Αλγόριθμοι και Πολυπλοκότητα

7ο εξάμηνο

Σ.Η.Μ.Μ.Υ. & Σ.Ε.Μ.Φ.Ε.

http://www.corelab.ece.ntua.gr/courses/

5η εβδομάδα: Δυναμικός Προγραμματισμός (Dynamic Programming): Discrete Knapsack, All-Pairs Shortest Paths, Traveling Salesman

Διδάσκοντες:

Στάθης Ζάχος – Άρης Παγουρτζής

Δυναμικός Προγραμματισμός

Αρχή της Βελτιστότητας (Optimality Principle)

Οποιαδήποτε υπακολουθία της βέλτιστης ακολουθίας αποφάσεων είναι βέλτιστη για το αντίστοιχο υποπρόβλημα.

Παράδειγμα

Συντομότερα μονοπάτια:

Αν v->w: v, … , u, …, w συντομότερο

τότε και u->w: u,…, w συντομότερο

Αναδρομική σχέση «προς τα εμπρός» (forward):

$$MinPathCost_{u \to w} = \min_{\forall k \ adjacent \ to \ u} (Cost(u,k) + MinPathCost_{k \to w})$$

Αναδρομική σχέση «προς τα πίσω» (backward):

$$MinPathCost_{u \to w} = \min_{\forall k \ adjacent \ to \ w} (MinPathCost_{u \to k} + Cost(k,w))$$

Πρόβλημα Σακιδίου Ακεραίων Τιμών (Discrete Knapsack Problem)

Ορισμός: όπως το Knapsack, αλλά κάθε αντικείμενο u_i μπαίνει στο σακίδιο είτε ολόκληρο (x_i=1) είτε καθόλου (x_i=0).

$$\begin{cases} \sum w_i x_i & \leq & M \\ \sum p_i x_i & = & MAXIMUM \end{cases}$$

Discrete Knapsack (συν.)

<u>Ισχύει η αρχή βελτιστότητας:</u>

Κάθε τελικό τμήμα

$$x_i, \ldots, x_n$$

της βέλτιστης ακολουθίας

$$x_1, x_2, \ldots, x_n$$

είναι βέλτιστη ακολουθία για το αντίστοιχο υποπρόβλημα με βάρος $M' = M - \sum_{k=1}^{i-1} w_k x_k$

Discrete Knapsack: αναδρομικές σχέσεις

Forward approach:

$$Knap(1..n, M) = \max\{0 + Knap(2..n, M), p_1 + Knap(2..n, M - w_1)\}$$

Backward approach

$$Knap(1..n, M) = \max\{0 + Knap(1..n-1, M), p_n + Knap(1..n-1, M - w_n)\}$$

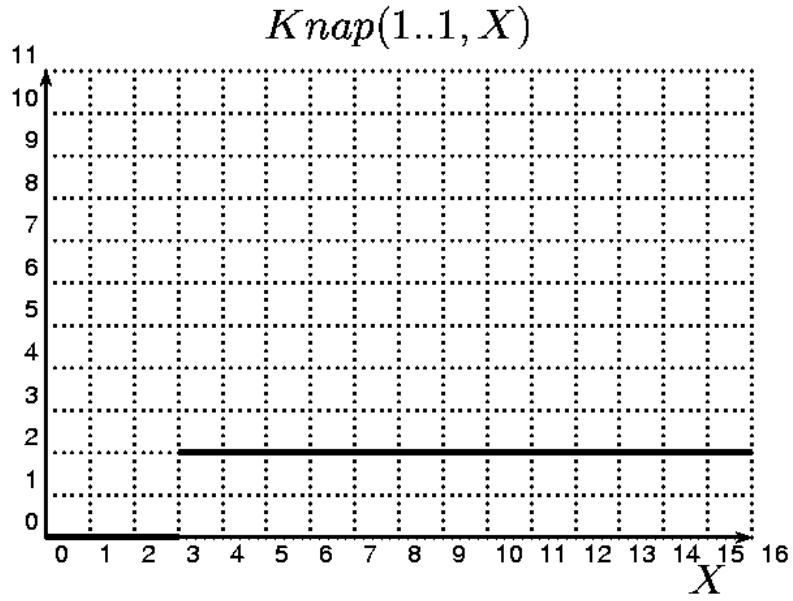
Γενικότερα:

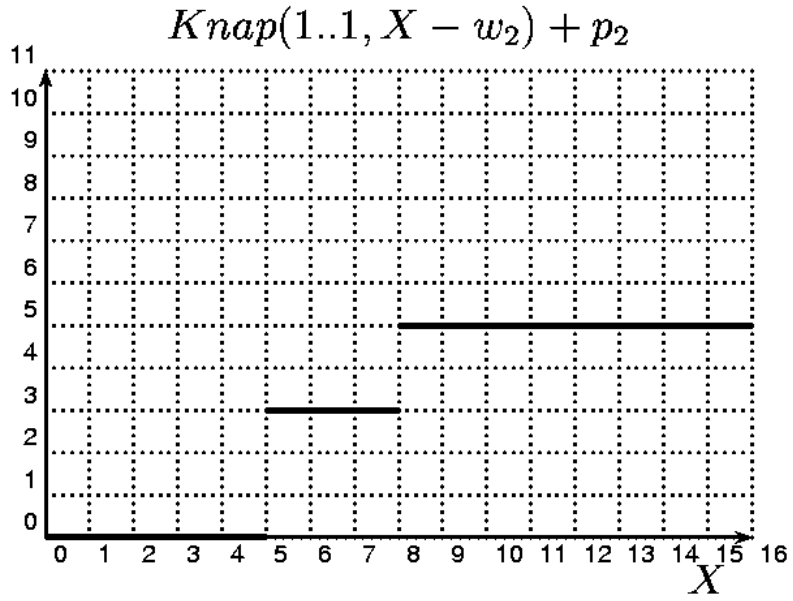
$$Knap(1..i, X) = \max\{0 + Knap(1..i-1, X), p_i + Knap(1..i-1, X - w_i)\}$$

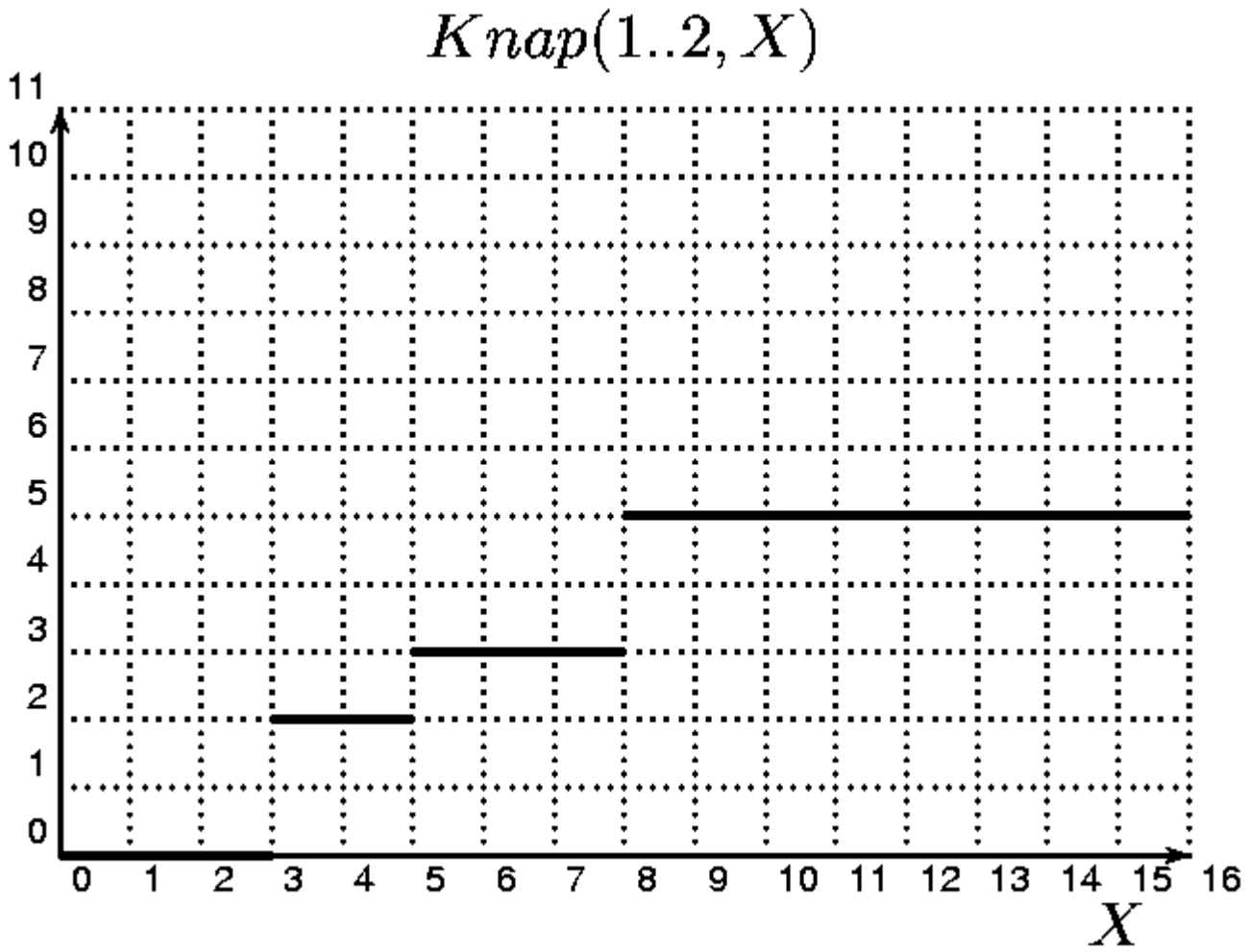
D. Knapsack: η μέθοδος των γραφικών παραστάσεων

$w_1 = 3 \quad w_2 = 5 \quad w_3 = 6$

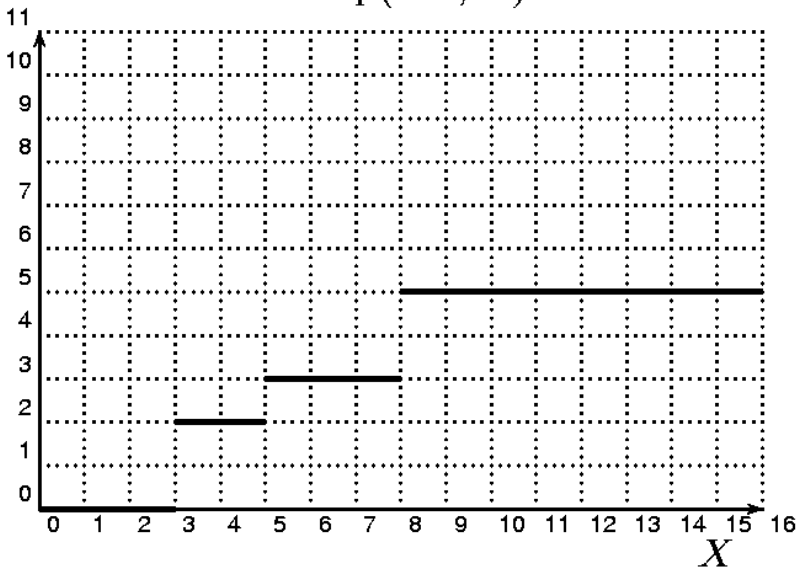
$p_1 = 2 \quad p_2 = 3 \quad p_3 = 5 \qquad M = 10$



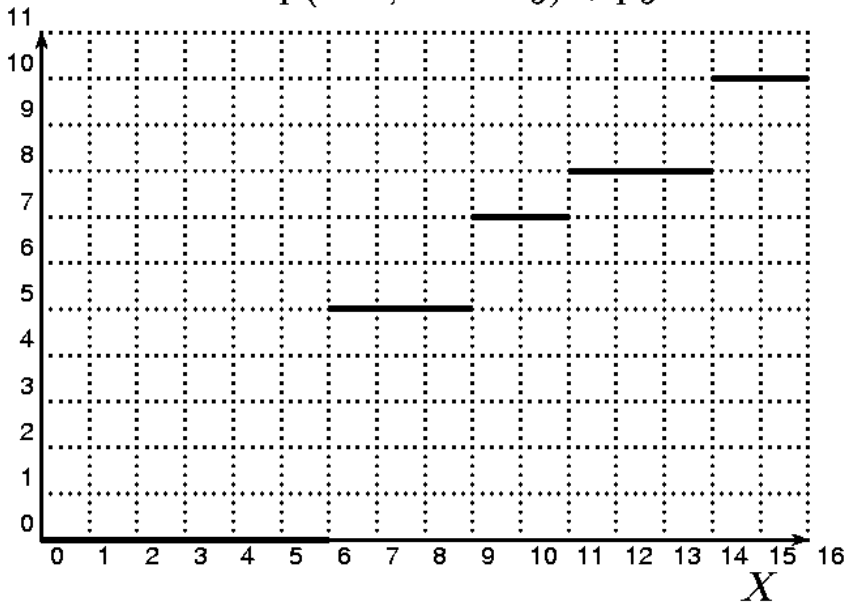



$Knap(1..2, X)$
11
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
X

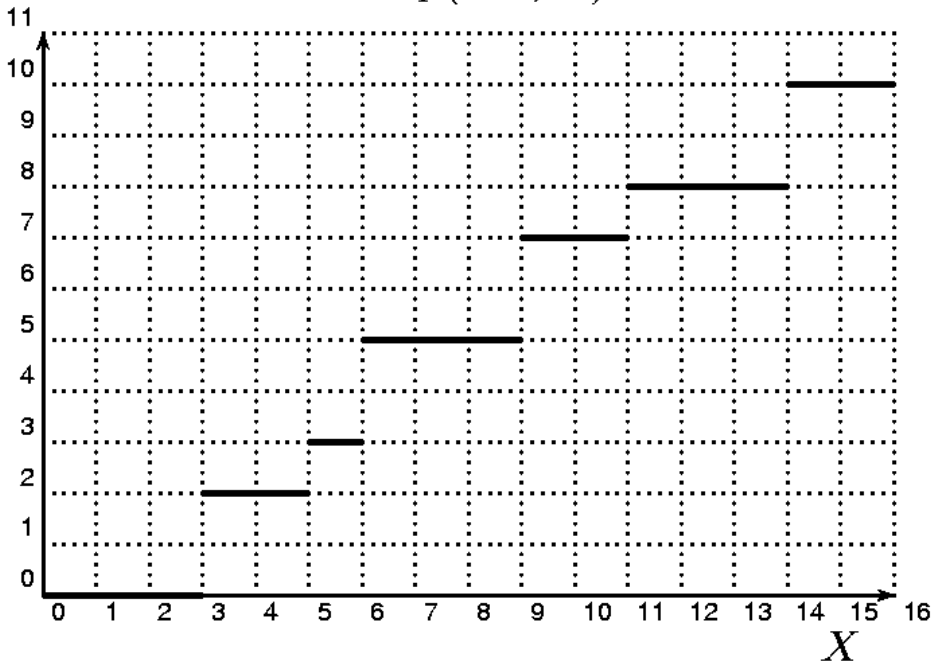
$Knap(1..2, X)$

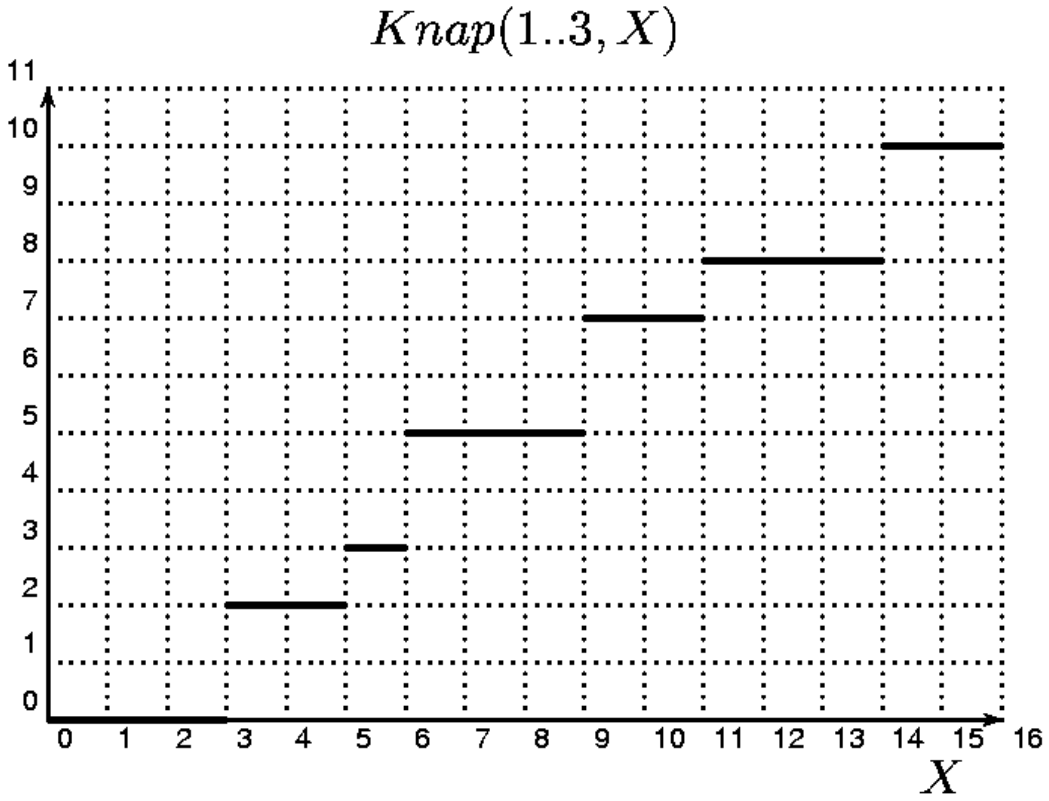


$Knap(1..2, X - w_3) + p_3$



$Knap(1..3, X)$





Σημαντικά ζεύγη:

$$(0,0),\ (3,2),\ (5,3),\ (6,5),\ (9,7),\ (11,8),\ (14,10)$$

D. Knapsack: η μέθοδος του πίνακα

$w_1 = 3$ $w_2 = 5$ $w_3 = 6$

$p_1 = 2$ $p_2 = 3$ $p_3 = 5$ M = 10

i											
(u_3) 3	0	0	0	2	2	3	**5**	5	5	**7**	7
(u_2) 2	0	0	0	2	2	**3**	3	3	**5**	5	5
(u_1) 1	0	0	0	**2**	2	2	2	2	2	2	2
0	**0**	0	0	0	0	0	0	0	0	0	0
	0	1	2	3	4	5	6	7	8	9	10
											X

$$K[i, X] = \max\{K[i-1, X], p_i + K[i-1, X - w_i]\}$$

K [i, X], ή *Knap* (1..*i*, *X*), είναι η μέγιστη αξία που μπορεί να επιτευχθεί με χρήση των αντικειμένων u_1, ..., u_i, και μέγεθος σακιδίου *X*.

Discrete Knapsack: Αλγόριθμος Merge-Discard

- $S_0^0 = \{(0,0)\}$
- for i:=1 to n do

$$S_1^i = \{(w + w_i, p + p_i) \mid (w,p) \in S_0^{i-1}\}.$$

$$S_0^i = \text{merge-discard}(S_0^{i-1}, S_1^i).$$

- traceback

$$\text{merge-discard}(S_0^{i-1}, S_1^i) =$$
$$S_0^{i-1} \cup S_0^i - \{(w,p) \mid \exists (w',p') : w' \leq w \wedge p' \geq p\} - \{(w,p) \mid w > M\}$$

Παράδειγμα Merge-Discard

i	S_0^i	S_1^i
0	{(0,0)}	
1		{(3,2)}
1	{(0,0),(3,2)}	
2		{(5,3),(8,5)}
2	{(0,0),(3,2),(5,3),(8,5)}	
3		{(6,5),(9,7),(11,8),(14,10)}
3	{(0,0),(3,2),(5,3),~~(8,5)~~,(6,5),(9,7),~~(11,8)~~,~~(14,10)~~}	

Διαδικασία Traceback

procedure Traceback;
begin
  A:=(the last pair **of** S_0^n that is the pair with maximum cost);
  for $i := n$ **downto** 1 **do**
    if A **in** S_1^i **then begin** $x_i := 1$; $A := (w_A - w_i, p_A - p_i)$ **end**
    else $x_i := 0$
end

Πολυπλοκότητα αλγορίθμου Discrete Knapsack

Χειρότερη περίπτωση: εκθετική

Συγκεκριμένα:

$O(\min(2^n, nW))$, όπου W το άθροισμα των βαρών.

All Pairs Shortest Paths

Είσοδος: Γράφος G(V,E) με n κόμβους και βάρη σε κάθε πλευρά u->w, C[u,w].

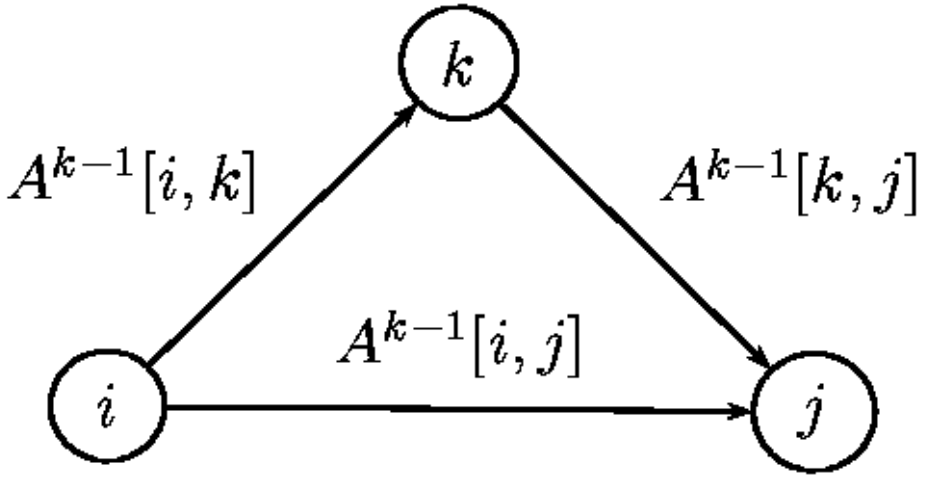
Εξοδος: συντομότερο μονοπάτι για κάθε ζεύγος κόμβων.

Αρχικοποίηση:

$$A[i,j] = \begin{cases} 0 & ,i = j \\ C[i,j] & ,i \neq j \& (i,j) \in E \\ \infty & ,\text{αλλιώς} \end{cases}$$

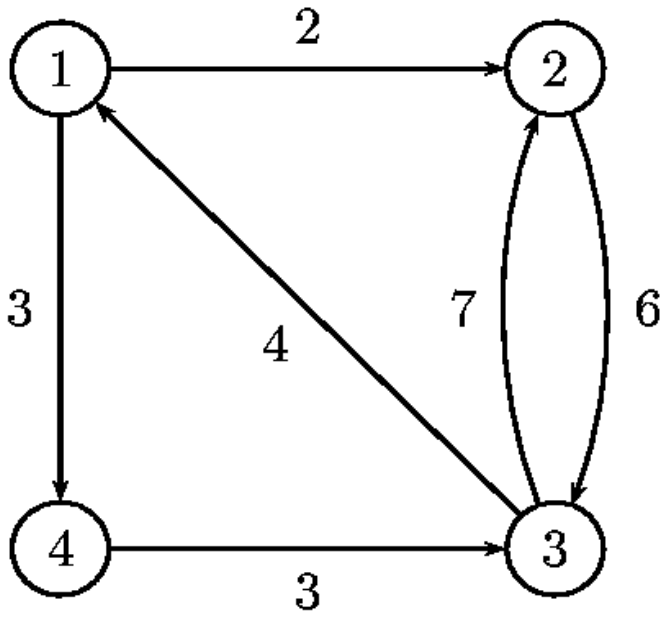
Αναδρομική σχέση

$$A^k[i,j] = \min(A^{k-1}[i,j], A^{k-1}[i,k] + A^{k-1}[k,j])$$



Αλγόριθμος Floyd

```
procedure AllShortestPaths (...);
begin
  for i:=1 to n do
    for j:=1 to n do A[i,j]:=Cost[i,j];
  for k:=1 to n do
    for i:=1 to n do
      for j:=1 to n do
        A[i,j]:= min(A[i,j], A[i,k]+A[k,j])
end
```

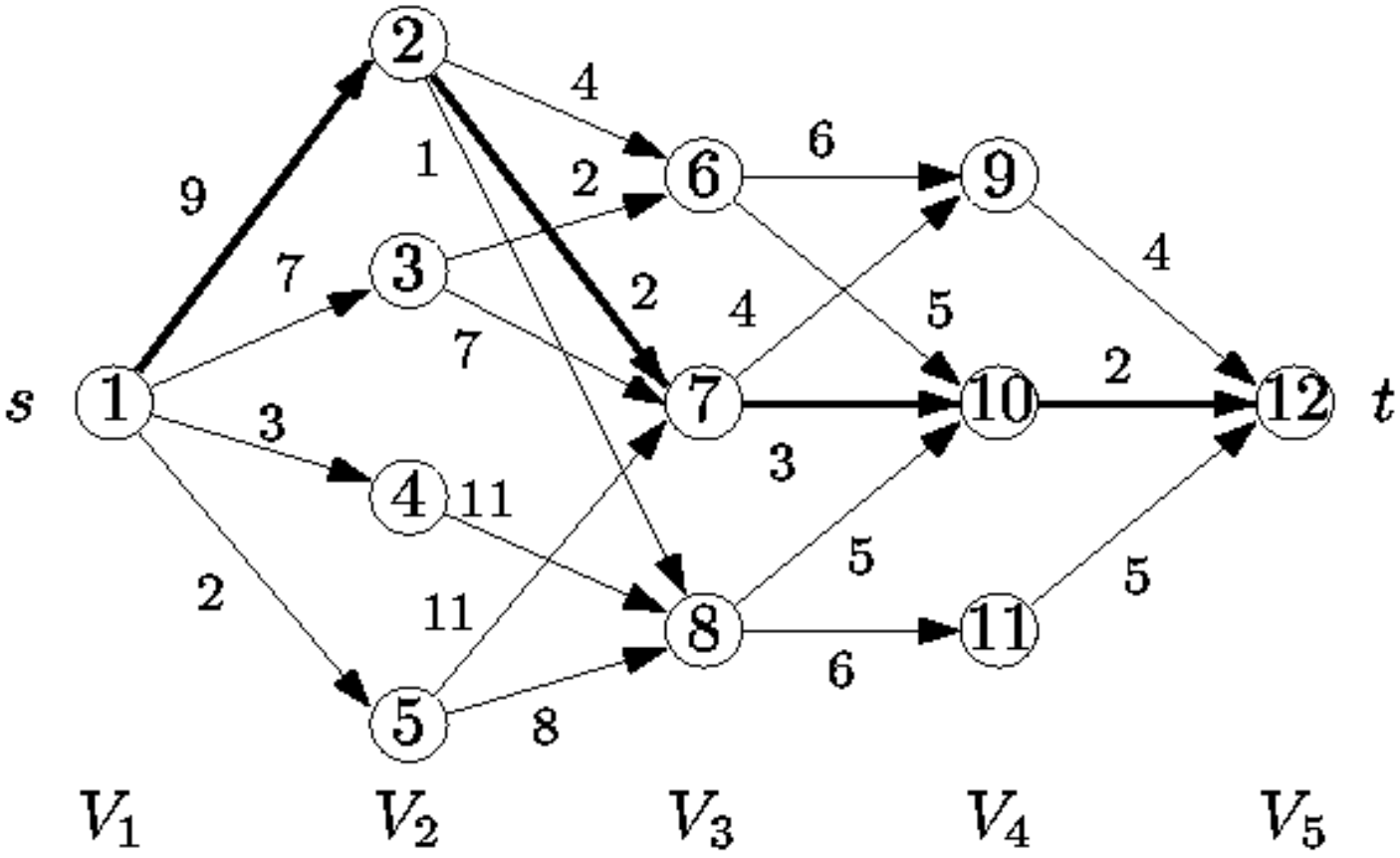


$$C = \begin{bmatrix} 0 & 2 & \infty & 3 \\ \infty & 0 & 6 & \infty \\ 4 & 7 & 0 & \infty \\ \infty & \infty & 3 & 0 \end{bmatrix}$$

$$A^1 = \begin{bmatrix} 0 & 2 & \infty & 3 \\ \infty & 0 & 6 & \infty \\ 4 & 6 & 0 & 7 \\ \infty & \infty & 3 & 0 \end{bmatrix}, A^2 = \begin{bmatrix} 0 & 2 & 8 & 3 \\ \infty & 0 & 6 & \infty \\ 4 & 6 & 0 & 7 \\ \infty & \infty & 3 & 0 \end{bmatrix}$$

$$A^3 = \begin{bmatrix} 0 & 2 & 8 & 3 \\ 10 & 0 & 6 & 13 \\ 4 & 6 & 0 & 7 \\ 7 & 9 & 3 & 0 \end{bmatrix}, A^4 = \begin{bmatrix} 0 & 2 & 6 & 3 \\ 10 & 0 & 6 & 13 \\ 4 & 6 & 0 & 7 \\ 7 & 9 & 3 & 0 \end{bmatrix}$$

Συντομότερο Μονοπάτι σε multistage graph



Αλγόριθμος για shortest path σε multistage graph

Αναδρομική σχέση:

$$OptCost[1,n] = \min_{r \in V_2}\{cost[1,r] + OptCost[r,n]\}$$

```
procedure ShortestPathInMultistageGraph (...)
begin
   DistFromN[n]:=0;
   for j:=n−1 downto 1 do
   begin
      select r with min{cost[j,r]+DistFromN[r]};
      DistFromN[j]:=cost[j,r]+DistFromN[r]; next[j]:=r;
   end
end
```

Πρόβλημα Πλανόδιου Πωλητή (Traveling Salesman Problem - TSP)

Είσοδος: Πλήρης γράφος με βάρη.

Εξοδος: Κύκλος ελαχίστου κόστους που περνάει από όλους τους κόμβους 1 φορά.

Αρχή βελτιστότητας:

$$MinCostTour_{1\to 1} = \min_{k\neq 1}\{cost[1,k] + MinCostTour_{k\to 1}\}$$

Αναδρομικές Σχέσεις

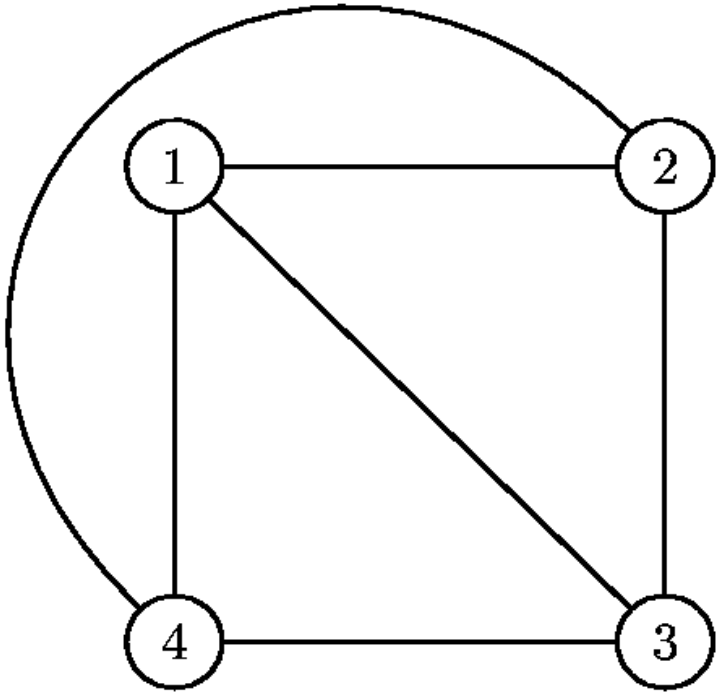
g(i,S) : κόστος συντομότερου μονοπατιού από τον i στον 1 που περνά από όλους τους κόμβους του S.

$$g(1, V - \{1\}) = \min_{2 \le k \le n} \{cost[1,k] + g(k, V - \{1,k\})\}$$

$$g(i,S) = \min_{j \in S} \{cost[i,j] + g(j, S - \{j\})\}$$

$$g(i,\emptyset) = cost[i,1].$$

Παράδειγμα



$$C = \begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix}$$

Επίλυση

|V|=0

$$
\begin{aligned}
g(2,\emptyset) &= Cost[2,1] = 5 \\
g(3,\emptyset) &= Cost[3,1] = 6 \\
g(4,\emptyset) &= Cost[4,1] = 8
\end{aligned}
$$

|V|=1

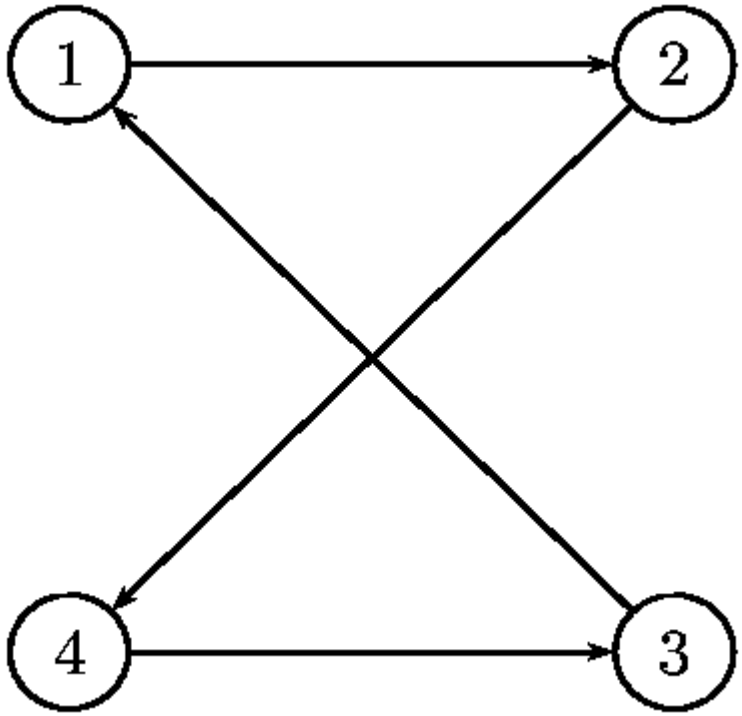
$$
\begin{aligned}
g(2,\{3\}) &= Cost[2,3] + g(3,\emptyset) = 9+6 = 15 \\
g(2,\{4\}) &= Cost[2,4] + g(4,\emptyset) = 10+8 = 18 \\
g(3,\{2\}) &= Cost[3,2] + g(2,\emptyset) = 13+5 = 18 \\
g(3,\{4\}) &= Cost[3,4] + g(4,\emptyset) = 12+8 = 20 \\
g(4,\{2\}) &= Cost[4,2] + g(2,\emptyset) = 8+5 = 13 \\
g(4,\{3\}) &= Cost[4,3] + g(3,\emptyset) = 9+6 = 15
\end{aligned}
$$

|V|=2

$$
\begin{aligned}
g(2,\{3,4^{\swarrow}\}) &= \min(Cost[2,3]+g(3,\{4\}),Cost[2,4]+g(4,\{3\})) \\
&= \min(9+20,10+15)=25 \\
g(3,\{2,4^{\swarrow}\}) &= \min(Cost[3,2]+g(2,\{4\}),Cost[3,4]+g(4,\{2\})) \\
&= \min(13+18,12+13)=25 \\
g(4,\{2^{\swarrow},3\}) &= \min(Cost[4,2]+g(2,\{3\}),Cost[4,3]+g(3,\{2\})) \\
&= \min(8+15,9+18)=23
\end{aligned}
$$

|V|=3

$$
\begin{aligned}
g(1,\{2^{\swarrow},3,4\}) &= \min(Cost[1,2]+g(2,\{3,4\}),Cost[1,3]+g(3,\{2,4\}), \\
&\quad Cost[1,4]+g(4,\{2,3\})) \\
&= \min(10+25,15+25,20+23)=35
\end{aligned}
$$



$$Space = (n-1)\sum_{k=0}^{n-2}\binom{n-2}{k} = (n-1)(1+1)^{n-2} = (n-1)\cdot 2^{n-2}$$