Αλγόριθμοι και Πολυπλοκότητα

Αρχοντία Γιαννοπούλου
Όλγα Φουρτουνέλλη

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Δυναμικός Προγραμματισμός

Ανεξάρτητο Σύνολο σε Δέντρα
Ο αλγόριθμος Bellman-Ford

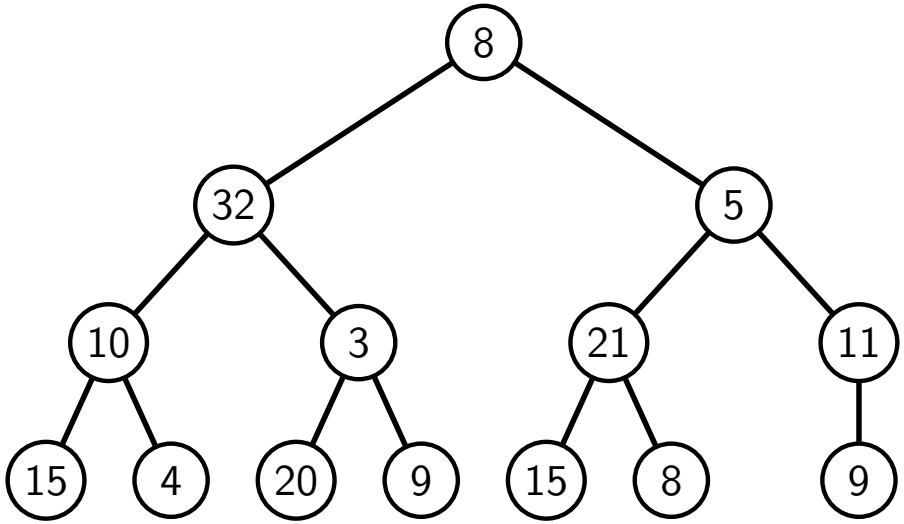
Ας οργανώσουμε ένα πάρτυ...

- Μας ζητάνε να οργανώσουμε ένα πάρτυ για μία εταιρεία που έχει αυστηρή ιεραρχική δομή.
- Η συμμετοχή κάθε υπάλληλου έχει μία (θετική) προσφορά v.
- Δεν θέλουμε στο πάρτυ να καλέσουμε ταυτόχρονα ένα υπάλληλο και τον άμεσο προϊστάμενό του.

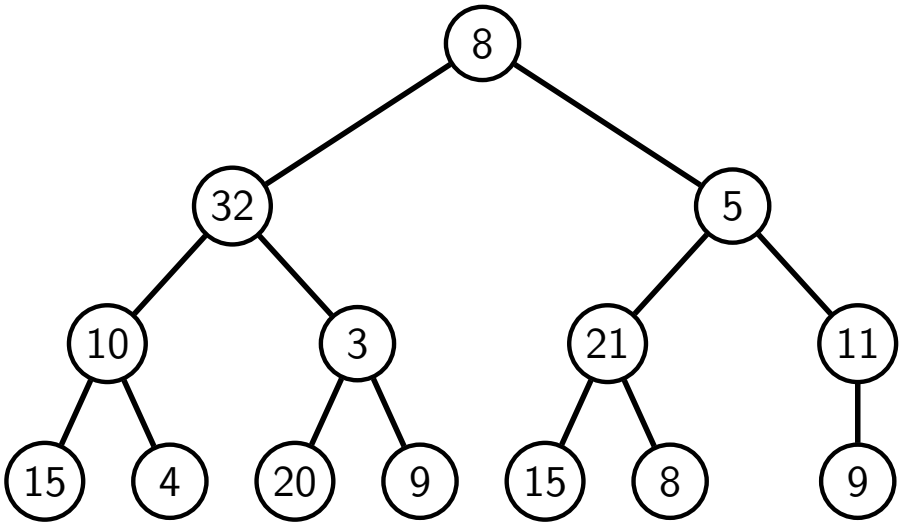
Δεδομένα: ένα δέντρο που περιγράφει ακριβώς την ιεραρχική δομή καθώς και την προσφορά κάθε υπάλληλου.

Ζητούμενο: να φτιάξουμε μία λίστα καλεσμένων με το μεγαλύτερο άθροισμα χιούμορ με βάσει τις παραπάνω προϋποθέσεις.

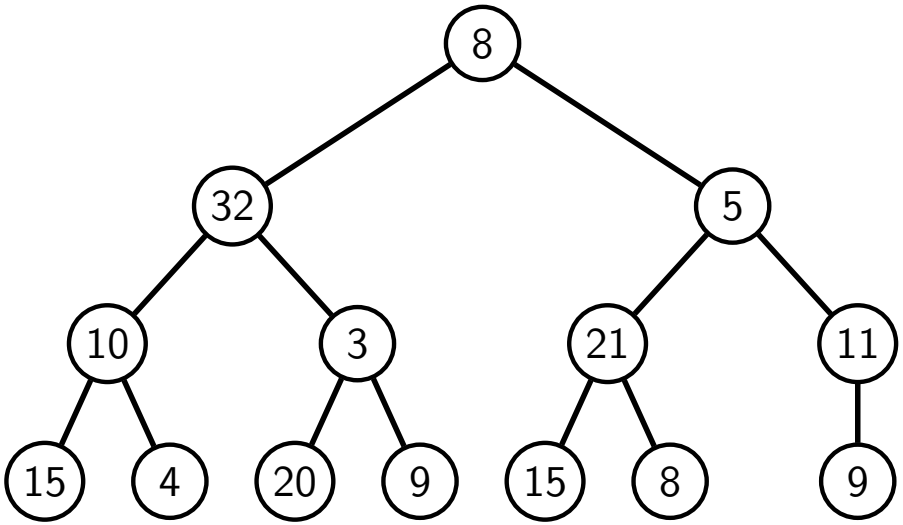
Για παράδειγμα



Ας προσπαθήσουμε να το λύσουμε

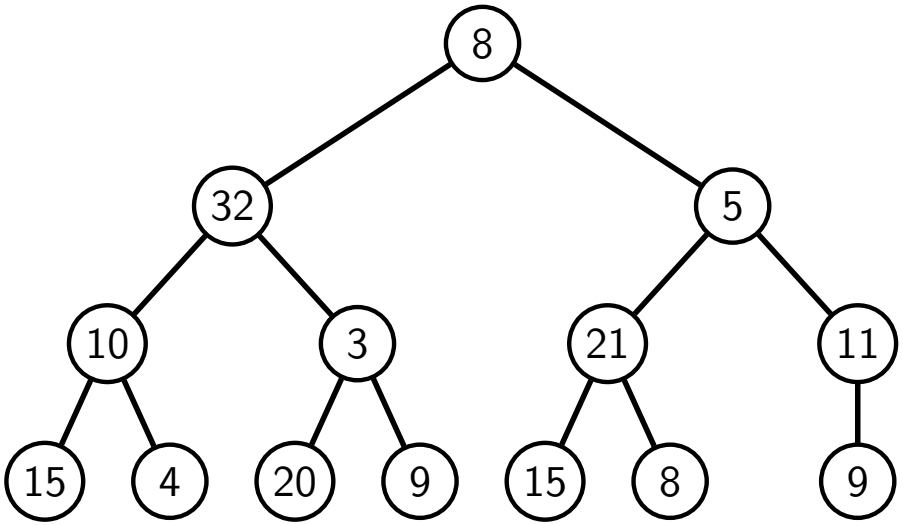


Ας προσπαθήσουμε να το λύσουμε



- Για κάθε υπάλληλο v και τους υφισταμένους του θα υπολογίσω τη μέγιστη προσφορά ενός υποσυνόλου (που πληροί τις προηγούμενες προϋποθέσεις), έστω $A[v]$.

Ας προσπαθήσουμε να το λύσουμε



- Για κάθε υπάλληλο v και τους υφισταμένους του θα υπολογίσω τη μέγιστη προσφορά ενός υποσυνόλου (που πληροί τις προηγούμενες προϋποθέσεις), έστω $A[v]$.
- Για κάθε υπάλληλο v και τους υφισταμένους του θα υπολογίσω τη μέγιστη προσφορά ενός υποσυνόλου (που πληροί τις προηγούμενες προϋποθέσεις) και δεν περιέχει τον v, έστω $B[v]$.

Δομή του προβλήματος

Έστω ένας υπάλληλος v και όλοι οι υφιστάμενοί του $v_1, v_2, \ldots, v_k$. Θέλουμε να υπολογίσουμε τις τιμές $A[v]$ (μέγιστο συνολικά) και $B[v]$ (μέγιστο χωρίς τον v). Για το $B[v]$:

Για το $A[v]$:

Δομή του προβλήματος

Έστω ένας υπάλληλος v και όλοι οι υφιστάμενοί του $v_1, v_2, \ldots, v_k$. Θέλουμε να υπολογίσουμε τις τιμές $A[v]$ (μέγιστο συνολικά) και $B[v]$ (μέγιστο χωρίς τον v). Για το $B[v]$: Αν ο v δεν μπει στη λίστα τότε δεν μπλοκάρει κανέναν από τους άμεσα ή έμμεσα) υφιστάμενους του.

Για το $A[v]$:

Δομή του προβλήματος

Έστω ένας υπάλληλος v και όλοι οι υφιστάμενοί του $v_1, v_2, \ldots, v_k$. Θέλουμε να υπολογίσουμε τις τιμές $A[v]$ (μέγιστο συνολικά) και $B[v]$ (μέγιστο χωρίς τον v). Για το $B[v]$: Αν ο v δεν μπει στη λίστα τότε δεν μπλοκάρει κανέναν από τους άμεσα ή έμμεσα) υφιστάμενους του.

Για το $A[v]$:

- Ο v δεν μπαίνει στη λίστα.
- Ο v μπαίνει στη λίστα.

Δομή του προβλήματος

Έστω ένας υπάλληλος v και όλοι οι υφιστάμενοί του $v_1, v_2, \ldots, v_k$. Θέλουμε να υπολογίσουμε τις τιμές $A[v]$ (μέγιστο συνολικά) και $B[v]$ (μέγιστο χωρίς τον v). Για το $B[v]$: Αν ο v δεν μπει στη λίστα τότε δεν μπλοκάρει κανέναν από τους άμεσα ή έμμεσα) υφιστάμενους του. Μπορούν εν δυνάμει όλοι οι άμεσοι υφιστάμενοι του να μπουν στην λίστα. Άρα

$$B[v] = \sum_{i=1}^{k} A[v_i].$$

Για το $A[v]$:

- Ο v δεν μπαίνει στη λίστα.
- Ο v μπαίνει στη λίστα.

Δομή του προβλήματος

Έστω ένας υπάλληλος v και όλοι οι υφιστάμενοί του $v_1, v_2, \ldots, v_k$. Θέλουμε να υπολογίσουμε τις τιμές $A[v]$ (μέγιστο συνολικά) και $B[v]$ (μέγιστο χωρίς τον v). Για το $B[v]$: Αν ο v δεν μπει στη λίστα τότε δεν μπλοκάρει κανέναν από τους άμεσα ή έμμεσα) υφιστάμενους του. Μπορούν εν δυνάμει όλοι οι άμεσοι υφιστάμενοι του να μπουν στην λίστα. Άρα

$$B[v] = \sum_{i=1}^{k} A[v_i].$$

Για το $A[v]$:

- Ο v δεν μπαίνει στη λίστα. Αυτή είναι η τιμή $B[v]$.
- Ο v μπαίνει στη λίστα.

Δομή του προβλήματος

Έστω ένας υπάλληλος v και όλοι οι υφιστάμενοί του $v_1, v_2, \ldots, v_k$. Θέλουμε να υπολογίσουμε τις τιμές $A[v]$ (μέγιστο συνολικά) και $B[v]$ (μέγιστο χωρίς τον v). Για το $B[v]$: Αν ο v δεν μπει στη λίστα τότε δεν μπλοκάρει κανέναν από τους άμεσα ή έμμεσα) υφιστάμενους του. Μπορούν εν δυνάμει όλοι οι άμεσοι υφιστάμενοι του να μπουν στην λίστα. Άρα

$$B[v] = \sum_{i=1}^{k} A[v_i].$$

Για το $A[v]$:

- Ο v δεν μπαίνει στη λίστα. Αυτή είναι η τιμή $B[v]$.
- Ο v μπαίνει στη λίστα.
  Τότε κανένας από τους άμεσα υφιστάμενους του $v_1, v_2, \ldots, v_k$ δεν θα μπουν στη λίστα.

Δομή του προβλήματος

Έστω ένας υπάλληλος v και όλοι οι υφιστάμενοί του $v_1, v_2, \ldots, v_k$. Θέλουμε να υπολογίσουμε τις τιμές $A[v]$ (μέγιστο συνολικά) και $B[v]$ (μέγιστο χωρίς τον v). Για το $B[v]$: Αν ο v δεν μπει στη λίστα τότε δεν μπλοκάρει κανέναν από τους άμεσα ή έμμεσα) υφιστάμενους του. Μπορούν εν δυνάμει όλοι οι άμεσοι υφιστάμενοι του να μπουν στην λίστα. Άρα

$$B[v] = \sum_{i=1}^{k} A[v_i].$$

Για το $A[v]$:

- Ο v δεν μπαίνει στη λίστα. Αυτή είναι η τιμή $B[v]$.
- Ο v μπαίνει στη λίστα.
  Τότε κανένας από τους άμεσα υφιστάμενους του $v_1, v_2, \ldots, v_k$ δεν θα μπουν στη λίστα.
  Αυτή η τιμή είναι η

$$\chi(v) + \sum_{i=1}^{k} B[v_i].$$

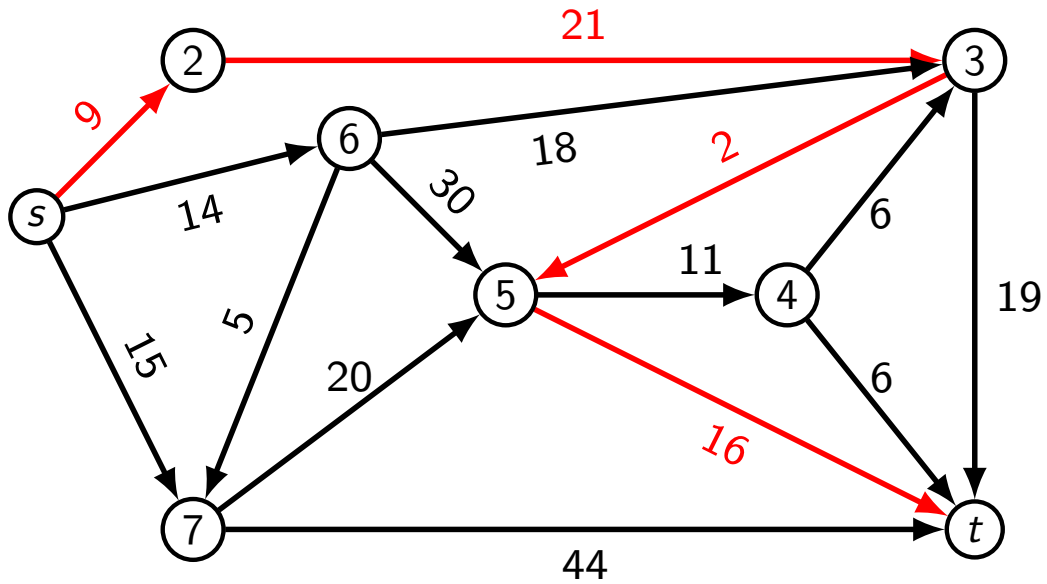
Συντομότερη Διαδρομή με Βάρη

Δεδομένα:

- Κατευθυνόμενο γράφημα $G = (V, A)$.
- Αφετηρία s, προορισμός t.
- ℓ_e πραγματικός αριθμός = μήκος της ακμής e.

Κόστος διαδρομής: άθροισμα μηκών των ακμών της διαδρομής.

Πρόβλημα συντομότερης διαδρομής: Να βρούμε τη συντομότερη σε μήκος κατευθυνόμενη διαδρομή από την κορυφή s στην κορυφή t.



Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

- Διατήρησε ένα σύνολο S εξερευνημένων κορυφών $\{u\}$ για το οποίο έχουμε προσδιορίσει την απόσταση $d(u)$ της συντομότερης διαδρομής από την s στην u.

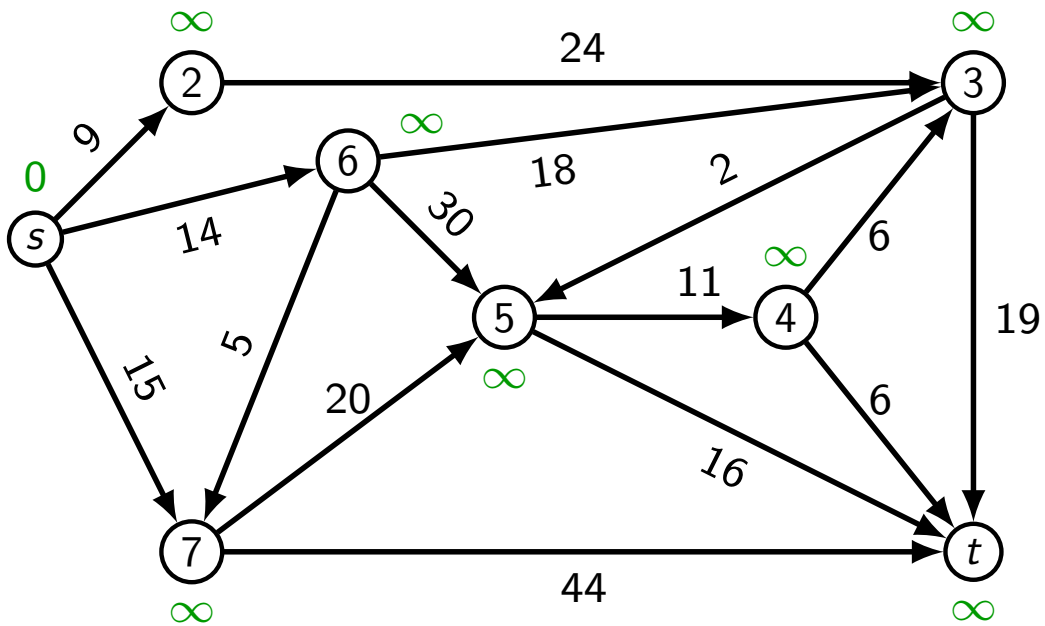
- Αρχικοποίησε $S = \{s\}$, $d(s) = 0$.

- Επαναληπτικά, διάλεξε την κορυφή v που δεν έχει εξερευνηθεί και ελαχιστοποιεί την τρέχουσα ελάχιστη απόσταση $\pi(v) = d(u) + \ell_{uv}$ από τις διαδρομές που περιέχουν την συντομότερη διαδρομή προς u στο S, ακολουθούμενη από ακμή $e = (u, v)$.

- Θέσε $d(v) = \pi(v)$ και πρόσθεσε την v στο S.

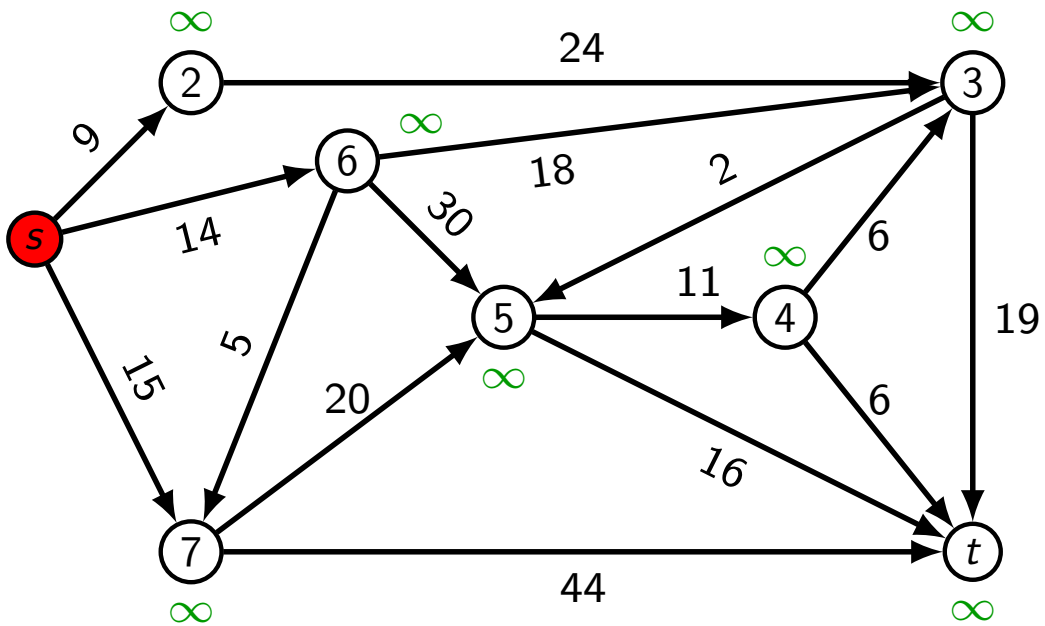
Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

$$\begin{aligned} S &= \{\} \\ PQ &= \{s, 2, 3, 4, 5, 6, 7, t\} \end{aligned}$$



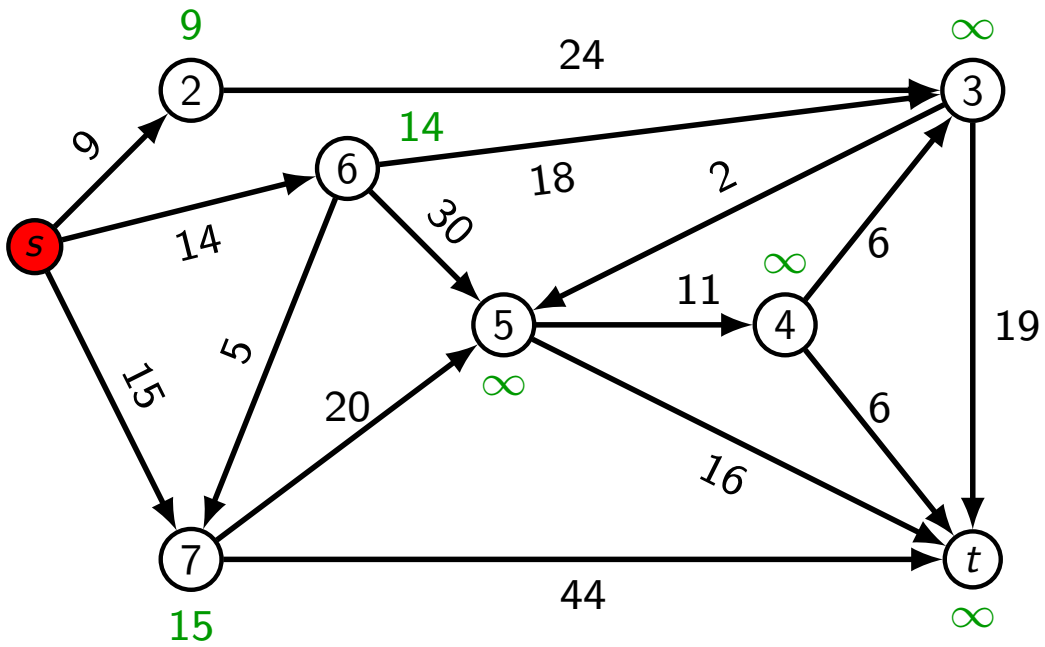
Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

$$\begin{aligned} S &= \{s\} \\ PQ &= \{2, 3, 4, 5, 6, 7, t\} \end{aligned}$$



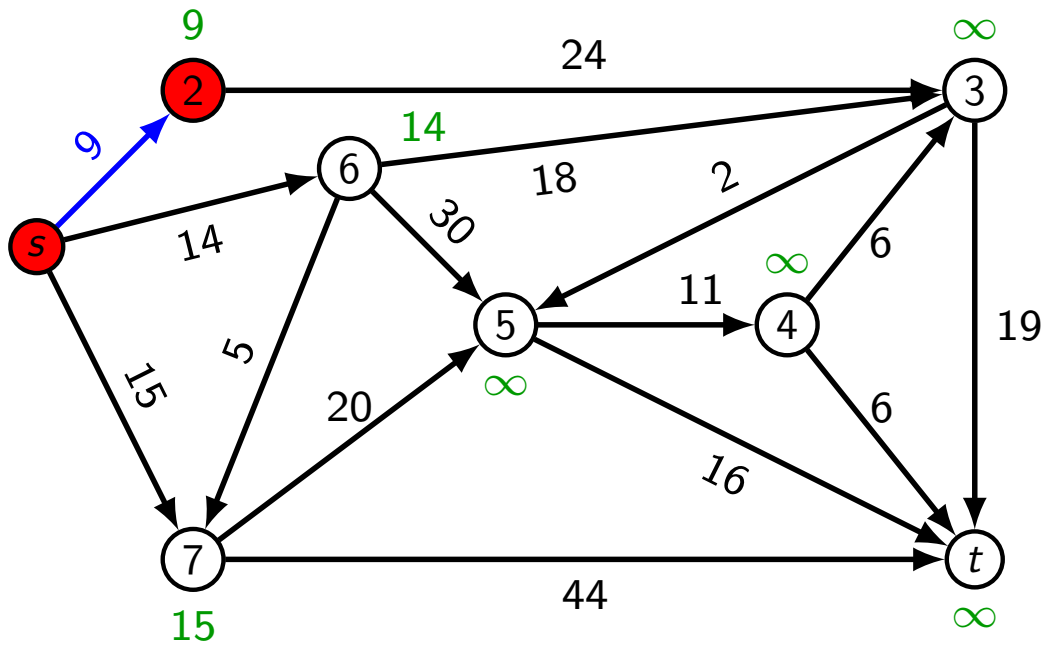
Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

$$\begin{aligned} S &= \{s\} \\ PQ &= \{2, 3, 4, 5, 6, 7, t\} \end{aligned}$$



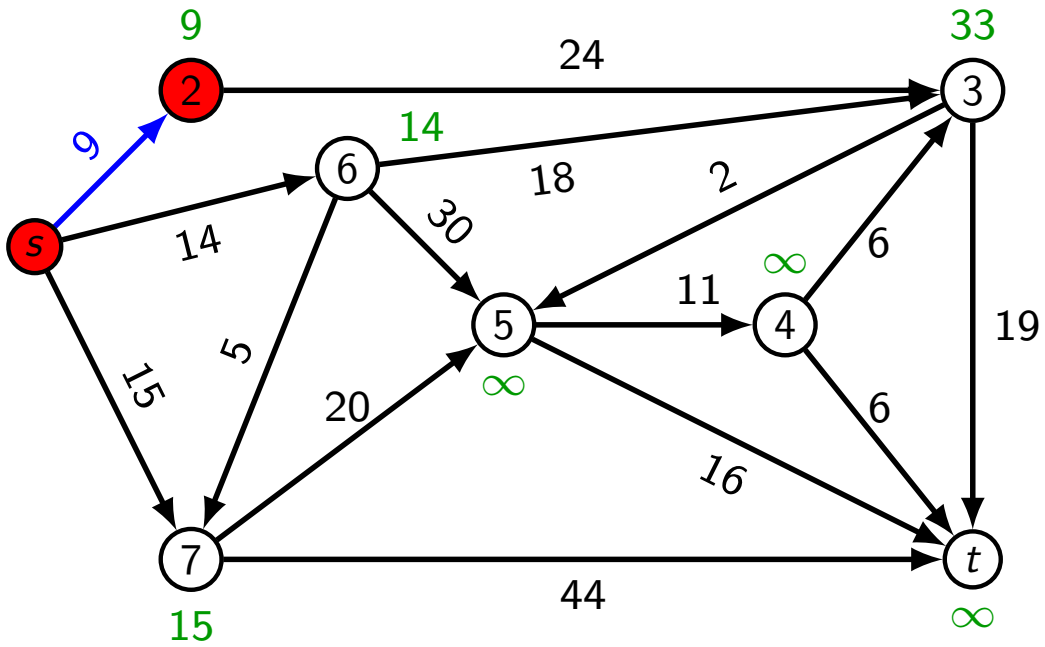
Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

$$S = \{s, 2\}$$
$$PQ = \{3, 4, 5, 6, 7, t\}$$



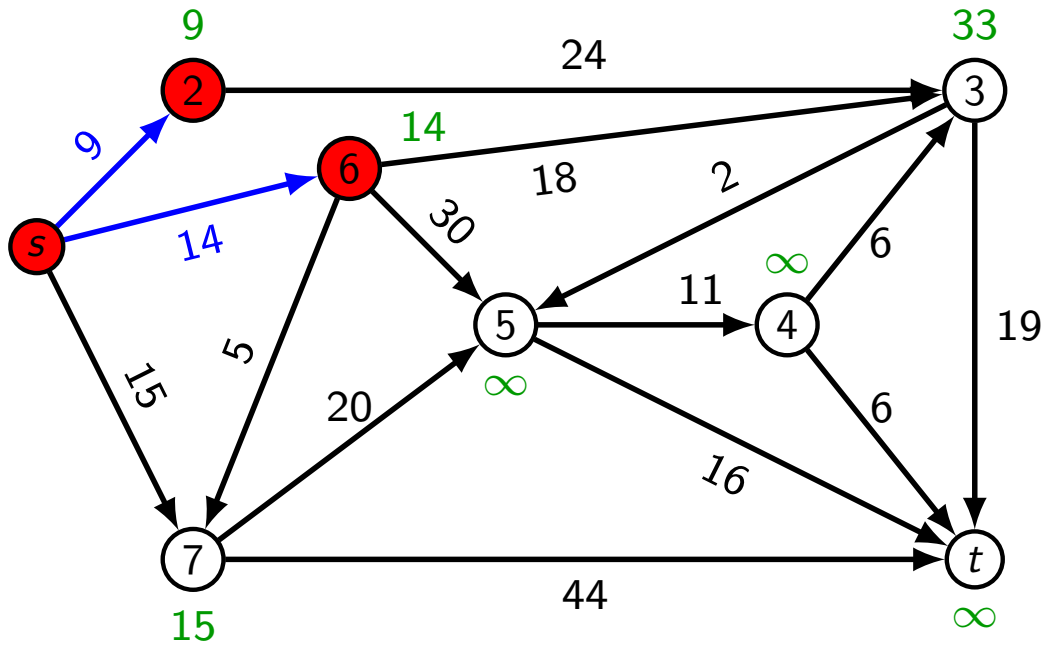
Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

$$\begin{aligned} S &= \{s, 2\} \\ PQ &= \{3, 4, 5, 6, 7, t\} \end{aligned}$$



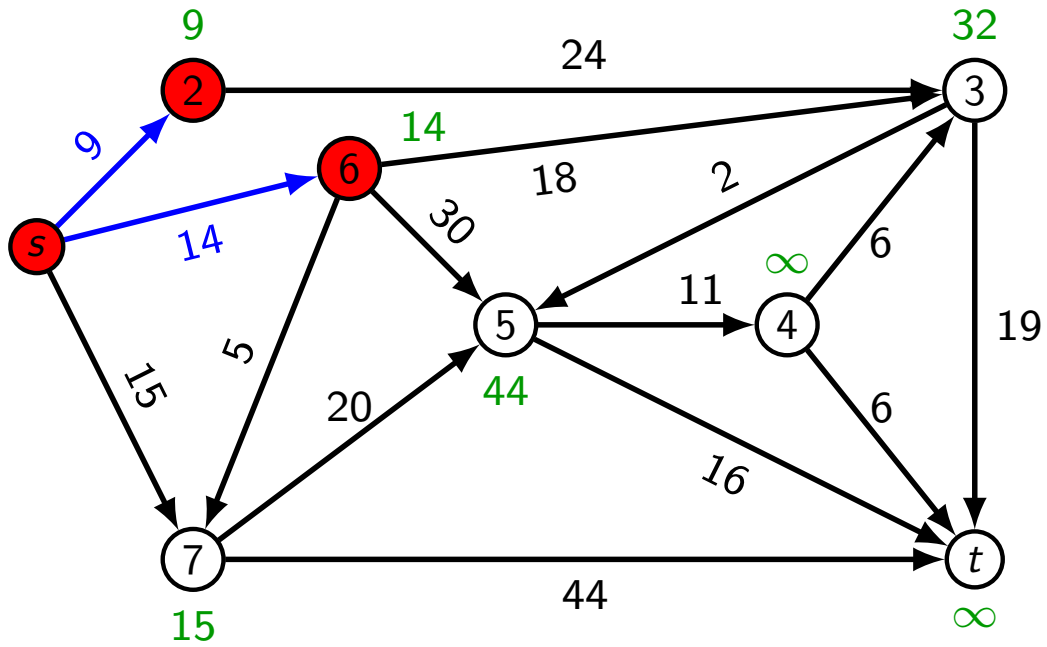
Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

$$\begin{aligned} S &= \{s, 2, 6\} \\ PQ &= \{3, 4, 5, 7, t\} \end{aligned}$$



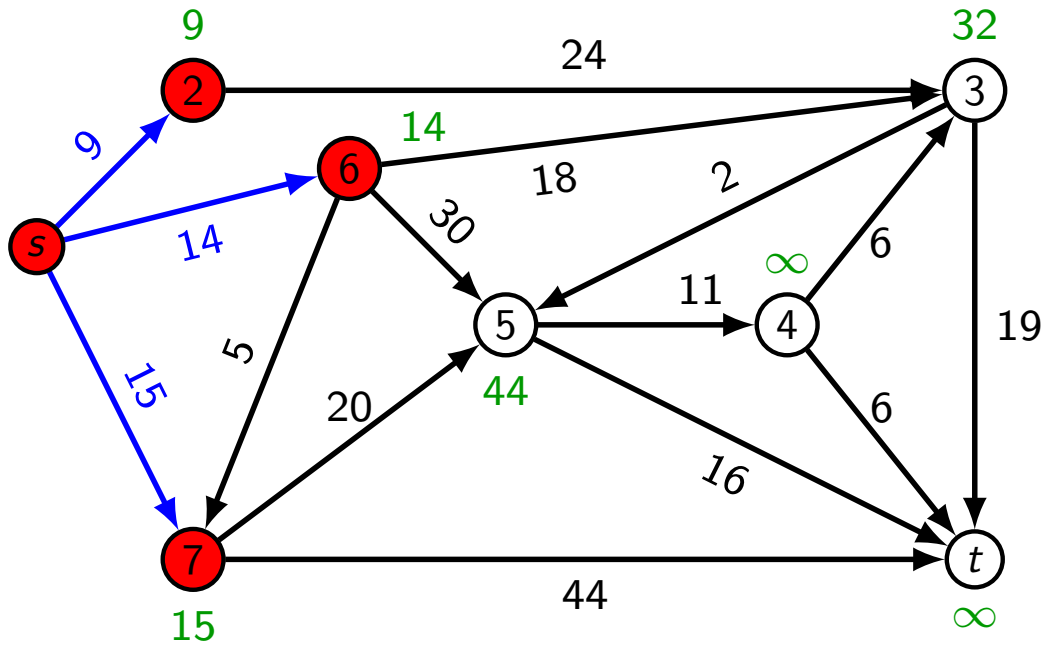
Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

$$\begin{aligned} S &= \{s, 2, 6\} \\ PQ &= \{3, 4, 5, 7, t\} \end{aligned}$$



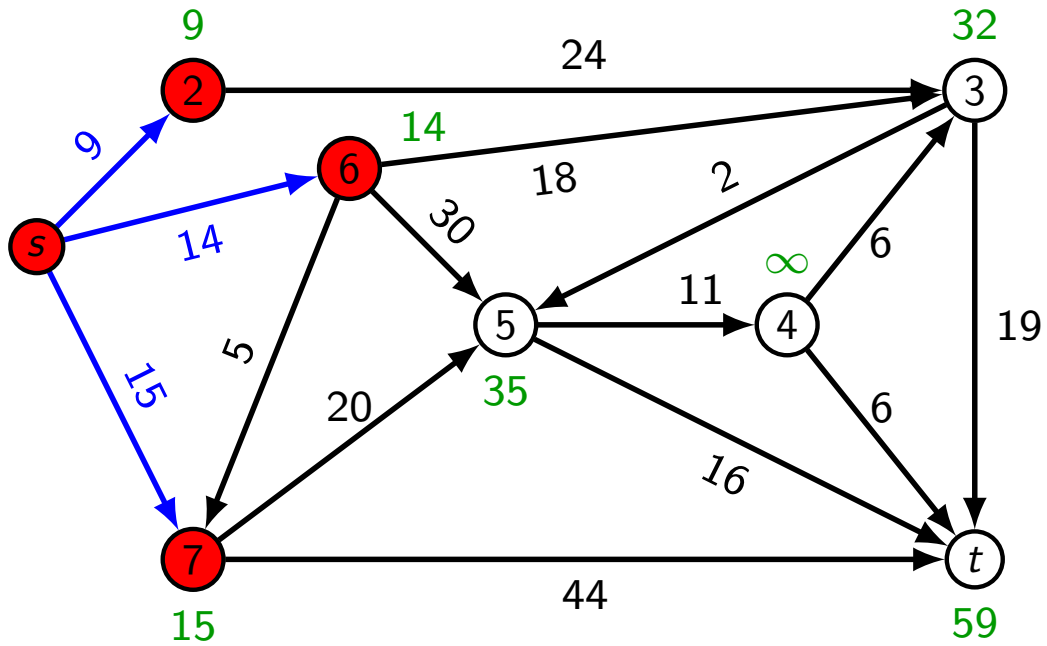
Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

$$S = \{s, 2, 6, 7\}$$
$$PQ = \{3, 4, 5, t\}$$



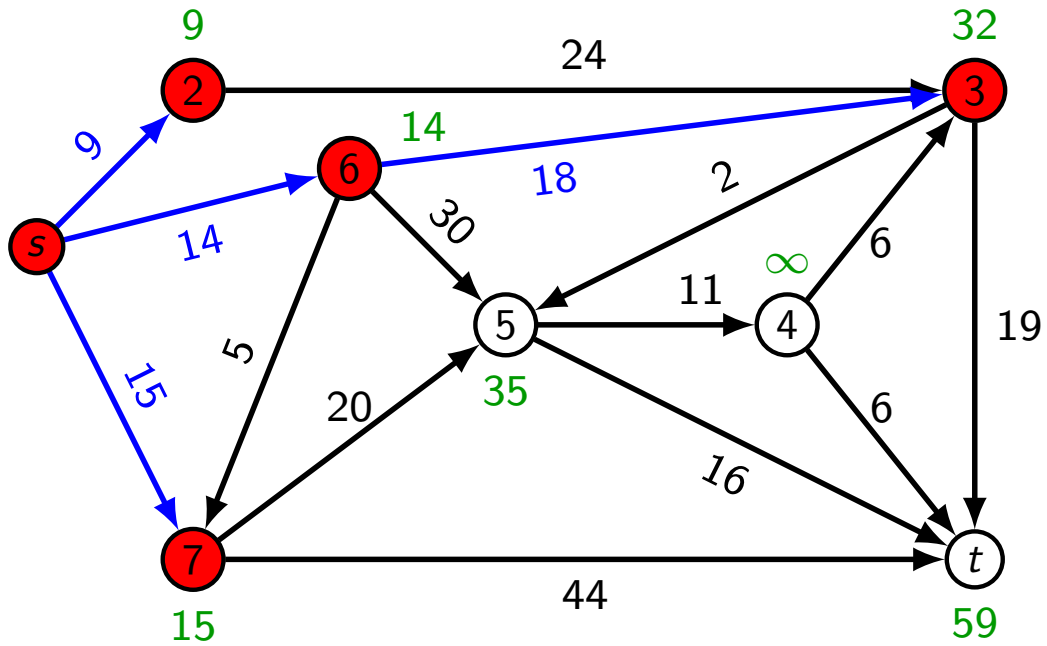
Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

$$S = \{s, 2, 6, 7\}$$
$$PQ = \{3, 4, 5, t\}$$



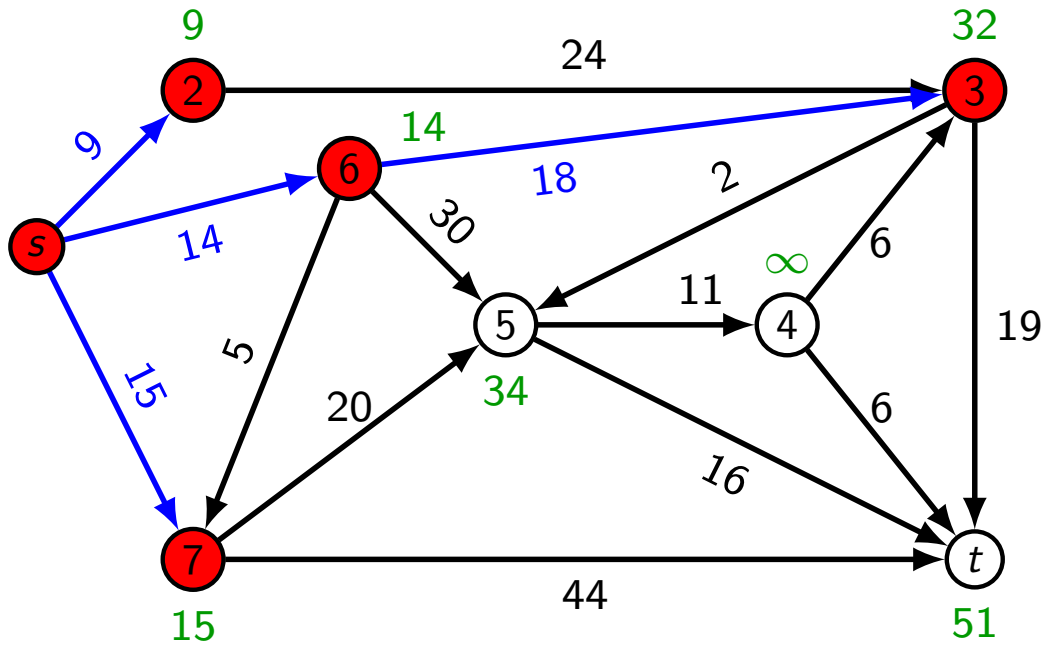
Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

$$S = \{s, 2, 6, 7, 3\}$$
$$PQ = \{4, 5, t\}$$



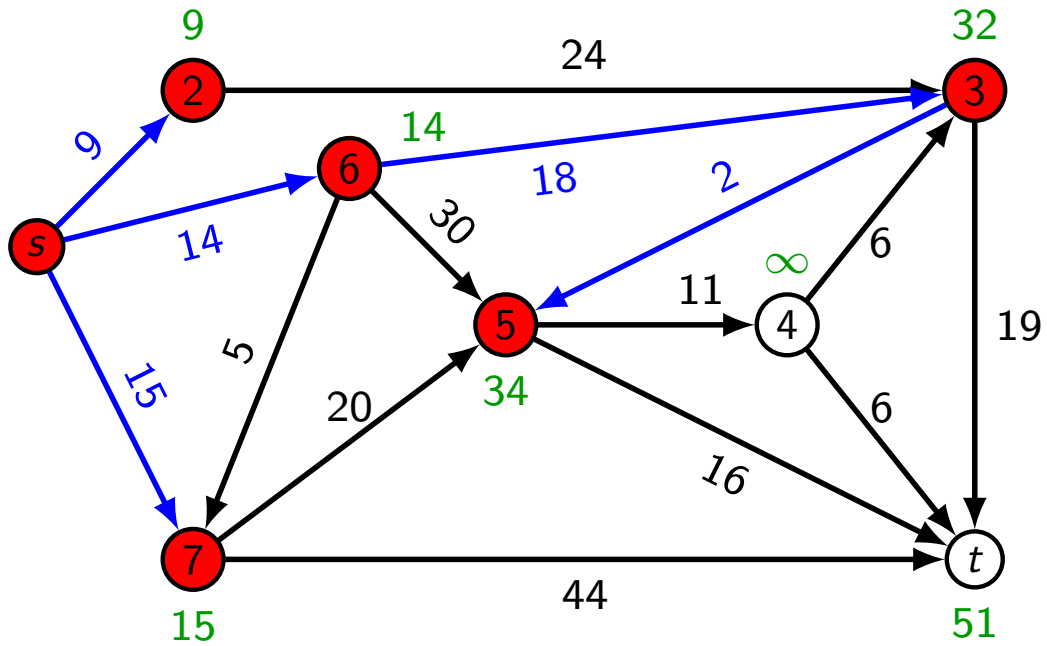
Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

$$S = \{s, 2, 6, 7, 3\}$$
$$PQ = \{4, 5, t\}$$



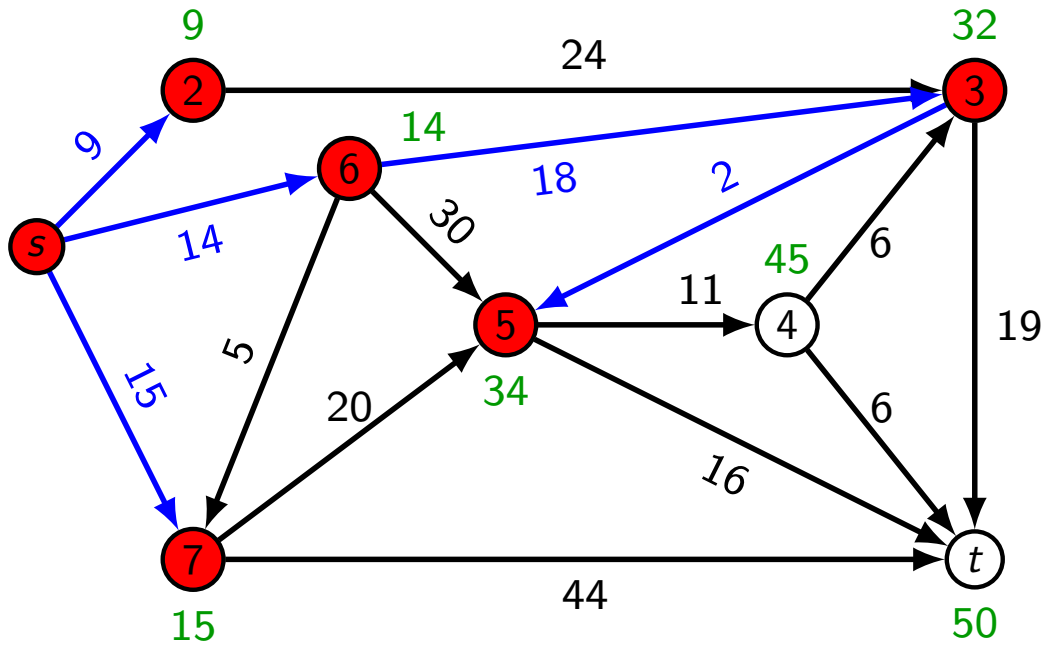
Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

$$S = \{s, 2, 6, 7, 3, 5\}$$
$$PQ = \{4, t\}$$



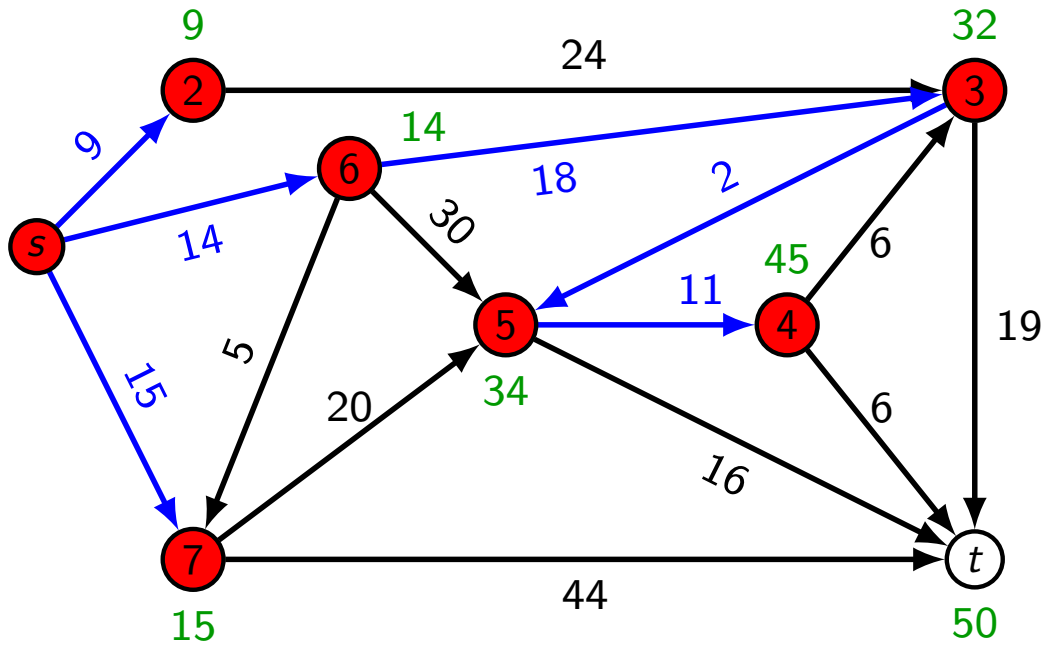
Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

$$S = \{s, 2, 6, 7, 3, 5\}$$
$$PQ = \{4, t\}$$



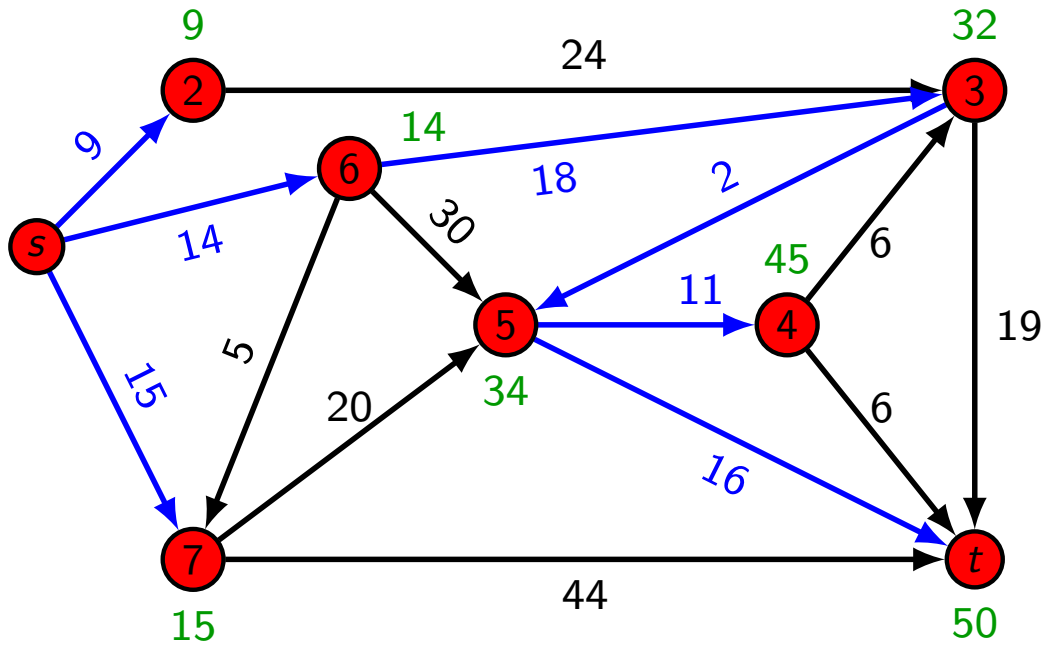
Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

$$S = \{s, 2, 6, 7, 3, 5, 4\}$$
$$PQ = \{t\}$$

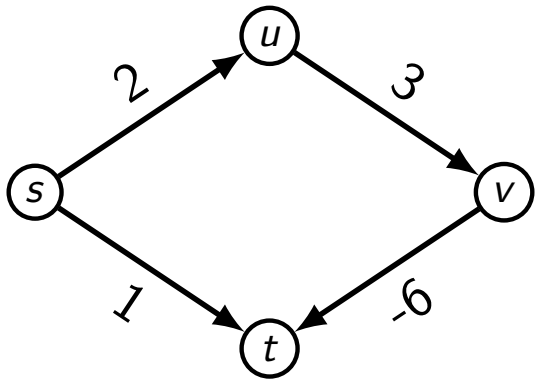


Μόνο θετικά βάρη - Αλγόριθμος Dijkstra

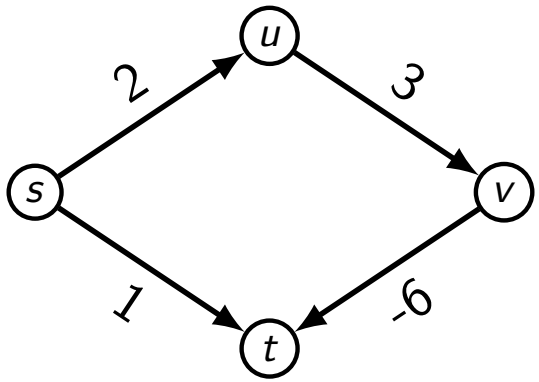
$$S = \{s, 2, 6, 7, 3, 5, 4, t\}$$
$$PQ = \{\}$$



Αν έχουμε και αρνητικά βάρη;



Αν έχουμε και αρνητικά βάρη;



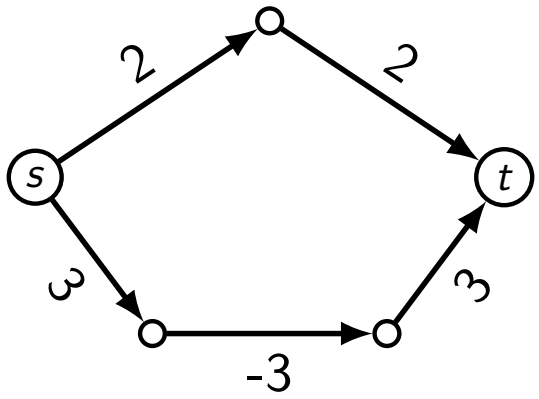
Ο αλγόριθμος του Dijkstra μπορεί να αποτύχει.

Απόπειρα 2: Προσθέτουμε σε όλα τα βάρη μία σταθερά

Αν προσθέσουμε κατάλληλη σταθερά σε όλα τα βάρη ώστε να γίνουν θετικά;

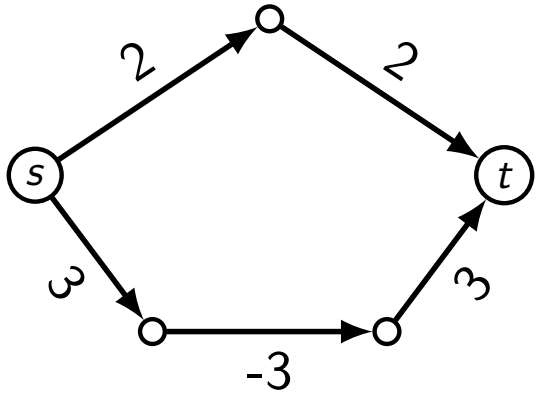
Απόπειρα 2: Προσθέτουμε σε όλα τα βάρη μία σταθερά

Αν προσθέσουμε κατάλληλη σταθερά σε όλα τα βάρη ώστε να γίνουν θετικά;



Απόπειρα 2: Προσθέτουμε σε όλα τα βάρη μία σταθερά

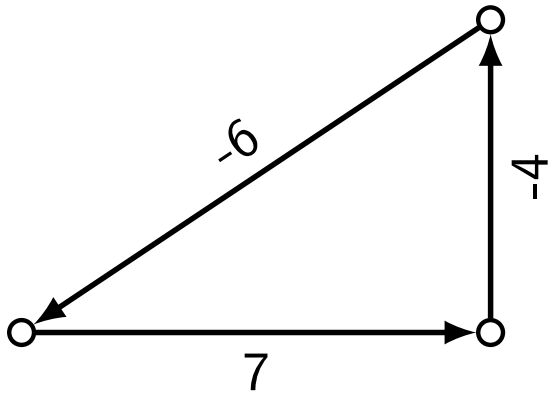
Αν προσθέσουμε κατάλληλη σταθερά σε όλα τα βάρη ώστε να γίνουν θετικά;



Ο αλγόριθμός μας μπορεί να αποτύχει.

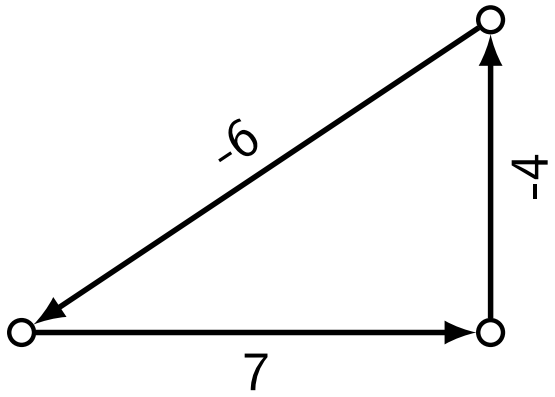
Αρνητικοί κύκλοι

Αρνητικός κύκλος: Το άθροισμα των βαρών των ακμών του είναι αρνητικό.



Αρνητικοί κύκλοι

Αρνητικός κύκλος: Το άθροισμα των βαρών των ακμών του είναι αρνητικό.



Εάν υπάρχει διαδρομή από κορυφή s σε κορυφή t η οποία περιέχει αρνητικό κύκλο τότε δεν υπάρχει συντομότερη $s - t$ διαδρομή. Διαφορετικά υπάρχει και είναι απλή.

Δομή του προβλήματος

$\mathrm{OPT}(i, v) =$ μήκος του συντομότερου $v - t$ μονοπατιού που χρησιμοποιεί το πολύ i ακμές.

Δομή του προβλήματος

$\mathrm{OPT}(i, v) =$ μήκος του συντομότερου $v - t$ μονοπατιού που χρησιμοποιεί το πολύ i ακμές.

- Το P έχει το πολύ $i - 1$ ακμές.

- Το P έχει ακριβώς i ακμές.

Δομή του προβλήματος

$\mathrm{OPT}(i, v) =$ μήκος του συντομότερου $v - t$ μονοπατιού που χρησιμοποιεί το πολύ i ακμές.

- Το P έχει το πολύ $i - 1$ ακμές.
  Τότε $\mathrm{OPT}(i, v) = \mathrm{OPT}(i - 1, v)$.

- Το P έχει ακριβώς i ακμές.

Δομή του προβλήματος

$\mathrm{OPT}(i, v) =$ μήκος του συντομότερου $v - t$ μονοπατιού που χρησιμοποιεί το πολύ i ακμές.

- Το P έχει το πολύ $i - 1$ ακμές.
  Τότε $\mathrm{OPT}(i, v) = \mathrm{OPT}(i - 1, v)$.

- Το P έχει ακριβώς i ακμές.
  Έστω ότι (v, w) είναι η πρώτη ακμή του μονοπατιού. Τότε αυτό αποτελείται από την (v, w) και το συντομότερο $w - t$ μονοπάτι που χρησιμοποιεί $i - 1$ ακμές.

Δομή του προβλήματος

$\mathrm{OPT}(i, v)$ = μήκος του συντομότερου $v - t$ μονοπατιού που χρησιμοποιεί το πολύ i ακμές.

- Το P έχει το πολύ $i - 1$ ακμές.
  Τότε $\mathrm{OPT}(i, v) = \mathrm{OPT}(i - 1, v)$.

- Το P έχει ακριβώς i ακμές.
  Έστω ότι (v, w) είναι η πρώτη ακμή του μονοπατιού. Τότε αυτό αποτελείται από την (v, w) και το συντομότερο $w - t$ μονοπάτι που χρησιμοποιεί $i - 1$ ακμές.

Αν δεν έχουμε αρνητικούς κύκλους η συντομότερη διαδρομή δεν έχει επαναλήψεις κορυφών. Άρα έχει το πολύ $n - 1$ ακμές.

Δομή του προβλήματος

$\mathrm{OPT}(i, v)$ = μήκος του συντομότερου $v - t$ μονοπατιού που χρησιμοποιεί το πολύ i ακμές.

- Το P έχει το πολύ $i - 1$ ακμές.
  Τότε $\mathrm{OPT}(i, v) = \mathrm{OPT}(i - 1, v)$.

- Το P έχει ακριβώς i ακμές.
  Έστω ότι (v, w) είναι η πρώτη ακμή του μονοπατιού. Τότε αυτό αποτελείται από την (v, w) και το συντομότερο $w - t$ μονοπάτι που χρησιμοποιεί $i - 1$ ακμές.

Αν δεν έχουμε αρνητικούς κύκλους η συντομότερη διαδρομή δεν έχει επαναλήψεις κορυφών. Άρα έχει το πολύ $n - 1$ ακμές.

Τί υπολογίζει η τιμή $M[n - 1, v]$ για κάποια κορυφή v;

Συνοψίζοντας

$$\mathrm{OPT}(i, v) = \begin{cases} 0 & i = 0 \\ \min\{\mathrm{OPT}(i-1, v), \min\limits_{(v,w)\in A} (\ell_{vw} + \mathrm{OPT}(i-1, w))\} & i > 0 \end{cases}$$

Αλγόριθμος

Συντομότερο Μονοπάτι(G, t)

Για κάθε $v \in V$
    $M[0, v] = \infty$
$M[0, t] = 0$

Για κάθε $i = 1$ έως $n - 1$
    Για κάθε $v \in V$
        $M[i, v] = M[i - 1, v]$
    Για κάθε $(v, w) \in A$
        $M[i, v] = \min(M[i, v], \ell_{vw} + M[i - 1, w])$

Αλγόριθμος

Συντομότερο Μονοπάτι(G, t)

Για κάθε $v \in V$
    $M[0, v] = \infty$
$M[0, t] = 0$

Για κάθε $i = 1$ έως $n - 1$
    Για κάθε $v \in V$
        $M[i, v] = M[i - 1, v]$
    Για κάθε $(v, w) \in A$
        $M[i, v] = \min(M[i, v], \ell_{vw} + M[i - 1, w])$

Πολυπλοκότητα: $\Theta(m \cdot n)$ χρόνο και $\Theta(n^2)$ χώρο.

Βελτιώσεις

- Διατηρούμε τον μονοδιάστατο πίνακα $M[v]$ = το μήκος του συντομότερου $v - t$ μονοπατιού που έχουμε βρει εως τώρα.

- Δεν ελέγχουμε ακμές v, w εκτός και αν το $M[w]$ άλλαξε την προηγούμενη φορά που το προσπελάσαμε.

Βελτιώσεις

- Διατηρούμε τον μονοδιάστατο πίνακα $M[v]$ = το μήκος του συντομότερου $v - t$ μονοπατιού που έχουμε βρει εως τώρα.

- Δεν ελέγχουμε ακμές v, w εκτός και αν το $M[w]$ άλλαξε την προηγούμενη φορά που το προσπελάσαμε.

Κατά τη διάρκεια που τρέχει ο αλγόριθμος το $M[v]$ είναι το μήκος κάποιου $v - t$ μονοπατιού. Μετά από i γύρους, η τιμή $M[v]$ δεν είναι μεγαλύτερη από το μήκος του συντομότερου $v - t$ μονοπατιού με το πολύ i ακμές.

Βελτιώσεις

- Διατηρούμε τον μονοδιάστατο πίνακα $M[v]$ = το μήκος του συντομότερου $v - t$ μονοπατιού που έχουμε βρει εως τώρα.

- Δεν ελέγχουμε ακμές v, w εκτός και αν το $M[w]$ άλλαξε την προηγούμενη φορά που το προσπελάσαμε.

Κατά τη διάρκεια που τρέχει ο αλγόριθμος το $M[v]$ είναι το μήκος κάποιου $v - t$ μονοπατιού. Μετά από i γύρους, η τιμή $M[v]$ δεν είναι μεγαλύτερη από το μήκος του συντομότερου $v - t$ μονοπατιού με το πολύ i ακμές.

Πολυπλοκότητα: $\mathcal{O}(m + n)$ χώρος και $\mathcal{O}(mn)$ χρόνος

Συντομότερη διαδρομή (ξανά)

Συντομότερο Μονοπάτι(G, s, t)

Για κάθε $v \in V$
$M[v] = \infty$
$S[v] = \emptyset$
$M[t] = 0$

Για κάθε $i = 1$ έως $n - 1$
Για κάθε $w \in V$
Εάν η τιμή του $M[w]$ ανανεώθηκε στην προηγούμενη επανάληψη
Για κάθε $(v, w) \in A$
Εάν $(M[v] > M[w] + \ell_{vw})$
$M[v] = M[w] + \ell_{vw}$
$S[v] = w$

Εάν καμμία τιμή δεν άλλαξε στην επανάληψη i, σταματάμε.

Παρατηρήσεις

Είδαμε ότι ο παραπάνω αλγόριθμος, δοσμένης μία απολητκτικής κορυφής t υπολογίζει το ελάχιστο μήκος που έχει ένα $v - t$ μονοπάτι για κάθε κορυφή v.

Μπορούμε να χρησιμοποιήσουμε τον παραπάνω αλγόριθμο για να βρούμε όλα τα ελάχιστα μήκη $s - v$ μονοπατιών που ξεκινάνε από μία αρχική κορυφή s;

Παρατηρήσεις

Είδαμε ότι ο παραπάνω αλγόριθμος, δοσμένης μία απολητκικής κορυφής t υπολογίζει το ελάχιστο μήκος που έχει ένα $v - t$ μονοπάτι για κάθε κορυφή v.

Μπορούμε να χρησιμοποιήσουμε τον παραπάνω αλγόριθμο για να βρούμε όλα τα ελάχιστα μήκη $s - v$ μονοπατιών που ξεκινάνε από μία αρχική κορυφή s;

Αν το γράφημά μας δεν έχει καθόλου κατευθυνόμενους κύκλους;

Παρατηρήσεις

Είδαμε ότι ο παραπάνω αλγόριθμος, δοσμένης μία απολητκτικής κορυφής t υπολογίζει το ελάχιστο μήκος που έχει ένα $v - t$ μονοπάτι για κάθε κορυφή v.

Μπορούμε να χρησιμοποιήσουμε τον παραπάνω αλγόριθμο για να βρούμε όλα τα ελάχιστα μήκη $s - v$ μονοπατιών που ξεκινάνε από μία αρχική κορυφή s;

Αν το γράφημά μας δεν έχει καθόλου κατευθυνόμενους κύκλους;

Αν μας ζητήσουν να υπολογίσουμε τα μακρύτερα αντί για τα κοντινότερα μονοπάτια;