23/04/2024 | LEZ 28

OGGI FAREMO:

- ANALISI ALGO MAX-FLOW (FORD-FULKERSON)
  - MIN-CUT/MAX-FLOW THEOREM
  - ANALISI COMPLESSITÀ

(1) CORRETTEZZA.

NELLA LEZ. PRIMA, ABBIAMO VISTO COME FUNZIONA L'ALGO.

MA XCHÉ FUNZIONA? → CIOÈ f IL FLUSSO MAX.

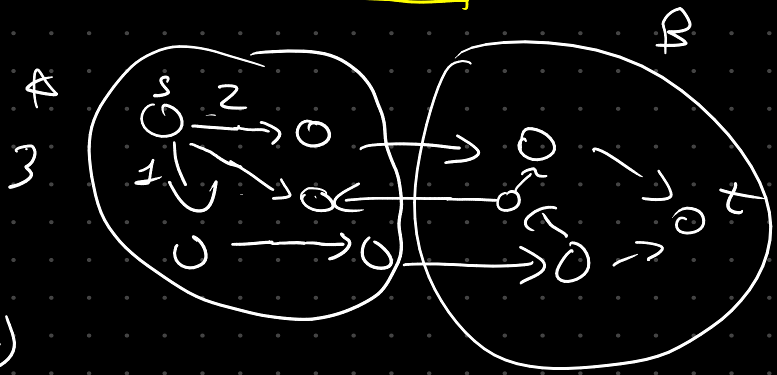
X DIM, USIAMO DIVERSE PROPIETÀ, CON UN TEOREMA:

- LEMMA DEL VALORE FLOW.

SIA ∀ CUT (A,B):

$$\boxed{VAL(f) = \sum_{e \text{ IN } s} f(e) - \sum_{e \text{ OUT } s} f(e) = \sum_{e \text{ IN } A} f(e) - \sum_{e \text{ OUT } A} f(e)}$$

DIM



- DALLA FORMULA

$$VAL(f) = \sum_{e \text{ OUT } 2} f(e) - \sum_{e \text{ IN } 2} f(e)$$

- LA POSSIAMO TRASFORMARE:

$$\sum_{v \in A} \left(\sum_{e \text{ OUT } v} f(e) - \sum_{e \text{ IN } v} f(e) \right) =$$

⇒ ∀ ARCO ∈ A → È SIA ARCO IN CHE ARCO OUT
↳ QUINDI LA DIFF = 0

$$= \sum_{e \text{ OUT } A} f(e) - \sum_{e \text{ IN } A} f(e)$$

↳ ARCHI U-V → U ∈ A ∧ V ∈ B → LO CONSIDERO SOLO COME e OUT

- STESSA COSA X ARCHI IN ↳ CON VALORE NEGATIVO.

• TEOREMA DUALITÀ DEBOLE

SIA F, e ∀ st-CUT. ALLORA

$$VAL(f) \leq CAP(A,B)$$

DIM

x LEMMA DI PRIMA

• $VAL(F) = \sum_{e \text{ OUT } A} f(e) - \sum_{e \text{ IN } A} f(e) \leq \sum_{e \text{ OUT } A} f(e) \leq \sum_{e \text{ OUT } A} c(e) = c(A,B)$

$\Downarrow$

DEBOLE XCHÈ È PIÙ UNA 'STROZZATURA' CHE UNA DELIMITAZIONE NETTA

COROLLARIO

SIA F, st-CUT

IF $VAL(F) = CAP(A,B) \rightarrow F = \text{MAX FLOW} \wedge (A,B) \text{ MIN CUT}$

WEAK DUAL

DIM

• X OGNI FLUSSO f': $VAL(f') \leq CAP(A,B) = VAL(F) =$

$= VAL(F') \leq VAL(F) \rightarrow$ VAL. MASSIMO.

• X OGNI CUT (A',B'): $CAP(A',B') \geq VAL(F) = CAP(A,B)$

$\Downarrow$

$CAP(A',B') \geq CAP(A,B) \rightarrow$ VAL MINIMO.

TEOREMA MAX-FLOW/MIN-CUT

$$\boxed{\text{VALORE MAX FLOW} = \text{CAPACITÀ MIN CUT}}$$

X DIM., DIMOSTRIAMO ALLO STESSO MOMENTO UN ALTRO TEOREMA

TEOREMA AUGM. PATH

$$\boxed{\text{UN FLUSSO } F \text{ SE NON HA AUGM. PATH} \Leftarrow \quad F = \text{MAX FLOW}}$$

DIM DIMOSTRIAMO CHE I 3 ENUNCIATI SONO EQUIVALENTI:

1. $\exists$ CUT(A,B) t.c. CAP(A,B) = VAL(F)
2. f IS MAX
3. $\nexists$ P TO f.

dobbiamo dimostrae che una implica l'altra(1 implica 2, 2 implica 3, 3 implica 1).

- 1 $\Rightarrow$ 2 GIÀ FATTO $\rightarrow$ COROLLARIO DUALITÀ DEBOLE
- 2 $\Rightarrow$ 3 [DIMOSTRO $\neg 3 \rightarrow \neg 2 = 3 \vee \neg 2 \Rightarrow 2 \Rightarrow 3$]
  - SIA P AUGM. PATH RISP. A F
  - ALLORA POSSIAMO ++ F
  - QUINDI F NON È MAX FLOW
- 3 $\Rightarrow$ 1
  - SIA f CON NONE AUGM. PATH
  - SIA A = SET DI NODI RAGGIUNGIBILI DA s NELLA RETE RESIDUA $\rightarrow G_F \rightarrow$ SET PERÒ LO STUDIAMO IN G OG.
    - X DEF DI A : $s \in A$
    - X // DI f : $t \notin A$.

FLOW LEMMA

QUANTO VALE F? $\quad VAL(F) = \sum_{e \text{ OUT } A} F(e) - \sum_{e \text{ IN } A} F(e)$ MA

- e IN A $\Rightarrow F(e) = 0 \rightarrow$ IF $\neq 0$, $G_F \; \exists$ ARCO REVERSE

$\Downarrow$

NODO $\in$ B APPART. AD A.

- e OUT A → F(e) = C(e) → IF < C(e), ∃ IN G_F PATH × REACH NODO ∈ B NODO ∈ A

QUINDI $VAL(F) = \sum_{e \text{ OUT } A} c(e) - 0 = \underline{CAP(A, B)}$

SONO TUTTE EQUIVALENTI

2 COMPLESSITÀ

IL RUNNING TIME DIPENDE FORTEMENTE DA COME IMPLEMENTIAMO LA RICERCA DEI CAMMINI AUGM.

BEFORE THAT, SOME ASSUMPTION:

1. ∀ c(e) HA VALORE x ∈ INT ⇒ $1 \le x \le C$
2. AD OGNI AUGM. DI F(e) (INTERO) ⇒ c(e) RESIDUA INTERO.
3. TEOREMA → × DARE UN'UPPER BOUND
   - ∀ AUGMENT, FLOW++ ALMENO DI 1.
   - QUINDI FORD-FULK. FINISCE ENTRO $VAL(F^*)$ MAX FLOW
     - $VAL(F^*) \le C \cdot n$ → n. NODI

3. COROLL

COSTO ALGO. FORD-FULKERSON, APPARENTEMENTE:

$$COST = O(m \; VAL(F^*)) = O(m \cdot n \cdot C)$$

QUINDI

$O(n \cdot m \cdot C)$ → MA È POLINOMIALE?

RISPOSTA → NOP

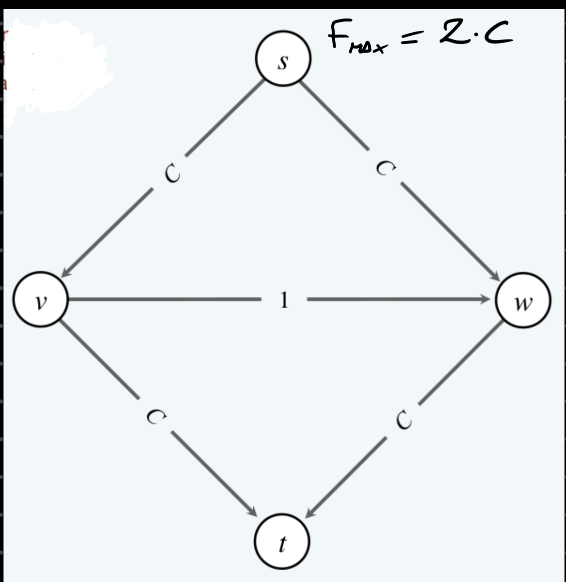
COSTO → PSEUDOPOLINOMIALE

→ DIPENDE DA GRANDEZZA C

→ BIT x REPRESENT $C = \lfloor \lg C \rfloor$

→ SE POCO ATTENTI CON PATH,
↳ COSTO EXPO!

ESEMPIO TIME EXP.



SE IL CAMMINO FATTA MALE:

- SCEGLIE $s \to v \to w \to t \Rightarrow F = 1$
- $v \to w$ ARCO REVERSE
- SCEGLIE $s \to w \to v \to t \Rightarrow F = 1$
- REPEAT

FAREBBE $2 \cdot C$ ⇒ C CAN BE EXPONENTIAL.

QUINDI:

$F(n) = \#$ BIT x RAPP. $C = \lg C \rightsquigarrow C = 2^n \Rightarrow$ TOT. NO. $= O(n \cdot m \cdot C) = O(2^n)$

$\boxed{\text{COME FARE?}}$ ⇝ FARE ATTENZIONE A CHE PATH CHOOSE

PATH DA CHOOSE

→ POSSONO PORTARE A POL. ALGO
→ // // = EXP. ALGO

OBB. ≈ 2 MODI DI CHOOSE P:

1. LARGE BOTT. NECK CAPACITY
2. POCHI ARCHI

1

- MANTENGO VAL. Δ (PARAM. SCALING)
- $G_f(\Delta) \to G_f \cup e$ t.c. $c(e) \geq \Delta$
- $\cup$ AUG. PATH → HA CAPACITY $\geq \Delta$

SCALING CODE

```
CAPACITY-SCALING(G)

FOREACH edge e ∈ E : f(e) ← 0.
Δ ← largest power of 2 ≤ C.

WHILE (Δ ≥ 1)
   Gf(Δ) ← Δ-residual network of G with respect to flow f.
   WHILE (there exists an s↝t path P in Gf(Δ))
      f ← AUGMENT(f, c, P).
      Update Gf(Δ).
   Δ ← Δ / 2.                                  Δ-scaling phase

RETURN f.
```

- # SCALING PHASE = $= 1 + \lfloor \lg C \rfloor \to \geq \Delta$
- # AUG. $\leq 2m$

QUINDI

AUGMENTATION TOT = $\underbrace{m \lg(C)}$

TIME = $\Theta(m^2 \lg C)$

② NEXT PATH U MENO ARCHI

↳ P BY BFS

CODE

SHORTEST-AUGMENTING-PATH(G)

FOREACH $e \in E: f(e) \leftarrow 0.$

$G_f \leftarrow$ residual network of G with respect to flow f.

WHILE (there exists an $s \rightsquigarrow t$ path in G_f)

$\boxed{P \leftarrow \text{BREADTH-FIRST-SEARCH}(G_f).}$

$f \leftarrow \text{AUGMENT}(f, c, P).$

Update G_f.

RETURN f.

\# TOT AUG. =

$= n \cdot m$

TIME = $O(m^2 n)$