11/04/2024 LEZ 26 LAST PROGRAMMAZIONE DINA.

# ALGORITMO DI BELLMAN-FORD-MOORE

ALGO x $\wp$ MIN. PATH.

QUINDI:

- GRAFO DIRETTO
- PESI ARCHI ANCHE NEGATIVI
- NODI $s, t$

$\rightsquigarrow$ ≠ DA PREMESSA OF DAJKSTRA

GOAL $\rightsquigarrow$ $\wp$ $s \rightsquigarrow t$ ∪ PESO MINIMO

DAJKSTRA FAIL ∪ $w(e) < 0$, $\wp$ ANOTHER MODO

**DEF** CICLO NEGATIVO:

CICLO ∪ $\sum w(e) < 0$

DEF IMPORTANTE x 2 LEMMI

**LEMMA 1**

SE PATH $s \rightsquigarrow t$ CONTIENE CICLO NEG., $\nexists$ S.P.

$\rightsquigarrow$ VADO A $\infty$ x ↓ PESO.

**LEMMA 2**

SE CICLO NEG $\notin G$, $\exists$ S.P. $v \rightsquigarrow t$ CHE È SEMPLICE, (È # EDGE $\leq n-1$)

↑ ∅ CONTIENE CICLI.

- DEI S.P., PRENDO QUELLO + CORTO.
- IF CONTIENE CICLO, CAN TOGLIERE SENZA ↑ LUNGHEZZA

# DEF PROBLEMA

PROBL: G DIRETTO, PESATO (w ANCHE NEG.), ∅ CICLI NEGATIVI:

GOAL: DATO t, ∃ SHORT-PATH DI TUTTI I PATH $s \rightsquigarrow t$

SIDE-MISSION: ∃ CICLI NEGATIVI IN G.

# ALGORITMO

IDEA ALGO: → PARTE + "EZ"

SOTTOP: OPT$(i, v)$ = LEN. S.P. $v \rightsquigarrow t$ CON |EDGE| $\leq i$

GOAL: OPT$(n-1, v)$ ∀ ARCO $v$ → CORR. × LEMMA 1.

CASI DI SCELTA

1. S.P. $v \rightsquigarrow t$ USE $\leq i-1$ EDGE
   $OPT(i, v) = OPT(i-1, v)$
2. S.P. $v \rightsquigarrow t$ USA $= i$ EDGE
   ∃ MIN OF
   $$\underbrace{w((v,w))}_{\ell_{vw}} + OPT(i-1, w)$$

PROPIETA' SOTTOSTRU. OTTIMA

Bellman equation.

$$OPT(i, v) = \begin{cases} 0 & \text{if } i = 0 \text{ and } v = t \\ \infty & \text{if } i = 0 \text{ and } v \neq t \\ \min\left\{ OPT(i-1, v), \min\limits_{(v,w)\in E} \{OPT(i-1, w) + \ell_{vw}\} \right\} & \text{if } i > 0 \end{cases}$$

# CODE

SHORTEST-PATHS($V$, $E$, $\ell$, $t$)

FOREACH node $v \in V$ :
  $M[0, v] \leftarrow \infty$. CASO (*)
$M[0, t] \leftarrow 0$. CASO (*)
FOR i = 1 TO $n - 1$
  FOREACH node $v \in V$ :
    $M[i, v] \leftarrow M[i-1, v]$.
    FOREACH edge $(v, w) \in E$ :
      $M[i, v] \leftarrow \min \{ M[i, v],\ M[i-1, w] + \ell_{vw} \}$.

SU M[u, v] ASSEGNO OPT(i, v)

- MATRICE $n \times n$ (Δ) → n NODI; → n EDGE × RIGA 2

→ AL PASSO i : SEE ALL (□) m EDGE

## COSTO

TIME: $\Theta(n \cdot m)$ → CICLO FOR (□)

SPACE: $\Theta(n^2)$ → (Δ)

## CODE: IMPLEMENTATION

1. BUILD SOLUZIONE

2 MODI:

- MEMORIZE SUCCESSOR[i, v], PUNTA AL NODO SUCC. DI UN $v \to t$ PATH CHE USA $\leq i$ e.
- DURING CALCOLO M[i, v] CONSIDERO SOLO ARCHI CON $M[i, v] = M[i-1, w] + \ell_{vw}$, FANNO PARTE DI S.P.

## 2 OTTIMIZZAZIONE SPACE

MANTIENI 2 ARRAY. (INVECE DI MATRIX).

2 ARRAY:
- $d[v]$ = LEN OF PATH $v \rightsquigarrow t$ FINORA
- SUCCESSOR $[v]$ = NEXT NODO OF A $v \rightsquigarrow t$

## 3 BOOST PERFORMANCE

- SE $d[w]$ NON È STATO AGGIORNATO NEL PASSO $i-1$, ALLORA È INUTILE CONSIDERARE ARCHI USCENTI DA $w$ NEL PASSO $i$.

# CODE UPDATE

```
BELLMAN–FORD–MOORE(V, E, ℓ, t)

FOREACH node v ∈ V :
    d[v] ← ∞.                 ⇝ CASO (*)
    successor[v] ← null.
d[t] ← 0.
FOR i = 1 TO n – 1
    FOREACH node w ∈ V :
        IF (d[w] was updated in previous pass)
            FOREACH edge (v, w) ∈ E :
                IF (d[v] > d[w] + ℓ_vw)
                    d[v] ← d[w] + ℓ_vw.
                    successor[v] ← w.
    IF (no d[·] value changed in pass i) STOP.
```

pass $i$
$O(m)$ time

DA QUI CAPIAMO 2 COSE:

1. $\forall v: d[v]$ = LEN DI UN PATH $v \rightsquigarrow t$
2. $\forall v: d[v]$ MONOTONA DECRESCENTE
   ↳ UPDATE SOLO X ↓ VALORE

CE N'È UN'ALTRA, IMPORTANTISSIMA!

**LEMMA**

DOPO PASSO $i$,

$d[v] \leq$ LEN S.P. $v \rightsquigarrow t$ CHE USA $\leq i$ e.

DIM × INDUZIONE SU $i$

- CASO BASE $i = 0 \rightsquigarrow$ OK $\infty = \infty$
- IP IND → VERA AL PASSO $i$
- $P$ = QUALSIASI $v \rightsquigarrow t$ $\cup \leq i+1$ EDGE
- SIA $(v,w)$ ($1°$ EDGE) e $P' = w \rightsquigarrow t$
- × IP. IND $d[w] \leq \ell(P') \rightsquigarrow P'$ = PATH $\cup \leq i$ EDGE
- AL PASSO $i+1$, CONSIDERANDO:

$$d[v] \leq \ell_{vw} + d[w] \leq \ell_{vw} + \ell(P') = \ell(P)$$

↓ E DA L.G.

$d[v]$ NON CRESCE

**COSTO** ⇝ SEMPRE ASSUNTO CHE ∄ CICLI NEGATIVI

TIME ⇝ $\Theta(nm)$ → + VELOCE IN PRATICA

SPACE ⇝ $\Theta(n)$ ⇝ XCHÉ → S.P. ∃ e $\leq n-1$ EDGE
→ DOPO $i$ → $d[v] \leq$ S.P. USA $i$ E.

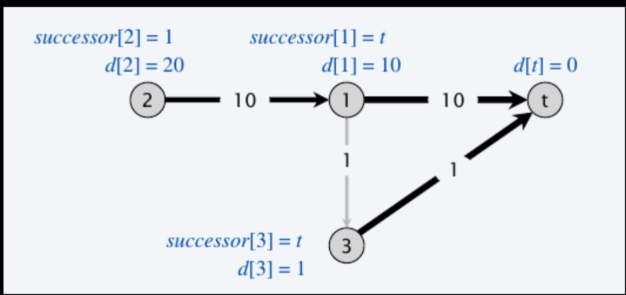
# BUILD SOL.

**POSS. APPROCCIO**: SEGUENDO PATH U SUCC.[V], FIND PATH $v \rightsquigarrow t$ U d[V], DURANTE EXE OF ALGO.

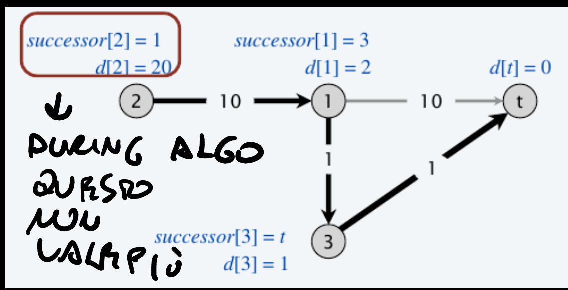
↓

**APP. FALSO**

**PROVA**




DOPO VISITA 3




↓ DURING ALGO QUESTO NON VALE PIÙ