ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΪΚΑ
ΜΑΘΗΜΑΤΑ

# ΑΛΓΟΡΙΘΜΟΙ

**Ενότητα 8**: Δυναμικός Προγραμματισμός

Ιωάννης Μανωλόπουλος, Καθηγητής
Αναστάσιος Γούναρης, Επίκουρος Καθηγητής
Τμήμα Πληροφορικής ΑΠΘ

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

# Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

# Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΪΚΑ
ΜΑΘΗΜΑΤΑ

# Δυναμικός Προγραμματισμός

## Διωνυμικοί συντελεστές, Αλγόριθμος Warshall, Αλγόριθμος Floyd, Βέλτιστα δυαδικά δέντρα αναζήτησης

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

# Δυναμικός προγραμματισμός

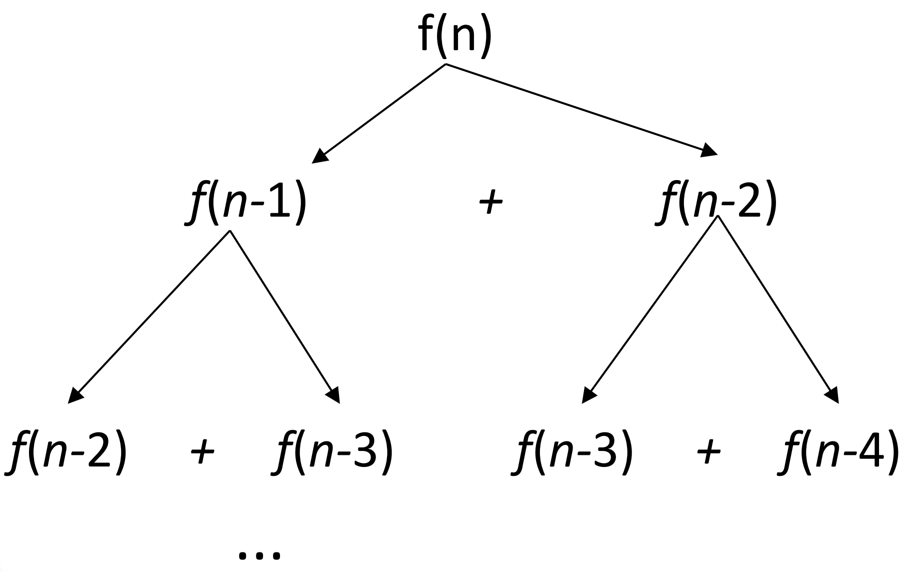
- *Ο Δυναμικός Προγραμματισμός* προτάθηκε από τον Αμερικανό μαθηματικό Richard Bellman περί το 1950 για την επίλυση προβλημάτων βελτιστοποίησης
- «Προγραμματισμός» εδώ σημαίνει «σχεδιασμός»
- Η κύρια ιδέα είναι:
  - να επιλύσουμε πολλά μικρότερα επικαλυπτόμενα προβλήματα
  - να καταγράψουμε τις λύσεις σε ένα πίνακα ώστε το κάθε υποπρόβλημα να επιλύεται μόνο μια φορά
  - να λάβουμε τη λύση στην τελική κατάσταση του πίνακα

# Παράδειγμα: αριθμοί Fibonacci

Ορισμός αριθμών Fibonacci:

- $f(0) = 0$
- $f(1) = 1$
- $f(n) = f(n\text{-}1) + f(n\text{-}2)$

Αναδρομικός υπολογισμός του *n*-οστού αριθμού Fibonacci (top-down)



# Παράδειγμα: αριθμοί Fibonacci (2)

Επαναληπτικός υπολογισμός του n-οστού αριθμού Fibonacci (bottom-up):

$f(0) = 0$

$f(1) = 1$

$f(2) = 0+1 = 1$

$f(3) = 1+1 = 2$

$f(4) = 1+2 = 3$

$f(n\text{-}2) =$

$f(n\text{-}1) =$

$f(n) = f(n\text{-}1) + f(n\text{-}2)$

# Παραδείγματα αλγορίθμων Δυν. Προγραμματισμού

- Υπολογισμός διωνυμικών συντελεστών
- Αλυσιδωτός πολλαπλασιασμός  πινάκων
- Κατασκευή δυαδικών δένδρων αναζήτησης
- Αλγόριθμος Warshall για μεταβατική κλειστότητα
- Αλγόριθμος Floyd για συντομότερα μονοπάτια
- Δύσκολα διακριτά προβλήματα βελτιστοποίησης:
    - εκδοχή προβλήματος περιοδεύοντας πωλητή
    - εκδοχή προβλήματος σάκου

# Διωνυμικοί συντελεστές

Αλγόριθμος βασισμένος στην ταυτότητα

$(a + b)^n = C(n,0)a^n b^0 + . . . + C(n,k)a^{n-k}b^k + . . . + C(n,n)a^0 b^n$

Αναδρομή: $C(n,k) = C(n-1,k) + C(n-1,k-1)$, $n > k > 0$

$C(n,0) = 1$, $C(n,n) = 1$ for $n \geq 0$

| | 0 | 1 | 2 | … | k-1 | k |
|---|---|---|---|---|---|---|
| 0 | 1 | | | | | |
| 1 | 1 | 1 | | | | |
| . | | | | | | |
| . | | | | | | |
| . | | | | | | |
| n-1 | | | | | C(n-1,k-1) | C(n-1,k) |
| n | | | | | | C(n,k) |

# Διωνυμικοί συντελεστές - ψευδοκώδικας

**<u>ALGORITHM Binomial(n,k)</u>**

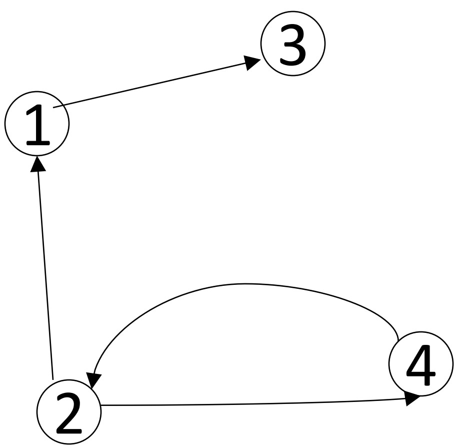
```
// Computes C(n,k) by the dynamic programming
algorithm
// Input: A pair of nonnegative integers n >= k
>= 0
// Output: The value of C(n,k)
   for i ← 0 to n do
      for j ← 0 to min(i,k) do
         if j = 0 or j = i
            C[i,j] ← 1
         else C[i,j] ← C[i-1, j-1] + C[i-1,j]
return C[n,k]
```

Χρονική αποδοτικότητα: Θ(*nk*)

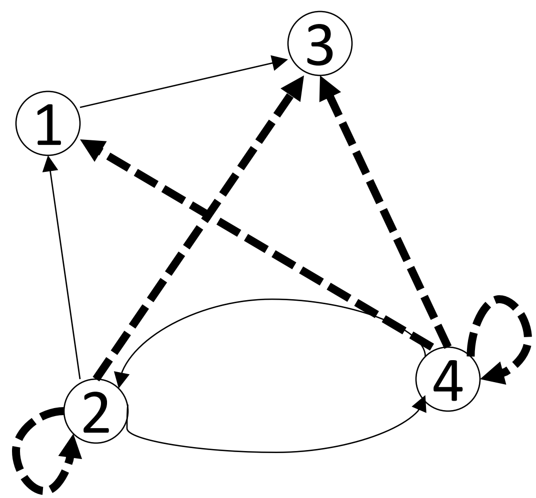
Χωρική αποδοτικότητα: Θ(*nk*)

# Αλγόριθμος Warshall

- Υπολογίζει τη μεταβατική κλειστότητα μιας σχέσης, δηλ. όλα τα μονοπάτια ενός κατευθυνόμενου γράφου
- Παράδειγμα:




$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$




$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & \underline{\mathbf{1}} & \underline{\mathbf{1}} & 1 \\ 0 & 0 & 0 & 0 \\ \underline{\mathbf{1}} & 1 & \underline{\mathbf{1}} & \underline{\mathbf{1}} \end{matrix}$$

# Αλγόριθμος Warshall (2)

Κύρια ιδέα: υπάρχει ένα μονοπάτι μεταξύ των κορυφών *i* και j αν

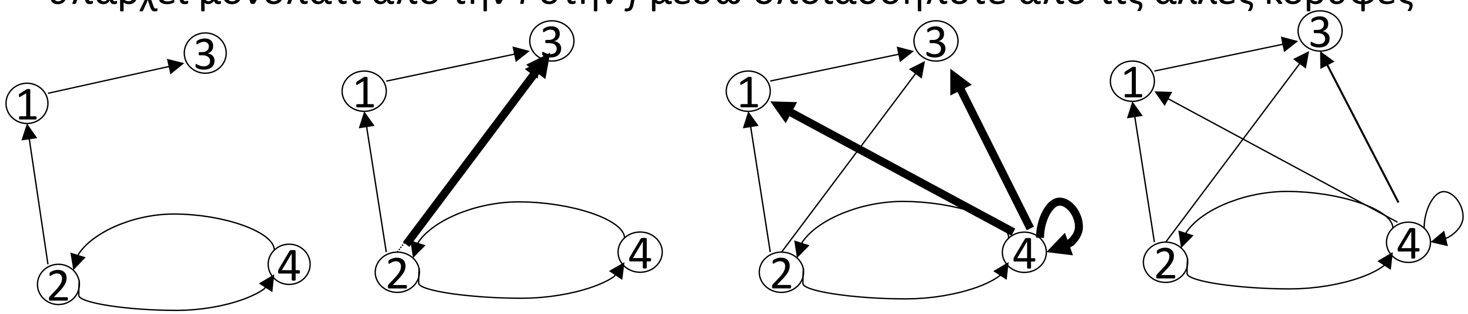
υπάρχει ακμή από την *i* στην *j*,

υπάρχει μονοπάτι από την *i* στην *j* μέσω της κορυφής 1;

υπάρχει μονοπάτι από την *i* στην *j* μέσω των κορυφών 1 ή/και 2;

...

υπάρχει μονοπάτι από την *i* στην *j* μέσω οποιασδήποτε από τις άλλες κορυφές




$R_0$

$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

$R_1$

$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & \underline{\mathbf{1}} & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

$R_2$

$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ \underline{\mathbf{1}} & 1 & \underline{\mathbf{1}} & \underline{\mathbf{1}} \end{matrix}$$

$R_3$

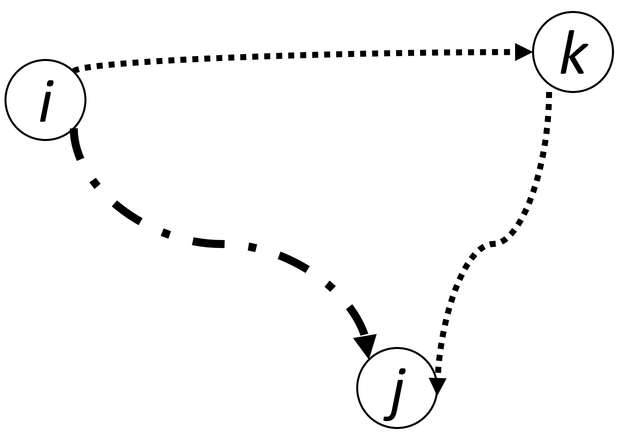
$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{matrix}$$

$R_4$

$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & \underline{\mathbf{1}} & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{matrix}$$

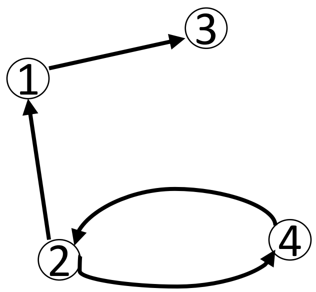
# Αλγόριθμος Warshall (3)

- Στο *k*-οστό στάδιο βρίσκουμε αν υπάρχει ένα μονοπάτι μεταξύ των κορυφών *i*, *j* χρησιμοποιώντας μόνο τις κορυφές 1,…,*k*
- $$R^{(k)}[i,j] = \begin{cases} R^{(k-1)}[i,j] & \text{(μονοπάτι μόνο με 1 ,…,}k\text{-1)} \\ \text{or} & \\ (R^{(k-1)}[i,k] \ \& \ R^{(k-1)}[k,j]) & \text{(μονοπάτι από } i \text{ σε } k \text{ και από } k \text{ σε } i \text{ χρησιμοποιώντας μόνο τις 1,…,}k\text{-1)} \end{cases}$$




$k^{th}$ stage

# Αλγόριθμος Warshall - παράδειγμα




$$R^{(0)} = \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix} \qquad R^{(1)} = \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & \mathbf{1} & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

$$R^{(2)} = \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ \mathbf{1} & 1 & \mathbf{1} & \mathbf{1} \end{matrix} \qquad R^{(3)} = \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{matrix} \qquad R^{(4)} = \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & \mathbf{1} & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{matrix}$$

# Αλγόριθμος Warshall - ψευδοκώδικας

**Algorithm Warshall(A[1..n,1..n])**

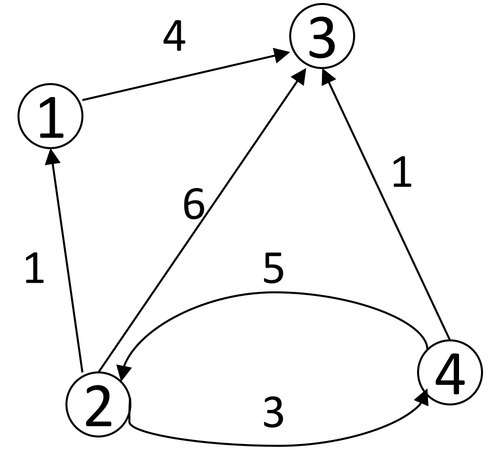
```
R(0) ← A
for k ← 1 to n do
   for i ← 1 to n do
      for j ← 1 to n do
         R(k)[i,j] ← R(k-1)[i,j] or R(k-1)[i,k] and R(k-1)[k,j]
return R(k)
```

Χρονική αποτελεσματικότητα: Θ($n^3$)

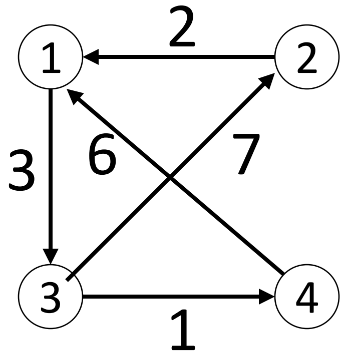
Χωρική αποτελεσματικότητα: Θ($n^2$) – επαναχρησιμοποίηση χώρου

# Αλγόριθμος Floyd: “All pairs shortest paths”

- Να βρούμε τα συντομότερα μονοπάτια μεταξύ όλων των κορυφών ενός ζυγισμένου γράφου
- Ιδέα: να βρούμε τη λύση μέσω μιας σειράς πινάκων D(0), D(1), … χρησιμοποιώντας ένα αρχικό υποσύνολο κορυφών ως ενδιάμεσες
- Παράδειγμα:



# Αλγόριθμος Floyd - παράδειγμα




$$D^{(0)} = \begin{array}{cccc} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{array}$$

$$D^{(1)} = \begin{array}{cccc} 0 & \infty & 3 & \infty \\ 2 & 0 & \mathbf{5} & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \mathbf{9} & 0 \end{array}$$

$$D^{(2)} = \begin{array}{cccc} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ \mathbf{9} & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{array}$$

$$D^{(3)} = \begin{array}{cccc} 0 & \mathbf{10} & 3 & \mathbf{4} \\ 2 & 0 & 5 & \mathbf{6} \\ 9 & 7 & 0 & 1 \\ 6 & \mathbf{16} & 9 & 0 \end{array}$$

$$D^{(4)} = \begin{array}{cccc} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ \mathbf{7} & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{array}$$

# Αλγόριθμος Floyd

**<u>Algorithm Floyd(W[1..n,1..n])</u>**

```
D←W
for k←1 to n do
  for i←1 to n do
        for j←1 to n do
     D[i,j]←min(D[i,j],D[i,k]+D[k,j])
return D
```

Χρονική και Χωρική αποτελεσματικότητα

Πότε δεν δουλεύει?

Αρχή της βελτιστότητας

# Βέλτιστα Δυαδικά Δένδρα Αναζήτησης

- Τα κλειδιά δεν αναζητούνται ισοπίθανα
- Έστωσαν τα κλειδιά A, B, C, D με πιθανότητες 0.1, 0.2, 0.4, 0.3
- Ποιο είναι το μέσο κόστος αναζήτησης σε κάθε πιθανή δομή ?
- Πόσες διαφορετικές δομές μπορούν να παραχθούν με *n* κόμβους ?
- Καταλανικός αριθμός C(n)=comb(2n,n)/(n+1)

# Βέλτιστα Δυαδικά Δένδρα Αναζήτησης (2)

- Το *C[i,j]* δίνει το μικρότερο μέσο κόστος αναζήτησης σε ένα δένδρο με τα στοιχεία από *i* μέχρι *j*
- Με βάση της αρχή της βελτιστότητας, το βέλτιστο δένδρο αναζήτησης παράγεται με βάση την αναδρομική εξίσωση:
  - $C[i,j] = \min\{C[i,k-1]+C[k+1,j]\} + \Sigma p_s$

    για $1 \leq i \leq j \leq n$ και $i \leq k \leq j$
  - $C[i,i] = p_i$

# Βέλτιστα Δυαδικά Δένδρα Αναζήτησης (3)

**Algorithm OptimalBST(P[1...n])**

```
//Finds an optimal binary search tree by dynamic
programming

//Input: An array P[1...n] of search probabilities for a
sorted list of n keys

//Output: Average number of comparisons in successful
searches in the optimal

// BST and table R of subtrees' roots in the optimal BST.

for i = 1 to n, do

      C[i,i-1]=0

      C[i,i]=P[i]

      R[i,i]=i

C[n+1,n]=0
```

# Βέλτιστα Δυαδικά Δένδρα Αναζήτησης (4)

**Algorithm OptimalBST(P[1...n]) - συνέχεια**

```
for d ← 1 to n-1 do //diagonal count
      for I ← 1 to n-d do
            j ← i+d
            minval ← ∞
            for k ← i to j do
                  if C[i,k-1]+C[k+1,j] < minval
                       minval ← C[i,k-1]+C[k+1,j]; kmin ← k
            R[i,j] ← kmin
            sum ← P[i]; for s ← i+1 to j do sum ← sum + P[s]
            C[i,j] ← minval + sum
return C[1,n], R
```

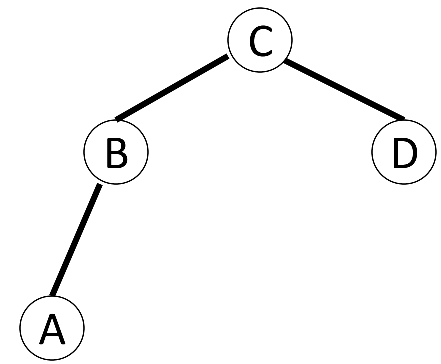
# Βέλτιστα Δυαδικά Δένδρα Αναζήτησης (5)

**Παράδειγμα: κλειδί** ***A B C D***

**πιθανότητα 0.1 0.2 0.4 0.3**

| $i$ \ $j$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **1** | 0 | .1 | .4 | 1.1 | 1.7 |
| **2** | | 0 | .2 | .8 | 1.4 |
| **3** | | | 0 | .4 | 1.0 |
| **4** | | | | 0 | .3 |
| **5** | | | | | 0 |

| $i$ \ $j$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **1** | | 1 | 2 | 3 | 3 |
| **2** | | | 2 | 3 | 3 |
| **3** | | | | 3 | 3 |
| **4** | | | | | 4 |
| **5** | | | | | |




**βέλτιστο BST**

Χρονική αποτελεσματικότητα: $\Theta(n^3)$ αλλά μπορεί να μειωθεί σε $\Theta(n^2)$ αν εκμεταλλευτούμε την μονοτονικότητα στον πίνακα ριζών.

Χωρική αποτελεσματικότητα: $\Theta(n^2)$

# Το πρόβλημα του σάκου

- Δίνονται $n$ στοιχεία με βάρη $w_1, w_2, \ldots, w_n$, αξίες $v_1, v_2, \ldots, v_n$ και χωρητικότητα σάκου $W$. Να βρεθεί το πολυτιμότερο υποσύνολο των στοιχείων που χωρούν στο σάκο
- Λύση με εξαντλητική αναζήτηση
- Έστω ότι το $V[i,j]$ δίνει τη βέλτιστη λύση για τα πρώτα $i$ στοιχεία και χωρητικότητα $j$.
- Σκοπός είναι ο υπολογισμός του $V[n,W]$
- Αναδρομή

$$V[i,j] = \begin{cases} \max\{V[i-1,j], v_i + V[i-1,j-w_i]\} & \text{if } j-w_i \geq 0 \\ V[i-1,j] & \text{if } j-w_i < 0 \end{cases}$$

# Το πρόβλημα του σάκου (2)

Παράδειγμα: *W*=5

| είδος | βάρος | αξία |
|---|---|---|
| 1 | 2 | 12 |
| 2 | 1 | 10 |
| 3 | 3 | 20 |
| 4 | 2 | 15 |

| *i,j* | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | | | | | |
| 2 | 0 | | | | | |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | ? |

Χρονική και Χωρική αποτελεσματικότητα

# Συναρτήσεις Μνήμης

- Ένα μειονέκτημα του δυναμικού προγραμματισμού είναι ότι επιλύει υποπροβλήματα που δεν χρειάζονται τελικά
- Μια εναλλακτική τεχνική είναι ο συνδυασμός της top-down με την bottom-up προσέγγιση (δηλ. αναδρομή συν ένα πίνακα προσωρινών αποτελεσμάτων)

# Συναρτήσεις Μνήμης (2)

**<u>Algorithm MFKnapsack(i,j)</u>**

```
if V[i,j]<0
        if j<Weights[i]
            value← MFKnapack[i-1,j]
        else
            value←max{MFKnapsack(i-1,j),
                     Values[i] + MFKnapsack[i-1,j-
Weights[i]]}
    V[i,j]←value
return V[i,j]
```

# Συναρτήσεις Μνήμης (3)

Παράδειγμα: $W$=5

| είδος | βάρος | αξία |
|---|---|---|
| 1 | 2 | 12 |
| 2 | 1 | 10 |
| 3 | 3 | 20 |
| 4 | 2 | 15 |

| *i,j* | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | | | | | |
| 2 | 0 | | | | | |
| 3 | 0 | | | | | |
| 4 | 0 | | | | | ? |

Χρονική και Χωρική αποτελεσματικότητα: παραμένουν ίδιες

# Σημείωμα Αναφοράς

# Σημείωμα Αδειοδότησης

# Τέλος ενότητας

Επεξεργασία: Ανδρέας Κοσματόπουλος
Θεσσαλονίκη, Αύγουστος 2015



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ
2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

# Σημειώματα

# Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.00.

# Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.