CAMMINI MINIMI TRA TUTTE LE COPPIE DI NODI
(ALL-PAIRS SHORTEST PATHS PROBLEM)

- CONVIENE EFFETTUARE LE SEGUENTI GENERALIZZAZIONI:

  $d[v] \rightsquigarrow d[s,v]$ (STIMA PER ECCESSO DELLA DISTANZA DA s A v)

  $Pred[v] \rightsquigarrow Pred[s,v]$ (PREDECESSORE DI v IN UN CAMMINO DA s A v)

IN QUANTO IN QUESTO CASO LA SORGENTE s NON E' ASSEGNATA

- SUPPONIAMO CHE

$$v_1 \to v_2 \to v_3 \to v_4 \to v_5 \to v_6 \to v_7$$

SIA UN CAMMINO MINIMO IN UN GRAFO PESATO (G, w)

- SUPPONIAMO CHE SIA GIA' NOTO CHE

  $d[v_1, v_4] = \delta(v_1, v_4)$

  $d[v_4, v_7] = \delta(v_4, v_7)$

- ALLORA VALE $d[v_1, v_7] \geq d[v_1, v_4] + d[v_4, v_7]$
- QUINDI, SE $d[v_1, v_7] > d[v_1, v_4] + d[v_4, v_7]$, CONVIENE PORRE

  $d[v_1, v_7] :=$?

  $Pred[v_1, v_7] :=$?

- SUPPONIAMO CHE

$$v_1 \to v_2 \to v_3 \to v_4 \to v_5 \to v_6 \to v_7$$

SIA UN <u>CAMMINO MINIMO</u> IN UN GRAFO PESATO (G, w)

- SUPPONIAMO CHE SIA GIA' NOTO CHE

$$d[v_1, v_4] = \delta(v_1, v_4)$$
$$d[v_4, v_7] = \delta(v_4, v_7)$$

- ALLORA VALE $d[v_1, v_7] \geq d[v_1, v_4] + d[v_4, v_7]$
- QUINDI, SE $d[v_1, v_7] > d[v_1, v_4] + d[v_4, v_7]$, CONVIENE PORRE

$$d[v_1, v_7] := d[v_1, v_4] + d[v_4, v_7]$$
$$Pred[v_1, v_7] := Pred[v_4, v_7]$$

- CIO' SUGGERISCE LA SEGUENTE GENERALIZZAZIONE DI RELAX:

```
G_RELAX (x, y, z; w)
  if d[x,z] > d[x,y] + d[y,z]  then
      d[x,z] := d[x,y] + d[y,z]
      Pred[x,z] := Pred[y,z]
```

- ANCHE LA FUNZIONE *Initialize-single-source* DEVE ESSERE OPPORTUNAMENTE GENERALIZZATA.

```
Initialize-All-Pairs(G, w)
  for u ∈ V[G] do
    for v ∈ V[G] do
      if (u,v) ∈ E[G] then
        d[u,v] := ?
        Pred[u,v] := ?
      else
        Pred[u,v] := ?
        if u ≠ v then
          d[u,v] := ?
        else
          d[u,v] := ?
```

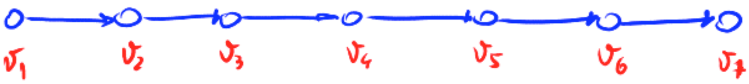
```
Initialize-All-Pairs(G, w)
  for u ∈ V[G] do
    for v ∈ V[G] do
      if (u,v) ∈ E[G] then
        d[u,v] := w(u,v)
        Pred[u,v] := u
      else
        Pred[u,v] := NIL
        if u ≠ v then
          d[u,v] := +∞
        else
          d[u,v] := 0
```

- UNA POSSIBILE SOLUZIONE PER IL PROBLEMA ALL-PAIRS SHORTEST PATHS SI OTTIENE ITERANDO L'ALGORITMO DI BELLMAN-FORD:

```
G_BELLMAN_FORD (G, w)
    Initialize_All_Pairs(G, w)
    for s ∈ V[G] do
        for i := 1 to |V[G]| do
            for (u,v) ∈ E[G] do
                G_RELAX (s, u, v)
```

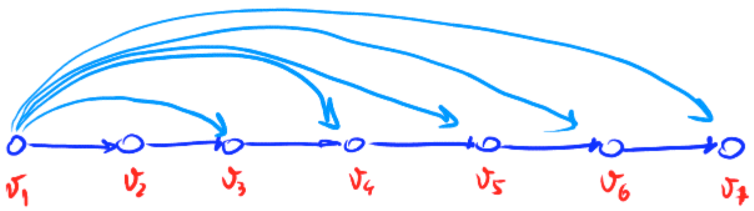
- TALE ALGORITMO HA COMPLESSITA': $O(V^2 E)$
- PER GRAFI DENSI, CIOE' GRAFI CARATTERIZZATI DA $E = \Theta(V^2)$, TALE COMPLESSITA' DIVENTA $O(V^4)$

- L'ALGORITMO G_BELLMAN_FORD NON E' MOLTO EFFICIENTE, IN QUANTO NON SFRUTTA AL MEGLIO I DATI PARZIALI CALCOLATI
- ESSENZIALMENTE, PER UN DATO CAMMINO MINIMO



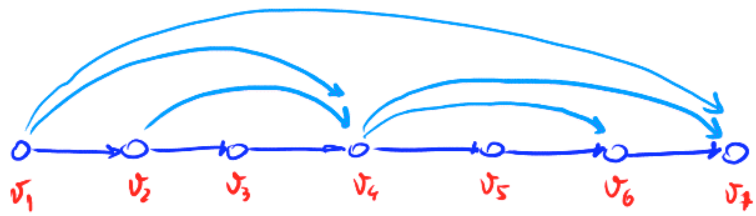
CALCOLA $d[v_1, v_7]$ MEDIANTE LA SEQUENZA DI CHIAMATE

G_RELAX(v_1, v_2, v_3)
G_RELAX(v_1, v_3, v_4)
G_RELAX(v_1, v_4, v_5)
G_RELAX(v_1, v_5, v_6)
G_RELAX(v_1, v_6, v_7)

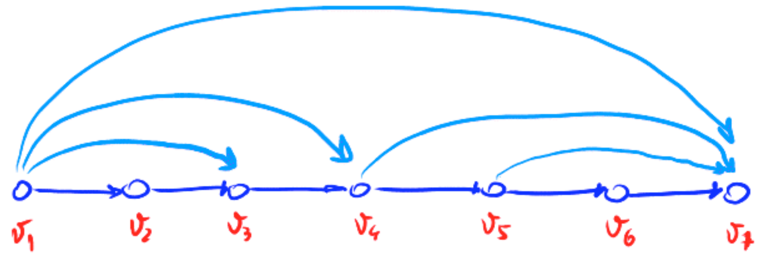


- MA PARECCHIE ALTRE SEQUENZE SONO POSSIBILI: [QUANTE?]

G_RELAX(v_4, v_5, v_6)
G_RELAX(v_2, v_3, v_4)
G_RELAX(v_4, v_6, v_7)
G_RELAX(v_1, v_2, v_4)
G_RELAX(v_1, v_4, v_7)



G_RELAX(v_5, v_6, v_7)
G_RELAX(v_1, v_2, v_3)
G_RELAX(v_4, v_5, v_7)
G_RELAX(v_1, v_3, v_4)
G_RELAX(v_1, v_4, v_7)



....

- QUAL E' UN MODO SISTEMATICO ED EFFICIENTE PER EFFETTUARE TALI CHIAMATE A G_RELAX ?

- BASTA EFFETTUARE I SEGUENTI BLOCCHI DI CHIAMATE

$$\left\{\begin{array}{l} \text{G\_RELAX}(-, y_1, -) \\ \vdots \\ \text{G\_RELAX}(-, y_1, -) \end{array}\right\} V^2$$

$$\left\{\begin{array}{l} \text{G\_RELAX}(-, y_2, -) \\ \vdots \\ \text{G\_RELAX}(-, y_2, -) \end{array}\right\} V^2$$

...

$$\left\{\begin{array}{l} \text{G\_RELAX}(-, y_n, -) \\ \vdots \\ \text{G\_RELAX}(-, y_n, -) \end{array}\right\} V^2$$

SEGUENDO UN "QUALUNQUE" ORDINAMENTO $y_1, y_2, \dots, y_n$ ASSEGNATO DEI NODI IN V

ALGORITMO DI FLOYD-WARSHALL

```
Floyd-Warshall (G, w)
  Initialize_All_Pairs(G, w)
  for y ∈ V[G] do
     for x ∈ V[G] do
        for z ∈ V[G] do
           G_RELAX(x, y, z)
```

COMPLESSITA': $O(V^3)$

CORRETTEZZA

- SIA $v_1, v_2, \ldots, v_n$ L'ORDINE IN CUI GLI ELEMENTI $y \in V[G]$ VENGONO SELEZIONATI NEL CICLO for PIÙ ESTERNO.
- PER $1 \leq i \leq n$ E $u, v \in V[G]$, PONIAMO:

  $\delta^{(i)}(u,v)$ = PESO DI UN CAMMINO/CICLO MINIMO SEMPLICE DA u A v I CUI NODI INTERNI SONO IN $\{v_1, \ldots, v_i\}$

- SI DIMOSTRA PER INDUZIONE CHE DOPO L' i-ESIMA ITERAZIONE DEL CICLO for PIÙ ESTERNO VALGONO:

  $d[u,v]$ = PESO DI UN CAMMINO π DA u A v I CUI NODI INTERNI SONO IN $\{v_1, \ldots, v_i\}$ E TALE CHE $w(\pi) \leq \delta^{(i)}(u,v)$

  $Pred[u,v]$ = PREDECESSORE DI v IN UN CAMMINO π DA u A v I CUI NODI INTERNI SONO IN $\{v_1, \ldots, v_i\}$ E TALE CHE $w(\pi) = d[u,v]$

- PERTANTO ALLA FINE DELL'ESECUZIONE DELL'ALGORITMO DI FLOYD-WARSHALL, SE IL GRAFO PESATO (G,w) DI PARTENZA AMMETTE CAMMINI MINIMI VARRA':

$$d[u,v] = \delta_{(G,w)}(u,v)$$

$Pred[u,v]$ = PREDECESSORE DI v IN UN CAMMINO MINIMO SEMPLICE DA u A v

- SI OSSERVI CHE IN TAL CASO VALE
$d[u,u] = 0$, PER OGNI $u \in V[G]$

- UN CAMMINO MINIMO DA u A v SARA' DATO DA

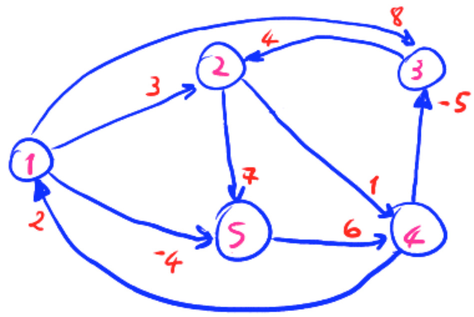
$$v'_k = u, v'_{k-1}, \ldots, v'_1, v'_0 = v,$$

DOVE

$$v'_{i+1} = Pred[u, v'_i] \quad (\neq NIL)$$

```
Build_Min_Path (u,v; G,Pred)
  if u ≠ v and Pred[u,v] = NIL then
     return "v NON E' RAGGIUNGIBILE DA u"
  if u = v then
     return ⟨u⟩
  else
     return Append(Build_Min_Path(u, Pred[u,v]; G,Pred), v)
```

ESEMPIO



$$d = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad Pred = \begin{pmatrix} NIL & 1 & 1 & NIL & 1 \\ NIL & NIL & NIL & 2 & 2 \\ NIL & 3 & NIL & NIL & NIL \\ 4 & NIL & 4 & NIL & NIL \\ NIL & NIL & NIL & 5 & NIL \end{pmatrix}$$

- SE INVECE IL GRAFO (G,w) NON AMMETTE CAMMINI MINIMI, ESISTERA' UN CICLO SEMPLICE π DI PESO NEGATIVO

- SIA v UN NODO SU π
- ALLA FINE DELL'ESECUZIONE DELL'ALGORITMO DI FLOYD-WARSHALL SI HA: $d[v,v] \leq \delta^{(n)}(v,v) \leq w(\pi) < 0$
- QUINDI: CONDIZIONE NECESSARIA E SUFFICIENTE PERCHE' UN GRAFO PESATO (G,w) AMMETTA CAMMINI MINIMI E' CHE ALLA FINE DELL'ESECUZIONE DELL'ALGORITMO DI FLOYD-WARSHALL SI ABBIA $d[u,u]=0$, PER OGNI $u \in V[G]$.

CHIUSURA TRANSITIVA

- DATO UN GRAFO $G=(V,E)$, LA CHIUSURA TRANSITIVA DI G E' IL GRAFO $G^*=(V,E^*)$ TALE CHE:

  $E^* = \{(u,v) \in V \times V : v$ E' RAGGIUNGIBILE DA u IN $G\}$

- UN MODO PER CALCOLARE LA CHIUSURA TRANSITIVA DI G E' IL SEGUENTE:
  - SIA $w: E \rightarrow \{1\}$ UNA FUNZIONE PESO COSTANTE
  - SIA d LA FUNZIONE CALCOLATA DA FLOYD-WARSHALL(G,w)
  - SI OSSERVI CHE $E^* = \{(u,v) \in V \times V : d[u,v] \neq \infty\}$

COMPLESSITA': $O(V^3)$

- UN ALGORITMO PIU' EFFICIENTE IN PRATICA SI OTTIENE MODIFICANDO L'ALGORITMO DI FLOYD-WARSHALL COME SEGUE

```
TRANSITIVE_CLOSURE(G)
  n := |V[G]|
  for i:=1 to n do
     for j:=1 to n do
        if i=j or (i,j)∈E[G] then tij:=1 else tij:=0
  for k:=1 to n do
     for i:=1 to n do
        for j:=1 to n do
           tij := tij ∨ (tik ∧ tkj)
```