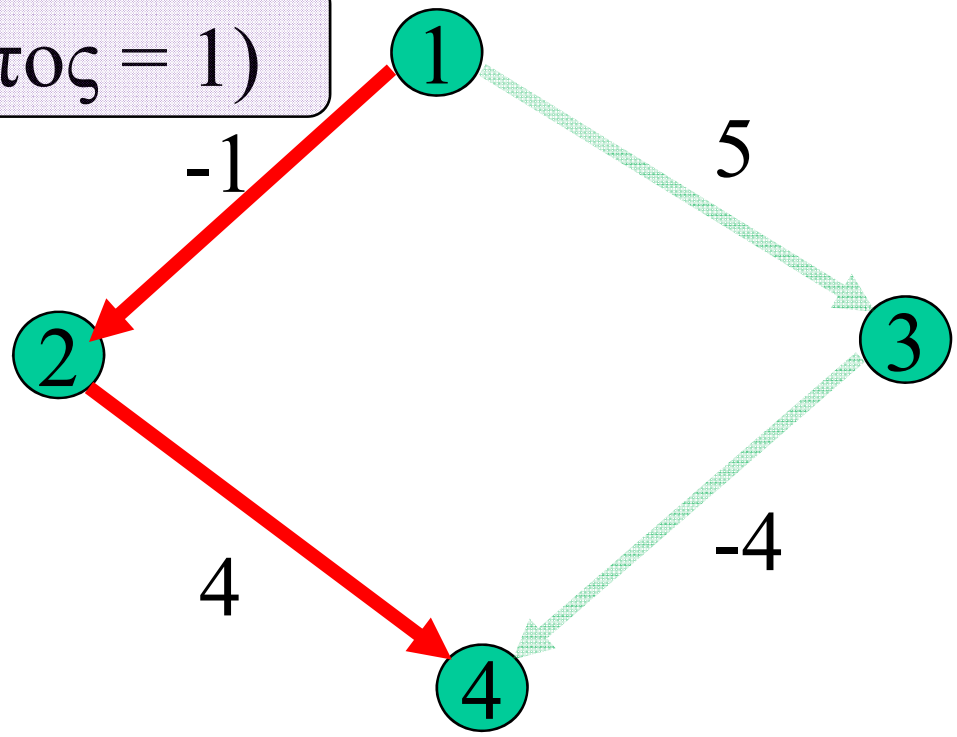
- Κωδικοποίηση: **Λίστες γειτνίασης**
- Topological Sort: **O(m)**
- Συνολική πολυπλοκότητα: **O(m)**

– Τροποποίηση για εύρεση **μέγιστου μονοπατιού** σε DAGs
– Εφαρμογές στο Χρονοπρογραμματισμό (Scheduling)

Αρνητικά βάρη

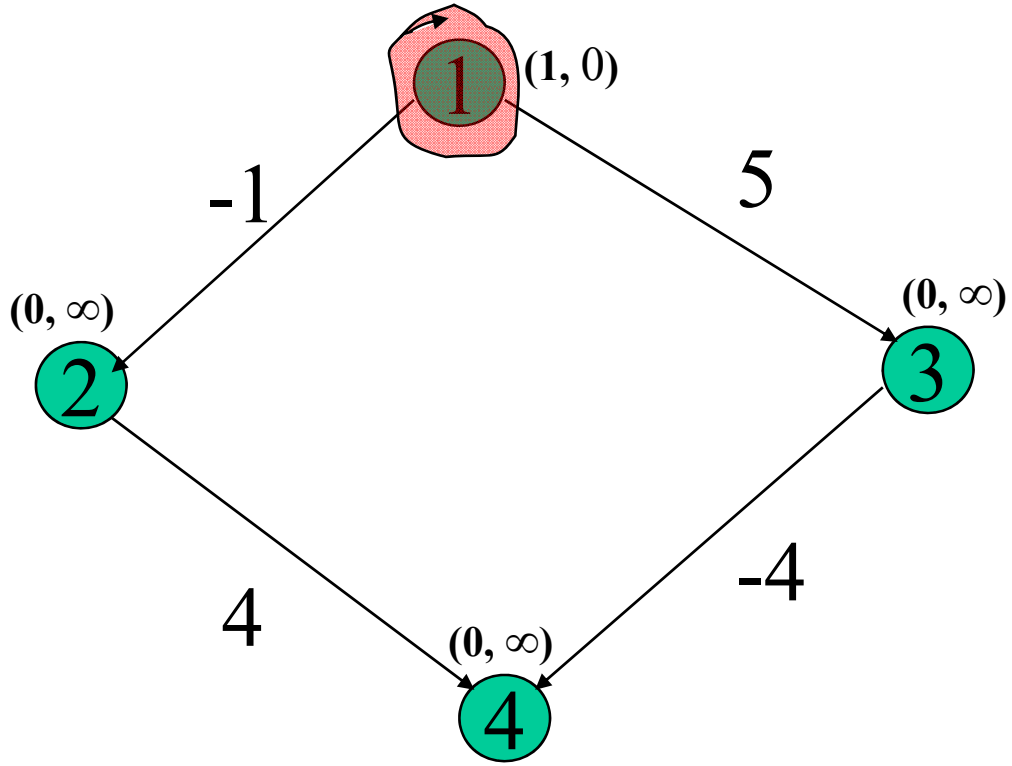
1 ⟶ 2 ⟶ 4 (κόστος = 3)

1 ⟶ 3 ⟶ 4 (κόστος = 1)



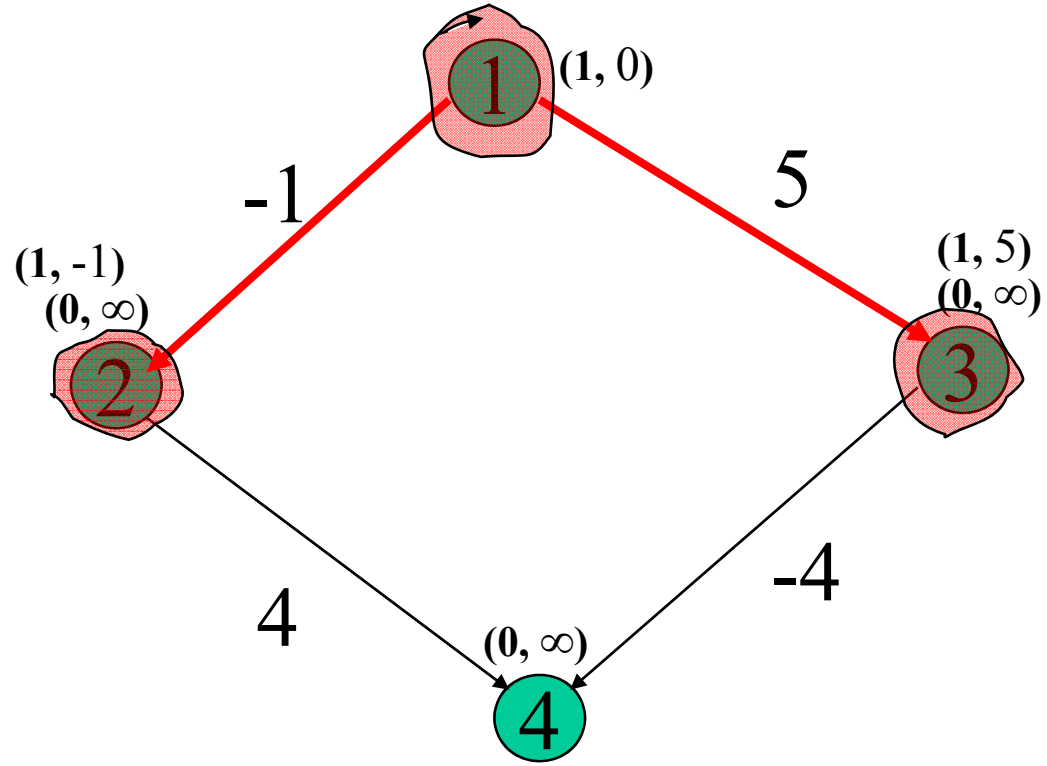
Συνομότερα μονοπάτια από το κόμβο 1 στο κόμβο 4

Αρνητικά βάρη



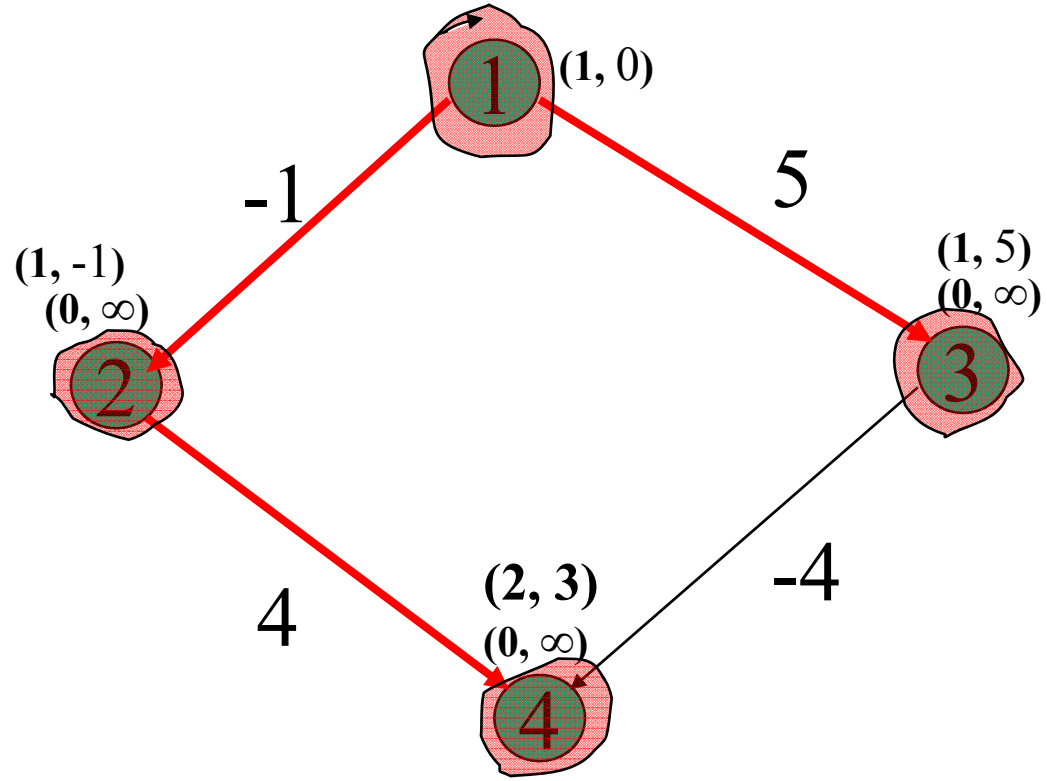
Συνομότερα μονοπάτια από το κόμβο 1 στο κόμβο 4

Αρνητικά βάρη



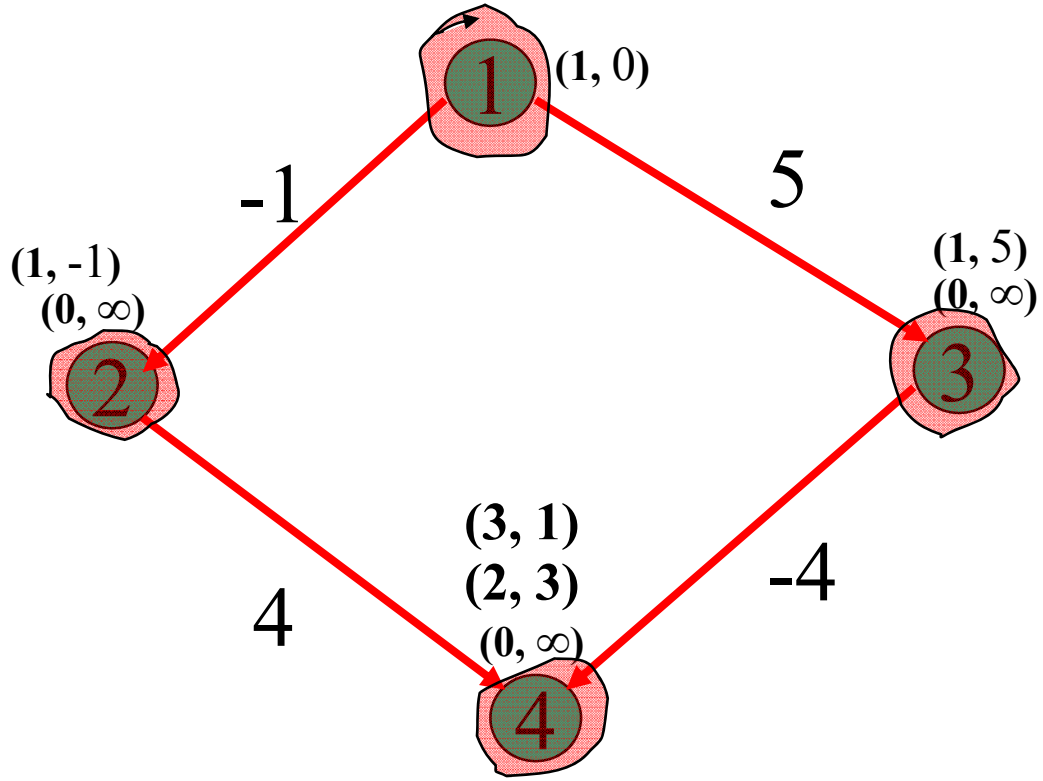
Συνομότερα μονοπάτια από το κόμβο 1 στο κόμβο 4

Αρνητικά βάρη



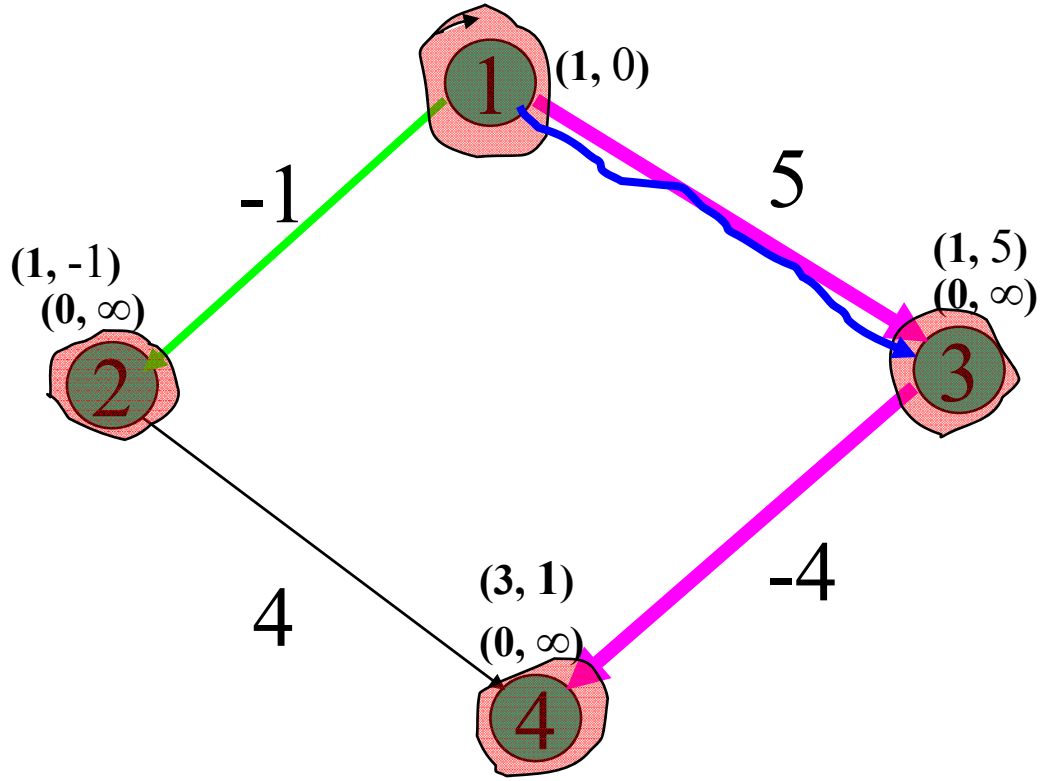
Συνομότερα μονοπάτια από το κόμβο 1 στο κόμβο 4

Αρνητικά βάρη



Συνομότερα μονοπάτια από το κόμβο 1 στο κόμβο 4

Αρνητικά βάρη

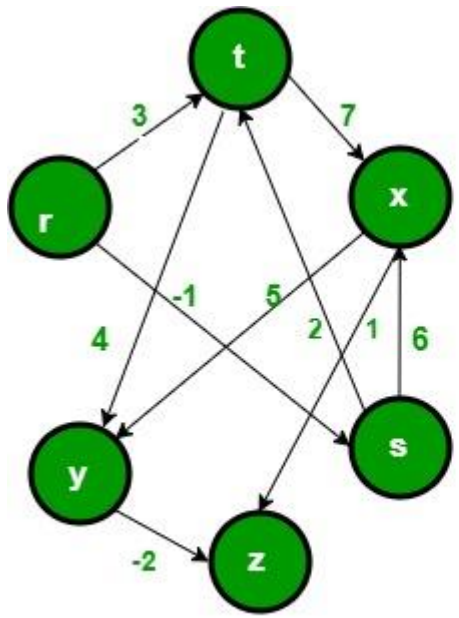


Συνομότερα μονοπάτια από το κόμβο 1 στο κόμβο 4

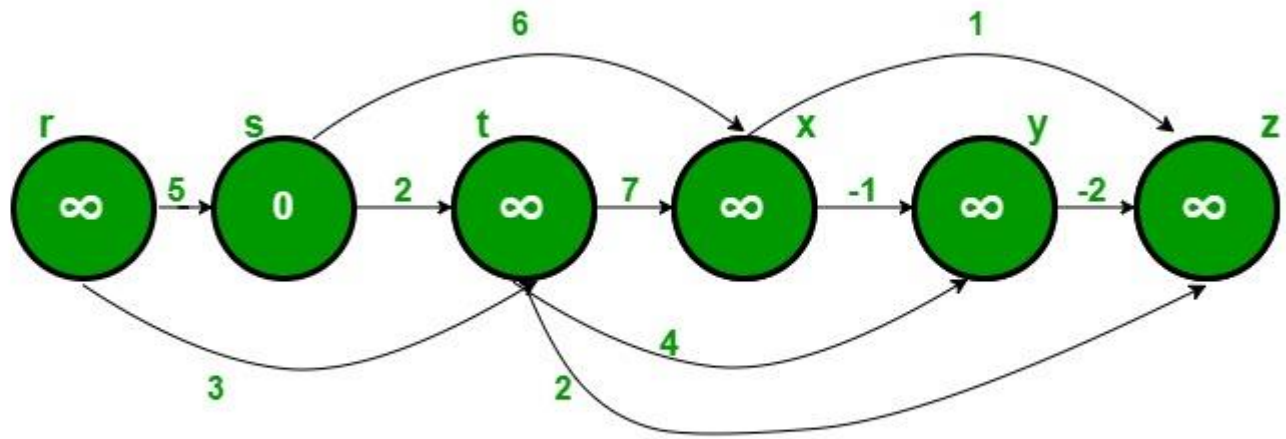
Σύνοψη

Για ένα γενικευμένο γράφο με βάρη, μπορούμε να υπολογίσουμε τις συντομότερες αποστάσεις μίας πηγής σε χρόνο O(VE) χρησιμοποιώντας τον αλγόριθμο Bellman-Ford. Για ένα γράφο χωρίς αρνητικά βάρη, μπορούμε να κάνουμε καλύτερα και να υπολογίσουμε τις συντομότερες αποστάσεις μιας πηγής στο χρόνο O(E + VLogV) χρησιμοποιώντας τον αλγόριθμο της Dijkstra. Για τον Κατευθυνόμενο Κυκλικό Γράφο (DAG), μπορούμε να υπολογίσουμε τις βραχύτερες αποστάσεις ενιαίας πηγής στο χρόνο O(V+E).

Η ιδέα είναι να χρησιμοποιηθεί η τοπολογική ταξινόμηση. Αρχικοποιούμε τις αποστάσεις σε όλους τους κόμβους τόσο άπειρες και την απόσταση από την πηγή (source) σε 0, τότε βρίσκουμε μια τοπολογική ταξινόμηση του γραφήματος. Η τοπολογική ταξινόμηση ενός γραφήματος αντιπροσωπεύει μια γραμμική ταξινόμηση του γράφου (βλέπε παρακάτω, το σχήμα β) είναι μια γραμμική αναπαράσταση του σχήματος (α)). Μόλις έχουμε τοπολογική σειρά (ή γραμμική αναπαράσταση), επεξεργαζόμαστε έναν προς έναν τους κόμβους με τοπολογική σειρά. Για κάθε κορυφή που επεξεργάζεται, ενημερώνουμε τις αποστάσεις του παρακείμενου χρησιμοποιώντας την απόσταση του τρέχοντος κόμβου.(E: πλευρές, V: κόμβοι).



(a)



(b)