Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Εθνικό Μετσόβιο Πολυτεχνείο

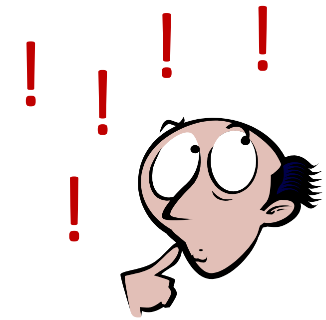
Θεμελιώδη Θέματα Επιστήμης Υπολογιστών

Δυναμικός Προγραμματισμός

Προσαρμογή από διαφάνειες
Σταύρου Νικολόπουλου
(Παν. Ιωαννίνων)

Δυναμικός Προγραμματισμός

- **Αλγοριθμική Τεχνική**



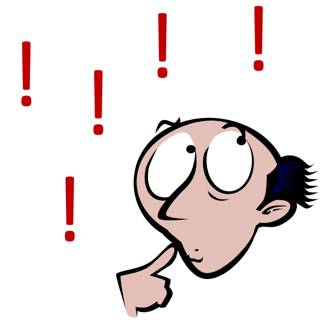
Δυναμικός Προγραμματισμός

Those who cannot remember the past
are condemned to repeat it.

-Dynamic Programming

Δυναμικός Προγραμματισμός

- **Αλγοριθμική Τεχνική**



Δυναμικός Προγραμματισμός

- **ΕΊΝΑΙ… Τεχνική Σχεδίασης Αλγορίθμων !!!**
  - ✓ με **ευρύτατο πεδίο εφαρμογών !**
  - ✓ και με χρήση εκεί όπου **άλλες** πιο ειδικευμένες **τεχνικές αποτυγχάνουν !**

και, όχι … Τεχνική Προγραμματισμού !!!

Δυναμικός Προγραμματισμός

- **Γιατί αυτός ο όρος και η ονομασία**

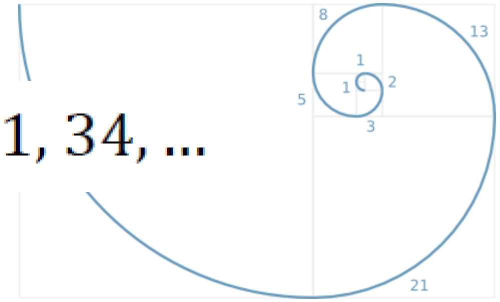
Δυναμικός Προγραμματισμός

- **Επινοήθηκε από τον Richard Bellman το 1950**
  - Τότε ο **προγραμματισμός** ήταν μια απόκρυφη, σπάνια δραστηριότητα την οποία ασκούσαν τόσο πολύ λίγοι που δεν άξιζε καν να την αναφέρουν.
  - Ο όρος «προγραμματισμός» την εποχή εκείνη σήμαινε **«σχεδιασμός ενεργειών»** (planning), ενώ
  - ο όρος «δυναμικός προγραμματισμός» σήμαινε **«πολυεπίπεδο, χρονικά μεταβαλλόμενο σχεδιασμό πολλαπλών διεργασιών»** !!!

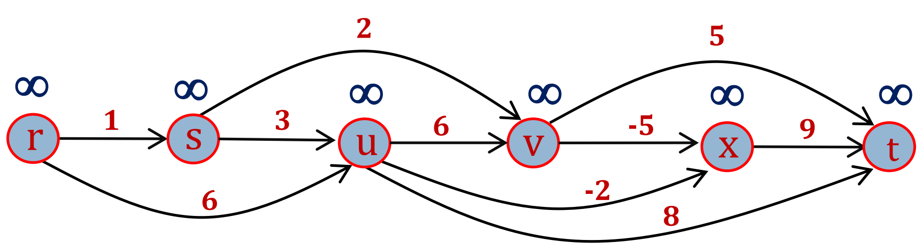
Δυναμικός Προγραμματισμός

- **Ας θυμηθούμε δύο γνωστά μας προβλήματα !**
  - **Ακολουθία Fibonacci**

    0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …

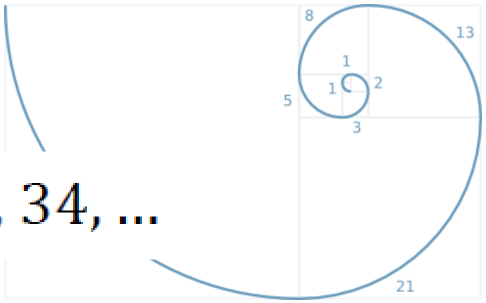


  - **Ελάχιστες διαδρομές σε DAG**



Δυναμικός Προγραμματισμός

- **Ακολουθία Fibonacci**

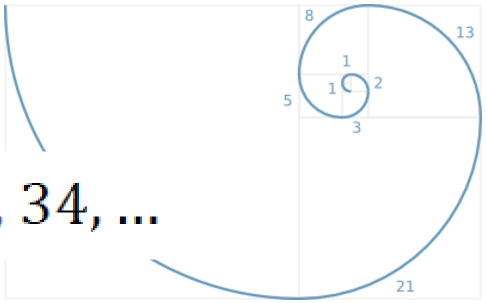


$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$$

$$F_n = \begin{cases} 0 & \text{εάν} \quad n = 0 \\ 1 & \text{εάν} \quad n = 1 \\ F_{n-1} + F_{n-2} & \text{εάν} \quad n > 1 \end{cases}$$

Δυναμικός Προγραμματισμός

- **Ακολουθία Fibonacci**



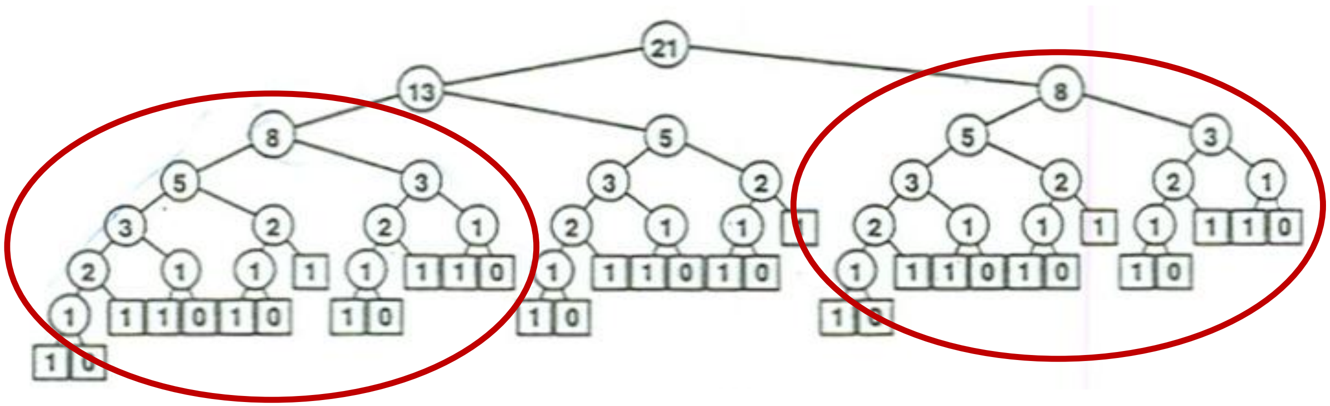
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …

```
fib1(n)
  if n = 0 then return 0
  if n = 1 then return 1
  return fib1(n-1) + fib1(n-2)
```

Δυναμικός Προγραμματισμός

Ακολουθία Fibonacci

```
fib1(n)
  if n = 0 then return 0
  if n = 1 then return 1
  return fib1(n-1) + fib1(n-2)
```

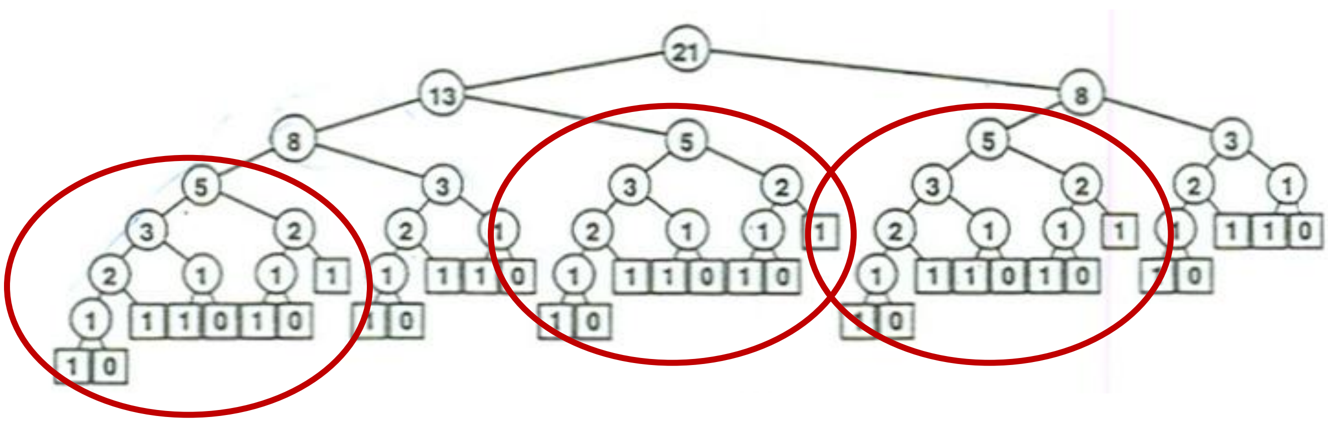


```
8 F(6)
 5 F(5)
  3 F(4)
   2 F(3)
    1 F(2)
     1 F(1)
     0 F(0)
    1 F(1)
   1 F(2)
    1 F(1)
    0 F(0)
  2 F(3)
   1 F(2)
    1 F(1)
    0 F(0)
   1 F(1)
 3 F(4)
  2 F(3)
   1 F(2)
    1 F(1)
    0 F(0)
   1 F(1)
  1 F(2)
   1 F(1)
   0 F(0)
```

Δυναμικός Προγραμματισμός

Ακολουθία Fibonacci

```
fib1(n)
  if n = 0 then return 0
  if n = 1 then return 1
  return fib1(n-1) + fib1(n-2)
```

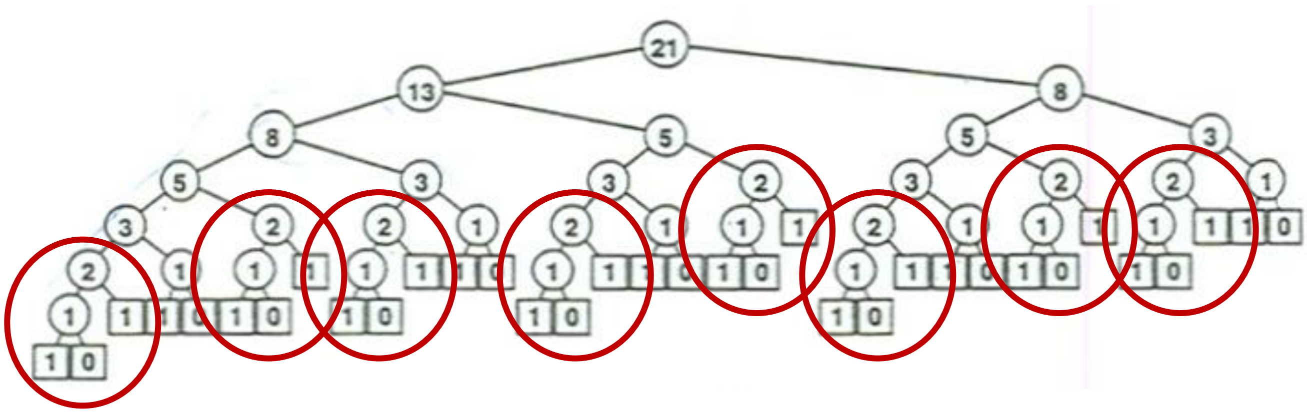


```
8 F(6)
  5 F(5)
    3 F(4)
      2 F(3)
        1 F(2)
          1 F(1)
          0 F(0)
        1 F(1)
      1 F(2)
        1 F(1)
        0 F(0)
    2 F(3)
      1 F(2)
        1 F(1)
        0 F(0)
      1 F(1)
  3 F(4)
    2 F(3)
      1 F(2)
        1 F(1)
        0 F(0)
      1 F(1)
    1 F(2)
      1 F(1)
      0 F(0)
```

Δυναμικός Προγραμματισμός

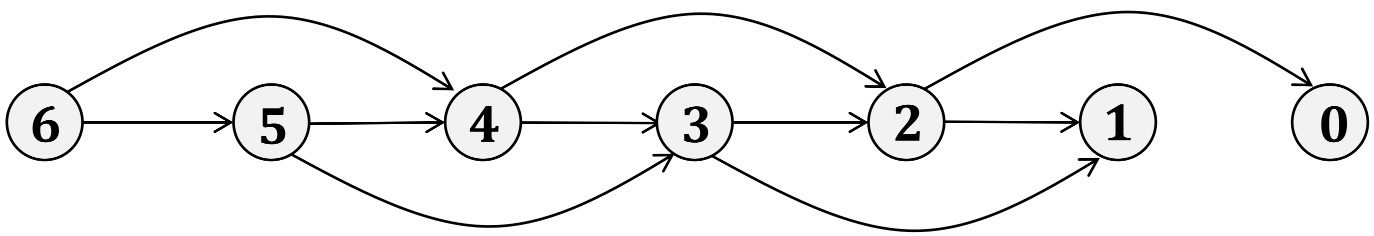
Ακολουθία Fibonacci

```
fib1(n)
  if n = 0 then return 0
  if n = 1 then return 1
  return fib1(n-1) + fib1(n-2)
```



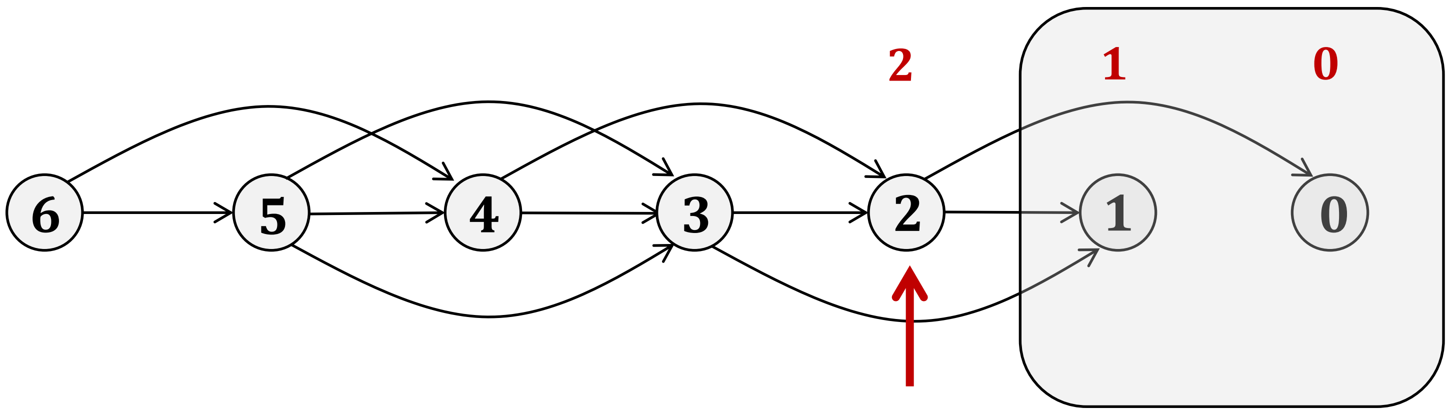
```
8 F(6)
  5 F(5)
    3 F(4)
      2 F(3)
        1 F(2)
          1 F(1)
          0 F(0)
        1 F(1)
      1 F(2)
        1 F(1)
        0 F(0)
    2 F(3)
      1 F(2)
        1 F(1)
        0 F(0)
      1 F(1)
  3 F(4)
    2 F(3)
      1 F(2)
        1 F(1)
        0 F(0)
      1 F(1)
    1 F(2)
      1 F(1)
      0 F(0)
```

Ακολουθία Fibonacci




```
fib(n) = fib(n-1) + fib(n-2)
```

Ακολουθία Fibonacci

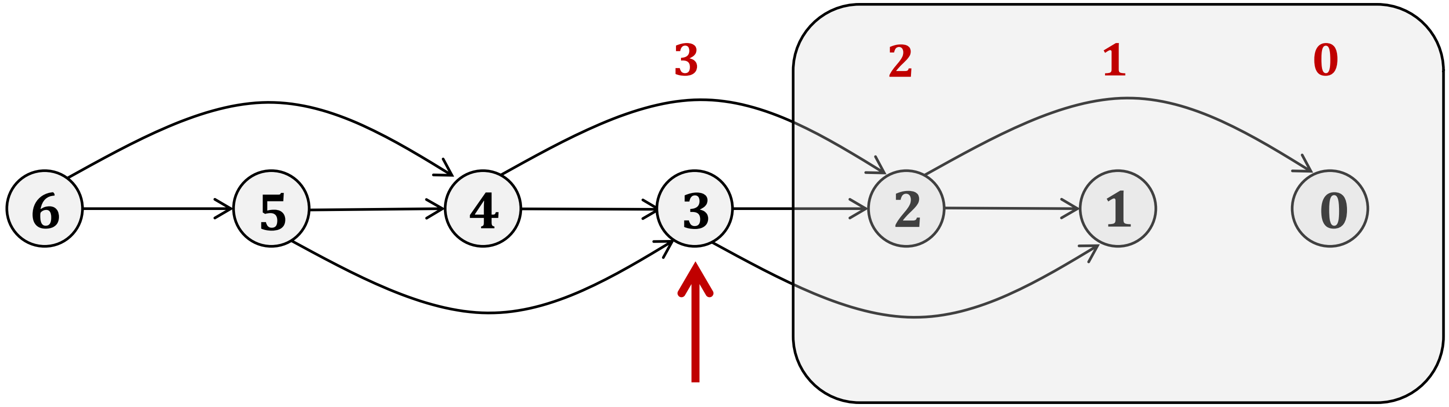


`fib(2) = fib(1) + fib(0)`

`fib(n) = fib(n-1) + fib(n-2)`

Δυναμικός Προγραμματισμός

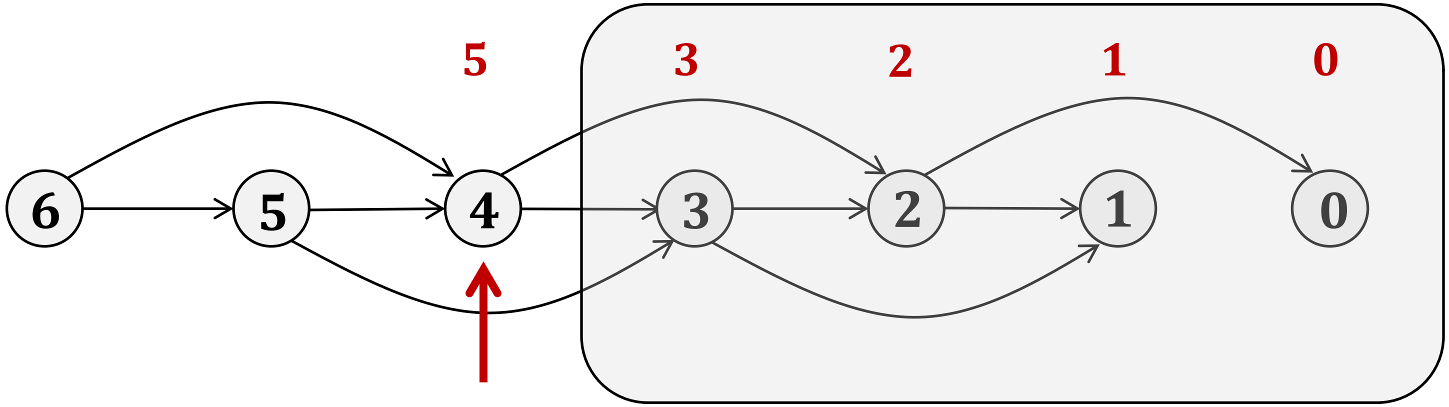
Ακολουθία Fibonacci



`fib(3) = fib(2) + fib(1)`

```
fib(n) = fib(n-1) + fib(n-2)
```

Ακολουθία Fibonacci

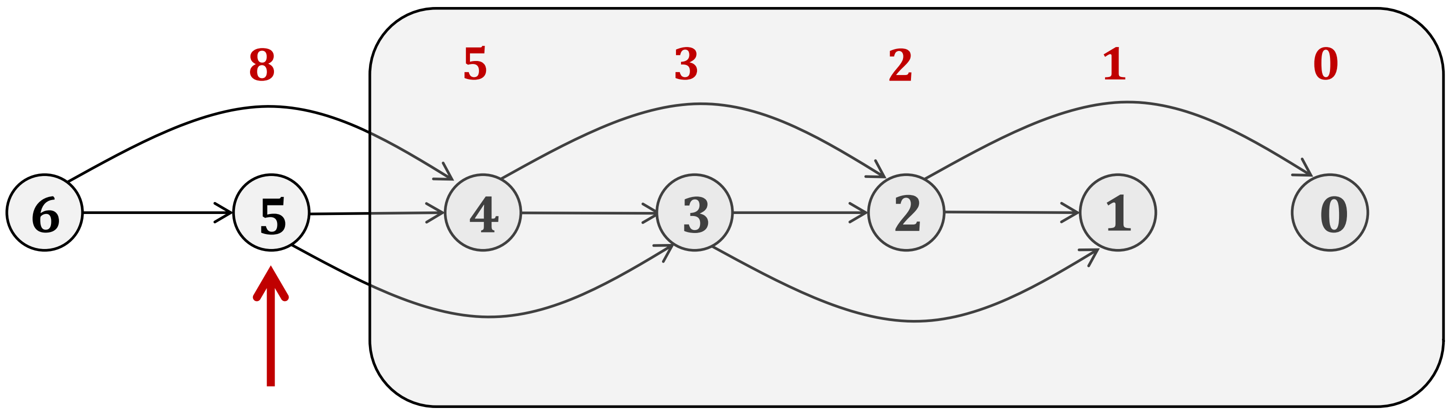



```
fib(4) = fib(3) + fib(2)
```

```
fib(n) = fib(n-1) + fib(n-2)
```

Δυναμικός Προγραμματισμός

Ακολουθία Fibonacci

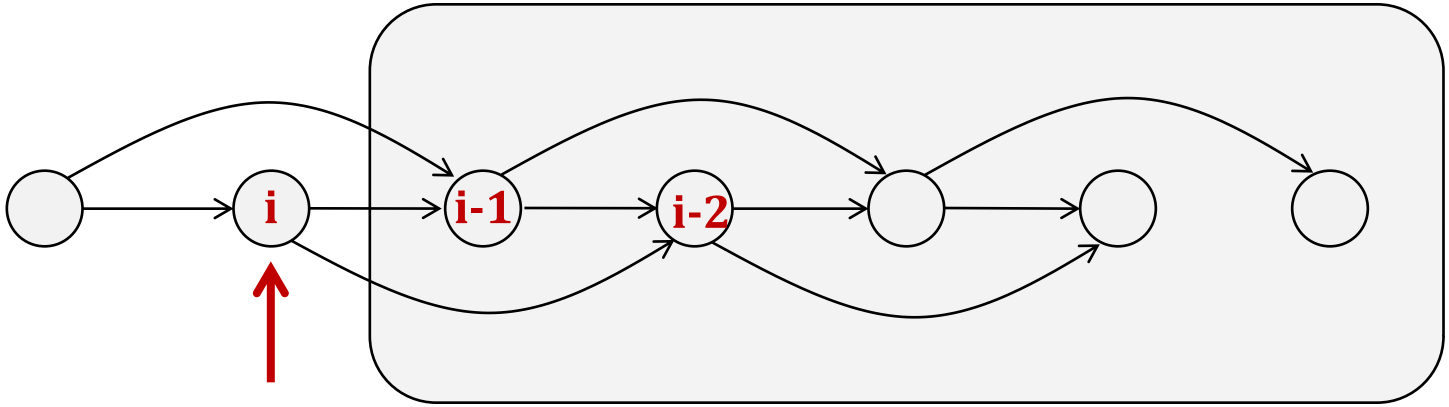


fib(5) = fib(4) + fib(3)

fib(n) = fib(n-1) + fib(n-2)

Δυναμικός Προγραμματισμός

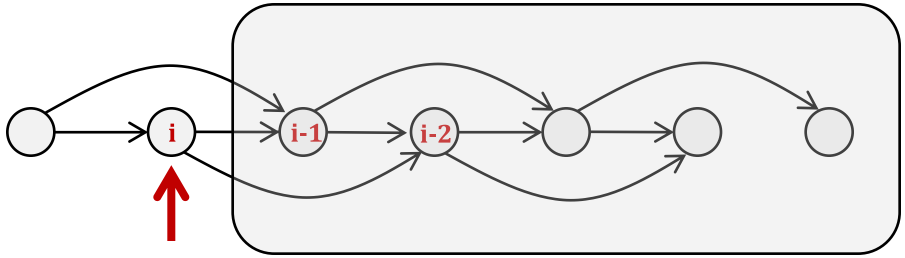
Ακολουθία Fibonacci



fib(i) = fib(i-1) + fib(i-2)

Μπορώ να λύσω ένα υποπρόβλημα εύκολα
εάν έχω **«βρει»** και **«κρατήσει»** τις λύσεις **«μικρότερων»**
υποπροβλημάτων !!!

Ακολουθία Fibonacci



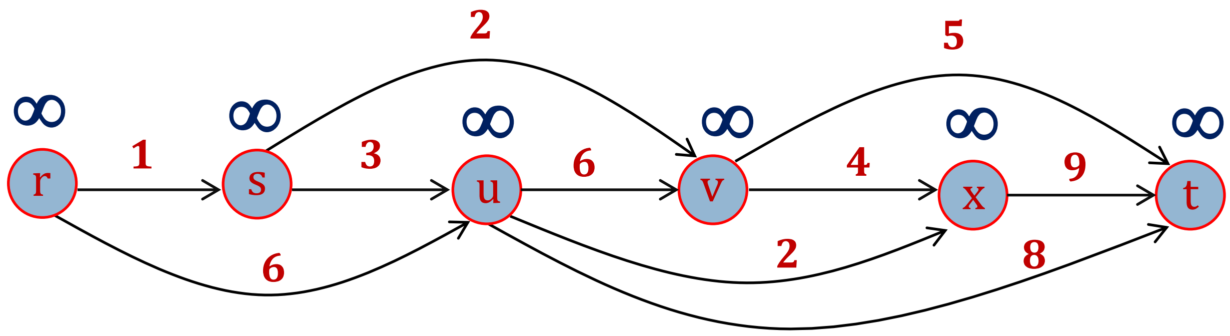
Αλγόριθμος FIBONACCI

```
fib2(n)
1. if n = 0 then return 0
2. if n = 1 then return 1
3. F(0) = 0, F(1) = 1 {F πίνακας μήκους n}
4. for i = 2, 3, …, n
           F(i) = F(i-1) + F(i-2)
5. return F(n)
```

Κρατήστε αυτά τα 2 στοιχεία του Αλγόριθμου !!!

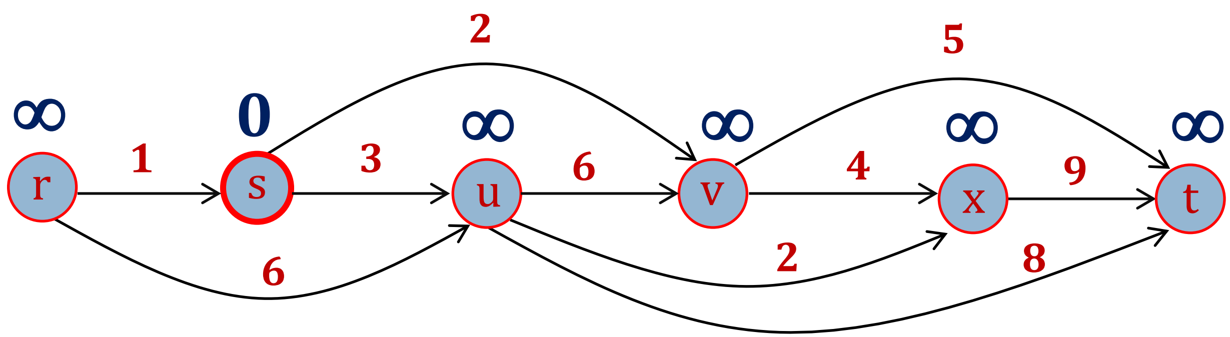
Δυναμικός Προγραμματισμός

- **Ελάχιστες διαδρομές σε DAG**
  - Χαρακτηριστικό ενός DAG: Τοπολογική Ταξινόμηση



Δυναμικός Προγραμματισμός

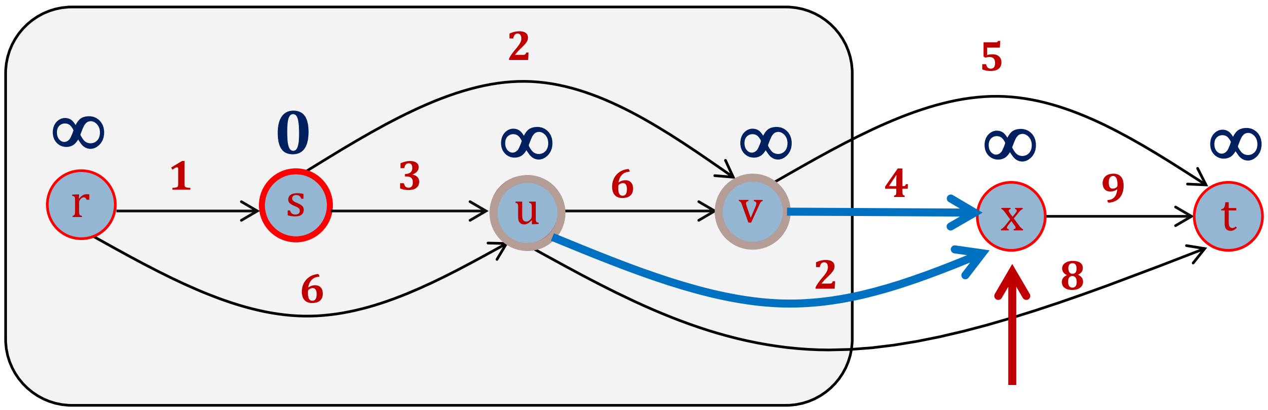
- **Ελάχιστες διαδρομές σε DAG**
  - Χαρακτηριστικό ενός DAG: Τοπολογική Ταξινόμηση



  - **Γιατί** αυτό το χαρακτηριστικό βοηθάει στον υπολογισμό των ε.δ. από ένα κόμβο, έστω τον **s** ?

Δυναμικός Προγραμματισμός

- **Ελάχιστες διαδρομές σε DAG**
  - Χαρακτηριστικό ενός DAG: Τοπολογική Ταξινόμηση

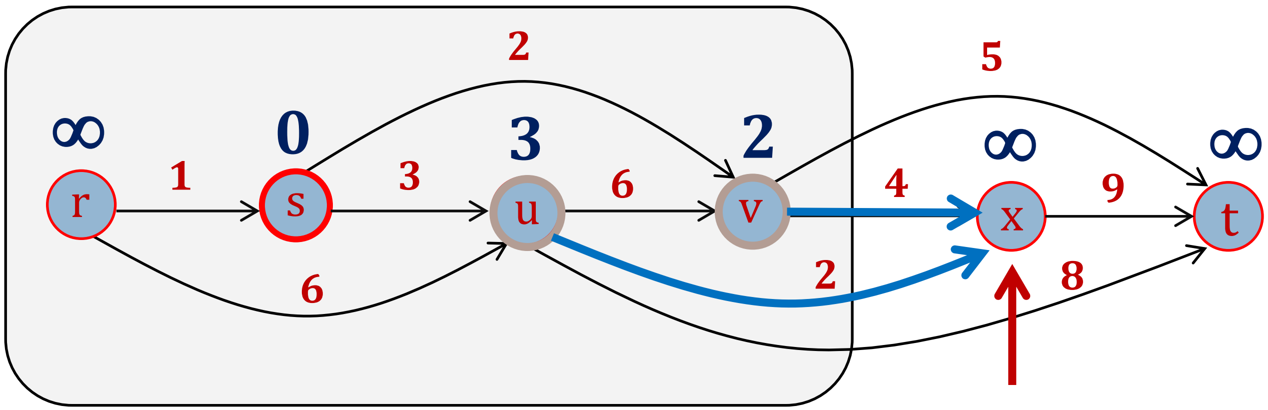


  - Ας εστιάσουμε σε ένα κόμβο, έστω στον κόμβο **x**

Ο μόνος τρόπος για να φθάσουμε στον x είναι μέσω των προκατόχων του: v ή u !!!

Δυναμικός Προγραμματισμός

- **Ελάχιστες διαδρομές σε DAG**
  - Χαρακτηριστικό ενός DAG: Τοπολογική Ταξινόμηση

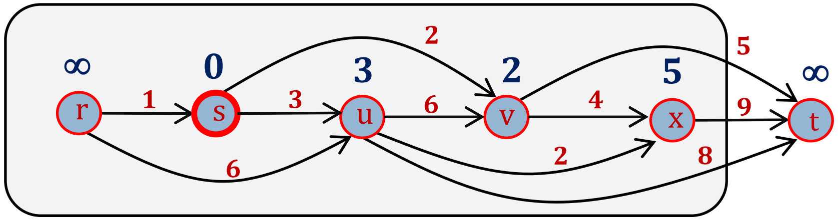


  - Επομένως, για να βρούμε την ε.δ. από τον **s** μέχρι τον **x**, **αρκεί μόνο να συγκρίνουμε τις δύο διαδρομές:**

```
d(x) = min{d(v)+4, d(u)+2}
```

Ελάχιστες διαδρομές σε DAG

- Μια τέτοια σχέση ισχύει για ∀ κόμβο !!!
  π.χ., για τον κόμβο **t**:



d(t) = min{ d(x)+9, d(v)+5, d(u)+8 }

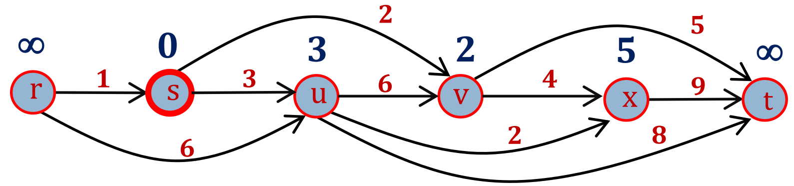
- Αν υπολογίσουμε τις τιμές **d(.)** με τη σειρά της Τοπολογικής Ταξινόμησης τότε:

> Όταν φθάσουμε στον κόμβο **x** θα έχουμε όλες τις πληροφορίες για τον υπολογισμό του **d(x)**.

Δυναμικός Προγραμματισμός

Ελάχιστες διαδρομές σε DAG

- Αλγόριθμος SP-DAG



1. Initialize(G,s)
2. Topological-Sorting(G)
3. για κάθε κόμβο $x \in V-\{s\}$ σε τοπολογική σειρά:

$$\mathbf{d(x) = min_{(u,x)\in E}\{d(u) + w(u,x)\}}$$

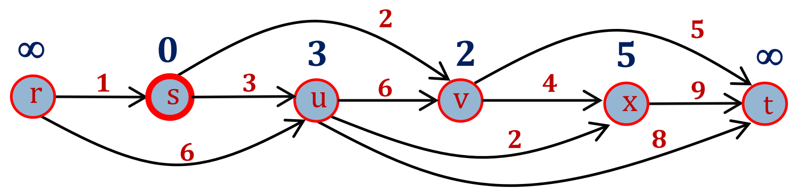
4. return d(.)

Κρατήστε αυτά τα 2
στοιχεία του Αλγόριθμου !!!

Δυναμικός Προγραμματισμός

Ελάχιστες διαδρομές σε DAG

Παρατηρήσεις !!!



- Ο αλγόριθμος SP-DAG
  επιλύει υποπροβλήματα, της μορφής:

$$\{\, d(x) \mid x \in V - \{s\} \,\}$$

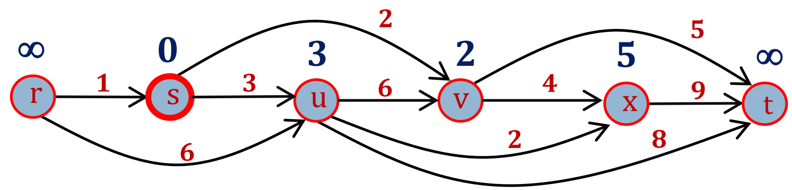
- **Αρχίζει** από το **«μικρότερο»** υποπρόβλημα και **προχωράει σε «μεγαλύτερα» υποπροβλήματα !!!**

> Ένα υποπρόβλημα το θεωρούμε **«μεγάλο»** εάν πρέπει να λύσουμε πολλά άλλα υποπροβλήματα πριν φθάσουμε σε αυτό !!!

Δυναμικός Προγραμματισμός

Ελάχιστες διαδρομές σε DAG

Παρατηρήσεις !!!



- Σε κάθε κόμβο **x**
  ο αλγόριθμος SP-DAG
  υπολογίζει μια συνάρτηση των τιμών των ε.δ των **προκατόχων** του κόμβου **x**

- Εδώ, η συνάρτηση μας είναι ένα **ελάχιστο αθροισμάτων !**

  Θα μπορούσε κάλλιστα να είναι μια συνάρτηση:

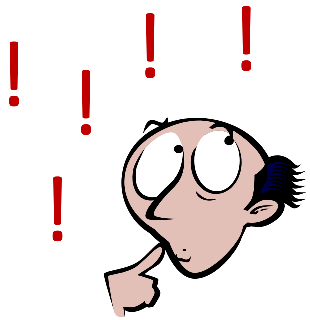
  ✓ **μεγίστου** οπότε θα υπολογίζαμε τις max διαδρομές, ή

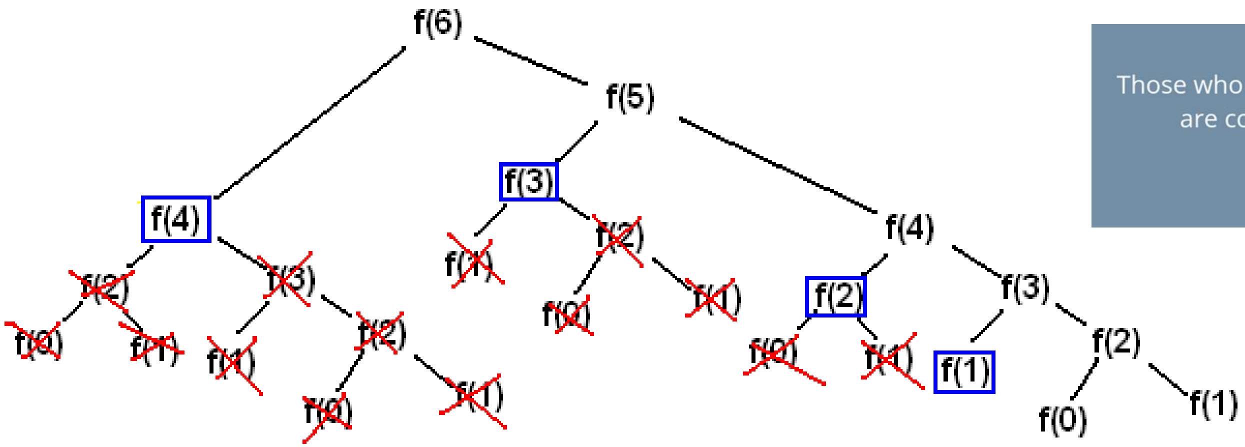
  ✓ **ελαχίστου γινομένων**
  οπότε θα υπολογίζαμε την διαδρομή με το ελάχιστο γινόμενο ακμών

Δυναμικός Προγραμματισμός

Χαρακτηριστικά Δυναμικού Προγραμματισμού

- **Η Βασική Ιδέα !!!**





Those who cannot remember the past are condemned to repeat it.

-Dynamic Programming

Δυναμικός Προγραμματισμός

Χαρακτηριστικά Δυναμικού Προγραμματισμού

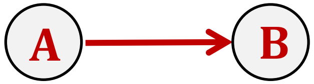
- **Η Βασική Ιδέα !!!...** Λύσης προβλήματος Π με ΔΠ:

1. **Υπολογίζουμε** ένα **σύνολο υποπροβλημάτων** του Π
2. **Επινοούμε** μια **σχέση** για την λύση ενός υποπροβλήματος
3. **Λύνουμε τα υποπρ/ματα** ξεκινώντας από «μικρότερα», **αποθηκεύουμε** τις λύσεις τους, και προχωράμε προς «μεγαλύτερα» **χρησιμοποιώντας τις αποθηκευμένες λύσεις των μικρότερων (bottom-up)** !!!
4. **Παίρνουμε** τη λύση του αρχικού μας προβλήματος Π, λύνοντας όλα τα υποπρ/ματα με **καθορισμένη σειρά** !!!

Δυναμικός Προγραμματισμός

Χαρακτηριστικά Δυναμικού Προγραμματισμού

- **Προσοχή !!!**
  - ✓ Στο ΔΠ το «**μοντέλο της τεχνικής**» θα μπορούσαμε να το θεωρήσουμε ότι είναι ένα γράφημα **G** τύπου DAG !!!
  - ✓ Οι κόμβοι του **G** αντιστοιχούν στα υποπροβλήματα και οι ακμές του στις εξαρτήσεις ανάμεσα σε αυτά:



    - Το **A** θεωρείται «μικρότερο» υποπρόβλημα από το **B,** ή
    - Για να λύσουμε το **B** χρειαζόμαστε την λύση του **A** !!!

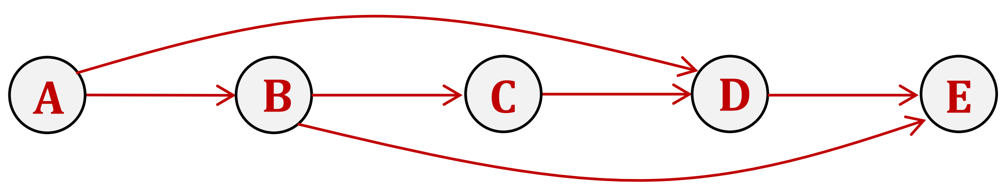
Δυναμικός Προγραμματισμός

Χαρακτηριστικά Δυναμικού Προγραμματισμού

- **Θεμελιώδης Ιδιότητα του ΔΠ !!!**

> Υπάρχει **διάταξη** των υποπροβλημάτων και μια **σχέση** που δείχνει **πως θα λυθεί** ένα υποπρόβλημα
>
> **έχοντας τις λύσεις «μικροτέρων»** υποπρ/των,
>
> δηλαδή, υποπρ/των που εμφανίζονται νωρίτερα στη διάταξη !

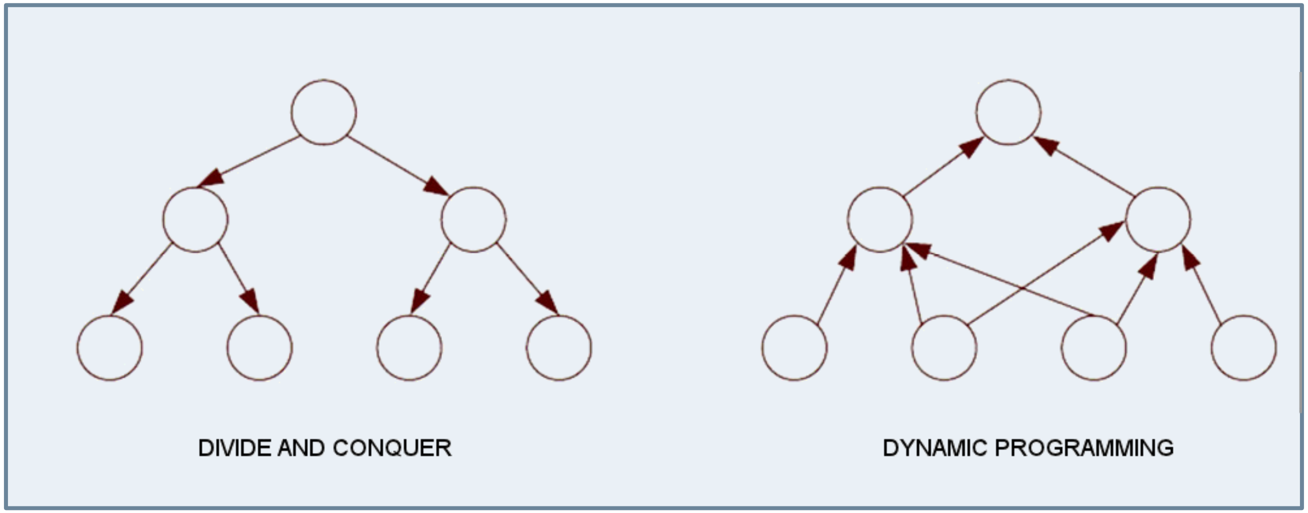
Διάταξη:



Σχέση: **Μεγάλο-ΥΠ = f(Μικρότερα-ΥΠ)**

Δυναμικός Προγραμματισμός

- **Χαρακτηριστικά Δυναμικού Προγραμματισμού**
  - **Διαίρει-και-Βασίλευε vs ΔΠ !!!**



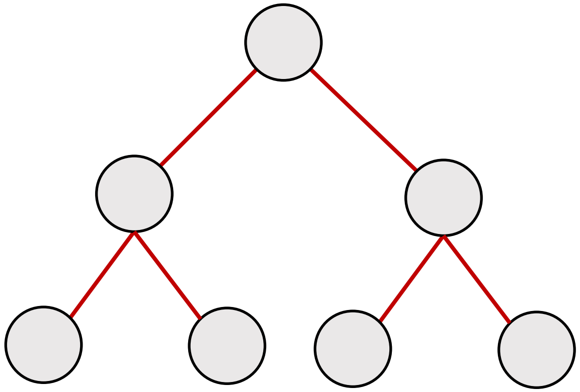
Δυναμικός Προγραμματισμός

Χαρακτηριστικά Δυναμικού Προγραμματισμού

- **Διαίρει-και-Βασίλευε vs ΔΠ !!!**
  - ✓ Στην τεχνική Δ-και-Β ένα Π μεγέθους **n** εκφράζεται συναντήσει υποπροβλημάτων Π(1), Π(2), …, Π(k) που είναι ***σημαντικά μικρότερα***, για παράδειγμα **n/2**, και ***δεν επικαλύπτονται*** !!!
  - ✓ Λόγω αυτής της απότομης μείωσης του μεγέθους του Π, το δένδρο της αναδρομής έχει:

    O(logn) βάθος

    O(n^c) πλήθος κόμβων !!!



Δυναμικός Προγραμματισμός

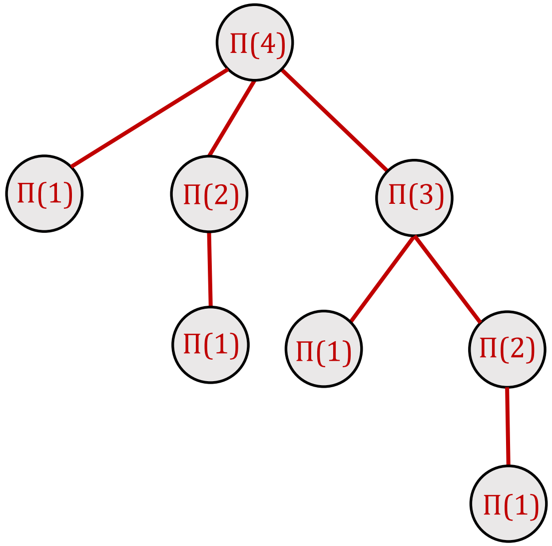
Χαρακτηριστικά Δυναμικού Προγραμματισμού

- **Διαίρει-και-Βασίλευε vs ΔΠ !!!**
  - ✓ Αντίθετα, στην τεχνική του ΔΠ τα υποπροβλημάτων Π(1), Π(2), …, Π(k) μπορεί να είναι ***ελάχιστα μικρότερα***, για παράδειγμα το **Π(i)** βασίζεται στο **Π(i-1)**, και να ***επικαλύπτονται*** !!!
  - ✓ Εδώ συνήθως, το δένδρο της αναδρομής έχει:

    O(n) βάθος

    $O(c^n)$ πλήθος κόμβων, c>1

    Εκθετικό πλήθος κόμβων !!!



Δυναμικός Προγραμματισμός

Χαρακτηριστικά Δυναμικού Προγραμματισμού

- **Ο ΔΠ είναι μια πολύ ισχυρή αλγοριθμική τεχνική !!!**
  με ευρύτατο πεδίο εφαρμογών !!!

  Συνοπτικά, στο ΔΠ:

  - υπολογίζουμε ένα σύνολο υποπροβλημάτων του Π,
  - λύνουμε κάθε υποπρόβλημα μία μόνο φορά,
  - αποθηκεύουμε τη λύση του, και
  - χρησιμοποιούμε αυτή, εάν χρειαστεί να το ξαναλύσουμε!!!

  Αποφεύγουμε έτσι να λύνουμε ξανά-και-ξανά πολλές φορές τα ίδια-και-ίδια υποπροβλήματα !!!

Προβλήματα Δυναμικού Προγραμματισμού

Μέγιστες Αύξουσες Υποακολουθίες

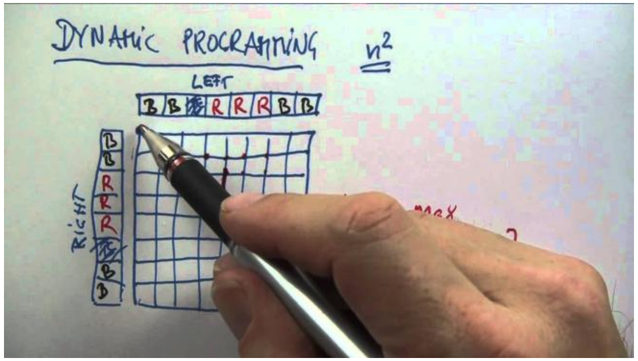
Διορθωτική Απόσταση

Πολλαπλασιασμός Ακολουθίας Πινάκων

Ελάχιστες Αξιόπιστες Διαδρομές

Πανζευκτικές Ελάχιστες Διαδρομές

Σακίδιο (Knapsack)



1 Μέγιστες Αύξουσες Υποακολουθίες

Πρόβλημα

Μας δίδεται μια ακολουθία n αριθμών

$$A = (a_1, a_2, \ldots, a_n)$$

και μας ζητείται να υπολογίσουμε μία αύξουσα υπακολουθία της A με το μέγιστο μήκος.

Μια ακολουθία $A' = (a_{i1}, a_{i2}, \ldots, a_{ik})$ είναι αύξουσα υπακολουθία της A εάν:

$$a_{i1} < a_{i2} < \ldots < a_{ik} \quad \text{και} \quad 1 \leq i_1 < i_2 < \ldots < i_k \leq n$$

1 **Μέγιστες Αύξουσες Υποακολουθίες**

Παράδειγμα

Η μέγιστη αύξουσα υπακολουθία της

$$A = (5, \mathbf{2}, 8, 6, \mathbf{3}, \mathbf{6}, \mathbf{9}, 7)$$

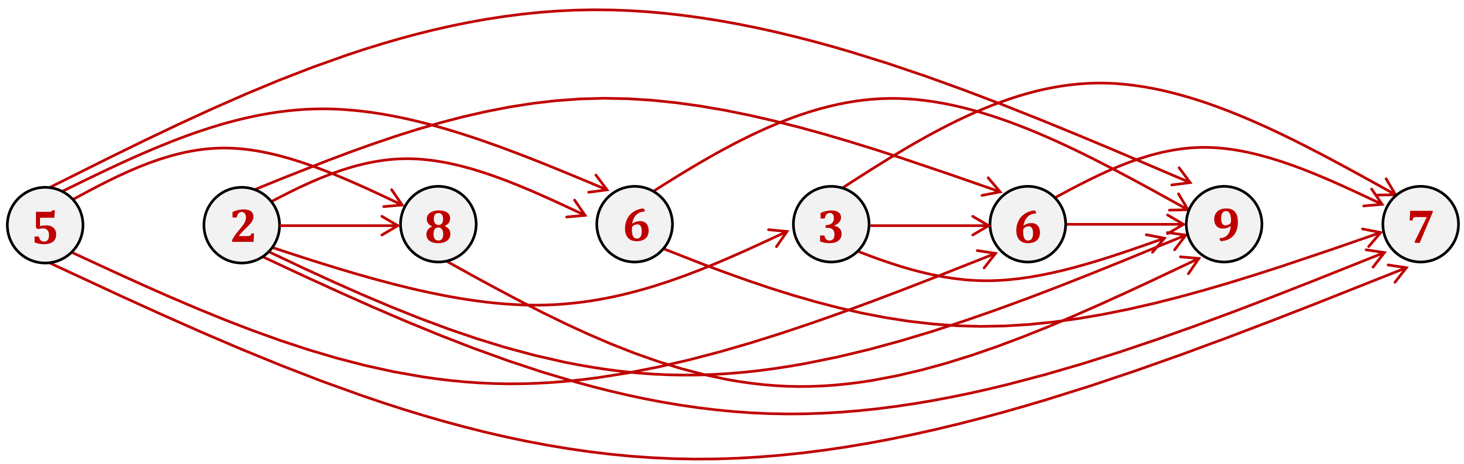
είναι η $A' = (2, 3, 6, 9)$ με μήκος 4.

Οι υπακολουθίες $B' = (2, 6, 3, 6, 9)$ και $C' = (5, 6, 6, 7)$ της A δεν είναι έγκυρες διότι δεν είναι (γνησίως) αύξουσες.

Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ

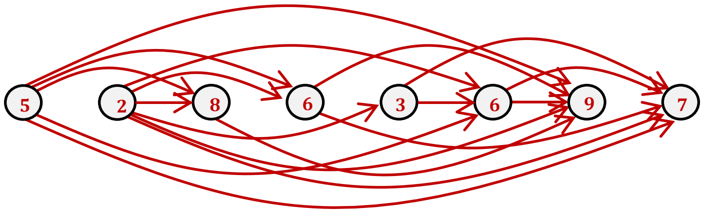
A = (5, 2, 8, 6, 3, 6, 9, 7)



Δημιουργούμε ένα γράφημα **G** με **ΟΛΕΣ** τις **δυνατές μεταβάσεις** (από αριθμό σε μεγαλύτερο που έπεται) !!!

Μέγιστες Αύξουσες Υποακολουθίες

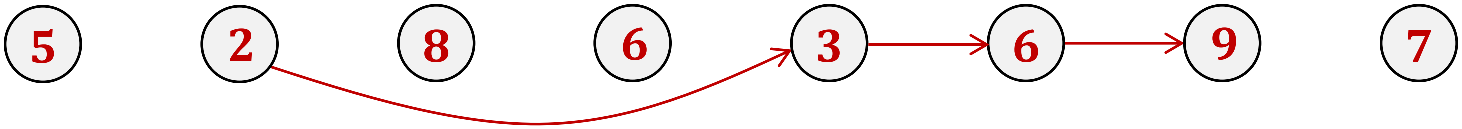
- Θα χρησιμοποιήσουμε ΔΠ



  - Παρατηρήστε ότι:

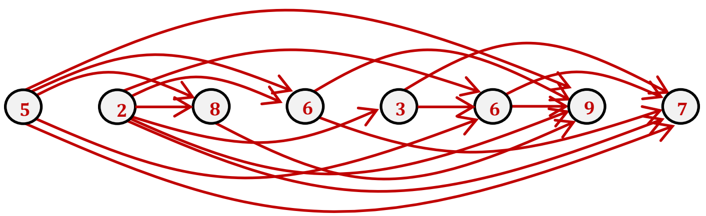
    (1) Το γράφημα G είναι **DAG** !!!

    (2) Υπάρχει 1-1 αντιστοιχία ανάμεσα στις **αύξουσες υπακολουθίες** και στις **διαδρομές αυτού του DAG** !!!



Μέγιστες Αύξουσες Υποακολουθίες

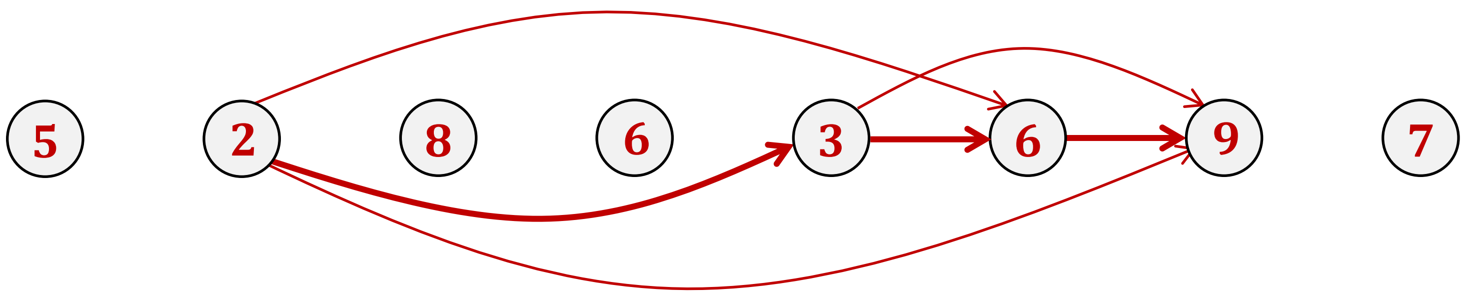
- Θα χρησιμοποιήσουμε ΔΠ



  - Παρατηρήστε ότι:

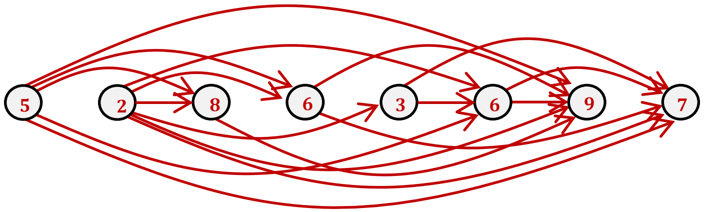
    (1) Το γράφημα G είναι **DAG** !!!

    (2) Υπάρχει 1-1 αντιστοιχία ανάμεσα στις **αύξουσες υπακολουθίες** και στις **διαδρομές αυτού του DAG** !!!



Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ



  - Παρατηρήστε ότι:

    (1) Το γράφημα G είναι **DAG** !!!

    (2) Υπάρχει 1-1 αντιστοιχία ανάμεσα στις **αύξουσες υπακολουθίες** και στις **διαδρομές αυτού του DAG** !!!

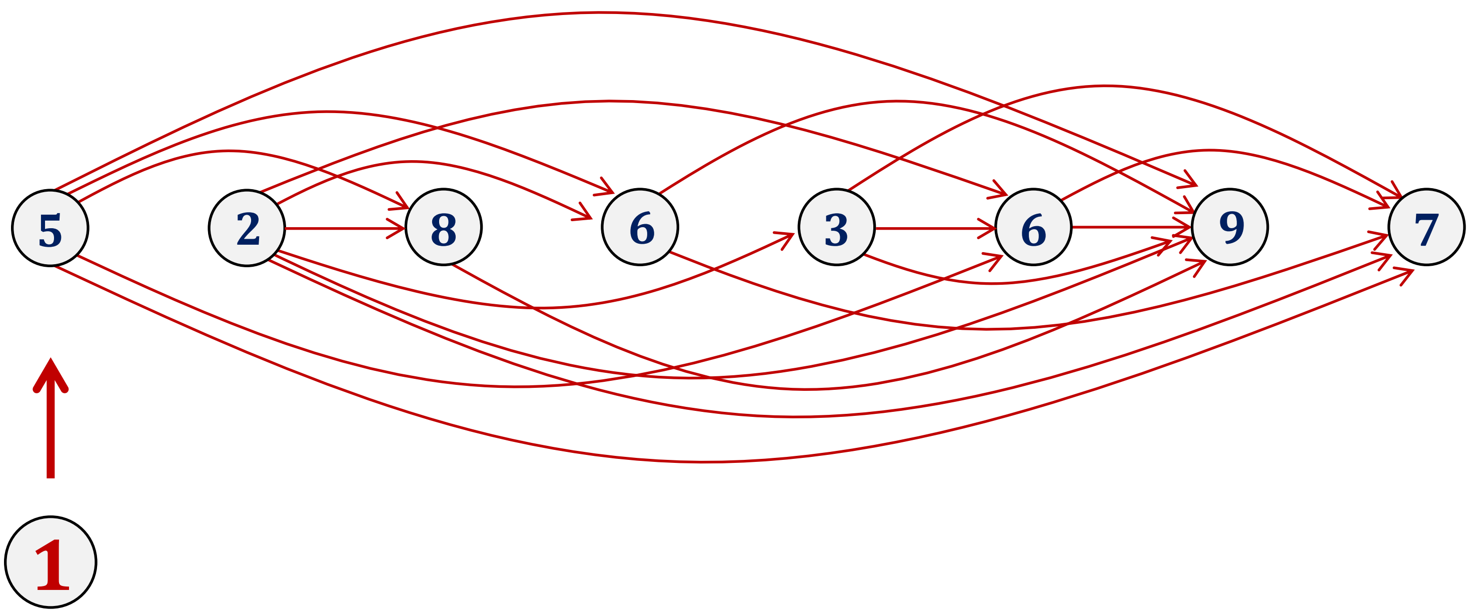
  - Επομένως, το πρόβλημά μας **ανάγεται** στον **υπολογισμό της μεγαλύτερης διαδρομής** στο γράφημα G !!!

Υπολογισμός:

Μεγαλύτερης διαδρομής σε DAG !!!

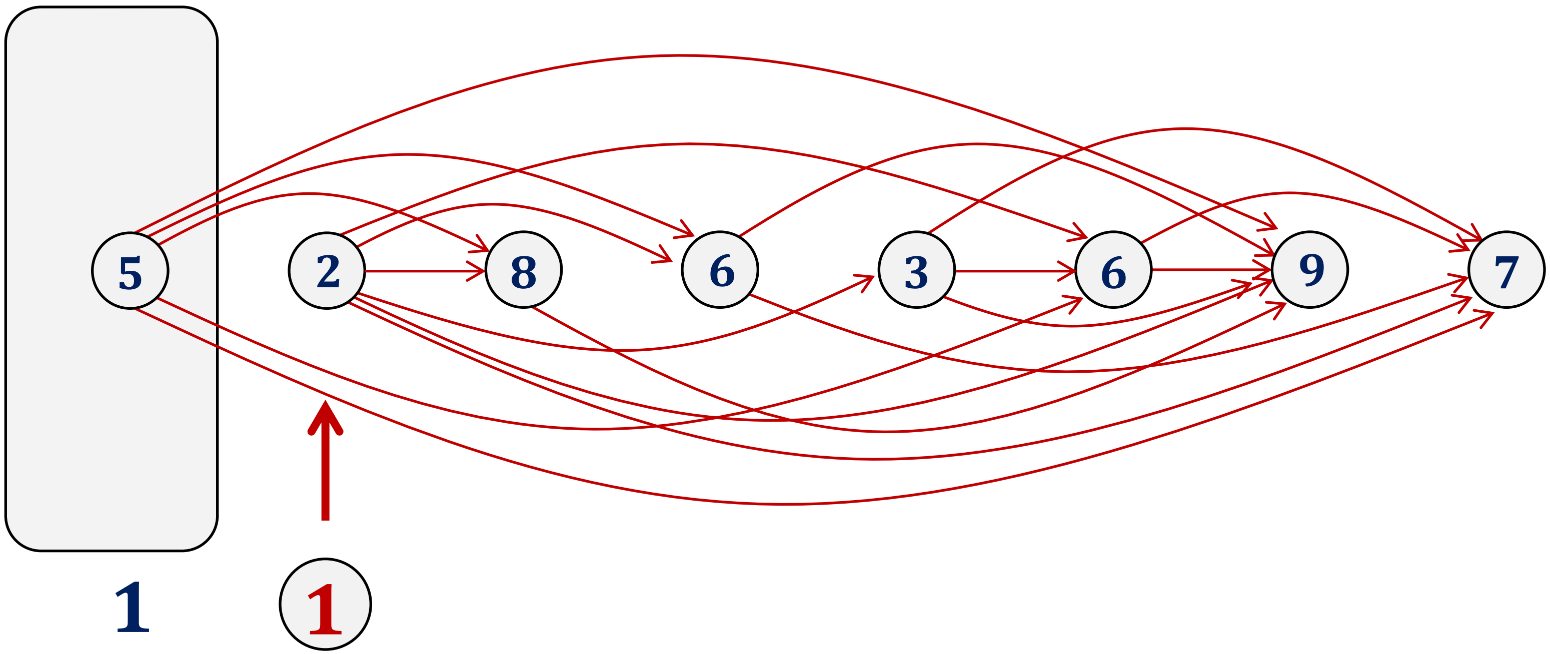
Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ



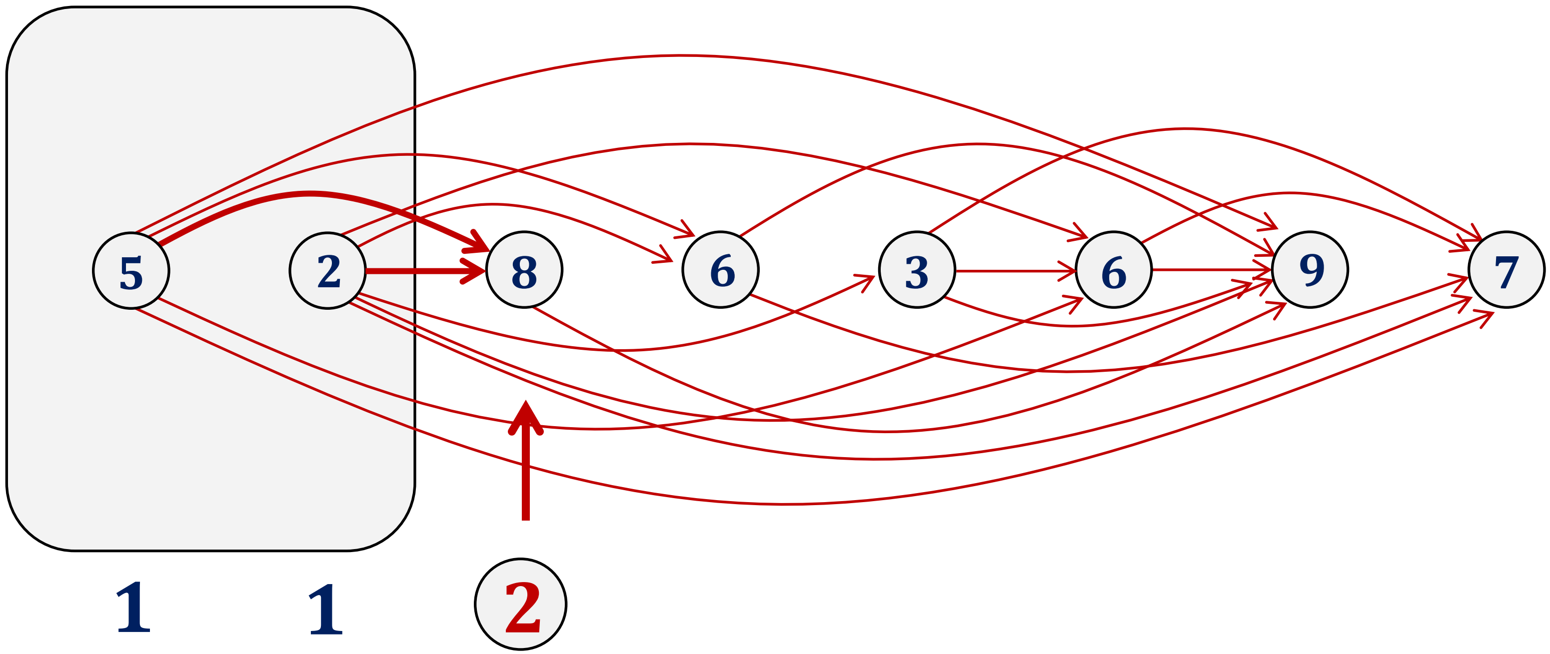
Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ



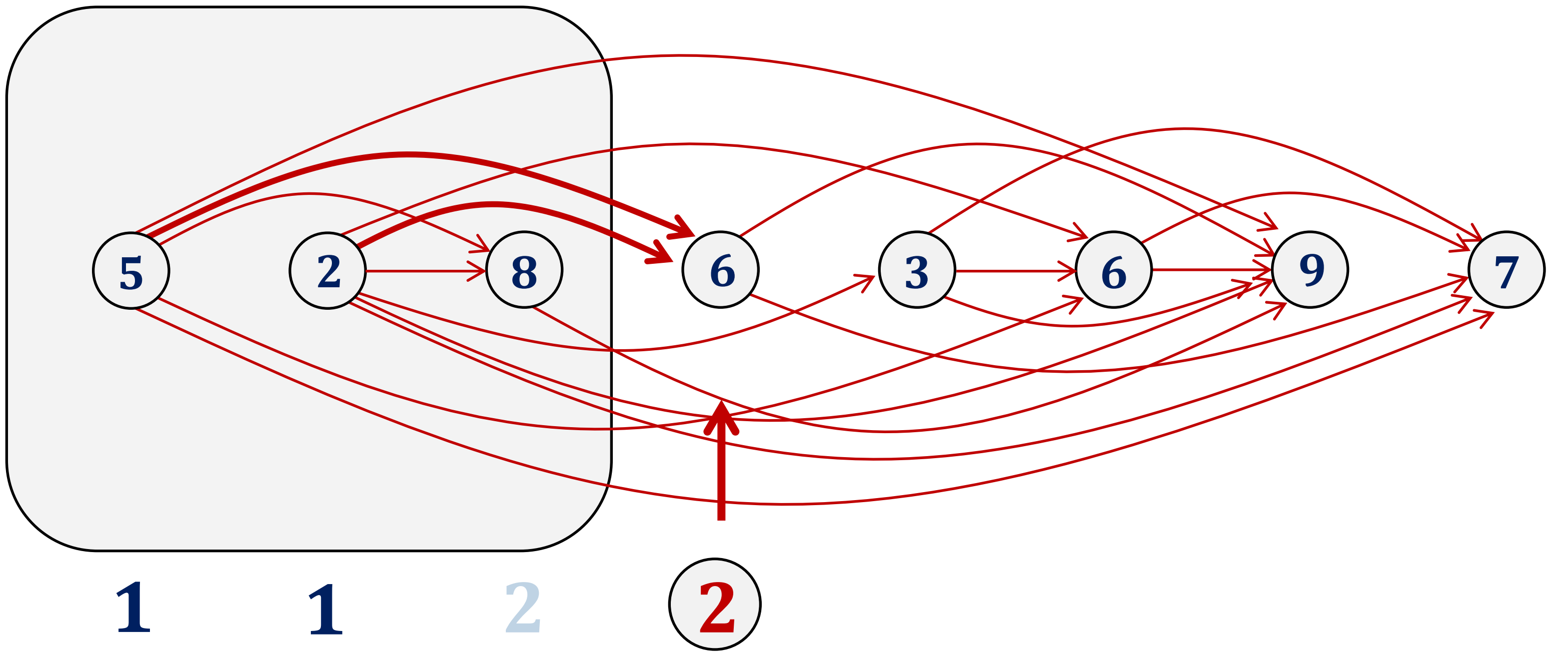
Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ



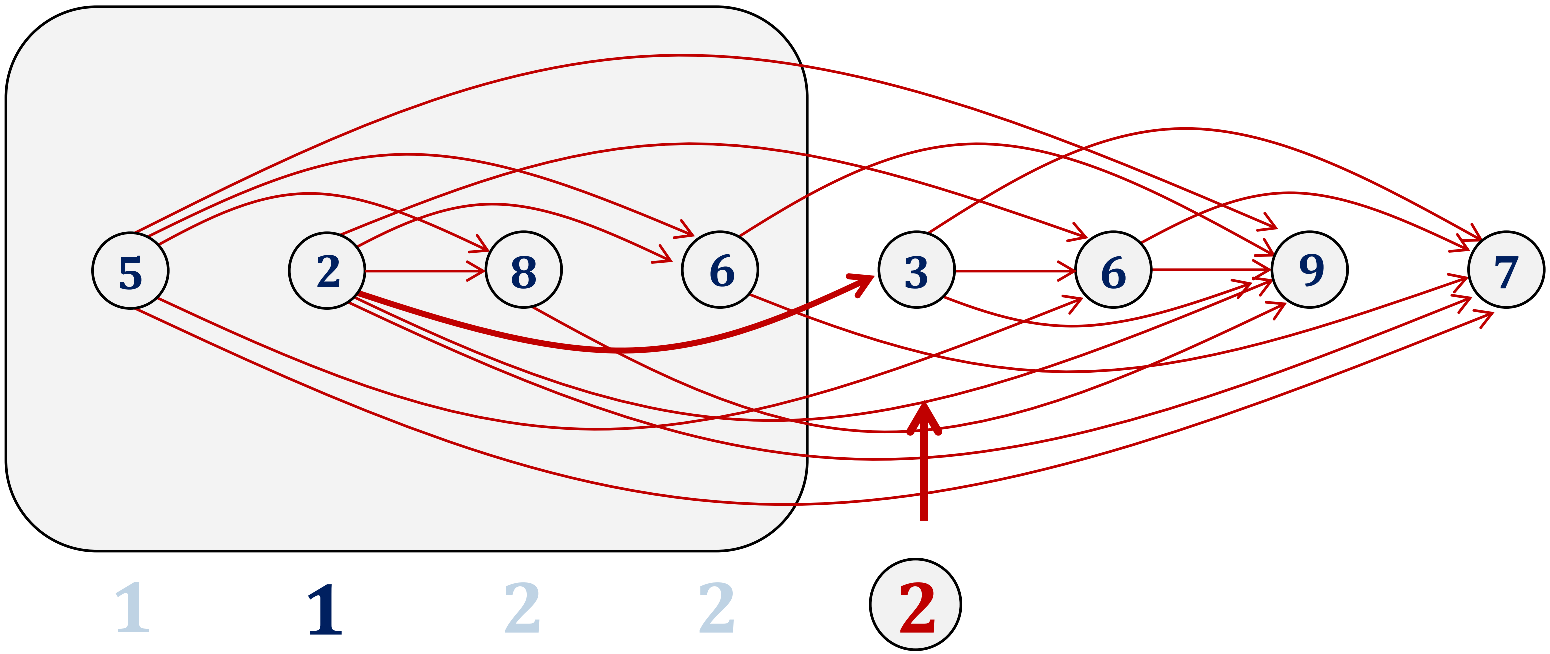
Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ



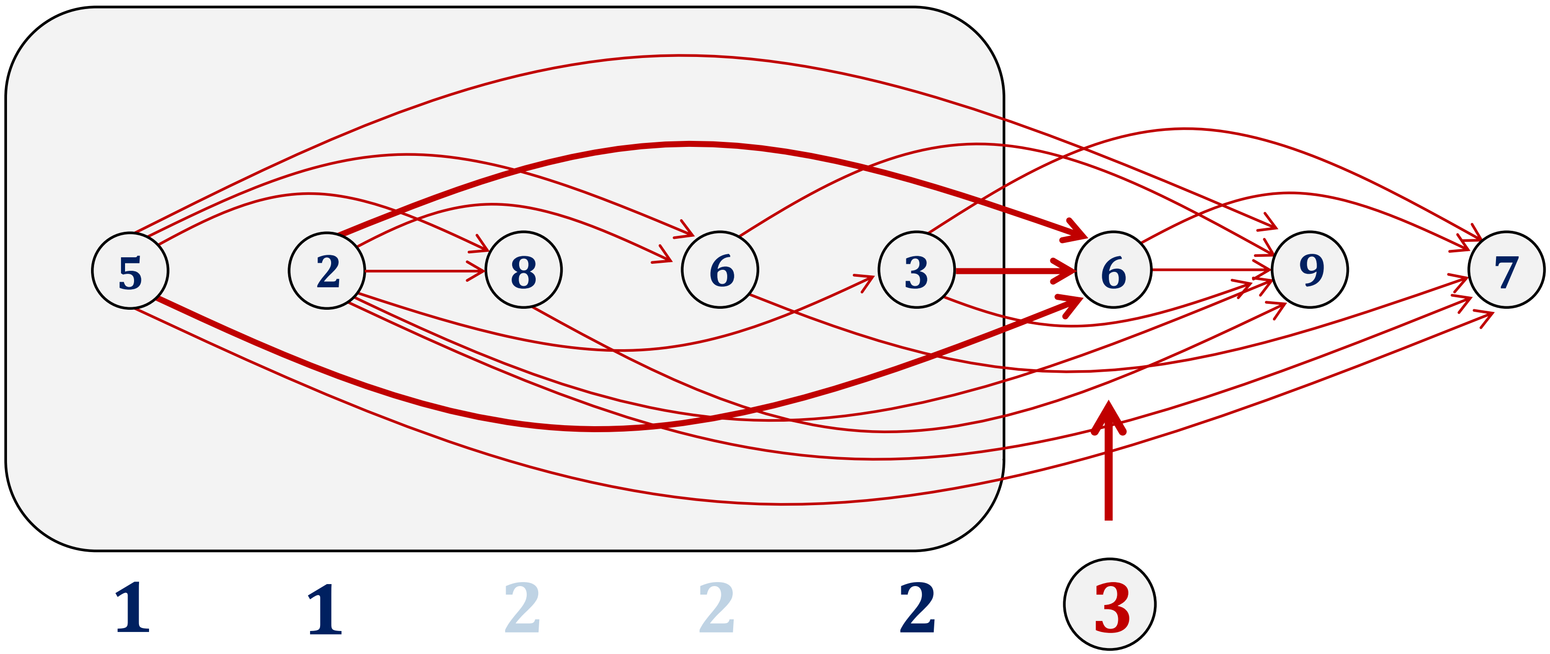
Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ



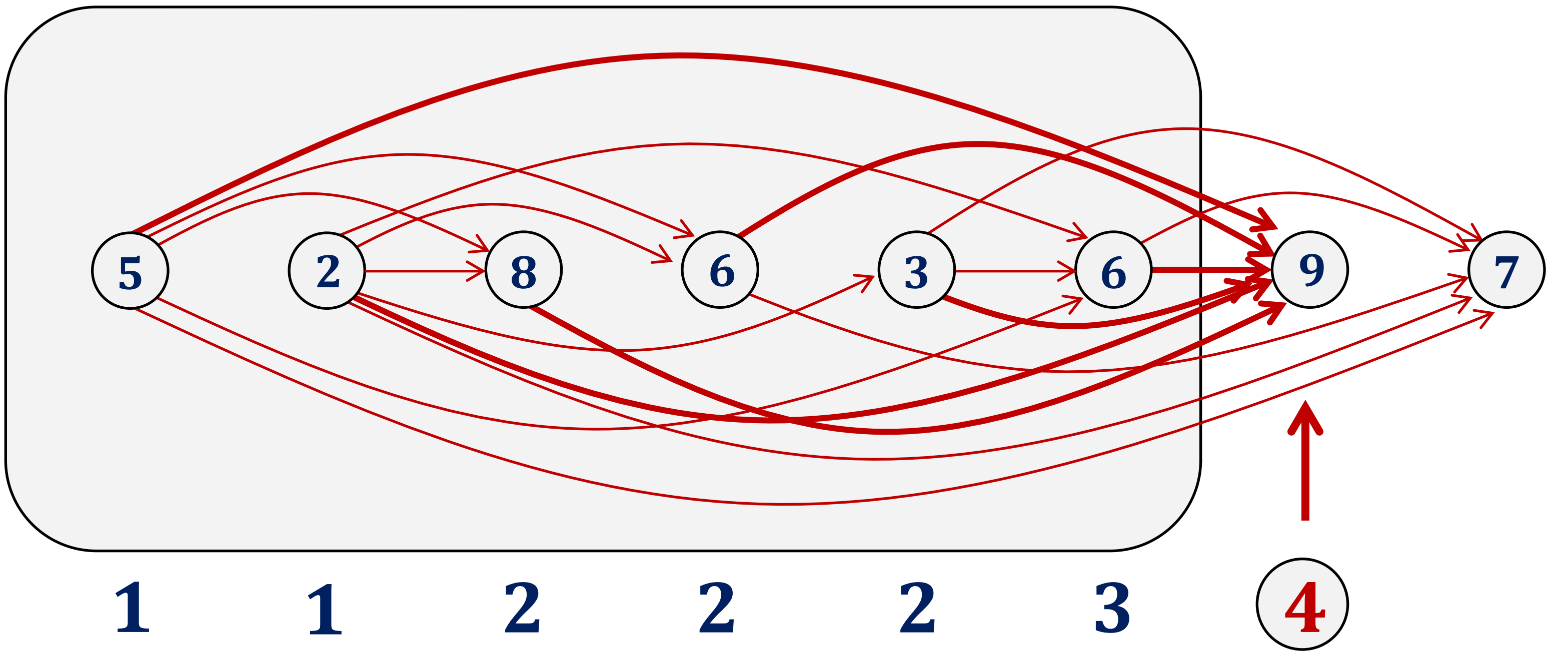
Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ



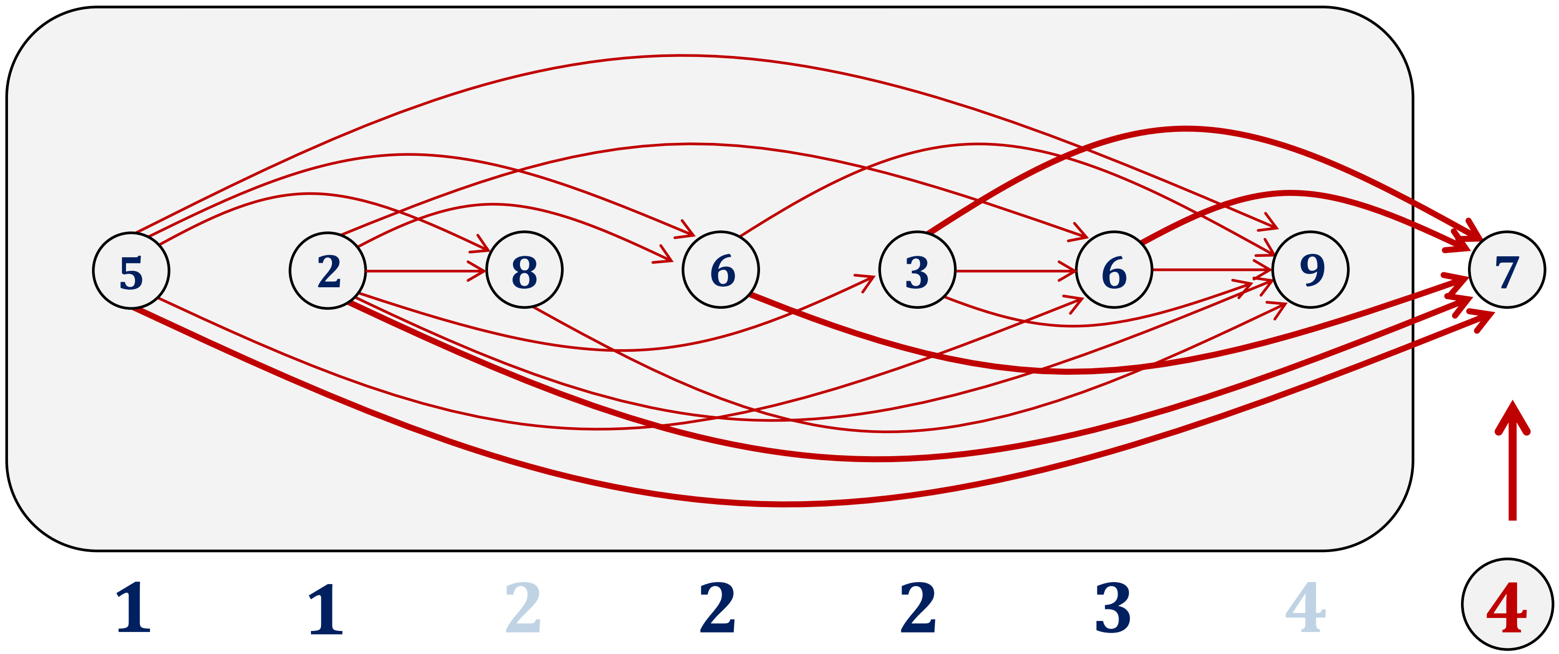
Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ



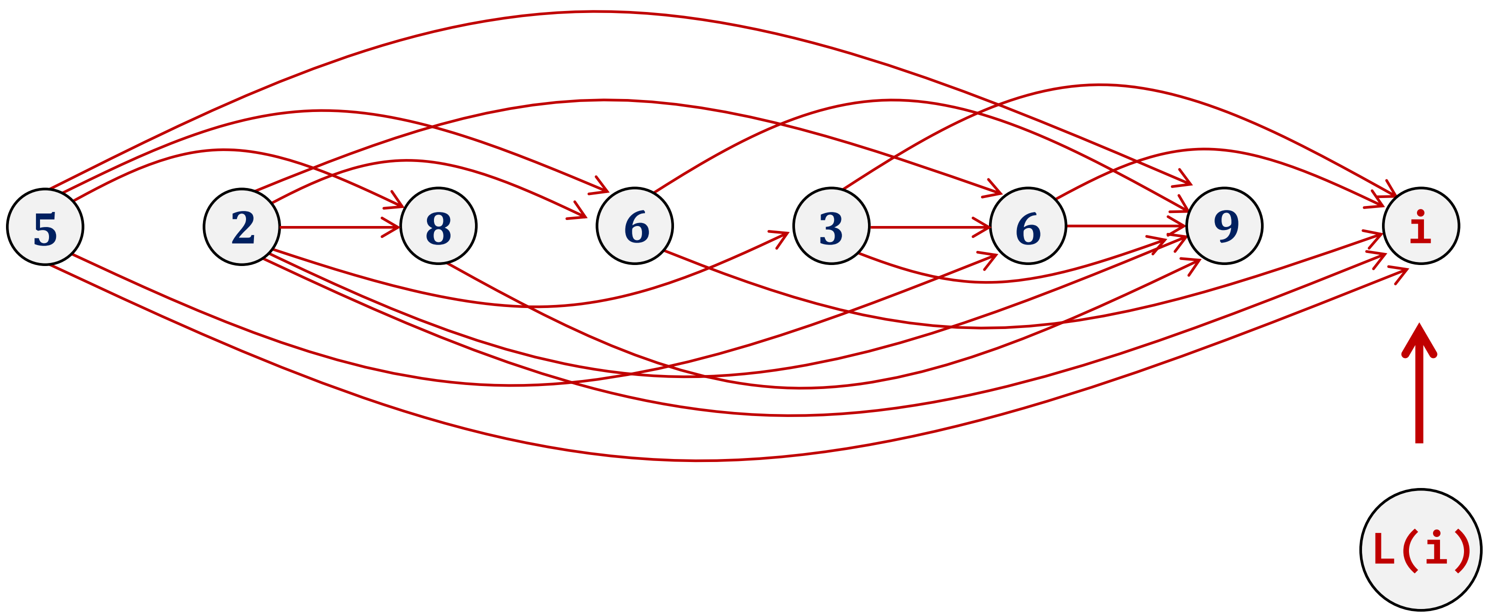
Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ



Μέγιστες Αύξουσες Υποακολουθίες

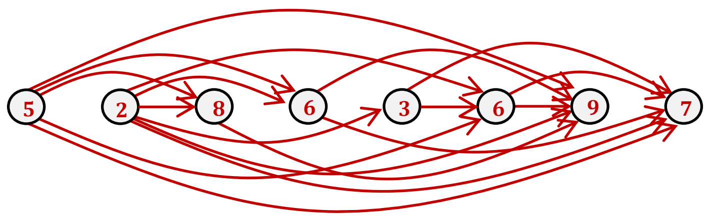
- Θα χρησιμοποιήσουμε ΔΠ




```
L(i) = 1 + max{L(j) : (j,i) ∈ E(G)}
```

Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ



  - Ορίστε ο αλγόριθμος:

```
1. για i = 1, 2, …, n:
```

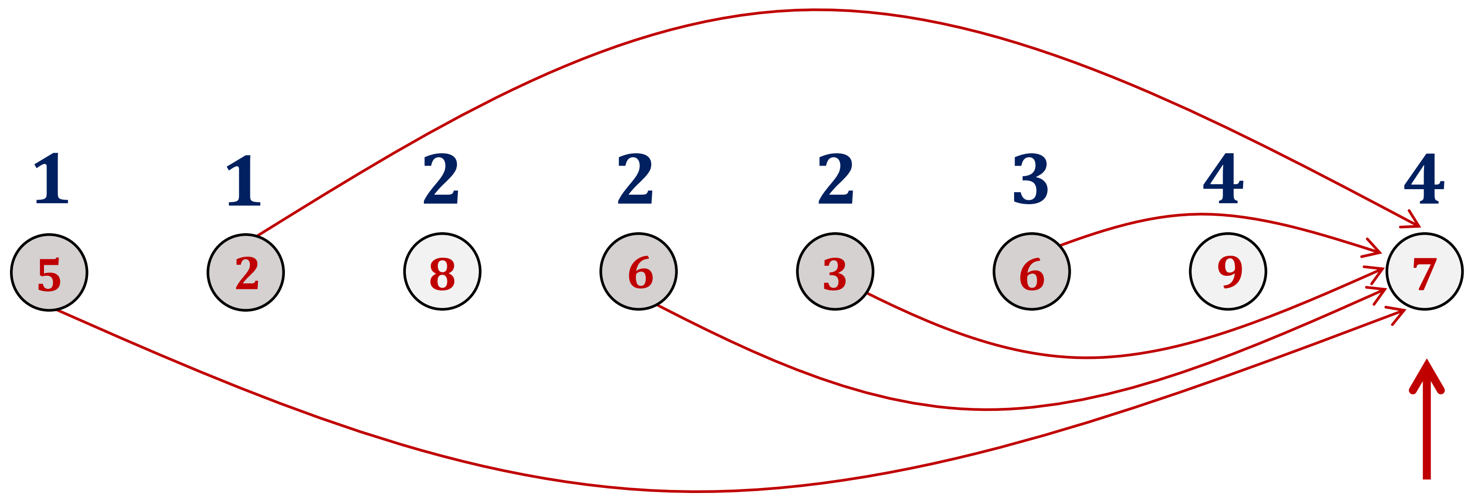
$$L(i) = 1 + \max\{L(j) : (j,i) \in E(G)\}$$

2. return $\max_{1 \le i \le n}\{L(i)\}$

L(i) = είναι το μήκος της μεγαλύτερης διαδρομής, ή ισοδύναμα, της μέγιστης αύξουσας υποακολουθίας, **που τελειώνει στον κόμβο i = 1, 2, …, n**

Μέγιστες Αύξουσες Υποακολουθίες

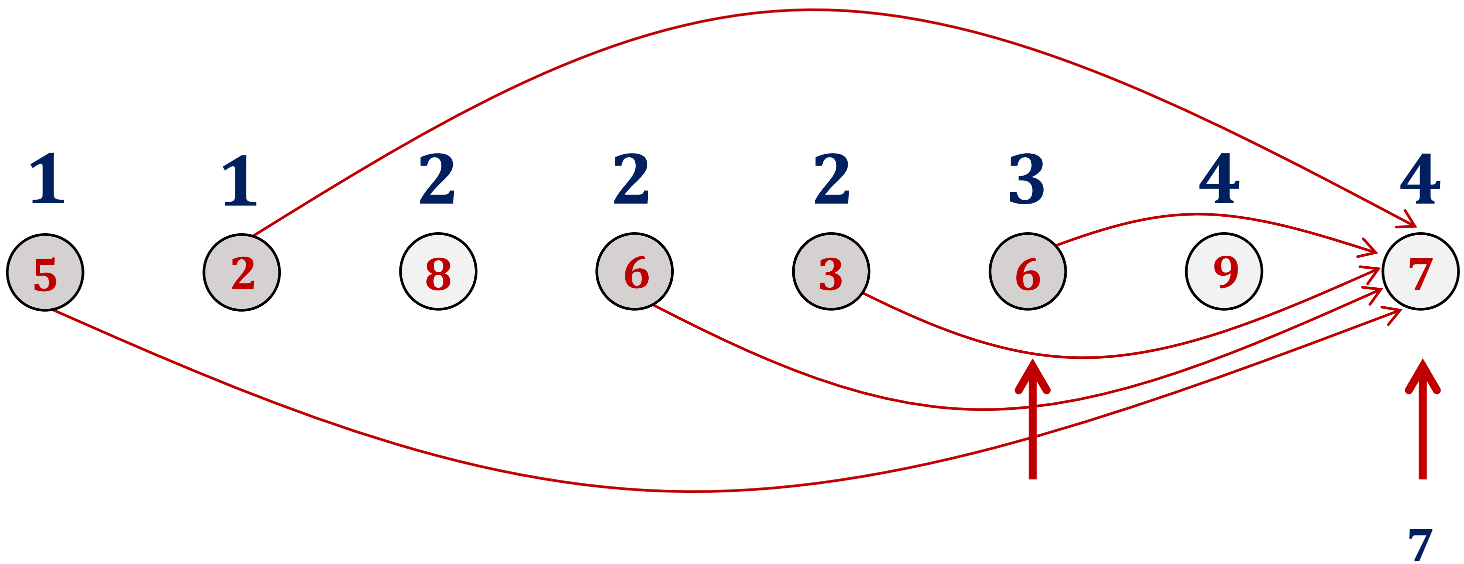
- Μέγιστη Αύξουσα Υποακολουθία: A = (5, 2, 8, 6, 3, 6, 9, 7)



Η μέγιστη αύξουσα υποακολουθία έχει μήκος 4

Μέγιστες Αύξουσες Υποακολουθίες

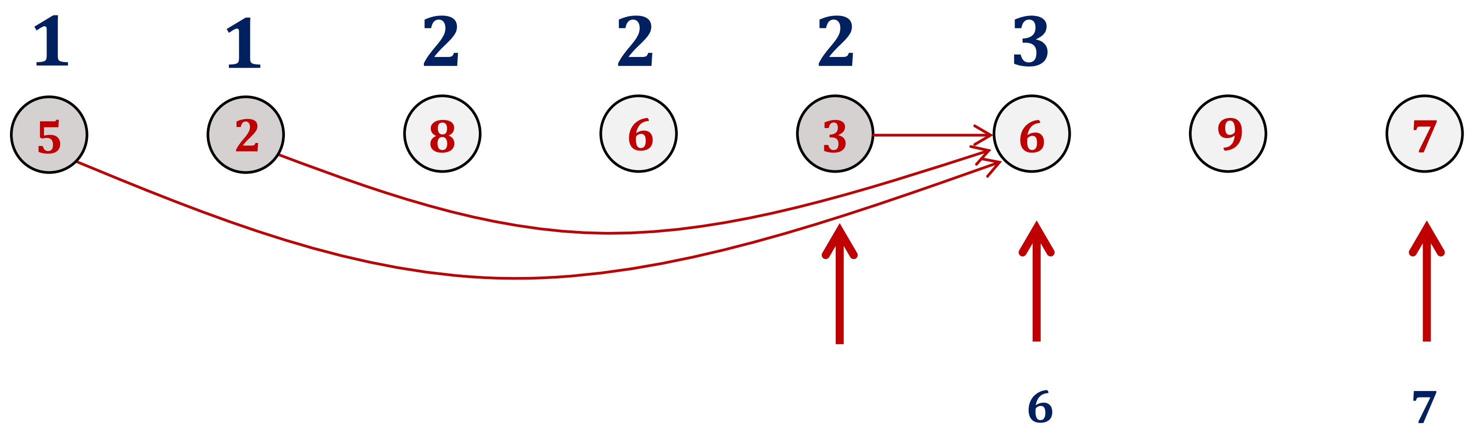
- Μέγιστη Αύξουσα Υποακολουθία: $A = (5, 2, 8, 6, 3, 6, 9, 7)$



Ποια είναι η υπακολουθία ?

Μέγιστες Αύξουσες Υποακολουθίες

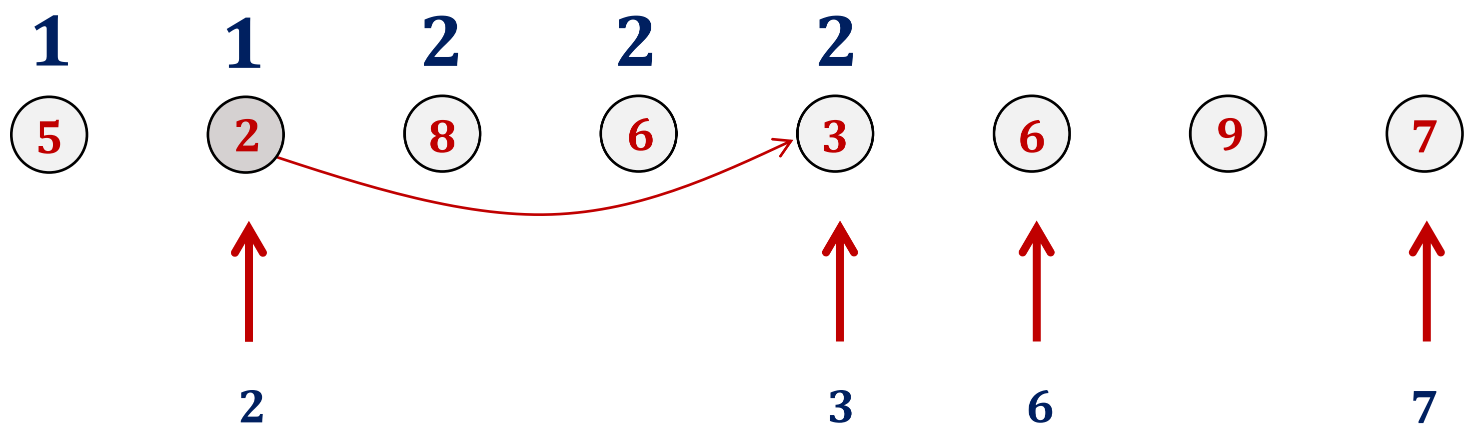
- Μέγιστη Αύξουσα Υπακολουθία: A = (5, 2, 8, 6, 3, 6, 9, 7)



Ποια είναι η υπακολουθία ?

Μέγιστες Αύξουσες Υποακολουθίες

- Μέγιστη Αύξουσα Υπακολουθία: A = (5, 2, 8, 6, 3, 6, 9, 7)



Ποια είναι η υπακολουθία ?

Μέγιστες Αύξουσες Υποακολουθίες

- Μέγιστη Αύξουσα Υπακολουθία: A = (5, 2, 8, 6, 3, 6, 9, 7)

Λύση !

(2, 3, 6, 7)

Ποια είναι η υπακολουθία ?

Μέγιστες Αύξουσες Υποακολουθίες

- **Γιατί είναι ΔΠ ?**
  - για i = 1, 2, …, n:

    $$L(i) = 1 + \max\{L(j) : (j,i) \in E(G)\}$$

  - Ορίσαμε, υποπροβλήματα: $\{L(i) : 1 \leq i \leq n\}$

    που έχουν τη Θεμελιώδη Ιδιότητα του ΔΠ !!!

> Υπάρχει μια **διάταξη** των υποπροβλημάτων,
> και μια **σχέση** που δείχνει **πως θα λυθεί** ένα υποπρόβλημα **έχοντας τις λύσεις «μικροτέρων»** υποπροβλημάτων !!!

Μέγιστες Αύξουσες Υποακολουθίες

Πολυπλοκότητα

- για i = 1, 2, …, n:

  $$L(i) = 1 + \max\{L(j) : (j,i) \in E(G)\}$$

- Το αρχικό μας πρόβλημα λύνεται με ένα «πέρασμα» των $n = |A|$ κόμβων

- Ο υπολογισμός του $L(i)$ απαιτεί χρόνο ανάλογο του βαθμού εισόδου του κόμβου i του G

  Συνολικά, χρόνο τάξης $|E(G)| \Rightarrow O(m)$

2 Διορθωτική Απόσταση

Κίνητρο

Όταν ένας ελεγκτής ορθογραφίας συναντά μια πιθανή ανορθόγραφη λέξη

$$\alpha_1\alpha_2 \ldots \alpha_n$$

εξετάζει το λεξικό του για άλλες λέξεις που είναι «κοντά» στην ανορθόγραφη!!

Λεξικό

$\beta_1\beta_2 \ldots \beta_m$

$\gamma_1\gamma_2 \ldots \gamma_k$

$\vdots$

$\omega_1\omega_2 \ldots \omega_p$

Ποια είναι η κατάλληλη έννοια της "εγγύτητας" δύο λέξεων ή συμβολοσειρών ?

2 Διορθωτική Απόσταση

Κίνητρο

Ένα μέτρο της "εγγύτητας" ανάμεσα σε δύο λέξεις ή συμβολοσειρές α και β, είναι οι **διορθώσεις** που πρέπει να κάνουμε, έτσι ώστε $\alpha \rightarrow \beta$ ή $\beta \rightarrow \alpha$

Ποιες διορθώσεις μπορούμε να κάνουμε ?

- Εισαγωγή συμβόλου
- Διαγραφή συμβόλου
- Αντικατάσταση συμβόλου

3 πράξεις σε συμβολοσειρές

2 Διορθωτική Απόσταση

Παράδειγμα

Έστω οι δύο λέξεις SNOWY και SUNNY.

Μπορούμε να πάρουμε την μία από την άλλη με διαφορετικά σύνολα αλλαγών/πράξεων

S	U	N	N	-	Y
S	–	N	O	W	Y

S	U	N	-	-	N	Y
-	S	N	O	W	-	Y

2 Διορθωτική Απόσταση

Παράδειγμα

Έστω οι δύο λέξεις SNOWY και SUNNY.

Μπορούμε να πάρουμε την μία από την άλλη με διαφορετικά σύνολα αλλαγών/πράξεων

S	U	N	N	-	Y
S	U	N	N	-	Y

S	U	N	-	-	N	Y
S	U	N	-	-	N	Y

2 Διορθωτική Απόσταση

Παράδειγμα

Έστω οι δύο λέξεις SNOWY και SUNNY.

Μπορούμε να πάρουμε την μία από την άλλη με διαφορετικά σύνολα αλλαγών/πράξεων

S - N O W Y   - S N O W - Y

S – N O W Y   - S N O W - Y

2 Διορθωτική Απόσταση

Ορίζουμε ως **κόστος διόρθωσης** δύο συμβολοσειρών α και β, το πλήθος των πράξεων που πρέπει να κάνουμε για να πάρουμε από την μία στην άλλη !!!

S	U	N	N	-	Y
S	–	N	O	W	Y

Κόστος **3**

S	U	N	-	-	N	Y
-	S	N	O	W	-	Y

Κόστος **5**

2 Διορθωτική Απόσταση

Το **ελάχιστο κόστος διόρθωσης** δύο συμβολοσειρών ονομάζεται **διορθωτική απόσταση (edit distance)** αυτών.

S	U	N	N	-	Y
S	–	N	O	W	Y

Κόστος **3**

S	U	N	-	-	N	Y
-	S	N	O	W	-	Y

Κόστος **5**

② Διορθωτική Απόσταση

Το **ελάχιστο κόστος διόρθωσης** δύο συμβολοσειρών ονομάζεται **διορθωτική απόσταση (edit distance)** αυτών.

S	U	N	N	-	Y
S	–	N	O	W	Y

Κόστος **3**

$\mathbf{E}(\mathtt{SUNNY}, \mathtt{SNOWY}) = \mathbf{3}$

2 Διορθωτική Απόσταση

Πρόβλημα

Μας δίνονται δύο συμβολοσειρές α και β με n και m σύμβολα, αντίστοιχα,

$$\alpha_1, \alpha_2, \ldots, \alpha_n \quad \text{και} \quad \beta_1, \beta_2, \ldots, \beta_m$$

και μας ζητείται να υπολογίσουμε την διορθωτική απόσταση

$$E(\alpha, \beta)$$

αυτών.

Διορθωτική Απόσταση

- Στο ΔΠ το πιο κρίσιμο ερώτημα είναι το εξής:

Ποια είναι τα υποπροβλήματα ?

- Όταν αυτά επιλέγονται ώστε να έχουν την

Θεμελιώδη Ιδιότητα του ΔΠ !!!

Υπάρχει μια **διάταξη** των υποπροβλημάτων,

και μια **σχέση** που δείχνει **πως θα λυθεί** ένα υποπρόβλημα **έχοντας τις λύσεις «μικροτέρων»** υποπροβλημάτων !!!

η διατύπωση ενός αλγόριθμου είναι εύκολη υπόθεση !!!

Τότε, επαναληπτικά επιλύουμε τα υποπροβλήματα κατά σειρά αυξανόμενου μεγέθους !

Διορθωτική Απόσταση

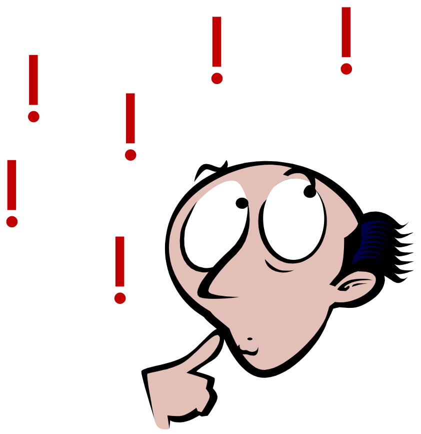
- Ποιο είναι ένα **καλό** υποπρόβλημα ?

  Θέλουμε την Ε(α,β) ανάμεσα στις δύο συμβολοσειρές

  $\alpha[1..n]=(\alpha_1, \alpha_2, \ldots, \alpha_n)$ και $\beta[1..m]= (\beta_1, \beta_2, \ldots, \beta_m)$

- Είναι **αυτό** που **«προχωράει»** τη λύση του συνολικού προβλήματος !!!

Πρέπει
να επινοήσουμε κάτι έξυπνο !



Τι θα λέγατε να εξετάσουμε την **Ε( )**
σε προθέματα (prefix) των **α** και **β** !

Διορθωτική Απόσταση

- Ας πάρουμε δύο προθέματα (prefix) των α και β, μήκους $i \le n$ και $j \le m$:

  $\alpha[1..i]=(\alpha_1, \alpha_2, \ldots, \alpha_i)$ και $\beta[1..j]= (\beta_1, \beta_2, \ldots, \beta_j)$

  και ας συμβολίσουμε $E(i,j)$ την διορθωτική απόσταση αυτού του υποπροβλήματος

  Παράδειγμα:

  $\boxed{\text{E X P O N E N}}$ T I A L   α[1..7]

  $\boxed{\text{P O L Y N}}$ O M I A L   β[1..5]

  Το υποπρόβλημα: $E(7,5)$

Διορθωτική Απόσταση

- Ας πάρουμε δύο προθέματα (prefix) των α και β, μήκους $i \le n$ και $j \le m$:

  $\alpha[1..i]=(\alpha_1, \alpha_2, \ldots, \alpha_i)$ και $\beta[1..j]= (\beta_1, \beta_2, \ldots, \beta_j)$

  και ας συμβολίσουμε E(i,j) την διορθωτική απόσταση αυτού του υποπροβλήματος

  Παράδειγμα:

$\boxed{\text{E X P O}}$ N E N T I A L	$\alpha[1..4]$
$\boxed{\text{P O}}$ L Y N O M I A L	$\beta[1..2]$

  Το υποπρόβλημα: E(4,2)

Διορθωτική Απόσταση

- Ας πάρουμε δύο προθέματα (prefix) των α και β, μήκους $i \leq n$ και $j \leq m$:

  $\alpha[1..i]=(\alpha_1, \alpha_2, \ldots, \alpha_i)$ και $\beta[1..j]= (\beta_1, \beta_2, \ldots, \beta_j)$

  και ας συμβολίσουμε $E(i,j)$ την διορθωτική απόσταση αυτού του υποπροβλήματος

  Παράδειγμα:

  $\boxed{\text{E}}$ X P O N E N T I A L $\qquad$ $\alpha[1..1]$

  $\boxed{\text{P O L Y N}}$ O M I A L $\qquad$ $\beta[1..5]$

  Το υποπρόβλημα: $E(1,5)$

Διορθωτική Απόσταση

- Ας πάρουμε δύο προθέματα (prefix) των α και β, μήκους $i \le n$ και $j \le m$:

$$\alpha[1..i]=(\alpha_1, \alpha_2, \ldots, \alpha_i) \quad \text{και} \quad \beta[1..j]= (\beta_1, \beta_2, \ldots, \beta_j)$$

και ας συμβολίσουμε E(i,j) την διορθωτική απόσταση αυτού του υποπροβλήματος

Παράδειγμα:

E X P O N E N T I A L	α[1..11]
P O L Y N O M I A L	β[1..10]

ΤΕΛΙΚΟΣ ΣΤΟΧΟΣ: **E(11, 10)**

Διορθωτική Απόσταση

- Ας πάρουμε δύο προθέματα (prefix) των α και β, μήκους i ≤ n και j ≤ m :

  $\alpha[1..i]=(\alpha_1, \alpha_2, \ldots, \alpha_i)$ και $\beta[1..j]= (\beta_1, \beta_2, \ldots, \beta_j)$

  και ας συμβολίσουμε E(i,j) την διορθωτική απόσταση αυτού του υποπροβλήματος

  Λύση:

E	X	P	O	N	E	N	-	T	I	A	L
-	-	P	O	L	Y	N	O	M	I	A	L

Βέλτιστη Λύση: **E(11, 10) = 6**

Διορθωτική Απόσταση

- Θα πρέπει να εκφράσουμε το υποπρόβλημα E(i,j) συναρτήσει μικρότερων υποπροβλημάτων !!!
- Τι γνωρίζουμε για την καλύτερη διόρθωση (στοίχιση) ανάμεσα στις συμβολοσειρές α[1..i] και β[1..j] ?
- Γνωρίζουμε ότι για την **δεξιότερη στήλη** ισχύει μία από τις ακόλουθες τρεις περιπτώσεις :

α[i]	ή	-	ή	α[i]
-		β[j]		β[j]

Διορθωτική Απόσταση

- Κόστος σε κάθε περίπτωση:

α[i] -	ή	- β[j]	ή	α[i] β[j]

Διορθωτική Απόσταση

- Κόστος σε κάθε περίπτωση:

α[i]	ή	**-**	ή	**α[i]**
-		**β[j]**		**β[j]**

- **1**η Περίπτωση: Επιφέρει κόστος **1**
  Και απομένει να διορθώσουμε (στοιχίσουμε) τις συμβολοσειρές

  α[1..i-1] και β[1..j]

Όμως, αυτό είναι ακριβώς το υποπρόβλημα **E(i-1, j)**

Διορθωτική Απόσταση

- Κόστος σε κάθε περίπτωση:

α[i] **-**	ή	**-** **β[j]**	ή	**α[i]** **β[j]**

- **2η** Περίπτωση: Επιφέρει κόστος **1**
  Και απομένει να διορθώσουμε (στοιχίσουμε) τις συμβολοσειρές

  α[1..i] και β[1..j-1]

Όμως, αυτό είναι ακριβώς το υποπρόβλημα **E(i, j-1)**

Διορθωτική Απόσταση

- Κόστος σε κάθε περίπτωση:

α[i]	ή	-	ή	α[i]
-		β[j]		β[j]

- **3η** Περίπτωση: Επιφέρει κόστος **1** (εάν α[i]≠β[j]) ή κόστος **0** (εάν α[i]=β[j]) Και απομένει να διορθώσουμε τις

  α[1..i-1] και β[1..j-1]

Τώρα, το υποπρόβλημα είναι ακριβώς το **E(i-1, j-1)**

Διορθωτική Απόσταση

- Επομένως, έχουμε εκφράσει το **E(i, j)** συναρτήσει ***τριών μικροτέρων*** υποπροβλημάτων:

E(i-1, j) **E(i, j-1)** **E(i-1, j-1)**

- Όμως, δεν γνωρίζουμε ποιο από αυτά είναι σωστό!!!

  Οπότε θα τα δοκιμάσουμε όλα και θα επιλέξουμε το καλύτερο:

$$E(i,j) = \min\{1+E(i-1,j),\ 1+E(i,j-1),\ \{0|1\} +E(i-1,j-1)\}$$

Διορθωτική Απόσταση

- **Αλγόριθμος**

```
Edit-distance(a[1..n],b[1..m])
1. Initialize()  →   for i ← 1 to n
                           E(i,0)=i
                     for j ← 1 to m
                           E(0,j)=j
2. for i ← 1 to n
       for j ← 1 to m
           E(i,j) ← min{1+E(i-1,j), 1+E(i,j-1),
                        {0|1}+E(i-1,j-1)}
3. return E(n,m)
```

Διορθωτική Απόσταση

- **Αλγόριθμος**

```
Edit-distance(a[1..n],b[1..m])
1. Initialize()
2. for i←1 to n
       for j←1 to m
           E(i,j) ← min{1+E(i-1,j), 1+E(i,j-1),
                        {0|1} +E(i-1,j-1)}
3. return E(n,m)
```

Συνολική Πολυπλοκότητα

O(nm)

Διορθωτική Απόσταση

Παράδειγμα

Ας πάρουμε τις συμβολοσειρές και το υποπρόβλημα

$\boxed{\texttt{E X P O}}$ `N E N T I A L`	**E(4, 3)**
$\boxed{\texttt{P O L}}$ `Y N O M I A L`	

- Το **E(4,3)** αντιστοιχεί στις συμβολοσειρές **EXPO** και **POL**

- **Εδώ, θέσαμε το Ερώτημα !!!**

  Τι ισχύει για την δεξιότερη στήλη σε μια βέλτιστη λύση του υποπροβλήματος E(4,3) ?

Διορθωτική Απόσταση

Παράδειγμα

Ας πάρουμε τις συμβολοσειρές και το υποπρόβλημα

E X P O N E N T I A L

P O L Y N O M I A L

E(4, 3)

- Τρεις περιπτώσεις μπορούν να ισχύουν:

$\begin{matrix} O \\ - \end{matrix}$ ή $\begin{matrix} - \\ L \end{matrix}$ ή $\begin{matrix} O \\ L \end{matrix}$

Άρα, $E(4,3) = \min\{1+E(3,3), 1+E(4,2), 1+E(3,2)\}$

Διορθωτική Απόσταση

Παράδειγμα

Οι λύσεις όλων των υποπρ/μάτων **E(i,j)** σχηματίζουν ένα **11×10** διδιάστατο πίνακα

		P	O	L	Y	N	O	M	I	A	L
	0	1	2	3	4	5	6	7	8	9	10
E	1	1	2	3	4	5	6	7	8	9	10
X	2	2	2	3	4	5	6	7	8	9	10
P	3	2	3	3	4	5	6	7	8	9	10
O	4	3	2	3	4	5	5	6	7	8	9
N	5	4	3	3	4	4	5	6	7	8	9
E	6	5	4	4	4	5	5	6	7	8	9
N	7	6	5	5	5	4	5	6	7	8	9
T	8	7	6	6	6	5	5	6	7	8	9
I	9	8	7	7	7	6	6	6	6	7	8
A	10	9	8	8	8	7	7	6	7	6	7
L	11	10	9	8	9	8	8	8	8	7	6

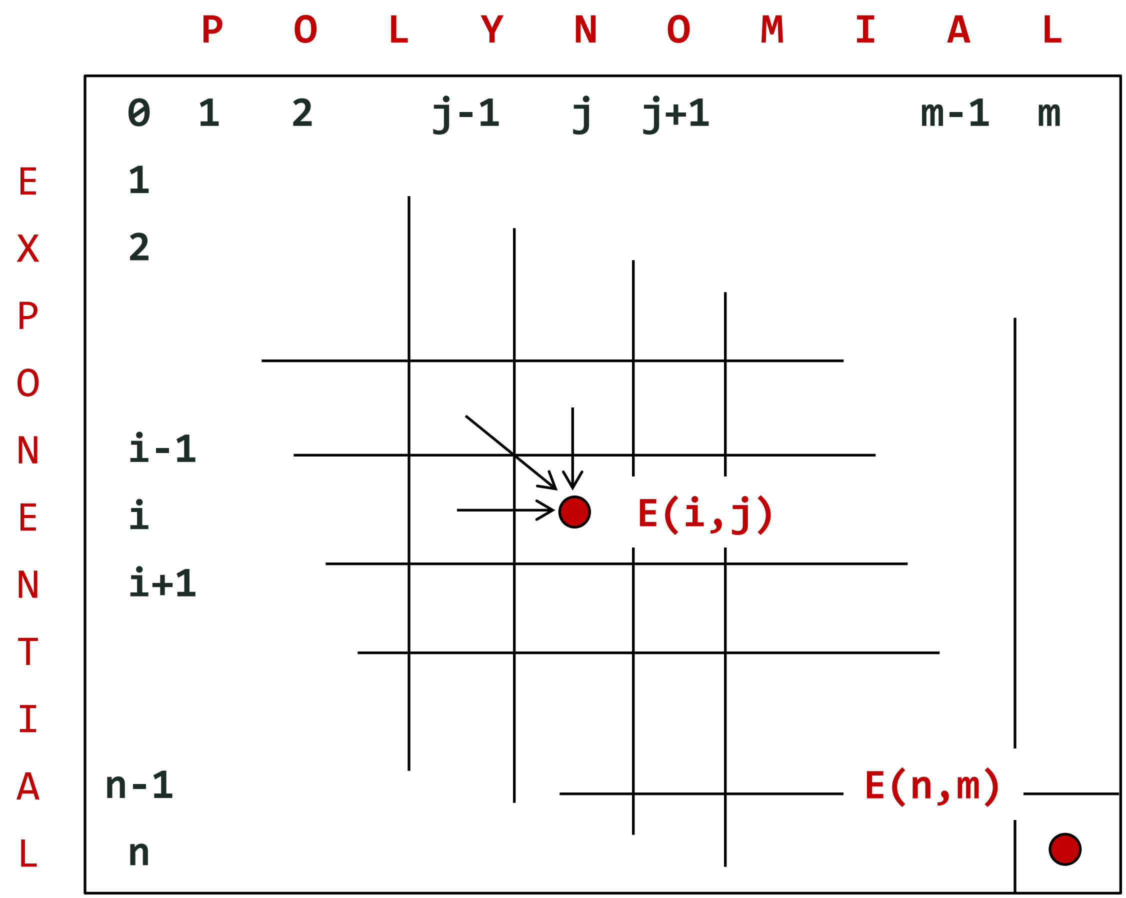
ΛΥΣΗ

E(11,10) = 6

Διορθωτική Απόσταση

Παράδειγμα

Συνολικά, τα υποπροβλήματα **E(i,j)** σχηματίζουν ένα **n×m** διδιάστατο πίνακα



ΛΥΣΗ

Διορθωτική απόσταση = E(n,m)

Διορθωτική Απόσταση

- Παράδειγμα

Αλγόριθμος

```
Edit-distance(a[1..11],b[1..10])
  a[1..11] = EXPONENTIAL
  b[1..10] = POLYNOMIAL
return E(11,10) = 6
```

E	X	P	O	N	E	N	-	T	I	A	L
-	-	P	O	L	Y	N	O	M	I	A	L

3 Πολλαπλασιασμός Ακολουθίας Πινάκων

Έστω ότι θέλουμε να πολλαπλασιάσουμε 4 πίνακες

$$A \times B \times C \times D$$

διαστάσεων

$$50 \times 20, \quad 20 \times 1, \quad 1 \times 10 \quad \text{και} \quad 10 \times 100$$

- Ο πολλαπλασιασμός πινάκων δεν είναι αντιμεταθετικός ($A \times B \neq B \times A$), αλλά είναι **προσεταιριστικός**, που σημαίνει: $(A \times B) \times C = A \times (B \times C)$

Μπορούμε να υπολογίσουμε το γινόμενο των 4 πινάκων μας με πολλούς διαφορετικούς τρόπους !

3 Πολλαπλασιασμός Ακολουθίας Πινάκων

Έστω ότι θέλουμε να πολλαπλασιάσουμε 4 πίνακες

$$A \times B \times C \times D$$

διαστάσεων

$$50 \times 20, \quad 20 \times 1, \quad 1 \times 10 \quad \text{και} \quad 10 \times 100$$

Ερώτημα !!! Υπάρχει διαφορά ?

Εάν ΝΑΙ : Ποιος τρόπος είναι ο καλύτερος ?

3 Πολλαπλασιασμός Ακολουθίας Πινάκων

Πολλαπλασιασμός

$$A \times B \times C \times D$$

A	50	×	20
B	20	×	1
C	1	×	10
D	10	×	100

Ο πολλ/σμός δύο πινάκων $A \times B$ διαστάσεων $n \times m$ και $m \times p$ απαιτεί $n \cdot m \cdot p$ πολλαπλασιασμούς

Διάταξη Παρενθέσεων	Υπολογισμός Κόστους	Κόστος
$A \times ((B \times C) \times D)$	$20 \cdot 1 \cdot 10 + 20 \cdot 10 \cdot 100 + 50 \cdot 20 \cdot 100$	120.200
$(A \times (B \times C)) \times D$	$20 \cdot 1 \cdot 10 + 50 \cdot 20 \cdot 10 + 50 \cdot 10 \cdot 100$	60.200
$(A \times B) \times (C \times D)$	$50 \cdot 20 \cdot 1 + 1 \cdot 10 \cdot 100 + 50 \cdot 1 \cdot 100$	7.000

3 Πολλαπλασιασμός Ακολουθίας Πινάκων

Πρόβλημα

Μας δίδεται μια ακολουθία n πινάκων (συμβατών διαστάσεων)

$$A_1, A_2, \ldots, A_n$$

και μας ζητείται να υπολογίσουμε το γινόμενο

$$A_1 \times A_2 \times \ldots \times A_n$$

κάνοντας το ελάχιστο πλήθος πολλαπλασιασμών.

Πολλαπλασιασμός Ακολουθίας Πινάκων

- Από το εισαγωγικό παράδειγμα είδαμε ότι η σειρά των πολλαπλασιασμών επηρεάζει πολύ τον χρόνο επίλυσης του προβλήματος !!!!

Διάταξη Παρενθέσεων	Υπολογισμός Κόστους	Κόστος
$A \times ((B \times C) \times D)$	$20{\cdot}1{\cdot}10 + 20{\cdot}10{\cdot}100 + 50{\cdot}20{\cdot}100$	120.200
$(A \times (B \times C)) \times D$	$20{\cdot}1{\cdot}10 + 20{\cdot}20{\cdot}10 + 50{\cdot}10{\cdot}100$	60.200
$(A \times B) \times (C \times D)$	$50{\cdot}20{\cdot}1 + 1{\cdot}10{\cdot}100 + 50{\cdot}1{\cdot}100$	7.000

- **Να ακολουθήσουμε άπληστη προσέγγιση ?** επιλέγοντας πάντοτε το φθηνότερο διαθέσιμο πολλ/σμό !

Αποτυχία !!!

Πολλαπλασιασμός Ακολουθίας Πινάκων

- **Πως** θα βρούμε τη **βέλτιστη σειρά** των πολλ/μών **?** όταν θέλουμε να υπολογίσουμε το γινόμενο n πινάκων

$$A_1 \times A_2 \times A_3 \times \ldots \times A_{n-1} \times A_n$$

διαστάσεων

$$m_0 \times m_1,\ \ m_1 \times m_2,\ \ m_2 \times m_3,\ \ldots,\ m_{n-2} \times m_{n-1},\ \ m_{n-1} \times m_n.$$

Παρατήρηση

Μια σειρά πολλ/μών μπορεί να αναπαρασταθεί πολύ φυσικά με ένα δυαδικό δένδρο:

- ✓ οι αρχικοί πίνακες αντιστοιχούν στα φύλλα
- ✓ οι ενδιάμεσοι κόμβοι είναι τα ενδιάμεσα γινόμενα
- ✓ η ρίζα είναι το τελικό αποτέλεσμα

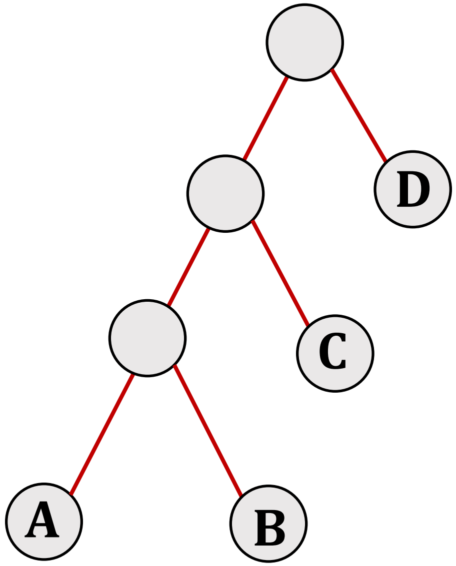
Πολλαπλασιασμός Ακολουθίας Πινάκων

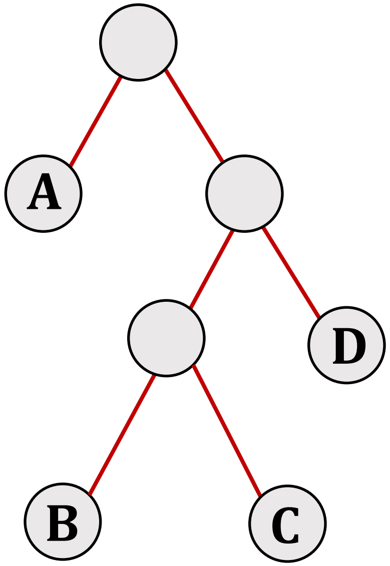
- Παράδειγμα

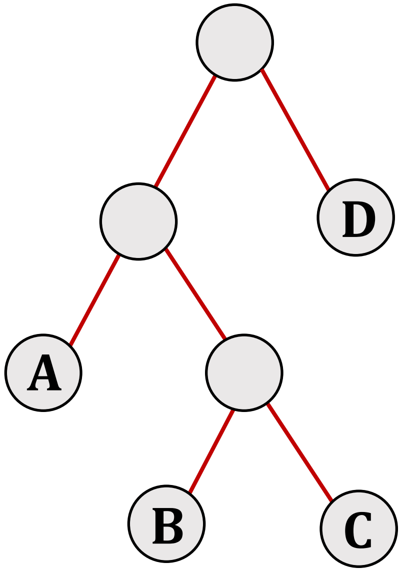
$((A \times B) \times C) \times D)$

$A \times ((B \times C) \times D)$

$(A \times (B \times C)) \times D$

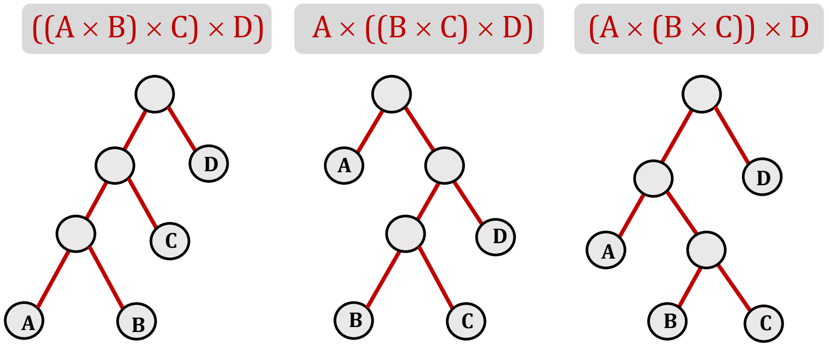






Πολλαπλασιασμός Ακολουθίας Πινάκων

Δένδρα - Διατάξεις



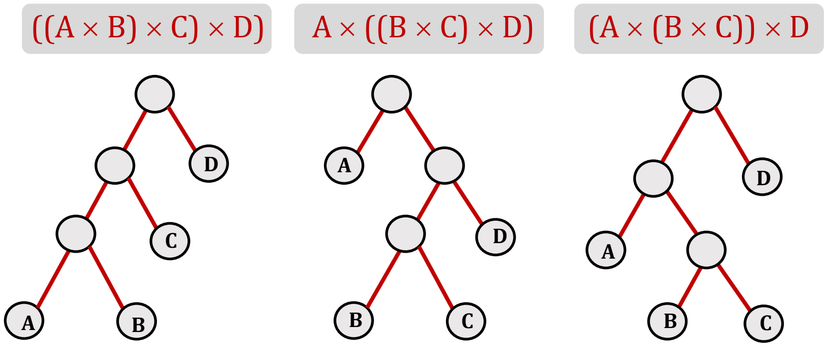
- Οι δυνατές διατάξεις με τις οποίες μπορεί να εκτελεστεί ο πολλ/σμός n πινάκων **αντιστοιχούν** στα διαφορετικά πλήρη δυαδικά δένδρα n κόμβων

- **Πόσα είναι τα δένδρα αυτά ?** **$\Omega(2^n)$**

  Δυστυχώς, το πλήθος τους είναι εκθετικό !!!

Πολλαπλασιασμός Ακολουθίας Πινάκων

Δένδρα - Διατάξεις



- Ένα από τα δένδρα αυτά είναι **βέλτιστο**!!
- Εάν το δένδρο **T** είναι βέλτιστο, τότε είναι βέλτιστα και τα υποδένδρα $\mathbf{T_1}$ και $\mathbf{T_2}$ της ρίζας του.

Ερώτηση! Ποια είναι τα υποπροβλήματα που αντιστοιχούν στα υποδένδρα $\mathbf{T_1}$ και $\mathbf{T_2}$ του βέλτιστου δένδρου **T ?**

Πολλαπλασιασμός Ακολουθίας Πινάκων

- Τα υποπροβλήματα που αντιστοιχούν στα υποδένδρα $\mathbf{T_1}$ και $\mathbf{T_2}$ της ρίζας ενός βέλτιστου δένδρου $\mathbf{T}$

  είναι γινόμενα της μορφής: $\mathbf{A_1 \times A_2 \times \ldots \times A_k}$ (T_1)

  $\mathbf{A_{k+1} \times \ldots \times A_n}$ (T_2)

- Γενικά, το υποπρόβλημα που αντιστοιχεί σε ένα υποδένδρο ενός βέλτιστου δένδρου $\mathbf{T}$

  είναι γινόμενο της μορφής: $\mathbf{A_i \times A_{i+1} \times \ldots \times A_j = A_{i..j}}$

Πολλαπλασιασμός Ακολουθίας Πινάκων

- **Ορίζουμε**

  C[i,j] = min κόστος του πολλ/σμού $A_i \times A_{i+1} \times \ldots \times A_j$

  όπου κόστος είναι το πλήθος των πολλαπλασιασμών αυτού του υποπροβλήματος

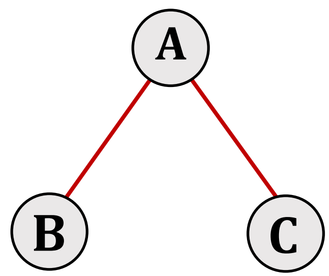
- Το μικρότερο υποπρόβλημα το παίρνουμε όταν **i = j**, όπου σε αυτή την περίπτωση δεν έχουμε κάτι να πολλαπλασιάσουμε, οπότε C[i,j]=0

Πολλαπλασιασμός Ακολουθίας Πινάκων

- Θεωρούμε το **βέλτιστο υποδένδρο** $\mathbf{T_{ij}}$, $j > i$, για το γινόμενο $A_i \times A_{i+1} \times \ldots \times A_j$

- Η ρίζα του υποδένδρου $\mathbf{T_{ij}}$ χωρίζει το γινόμενο

$$\mathbf{A_i \times A_{i+1} \times \ldots \times A_j}$$



$\mathbf{A_i \times A_{i+1} \times \ldots \times A_k}$ $\mathbf{A_{k+1} \times A_{k+2} \times \ldots \times A_j}$

σε δύο τμήματα, για κάποιο k, $i \leq k \leq j$.

Πολλαπλασιασμός Ακολουθίας Πινάκων

- Τότε, το κόστος C[i,j] του υποδένδρου $\mathbf{T_{ij}}$ θα είναι:

το κόστος υπολογισμού των μερικών γινομένων

$$\mathbf{A_i \times A_{i+1} \times \ldots \times A_k} \quad \text{και} \quad \mathbf{A_{k+1} \times A_{k+2} \times \ldots \times A_j}$$

συν το κόστος του πολλ/σμού των πινάκων

$$\mathbf{A_{i..k}} \text{ και } \mathbf{A_{k+1..j}}$$

που είναι $m_{i-1} \cdot m_k \cdot m_j$.

Διάσταση $\mathbf{A_i}$

$m_{i-1} \times m_i$

Πρέπει απλώς να βρούμε το σημείο χωρισμού **k** για το οποίο το κόστος C[i,j] είναι το μικρότερο!!!

Πολλαπλασιασμός Ακολουθίας Πινάκων

- Τότε, το κόστος C[i,j] του υποδένδρου $\mathbf{T_{ij}}$ θα είναι:

το κόστος υπολογισμού των μερικών γινομένων

$$\mathbf{A_i \times A_{i+1} \times \ldots \times A_k} \quad \text{και} \quad \mathbf{A_{k+1} \times A_{k+2} \times \ldots \times A_j}$$

συν το κόστος του πολλ/σμού των πινάκων

$$\mathbf{A_{i\,..k}} \text{ και } \mathbf{A_{k+1\,..j}}$$

που είναι $m_{i-1} \cdot m_k \cdot m_j$.

$$\mathbf{C[i,j] = \min_{i \le k \le j} \{C[i,k] + C[k+1,j] + m_{i-1} \cdot m_k \cdot m_j\}}$$

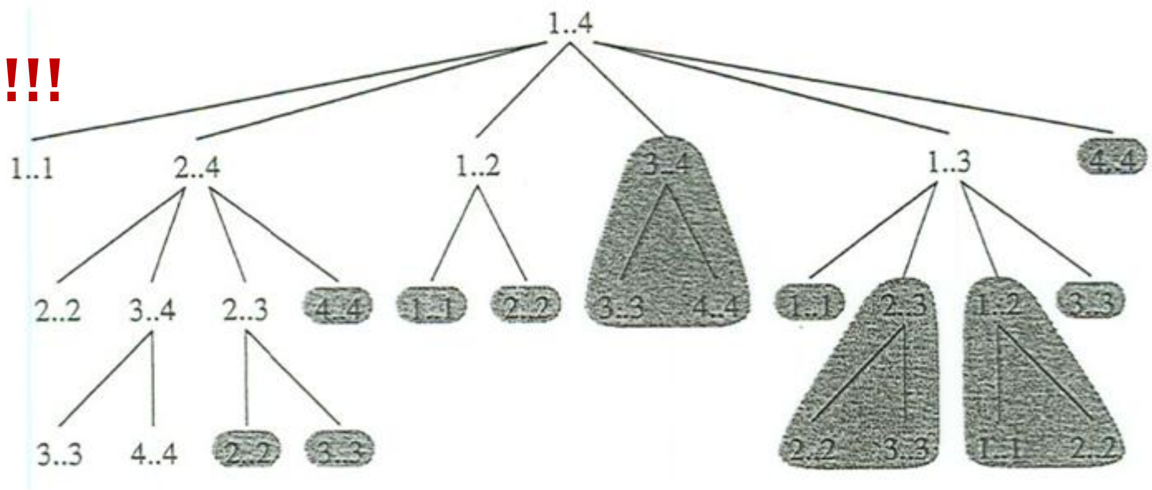
Πολλαπλασιασμός Ακολουθίας Πινάκων

- Σχέση που δίδει τη λύση ενός υποπροβλήματος από «μικρότερα» υποπροβλήματα

$$C[i,j] = \min_{i \le k \le j} \{C[i,k] + C[k+1,j] + m_{i-1} \cdot m_k \cdot m_j\}$$

- **Αναδρομή?**
  ΌΧΙ, ευχαριστώ!!!

  $\Omega(2^n)$



Πολλαπλασιασμός Ακολουθίας Πινάκων

- Σχέση που δίδει τη λύση ενός υποπροβλήματος από «μικρότερα» υποπροβλήματα

$$C[i,j] = \min_{i \leq k \leq j} \{C[i,k] + C[k+1,j] + m_{i-1} \cdot m_k \cdot m_j\}$$

- **Δυναμικός Προγραμματισμός ?**
  ΝΑΙ, θα πάρω!!!

Ερώτηση: Πόσα υποπροβλήματα έχω ?

Ένα για κάθε ζεύγος (i, j) $\Rightarrow$ $\mathbf{O(n^2)}$

Πολλαπλασιασμός Ακολουθίας Πινάκων

- **Αλγόριθμος**

```
Matrix-Chain()
1. for i←1 to n: C(i,i)=0
2. for s←1 to n-1
       for i←1 to n-s
         j ←i + s
         for k←i to j
           C[i,j]←min{C[i,k]+C[k+1,j]+m_{i-1}·m_k·m_j}
3. return C[1,n]
```

Συνολική Πολυπλοκότητα

$O(n^3)$

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30	×	35
A_2	35	×	15
A_3	15	×	5
A_4	5	×	10
A_5	10	×	20
A_6	20	×	25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

C[1,1] = 0

C[2,2] = 0

...

C[6,6] = 0

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30	×	35
A_2	35	×	15
A_3	15	×	5
A_4	5	×	10
A_5	10	×	20
A_6	20	×	25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

C[1,2] = 30 · 35 · 15 = 15,750

C[2,3] = 35 · 15 · 5 = 2,625

...

C[5,6] = 10 · 20 · 25 = 5,000

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30	×	35
A_2	35	×	15
A_3	15	×	5
A_4	5	×	10
A_5	10	×	20
A_6	20	×	25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30	×	35
A_2	35	×	15
A_3	15	×	5
A_4	5	×	10
A_5	10	×	20
A_6	20	×	25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

$$C[2,5] = \min \begin{cases} C[2,2]+C[3,5]+m_1 \cdot m_2 \cdot m_5 = 13,000 \\ C[2,3]+C[4,5]+m_1 \cdot m_3 \cdot m_5 = 7,125 \\ C[2,4]+C[5,5]+m_1 \cdot m_4 \cdot m_5 = 11,375 \end{cases}$$

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30	×	35
A_2	35	×	15
A_3	15	×	5
A_4	5	×	10
A_5	10	×	20
A_6	20	×	25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

Τελικό Αποτέλεσμα: C[1,6] = 15,125

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30	×	35
A_2	35	×	15
A_3	15	×	5
A_4	5	×	10
A_5	10	×	20
A_6	20	×	25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

Ποια είναι η βέλτιστη λύση της $A_1 \times A_2 \times \ldots \times A_6$?

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30	×	35
A_2	35	×	15
A_3	15	×	5
A_4	5	×	10
A_5	10	×	20
A_6	20	×	25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

$$C[2,5] = \min \begin{cases} C[2,2]+C[3,5]+m_1 \cdot m_2 \cdot m_5 = 13{,}000 \\ C[2,3]+C[4,5]+m_1 \cdot m_3 \cdot m_5 = 7{,}125 \\ C[2,4]+C[5,5]+m_1 \cdot m_4 \cdot m_5 = 11{,}375 \end{cases}$$

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30	×	35
A_2	35	×	15
A_3	15	×	5
A_4	5	×	10
A_5	10	×	20
A_6	20	×	25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	3	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

$$C[2,5] = \min \begin{cases} C[2,2]+C[3,5]+m_1 \cdot m_2 \cdot m_5 = 13{,}000 \\ C[2,3]+C[4,5]+m_1 \cdot m_3 \cdot m_5 = 7{,}125 \\ C[2,4]+C[5,5]+m_1 \cdot m_4 \cdot m_5 = 11{,}375 \end{cases}$$

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30	×	35
A_2	35	×	15
A_3	15	×	5
A_4	5	×	10
A_5	10	×	20
A_6	20	×	25

	1	2	3	4	5	6
1	0	1	1	3	3	3
2		0	2	3	3	3
3			0	3	3	3
4				0	4	5
5					0	5
6						0

C[1,6] = 15,125

$$A_1 \times A_2 \times A_3 \times A_4 \times A_5 \times A_6$$

$$(A_1 \times A_2 \times A_3) \times (A_4 \times A_5 \times A_6)$$

$$(A_1 \times (A_2 \times A_3)) \times ((A_4 \times A_5) \times A_6)$$

4 Ελάχιστες Αξιόπιστες Διαδρομές

Κίνητρο

Σε εφαρμογές συμβαίνει συχνά τα βάρη των ακμών και οι ελάχιστες διαδρομές να **μην** απεικονίζουν **ολόκληρη την αλήθεια** !!!

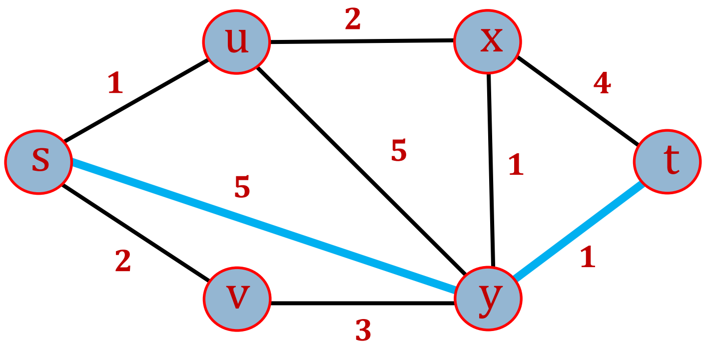
Για παράδειγμα, σε ένα δίκτυο επικοινωνιών, ακόμα και αν τα βάρη των ακμών απεικονίζουν πιστά τις καθυστερήσεις μετάδοσης, μπορεί να υπάρχουν άλλοι παράγοντες που εμπλέκονται σε μια ε.δ !!!

4 Ελάχιστες Αξιόπιστες Διαδρομές

Κίνητρο

Θα μπορούσε, για παράδειγμα, κάθε επιπλέον ακμή στην ε.δ να είναι ένας παράγοντας «αβεβαιότητας» που ελλοχεύει κίνδυνους απώλειας πακέτων!

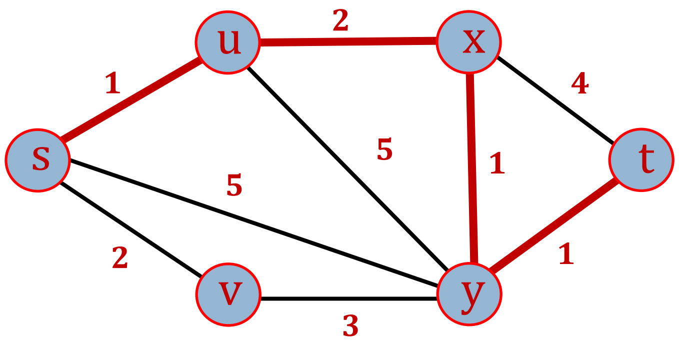
Τότε, θα θέλαμε να αποφύγουμε διαδρομές με πάρα πολλές ακμές!



4 Ελάχιστες Αξιόπιστες Διαδρομές

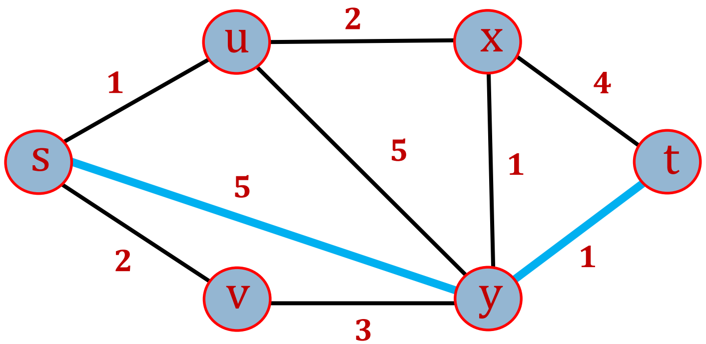
d(s,t)=(s,u,x,y,t)

Κόστος 5
Πλήθος ακμών 4



δ(s,t)=(s,y,t)

Κόστος 6
Πλήθος ακμών 2



4 Ελάχιστες Αξιόπιστες Διαδρομές

Πρόβλημα

Έστω ένα έμβαρο γράφημα G, δύο κόμβοι του **s** και **t**, και ένας ακέραιος ***k***.

Θέλουμε να υπολογίσουμε μια ε.δ στο G

$$\mathbf{s} \rightsquigarrow \mathbf{t}$$

*η οποία να χρησιμοποιεί το πολύ **k** ακμές* !!!

Ελάχιστες Αξιόπιστες Διαδρομές

- Υπάρχει κάποιος αποτελεσματικός τρόπος να προσαρμόσουμε τον αλγόριθμο του Dijkstra στο νέο μας πρόβλημα ?

- Μάλλον **Όχι** !!!

  Ο αλγόριθμος του Dijkstra εστιάζει στο μήκος κάθε ε.δ. χωρίς να «θυμάται» το πλήθος των ακμών στη διαδρομή !

  Ακόμα και να το θυμόταν, δεν θα μπορούσε να επιλέξει την ελάχιστη με το πολύ k ακμές !!!

Ελάχιστες Αξιόπιστες Διαδρομές

- Μήπως ο Δυναμικός Προγραμματισμός μας λύσει το πρόβλημα ?

- Ας ορίσουμε για κάθε κόμβο **v** και κάθε ακέραιο $\mathbf{i \leq k}$

  d(s,v,i) = d(v,i)

  να είναι το κόστος της ελάχιστης διαδρομής $\mathbf{s} \rightsquigarrow \mathbf{v}$ *η οποία να χρησιμοποιεί το πολύ* **i** *ακμές*

  Προφανώς, ισχύει:

  d(s,0) = 0
  d(v,0) = ∞ ∀ v∈V-{s}

Ελάχιστες Αξιόπιστες Διαδρομές

- Πολύ φυσικά, η γενική εξίσωση ενημέρωσης, είναι:

$$\mathbf{d(v,i)} = \min\{\mathbf{d(v,i-1)}, \min_{(u,v)\in E}\{\mathbf{d(u,i-1)} + \mathbf{w(u,v)}\}\}$$

- **Αυτό ήταν… Τελειώσαμε !!!**

  Ο αλγόριθμος πλέον γράφεται πολύ εύκολα,
  σχεδόν μόνος του !!!

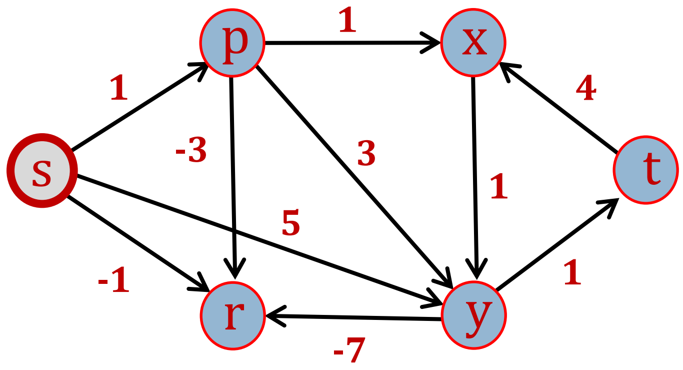
Ελάχιστες Αξιόπιστες Διαδρομές

- **Αλγόριθμος**

```
K-shortest-path(G,w, k)
1. Initialize()  ──→  for i←1 to n
                              C(i,0)=∞
                      C(s,0)=0
2. for i←1 to k
       for each v∈V-{s}
           for each u∈Adj(v)
               d(v,i) ← min{C(v, i-1),C(u, i-1)+w(u,v)}
3. return C(k)
```

Ελάχιστες Αξιόπιστες Διαδρομές

Παράδειγμα

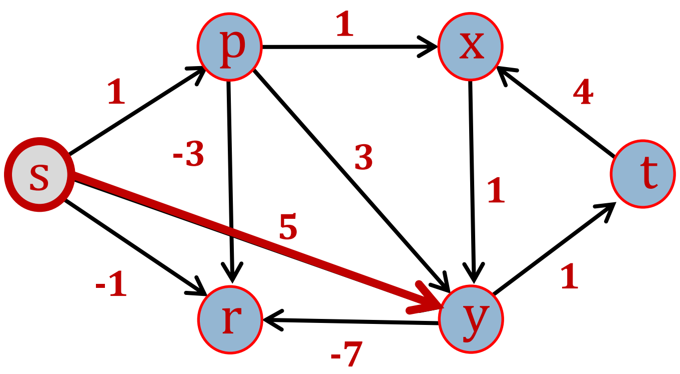


s	p	r	x	y	t
0	∞	∞	∞	∞	∞

d(v,0) =

Ελάχιστες Αξιόπιστες Διαδρομές

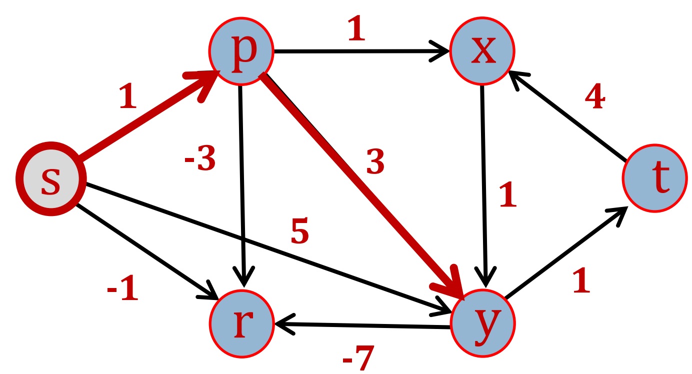
- Παράδειγμα



	s	p	r	x	y	t
d(v,0) =	0	∞	∞	∞	∞	∞
d(v,1) =	0	1	-1	∞	5	∞

Ελάχιστες Αξιόπιστες Διαδρομές

- Παράδειγμα



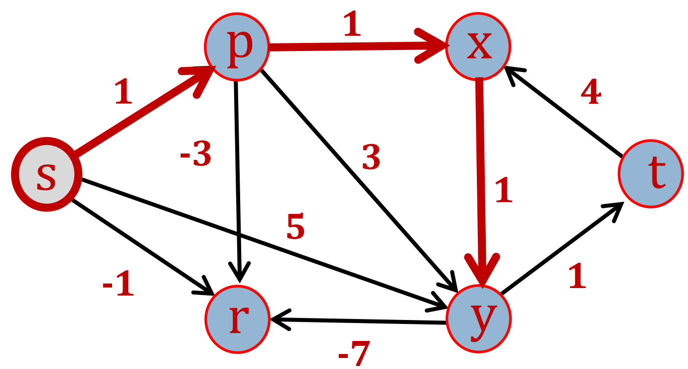
	s	p	r	x	y	t
d(v,0) =	0	∞	∞	∞	∞	∞
d(v,1) =	0	1	-1	∞	5	∞
d(v,2) =	0	1	-2	2	4	6

$$d(y,2) = \min\{d(y,1), \min_{(u,y)\in E}\{d(u,1)+w(u,y)\}\}$$

$$= \min\{5, \min\{0+5,\ 1+3,\ \infty+1\}\} = \mathbf{4}$$

Ελάχιστες Αξιόπιστες Διαδρομές

- Παράδειγμα

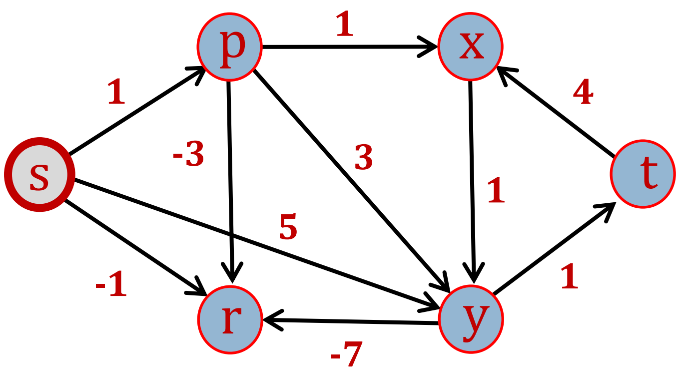


	s	p	r	x	y	t
d(v,0) =	0	∞	∞	∞	∞	∞
d(v,1) =	0	1	-1	∞	5	∞
d(v,2) =	0	1	-2	2	4	6
d(v,3) =	0	1	-3	2	3	6

d(y,3) = min{4, min{0+5, 1+3, 2+1}} = **3**

Ελάχιστες Αξιόπιστες Διαδρομές

- Παράδειγμα



	s	p	r	x	y	t
d(s,v,0)	0	∞	∞	∞	∞	∞
d(s,v,1)	0	1	-1	∞	5	∞
d(s,v,2)	0	1	-2	2	4	6
d(s,v,3)	0	1	-3	2	3	6
d(s,v,4)	0	1	-4	2	3	4

5 Πανζευκτικές Ελάχιστες Διαδρομές

Κίνητρο

Σε πολλές εφαρμογές, χρειαζόμαστε να έχουμε όχι μόνο την ε.δ μεταξύ **s** $\rightsquigarrow$ **t** ενός G τάξης n, αλλά μεταξύ κάθε ζεύγους κόμβων του!!!

Μπορούμε να λύσουμε το πρόβλημα ?

Φυσικά ΝΑΙ !!!
Εφαρμόζοντας n φορές τον αλγόριθμο των Bellman-Ford (επιτρέπονται αρνητικά βάρη)

5 Πανζευκτικές Ελάχιστες Διαδρομές

Σε πόσο χρόνο ?

Ο Bellman-Ford εκτελείται σε $O(nm)$ χρόνο, και επομένως:

$$\mathbf{O(n^2m)}$$

Μπορούμε καλύτερα ?

ΝΑΙ !!! με τον αλγόριθμο των **Floyd-Watshall**

ο οποίος βασίζεται σε ΔΠ, σε χρόνο: $\mathbf{O(n^3)}$

5 Πανζευκτικές Ελάχιστες Διαδρομές

Πρόβλημα

Έστω ένα έμβαρο γράφημα G (με αρνητικά ακμικά βάρη) τάξης n και μεγέθους m.

Θέλουμε να υπολογίσουμε τις ε.δ στο G

$$\mathbf{x} \rightsquigarrow \mathbf{y}$$

για κάθε ζεύγος κόμβων (x, y), x, y $\in$ *V(G)* !!!

Πανζευκτικές Ελάχιστες Διαδρομές

- Το να λύσουμε το πρόβλημα υπολογίζοντας όλο και περισσότερα ζεύγη κόμβων δε βοηθά, καθώς οδηγεί πίσω στον αλγόριθμο **$O(n^2m)$** !!!

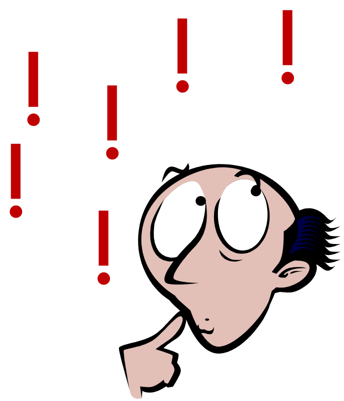
- **Κρίσιμο Ερώτημα ΔΠ**

  Υπάρχει κάποιο καλό υποπρόβλημα για τον υπολογισμό των αποστάσεων μεταξύ κάθε ζεύγους κόμβων ?

Πρέπει

να επινοήσουμε κάτι έξυπνο !

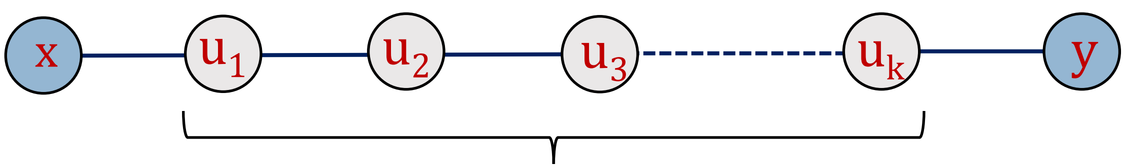
να βρούμε μια νέα ιδέα !



Πανζευκτικές Ελάχιστες Διαδρομές

- **Να μια ιδέα !!!**

  Ας πάρουμε μια ε.δ $x \rightsquigarrow y$



Ενδιάμεσοι Κόμβοι

- Υποθέστε ότι δεν επιτρέπουμε ενδιάμεσους κόμβους !!!

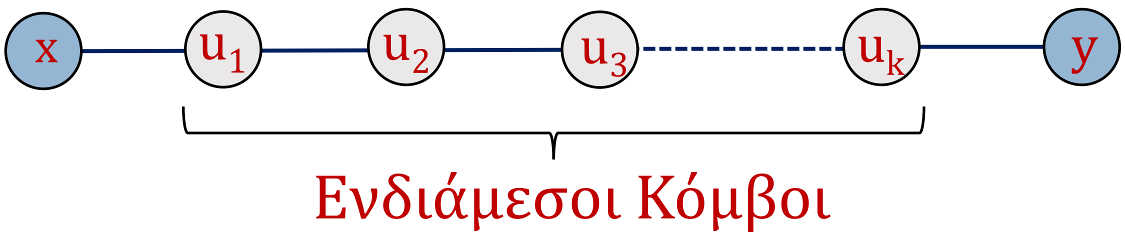
  Τότε, μπορούμε να λύσουμε το πρόβλημα των π.ε.δ, καθώς για κάθε ζεύγος κόμβων x, y του G:

  η ε.δ $x \rightsquigarrow y$ είναι η ακμή (x, y), εάν υπάρχει !!!

Πανζευκτικές Ελάχιστες Διαδρομές

- **Να μια ιδέα !!!**

  Ας πάρουμε μια ε.δ $x \rightsquigarrow y$



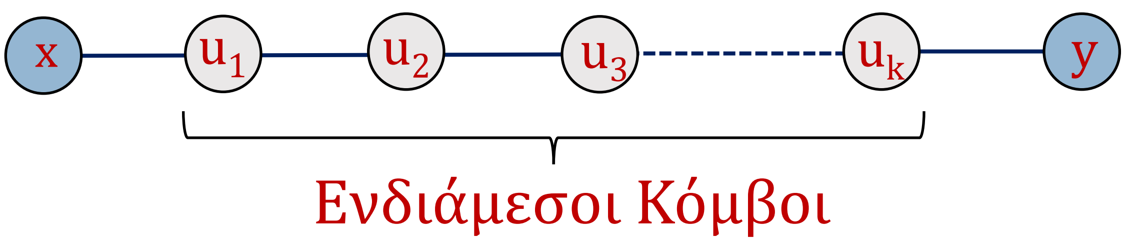
- Αν διευρύνουμε σταδιακά το **σύνολο των ενδιάμεσων κόμβων**, **έναν κάθε φορά**, **ενημερώνοντας τα μήκη των ε.δ** σε κάθε διεύρυνση,

  τι μπορούμε να πετύχουμε !!!

Πανζευκτικές Ελάχιστες Διαδρομές

- **Να μια ιδέα !!!**

  Ας πάρουμε μια ε.δ $x \rightsquigarrow y$



- Μπορούμε να διευρύνουμε το σύνολο των ενδιάμεσων κόμβων **έως να γίνει όλο το σύνολο κόμβων V !!!**

Τότε όμως έχουμε λύσει το πρόβλημα των π.ε.δ !!!

Πανζευκτικές Ελάχιστες Διαδρομές

- **Τυποποίηση της ιδέας !!!**
  - Αριθμούμε τους κόμβους του **V** ως **{1,2, …,n}**
  - d(i,j,k) συμβολίζουμε το μήκος της ε.δ i ⇝ j

    με ενδιάμεσους κόμβους **ΜΟΝΟ** από το σύνολο **{1,2, …,k}**
  - Επομένως,

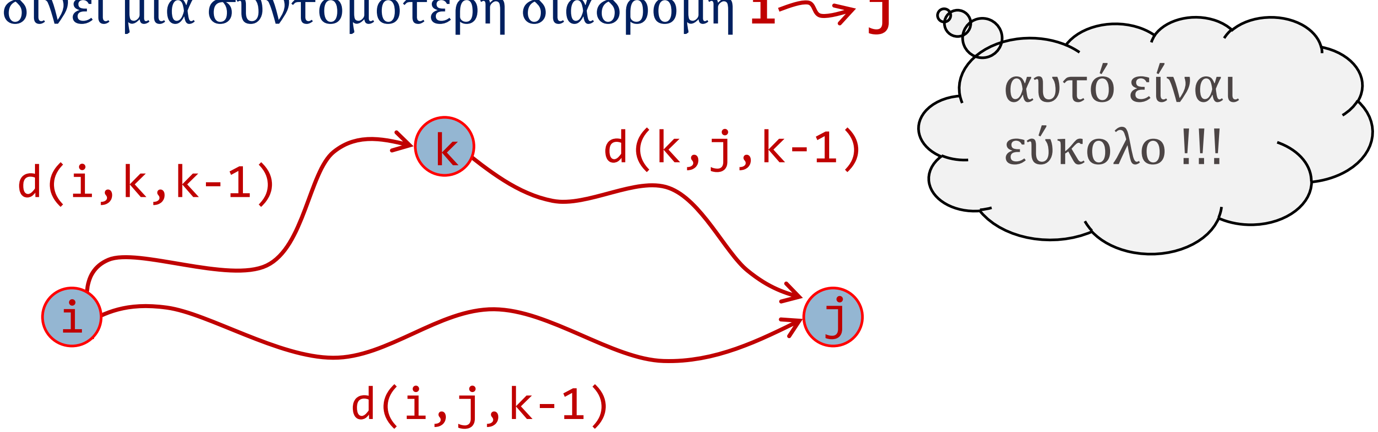
    d(i,j,0) = w(i,j) εάν υπάρχει η ακμή (i,j)

Πανζευκτικές Ελάχιστες Διαδρομές

- Τι πρέπει να ελέγξουμε όταν επεκτείνουμε το ενδιάμεσο σύνολο κατά ένα κόμβο ?

  `{1,2, …,k-1}` → `{1,2, …,k}`

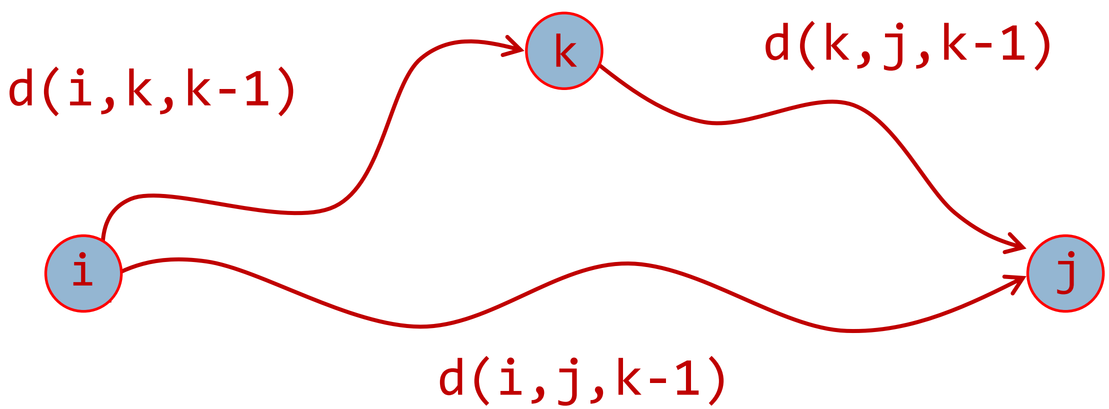
  - Πρέπει να **επανεξετάσουμε** όλα τα ζεύγη κόμβων **i, j** και να **ελέγξουμε** εάν η χρήση του **k** ως ενδιάμεσου κόμβου μας δίνει μια συντομότερη διαδρομή **i⇝j**



Πανζευκτικές Ελάχιστες Διαδρομές

- Μια ε.δ i $\rightsquigarrow$ j η οποία χρησιμοποιεί τον k μαζί με άλλους κόμβους χαμηλότερης αρίθμησης {1,2, …,k-1}, περνάει από τον k **μία μόνο φορά** !!!

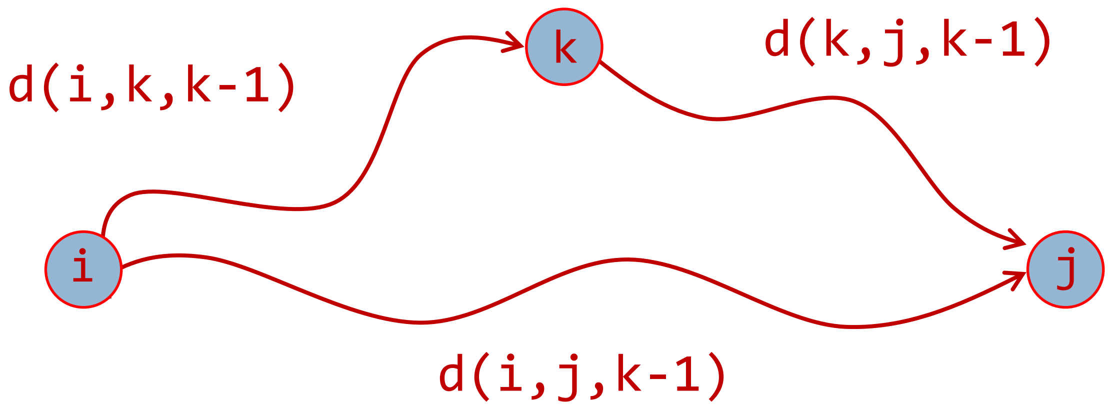
γιατί δεν υπάρχουν αρνητικοί κύκλοι



Πανζευκτικές Ελάχιστες Διαδρομές

- Έτσι, η χρήση του **k** ως ενδιάμεσου κόμβου μας δίνει μια συντομότερη διαδρομή **i** ⇝ **j** εάν και μόνο εάν:

```
d(i,k,k-1) + d(k,j,k-1) < d(i,j,k-1)
```



οπότε η τιμή:

`d(i,j,k)` θα πρέπει να ενημερωθεί ανάλογα !!!

Πανζευκτικές Ελάχιστες Διαδρομές

- **Σχέση Επίλυσης Υποπροβλημάτων**
  - `d(i,j,0) = w(i,j)` εάν υπάρχει η ακμή `(i,j)`
  - Αναδρομική σχέση:

$$
d(i,j,k) = \begin{cases} w(i,j) & \text{if } k=0 \\ \min\{d(i,j,k-1), \\ \quad d(i,k,k-1) + d(k,j,k-1)\} & \text{if } k \geq 1 \end{cases}
$$

- **Αλγόριθμος Floyd-Warshall**

```
All-pair-Shortest-Paths(G,w)
1. Initialize(G)
2. for k←1 to n
       for i←1 to n
           for j←1 to k
               d(i,j,k) ← min{d(i,j,k-1),
                              d(i,k,k-1) + d(k,j,k-1)}
3. return d(i,j,n)
```

```
for i←1 to n
    for j←1 to n
        C(i,i)=0
for all (i,j)∈E
    d(i,j,0)=w(i,j)
```

Πανζευκτικές Ελάχιστες Διαδρομές

- **Πολυπλοκότητα**

Το πιο χρονοβόρο βήμα:

```
2. for k ← 1 to n
     for i ← 1 to n
       for j ← 1 to k
         d(i,j,k) ← min {d(i,j,k-1),
                         d(i,k,k-1) + d(k,j,k-1)}
```

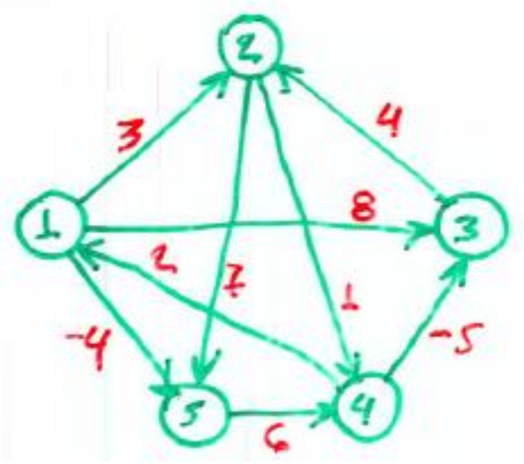
Συνολική
Πολυπλοκότητα Αλγόριθμου: **$O(n^3)$**

Πανζευκτικές Ελάχιστες Διαδρομές

Παράδειγμα

d(i,j,0)

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0

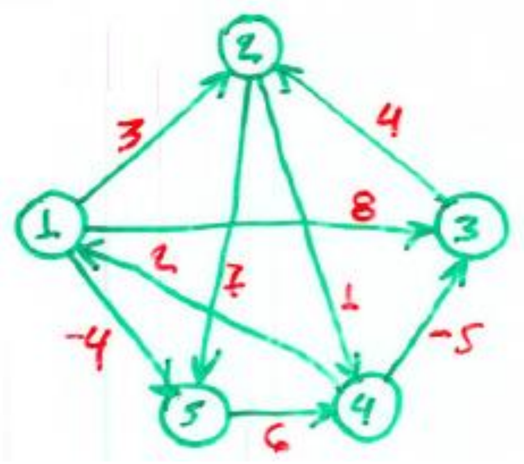


Πανζευκτικές Ελάχιστες Διαδρομές

Παράδειγμα

d(i,j,1)

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



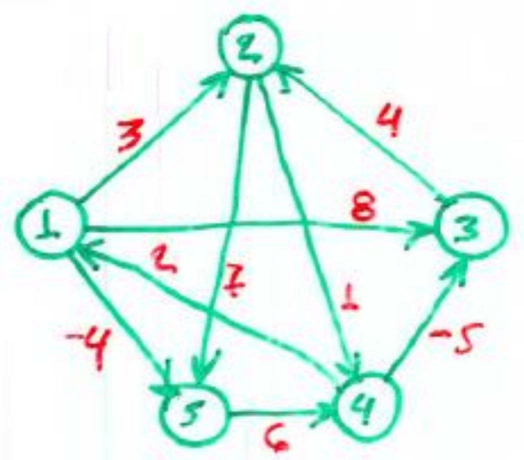
$$d(4,2,1) = \min\{d(4,2,0), d(4,1,0)+d(1,2,0)\}$$
$$= \min\{\infty, 2+3\} = \mathbf{5}$$

Πανζευκτικές Ελάχιστες Διαδρομές

Παράδειγμα

d(i,j,1)

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



d(4,2,2)

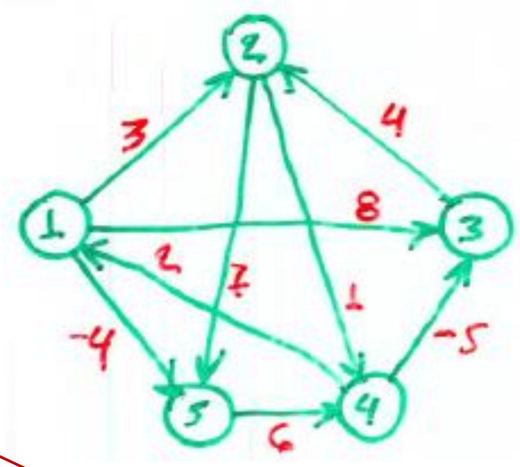
$$d(4,2,2) = \min\{d(4,2,1), d(4,2,1) + d(2,2,1)\}$$
$$= \min\{5, 5+0\} = \mathbf{5}$$

Πανζευκτικές Ελάχιστες Διαδρομές

Παράδειγμα

d(i,j,1)

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



d(3,4,2)

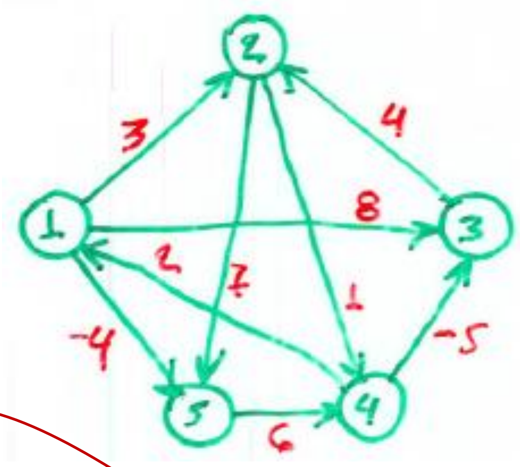
$$d(3,4,2) = \min\{d(3,4,1), d(3,2,1)+d(2,4,1)\}$$
$$= \min\{\infty, 4+1\} = \mathbf{5}$$

Πανζευκτικές Ελάχιστες Διαδρομές

Παράδειγμα

d(i,j,1)

$$
\begin{array}{c|ccccc|}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 0 & 3 & 8 & \infty & -4 \\
2 & \infty & 0 & \infty & 1 & 7 \\
3 & \infty & 4 & 0 & \infty & \infty \\
4 & 2 & 5 & -5 & 0 & -2 \\
5 & \infty & \infty & \infty & 6 & 0 \\
\hline
\end{array}
$$



d(3,5,2)

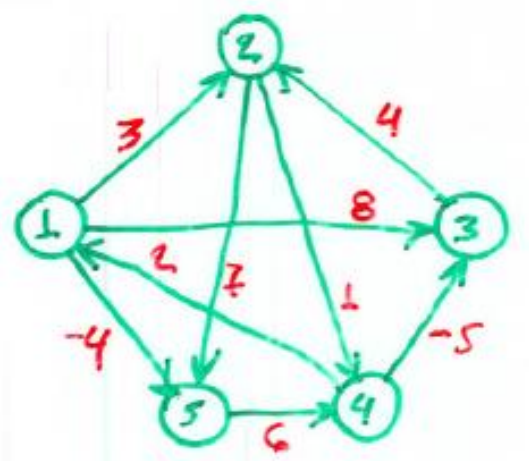
$$d(3,5,2) = \min\{d(3,5,1), d(3,2,1) + d(2,5,1)\}$$
$$= \min\{\infty, 4+7\} = \mathbf{11}$$

Πανζευκτικές Ελάχιστες Διαδρομές

Παράδειγμα

d(i,j,5)

	1	2	3	4	5
1	0	3	-3	2	-4
2	3	0	-4	1	-1
3	6	4	0	5	3
4	2	-1	-5	0	-2
5	7	5	1	6	0



6 Σακίδιο

Κατά την διάρκεια μιας κλοπής, ένας διαρρήκτης βρίσκει περισσότερα λάφυρα απ΄ όσα περίμενε και πρέπει να αποφασίσει τι θα πάρει !!!

Το «σακίδιό» του (knapsack) μπορεί να μεταφέρει **W** κιλά !!!

Υπάρχουν **n** είδη

με βάρη $w_1, w_2, \ldots, w_n$

και αξία $v_1, v_2, \ldots, v_n$

6 Σακίδιο

Το ερώτημα που τον απασχολεί !!!
Ποιος είναι ο πλέον πολύτιμος συνδυασμός ειδών που μπορεί να βάλει στο σακίδιό του ?

Το «σακίδιό» του (knap-sack) μπορεί να μεταφέρει **W** κιλά !!!

Υπάρχουν **n** είδη

με βάρη $w_1, \; w_2, \; \ldots, \; w_n$
και αξία $v_1, \; v_2, \; \ldots, \; v_n$

6 Σακίδιο

Παράδειγμα

Έστω ότι το σακίδιο χωράει **W = 10** κιλά, και

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

6 Σακίδιο

Παράδειγμα

Έστω **W=10** κιλά, και

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Δύο Εκδοχές

Υπάρχουν απεριόριστες ποσότητες από κάθε είδος

Υπάρχει μία μόνο ποσότητα από κάθε είδος

6 Σακίδιο

Παράδειγμα

Έστω **W=10** κιλά, και

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Δύο Εκδοχές

Υπάρχουν απεριόριστες ποσότητες από κάθε είδος

Βέλτιστη Επιλογή

1 από είδος Α w=6 30€
2 από είδος Δ w=4 18€

Συνολική Τιμή **48€**

6 Σακίδιο

Παράδειγμα

Έστω **W=10** κιλά, και

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Δύο Εκδοχές

1 από είδος Α 30€
1 από είδος Γ 16€ **46€**

Βέλτιστη Επιλογή

Υπάρχει μία μόνο ποσότητα από κάθε είδος

6 Σακίδιο (Knapsack)

Πρόβλημα

Μας δίδεται ένα σύνολο n αντικειμένων

$$S = \{a_1, a_2, \ldots, a_n\}$$

με ακέραια βάρη w_i και αξίες v_i $(1 \le i \le n)$, και μία τιμή W,

και μας ζητείται να υπολογίσουμε ένα υποσύνολο αντικειμένων $S' \subseteq S$ με μέγιστη συνολική αξία C και με συνολικό βάρος $\le \mathbf{W}$, δηλαδή,

$$\max C = \sum_{a_i \in S'} v_i \quad \text{και} \quad \sum_{a_i \in S'} w_i \le W$$

Σακίδιο : Με επανάληψη

- Ας δούμε την εκδοχή που επιτρέπει επανάληψη !

  Και, όπως πάντα, το κρίσιμο ερώτημα στο ΔΠ είναι:

  Ποια είναι τα υποπροβλήματα ?

- Μπορούμε να **«συρρικνώσουμε»** το αρχικό πρόβλημα με δύο τρόπους:

(1) Μπορούμε να εξετάσουμε την περίπτωση σακιδίων μικρότερης χωρητικότητας $\mathbf{w \leq W}$

ή

(2) Την περίπτωση λιγότερων ειδών $\mathbf{1, 2, \ldots, i \leq n}$

Σακίδιο : Με επανάληψη

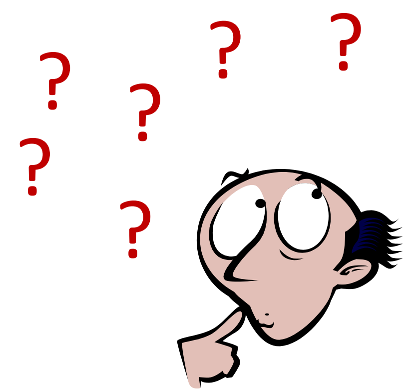
- Ο πρώτος περιορισμός επιβάλει να έχουμε μικρότερες χωρητικότητες !

  Ορίζουμε:

  `C(w)` = **Max Τιμή που παίρνουμε με σακίδιο χωρητικότητας `w`**

- **Ερώτημα !!!**

  Μπορούμε να εκφράσουμε το υποπρόβλημα **C(w)** συναρτήσει μικρότερων υποπροβλημάτων ?



Σακίδιο : Με επανάληψη

- Έστω μια βέλτιστη λύση του υποπροβλήματος **C(w),** που περιλαμβάνει :
  - ✓ τα είδη $p, \ldots, i, \ldots, q$
  - ✓ τέτοια ώστε $w_p + \ldots + w_i + \ldots + w_q \leq w$
  - ✓ με max τιμή $C(w) = v_p + \ldots + v_i + \ldots + v_q$
- Εάν η βέλτιστη λύση του **C(w)** περιλαμβάνει το είδος **i** βάρους $\mathbf{w_i}$, τότε η αφαίρεση του **i** από το σακίδιο δίδει μια βέλτιστη λύση για το υποπρόβλημα **C(w-$\mathbf{w_i}$) !!!**

Με άλλα λόγια:

> Το υποπρ/μα **C(w)** είναι απλώς το **C(w- $\mathbf{w_i}$) + $\mathbf{v_i}$**, για κάποιο **i**

Σακίδιο : Με επανάληψη

- Με άλλα λόγια:

> Το υποπρ/μα **C(w)** είναι απλώς το **C(w- w$_i$) + v$_i$**, για κάποιο **i**.

Ποιο είναι αυτό το **i** ? ... δεν γνωρίζουμε !!!

Τότε**,** δεν έχουμε παρά να πάρουμε όλες τις δυνατότητες!

- Ορίστε πως !!!

$$C(w) = \max_{i\,:\,w_i \le w} \{C(w-w_i) + v_i\}$$

όπου, ως συνήθως, η σύμβασή μας είναι ότι η μέγιστη τιμή του κενού συνόλου είναι 0 !!!

Σακίδιο : Με επανάληψη

- Τελειώσαμε ! … ο αλγόριθμος τώρα γράφεται μόνος του και είναι ιδιαιτέρα κομψός !!!

Αλγόριθμος

```
Knapsack1(w[1..n], v[1..n], W)
1. C(0)=0
2. for w ← 1 to W
        C(w) ← max {C(w-w(i)) + v(i) : w(i) ≤ w}
                i
3. return C(W)
```

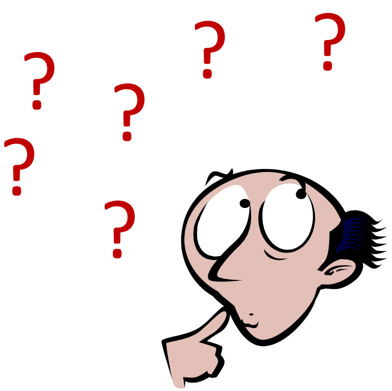
Συνολική Πολυπλοκότητα **O(nW)**

Σακίδιο : Με επανάληψη

- **Ας μείνουμε λίγο ακόμα** σε αυτή την εκδοχή του προβλήματος !
  - ✓ Γνωρίζουμε ότι ένα πρόβλημα ΔΠ εκφράζεται με ένα γράφημα **G** τύπου DAG !!!

Εύλογο ερώτημα !!!

Ποιο είναι το **DAG** του προβλήματος **του σακιδίου με επανάληψη ?**



Σακίδιο : Με επανάληψη

- **Το DAG του προβλήματος**

Χωρητικότητα **W = 10**

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

0 1 2 3 4 5 6 7 8 9 10

Κόμβοι DAG ⇔ Χωρητικότητα Σακιδίου **W**

Σακίδιο : Με επανάληψη

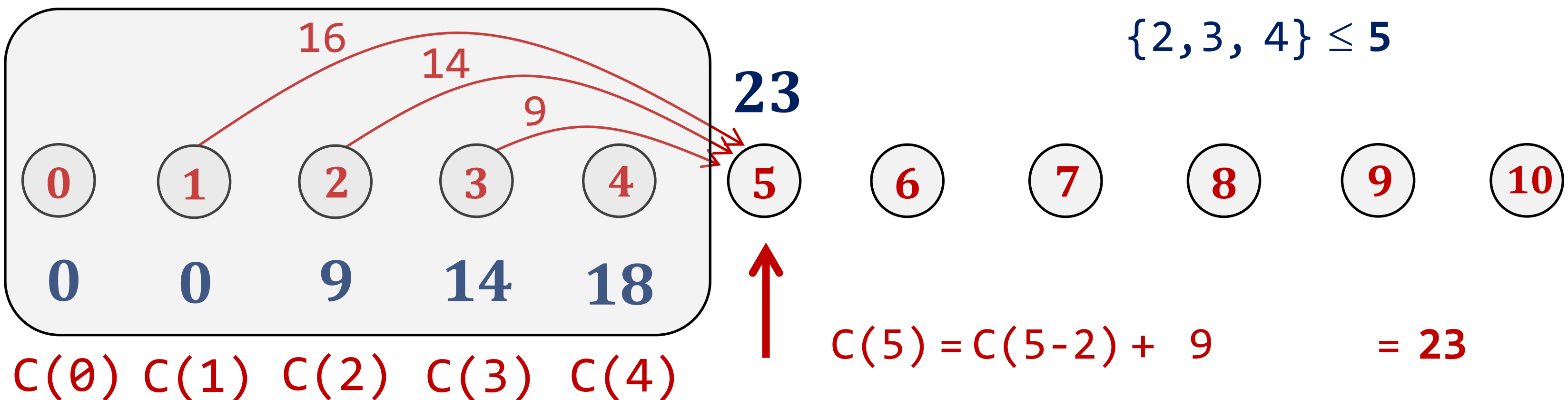
Το DAG του προβλήματος

Χωρητικότητα **W = 10**

$$C(w) = \max\{C(w-w(i))+v(i) : w(i) \leq w\}$$

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

$\{2,3,4\} \leq 5$



$C(5) = C(5-2) + 9 = 23$

$C(5) = C(5-3) + 14 = 23$

$C(5) = C(5-4) + 16 = 16$

Σακίδιο : Με επανάληψη

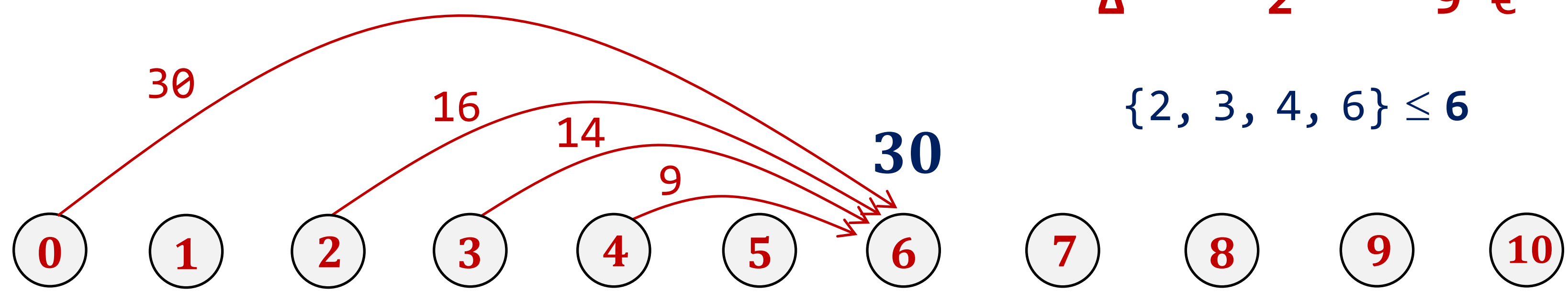
Το DAG του προβλήματος

Χωρητικότητα **W = 10**

$$C(w) = \max\{C(w-w(i)) + v(i) : w(i) \le w\}$$

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

$\{2, 3, 4, 6\} \le \mathbf{6}$



0	**0**	**9**	**14**	**18**	**23**
C(0)	C(1)	C(2)	C(3)	C(4)	C(5)

C(6) = C(6-2) + 9 = **27**

C(6) = C(6-3) + 14 = **28**

C(6) = C(6-4) + 16 = **25**

C(6) = C(6-6) + 30 = **30**

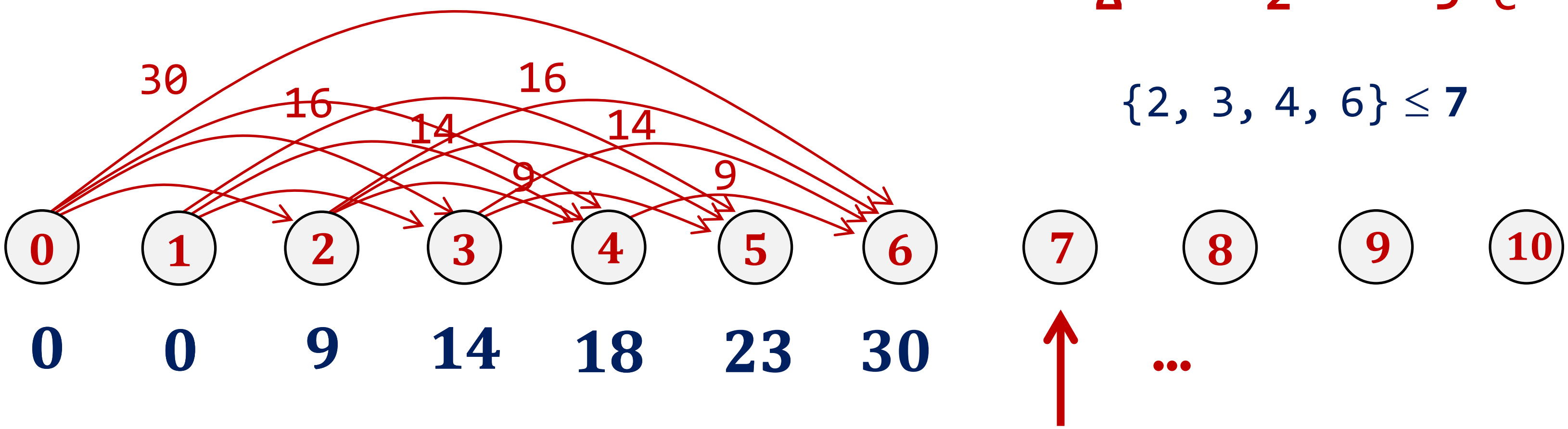
Το DAG του προβλήματος

Είδος	Βάρος	Τιμή
A	6	30 €
B	3	14 €
Γ	4	16 €
Δ	2	9 €

Χωρητικότητα **W = 10**

$C(w) = \max\{C(w - w(i)) + v(i) : w(i) \le w\}$

$\{2, 3, 4, 6\} \le \mathbf{7}$



Σακίδιο με επανάληψη ⇔ Max διαδρομή σε DAG

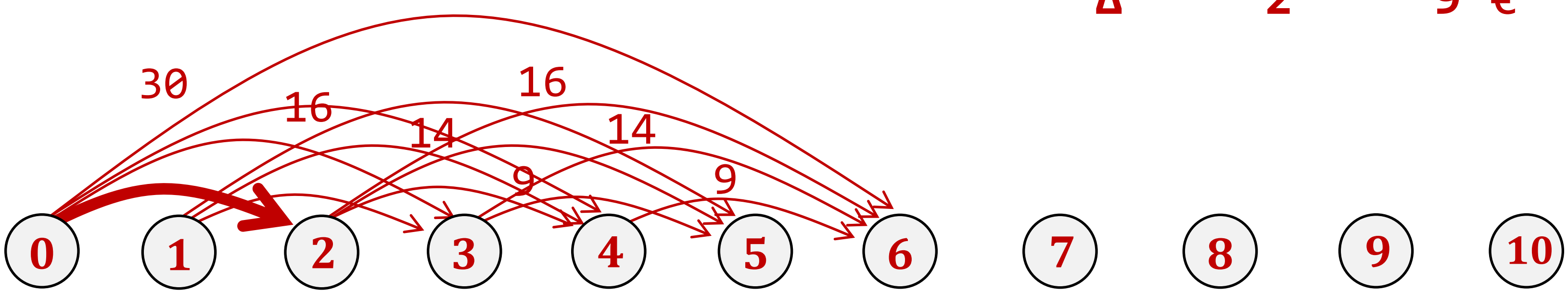
Σακίδιο : Με επανάληψη

Το DAG του προβλήματος

Χωρητικότητα **W = 10**

{Δ} λύση για w = 2 με C(w) = 9

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €



0 0 9 14 18 23 30 ↑ ...

Σακίδιο με επανάληψη ⇔ Max διαδρομή σε DAG

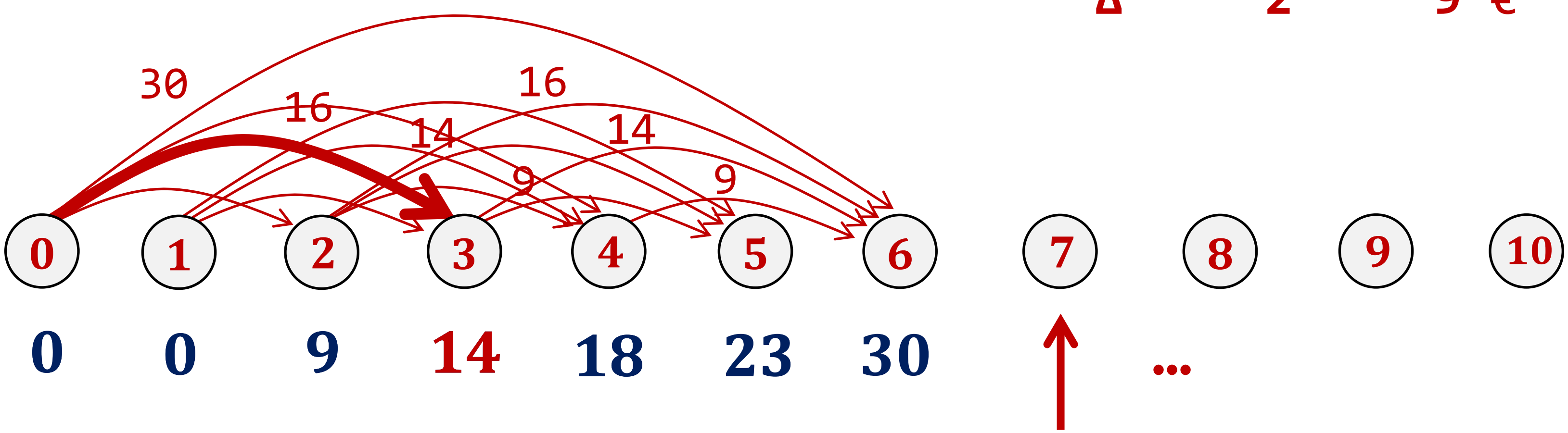
Σακίδιο : Με επανάληψη

Το DAG του προβλήματος

Χωρητικότητα **W = 10**

{B} λύση για $w = 3$ με $C(w) = 14$

Είδος	Βάρος	Τιμή
A	6	30 €
B	3	14 €
Γ	4	16 €
Δ	2	9 €



Σακίδιο με επανάληψη ⇔ Max διαδρομή σε DAG

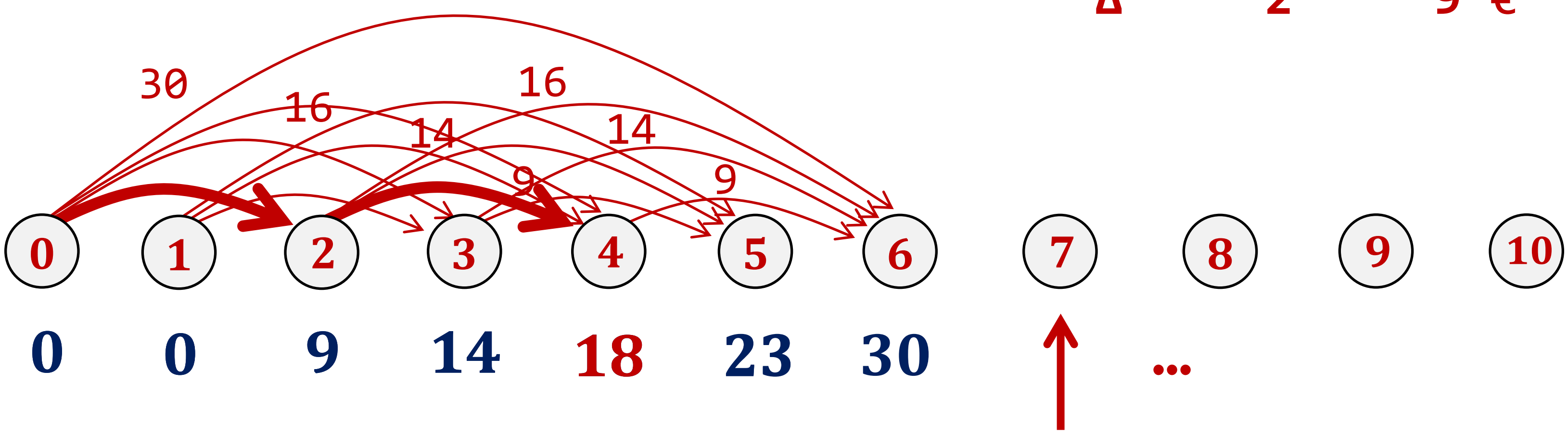
Σακίδιο : Με επανάληψη

- **Το DAG του προβλήματος**

Χωρητικότητα **W = 10**

{Δ,Δ} λύση για w = 4 με C(w) = 18

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €



Σακίδιο με επανάληψη ⇔ Max διαδρομή σε DAG

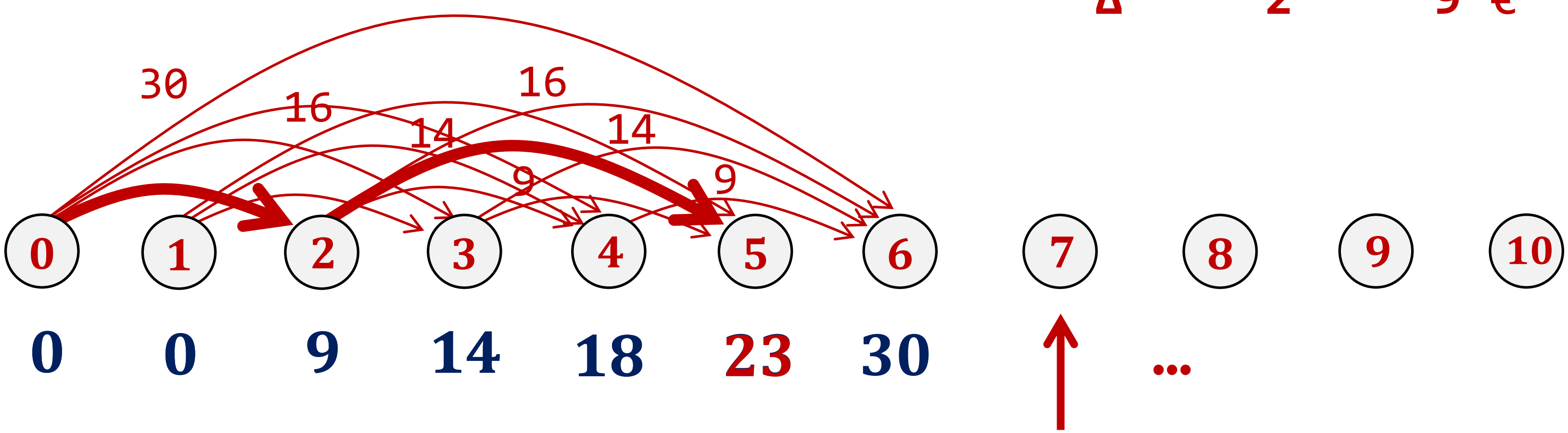
Σακίδιο : Με επανάληψη

Το DAG του προβλήματος

Χωρητικότητα **W = 10**

{Δ, Β} λύση για w = 5 με C(w) = 23

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €



Σακίδιο με επανάληψη ⇔ Max διαδρομή σε DAG

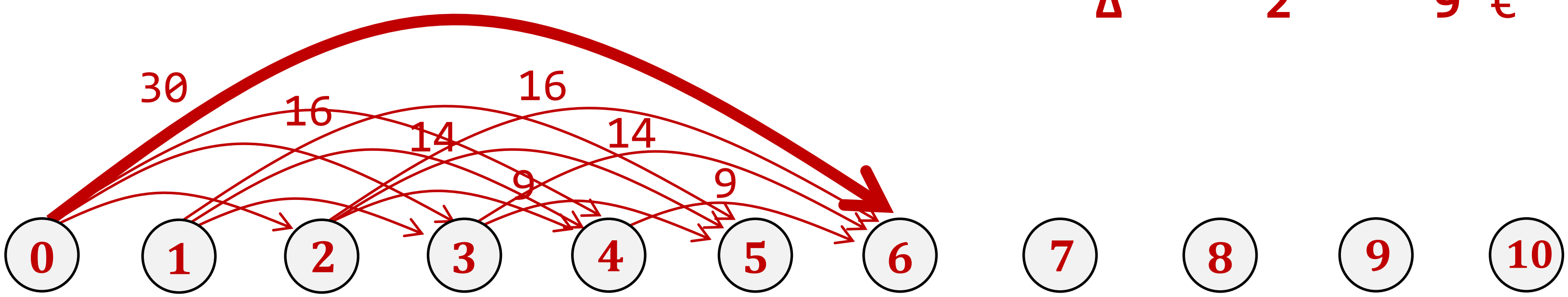
Σακίδιο : Με επανάληψη

- **Το DAG του προβλήματος**

Χωρητικότητα **W = 10**

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

`{A}` λύση για `w = 6` με `C(w) = 30`



0 0 9 14 18 23 30 ...

Σακίδιο με επανάληψη ⇔ Max διαδρομή σε DAG

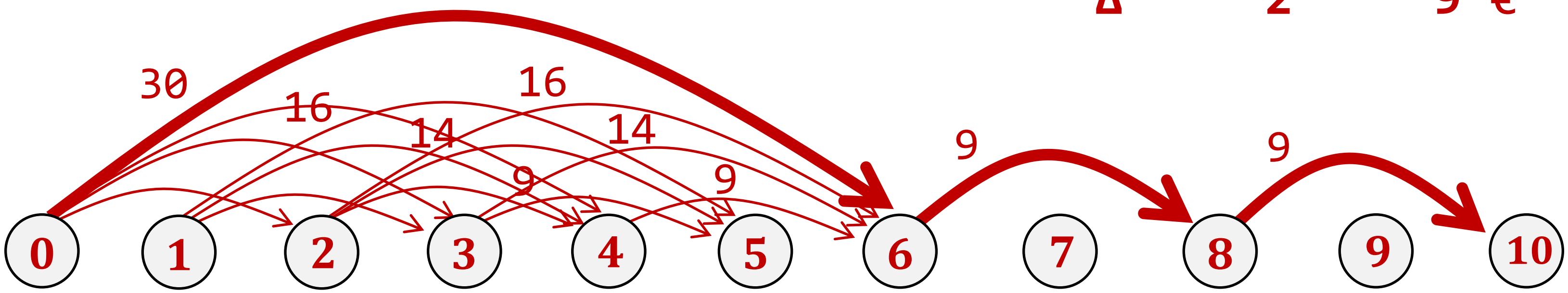
Σακίδιο : Με επανάληψη

- **Το DAG του προβλήματος**

Χωρητικότητα **W = 10**

{A,Δ,Δ} τελική λύση, C(w) = 48

Είδος	Βάρος	Τιμή
A	6	30 €
B	3	14 €
Γ	4	16 €
Δ	2	9 €



0 0 9 14 18 23 30 … 48

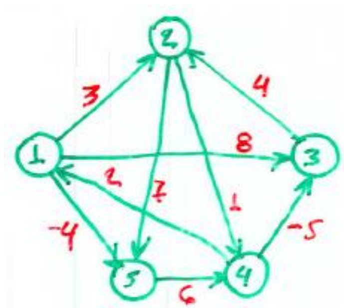
Σακίδιο με επανάληψη ⇔ Max διαδρομή σε DAG

Ποιες Αλγοριθμικές Τεχνικές έχουμε δει έως τώρα ?

- Αλγοριθμικές Τεχνικές, όπως:
  - ✓ Διαίρει-και-Βασίλευε
  - ✓ Άπληστη Επιλογή
- Πρόσφατα, προσθέσαμε και την τεχνική
  - ✓ **Δυναμικός Προγραμματισμός**

S	U	N	N	-	Y
S	–	N	O	W	Y

A	50	×	20
B	20	×	1
C	1	×	10
D	10	×	100



Αλγοριθμικές Τεχνικές: Επισκόπηση

- **Divide-and-Conquer** (διαίρει-και-κυρίευε):

  διαίρεση σε ανεξάρτητα υποπροβλήματα, αναδρομικές σχέσεις, υλοποίηση με αναδρομή κυρίως (top-down), σπανίως επαναληπτικά (δύσκολη υλοποίηση)

- **Dynamic Programming** (δυναμικός προγραμματισμός):

  αρχή βελτιστότητας υπολύσεων, διαίρεση σε επικαλυπτόμενα υποπροβλήματα (DAG / πίνακας), αναδρομικές σχέσεις, υλοποίηση επαναληπτικά κυρίως (bottom up), κάποιες φορές και με αναδρομή (με προσοχή: χρήση *memoization* !)

- **Greedy** (άπληστη μέθοδος):

  άπληστη αμετάκλητη επιλογή, χτίσιμο λύσης βήμα-βήμα, βέλτιστη επίλυση σταδιακά αυξανόμενων υποπροβλημάτων, υλοποίηση επαναληπτικά κυρίως