Δυναμικός Προγραμματισμός

Πότε εφαρμόζεται;

Σε προβλήματα με:

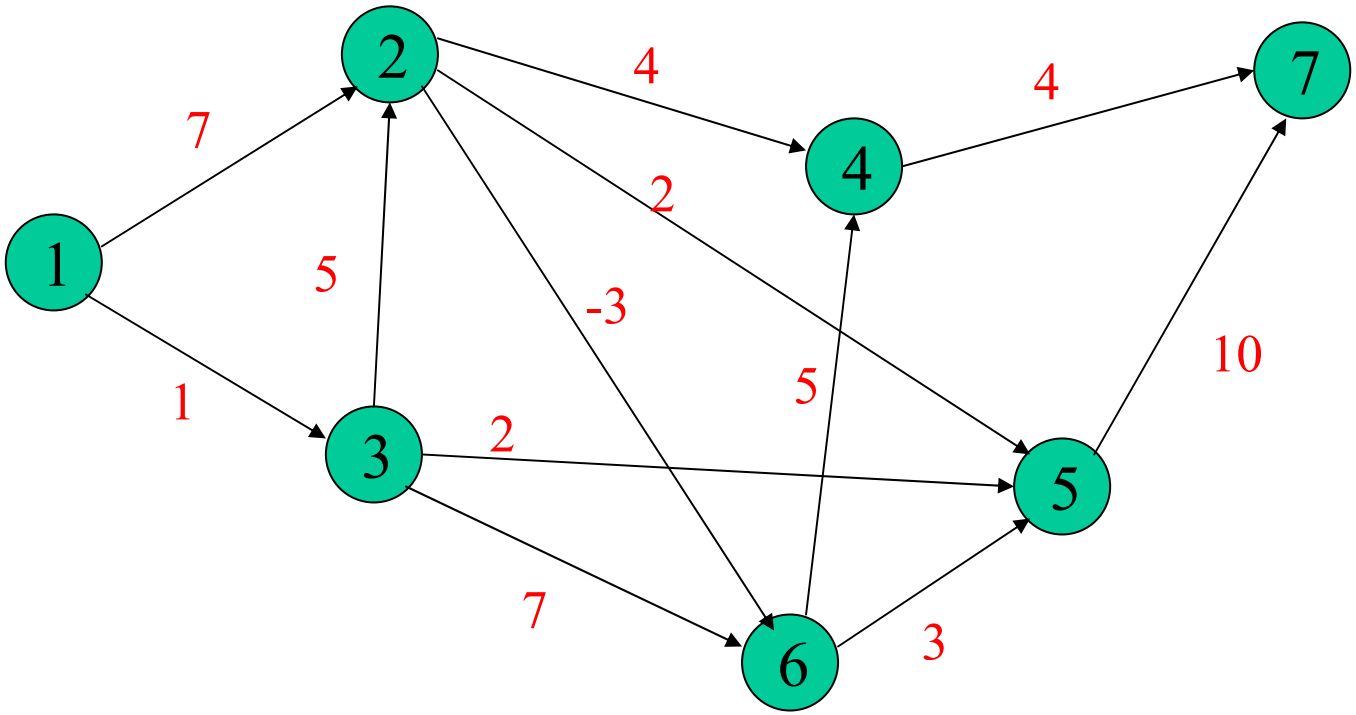
- **Βέλτιστη υπο-δομή**: η βέλτιστη λύση του προβλήματος συνίσταται από βέλτιστες λύσεις υπο-προβλημάτων

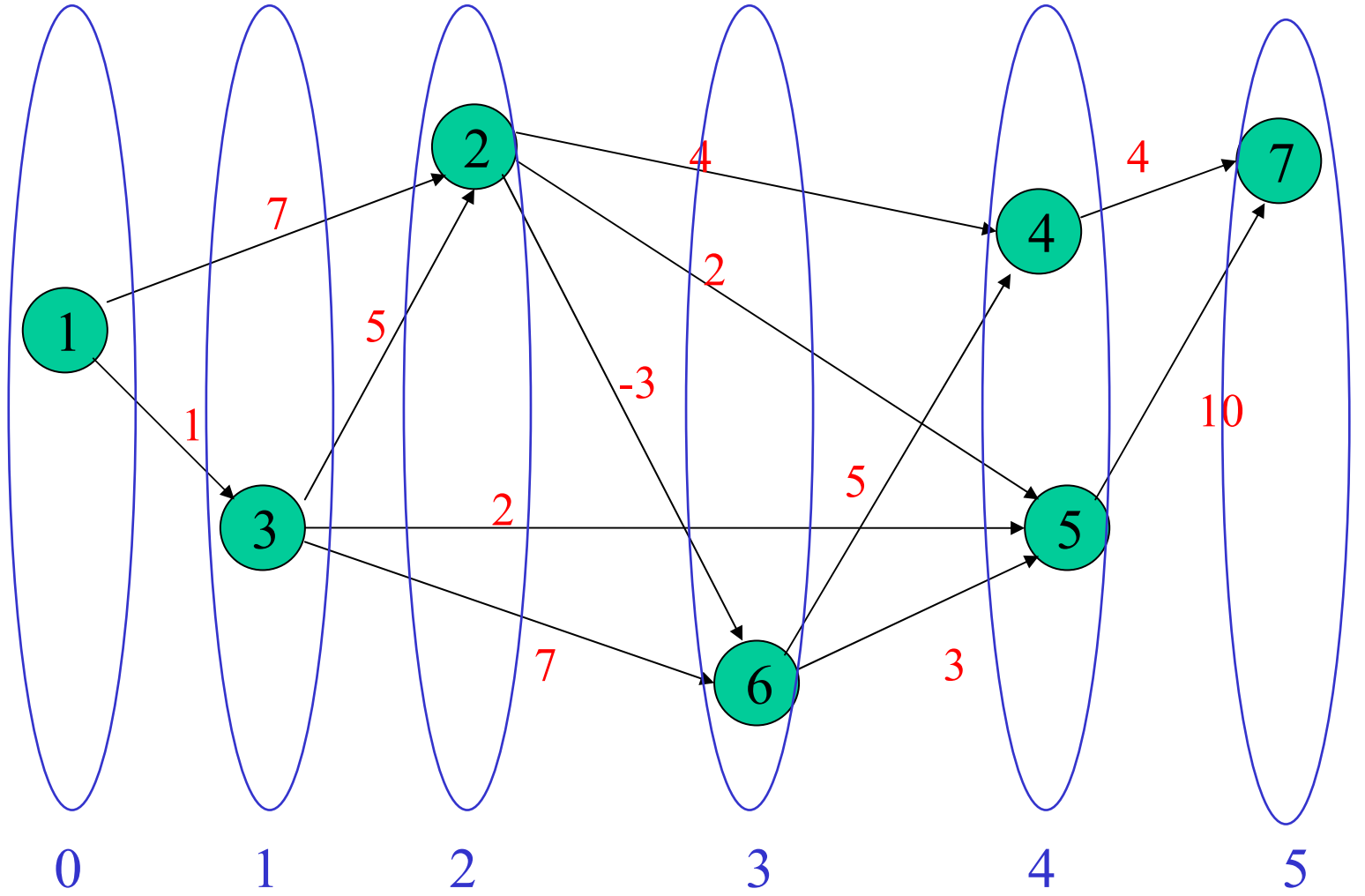
  «Κάθε υπο-πολιτική μιας βέλτιστης πολιτικής είναι η ίδια βέλτιστη»
- **Επικαλυπτόμενα υποπροβλήματα**: λίγα συνολικά υπο-προβλήματα, πολλά αναδρομικά στιγμιότυπα του καθενός

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


1
2
3
4
5
6
7
7
1
5
4
2
-3
5
2
7
3
4
10



$V(1) = 0$ → $V(3)$ → $V(2)$ → … → $V(n)$

Στο γράφο $G=(X,U)$ $\quad |X|=n$

ή $\quad (G=(X,\Gamma))$

n − προβλήματα

$1 \longrightarrow 1 \quad (P(1))$

$1 \longrightarrow 2 \quad (P(2))$

$\vdots$

$1 \longrightarrow n \quad (P(n))$

$P(i): 1 \longrightarrow i$, $V(i)$ βέλτιστη τιμη λύσης

$$V(i)=\min_{j\in\Gamma^{-1}(i)}\{V(j)+w_{ji}\}$$

Αλγόριθμος Bellman για DAGs

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

Initialization V to ∞; Initialization P to
0;

V[s] := 0; P[s] := 0;

for all successors of s

      V[x] := Wsx


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)

Scheduling και μέγιστα μονοπάτια

Ένα σύνολο Χ από Ν εργασίες με γνωστή διάρκεια P_i (i =1, …, N) που ορίζουν ένα Project που πρόκειται να εκτελεσθεί κάτω από περιορισμούς προτεραιότητας, θέλουμε να αποπερατωθεί στον ελάχιστο χρόνο.

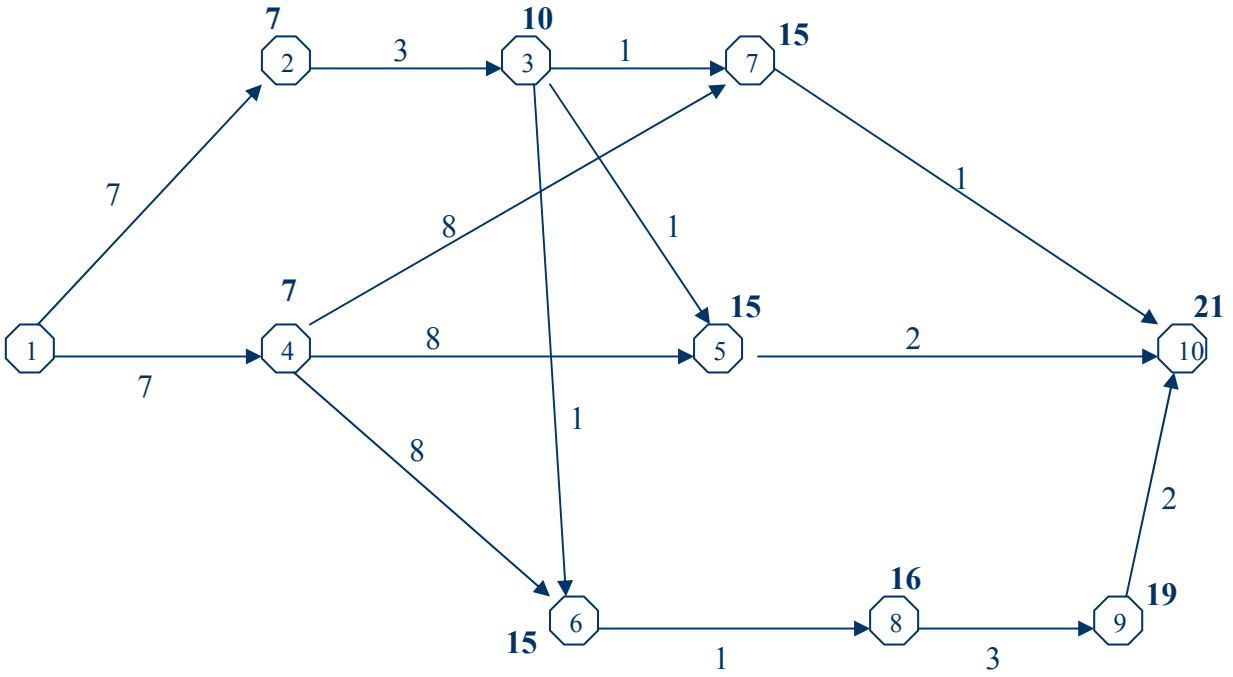
→ **Γράφος σταθμισμένος G = (X, A, P)**

→ X οι εργασίες

→ (i, j) $\in$ A ενώνει την εργασία i με την εργασία j αν h i πρέπει να τελειώσει πριν αρχίσει η εργασία j.

→ βάρος(i, j) = P_i = ελάχιστη διάρκεια που χωρίζει τους χρόνους έναρξης των δύο εργασιών.

Scheduling και μέγιστα μονοπάτια (παράδειγμα)



- **G χωρίς κύκλο → διαίρεση σε επίπεδα σε O(m) με m = |A|**
- **t_i =** ελάχιστος χρόνος έναρξης της εργασίας i

Scheduling και μέγιστα μονοπάτια

- Πρόσθεση δύο πλασματικών εργασιών a και w διάρκειας 0
- Προσθέτουμε τόξα με βάρη μηδέν:

$$(a,i), \forall i \mid \Gamma^{-1}(\imath) = \varnothing$$

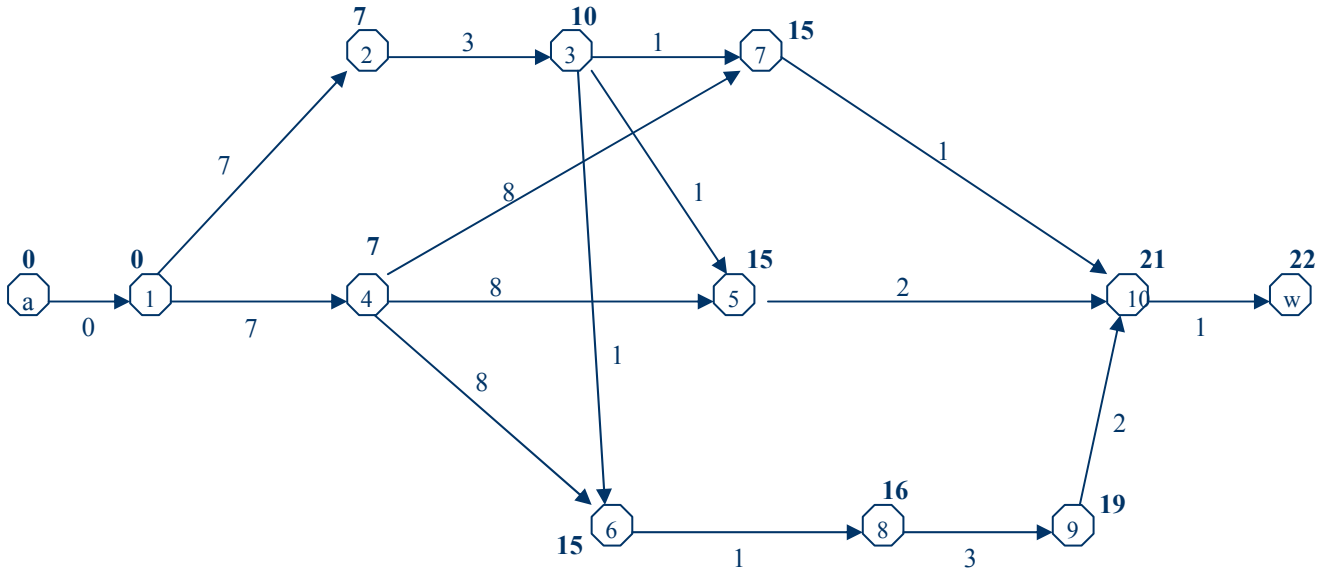
- Προσθέτουμε τόξα με βάρη Pi:

$$(i,w), \forall i \mid \Gamma(\imath) = \varnothing$$

Το πρόβλημα ανάγεται στην ελαχιστοποίηση του tw

Scheduling και μέγιστα μονοπάτια

→ Μια εργασία ΔΕΝ μπορεί να αρχίσει παρά μόνο αν όλες οι προηγούμενές της έχουν τελειώσει.



→ **Εύρεση μονοπατιού** μέγιστης διάρκειας από a προς i.
(Critical Path)

Αλγόριθμος (Critical Path)

```
Init V με -∞
V[a] := 0
P[a] := a
for i := 1 to N
        x := Sorted[i]
        for all successors y of x such that V[x]+W[x,y]>V[y]
                V[y] := V[x] + W(x, y)
                P[y] := x
        endfor
endfor
```

⇒ **Πολυπλοκότητα O(m)**