Algoritmi e Strutture Dati

Lezione 19

5 novembre 2025

Problema UNION-FIND

Rappresentare una collezione di insiemi disgiunti, con le operazioni:

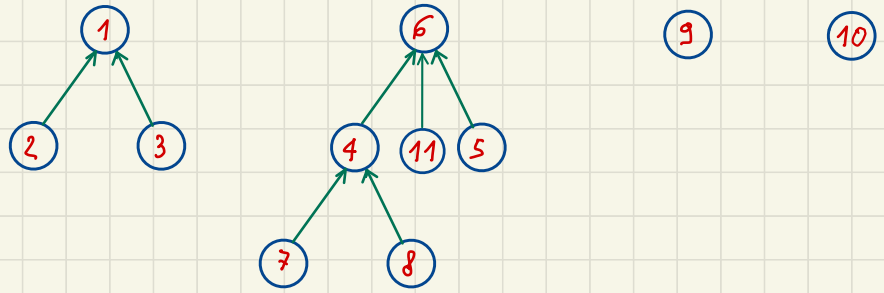
$\{\{1\}\ \{2,3,7\}\ \{9,5,6\}\}$

- UNION(A,B) — unisce gli insiemi A e B in un unico insieme di nome A
- FIND(x) — restituisce il nome dell'insieme che contiene l'elemento x
- MAKESET(x) — crea un nuovo insieme $\{x\}$ di nome x con x nuovo elemento

ALGORITMI "QuickUnion"

- ALBERI DI VARIE ALTEZZE
- Elementi dell'insieme ⟷ NODI
- Elemento che dà il nome all'insieme ⟷ RADICE

Esempio



{1, 2, 3} {7, 8, 4, 11, 5, 6} {9} {10}

"QuickUnion": operazioni.

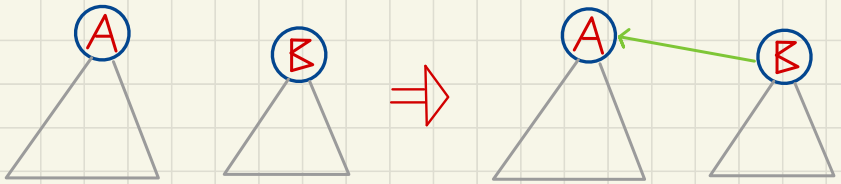
Tempo.

MAKESET (elemento e)

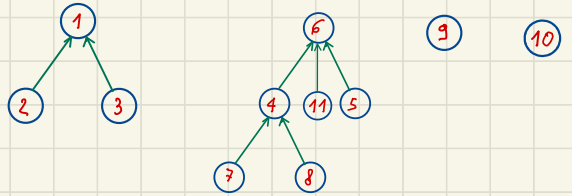
$T(n) = O(1)$

MAKESET (7)

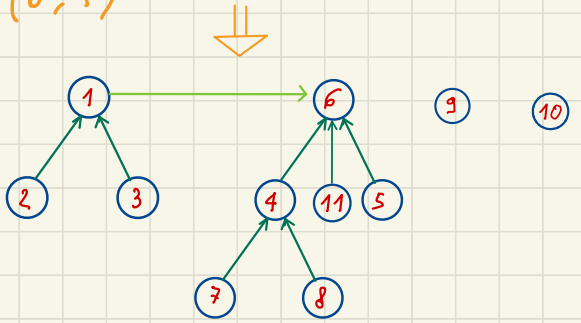
UNION (nome A, nome B)



$T(n) = O(1)$

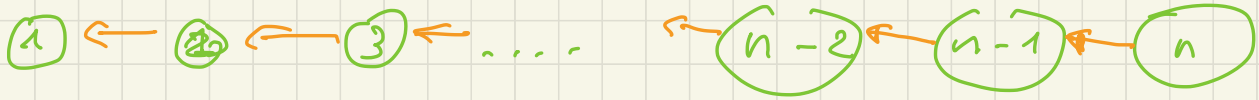


UNION (6, 1)



NOTA: albero "peggiore"

MAKESET(1), MAKESET(2), ..., MAKESET(n)



UNION (n-1, n)

UNION (n-2, n-1)

UNION (n-3, n-2)

⋮

UNION (1, 2)

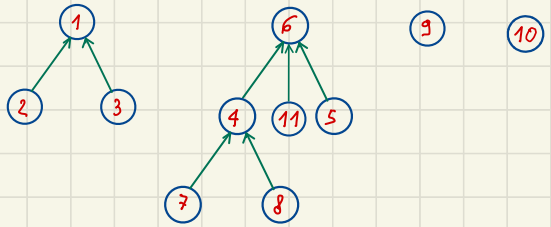
ALBERO di altezza n-1

"QuickUnion": operazioni

FIND (elemento e) → nome

risale da e
fino alla radice

FIND (8)



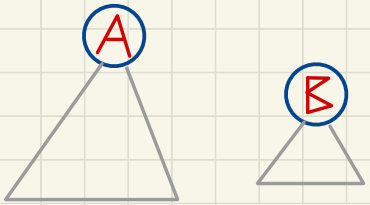
passi: altezza dell'albero

$$T(n) = O(n)$$

"QuickUnion" con UNION bilanciata

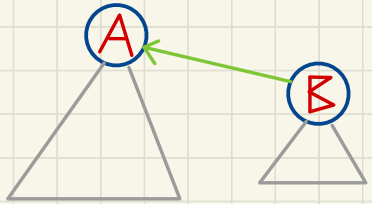
UNION: attacchiamo la radice dell'albero più basso sotto quella dell'albero più alto

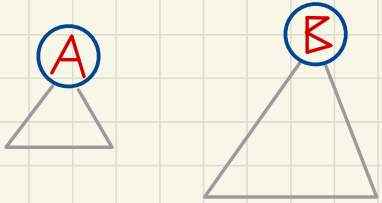
"rank" = altezza



rank(A) ≥ rank(B)

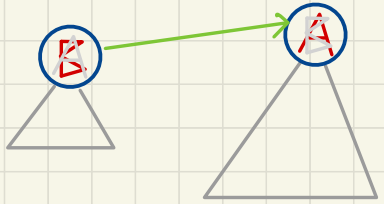
UNION (A,B)





rank(B) > rank(A)

UNION (A,B)



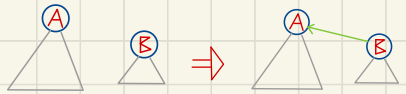
Scambio radici

UNION by rank

Nella radice si memorizza l'altezza dell'albero (rank)

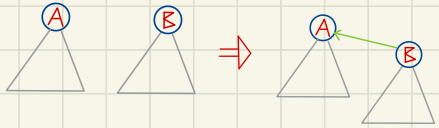
UNION (nome A, nome B)

IF rank(A) > rank(B) THEN
  rendi la radice di B figlia di quella di A



ELSE IF rank(B) > rank(A) THEN
  rendi la radice di A figlia di quella di B
  scambia i contenuti dei nodi A e B

ELSE
  rendi la radice di B figlia di quella di A
  rank(A) ← rank(A) + 1



$$T(n) \simeq O(1)$$

"QuickUnion" con UNION bilanciata: operazioni

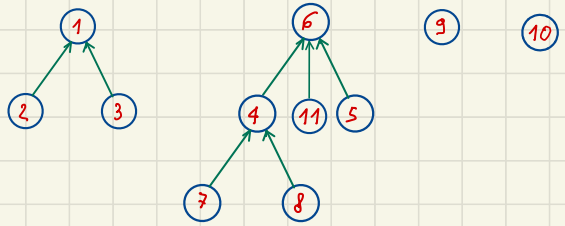
MAKESET (elemento e)

MAKESET (7)

occorre memorizzare nella radice che il rank è 0

FIND (elemento e) → nome

FIND (8)



- occorre risalire nell'albero
- il tempo dipende dall'altezza dell'albero

$T(n) = O$ (altezza max alberi)

Lemma Ogni albero "QuickUnion" costruito effettuando operazioni di UNION bilanciate contiene almeno $2^{rank(x)}$ nodi, x radice

$$\#nodi(x) \geq 2^{rank(x)}$$

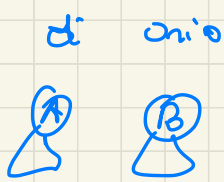
Dim Sia k il n° di operazioni di UNION con cui l'albero  è stato ottenuto

Induzione su k

Base $k = 0$  $\#nodi(x) = 1$ $rank(x) = 0$ $1 \geq 2^0$

Passo $< k \rightarrow k$

sia UNION (A, B) l'ultima op. di union con cui è stato ottenuto 

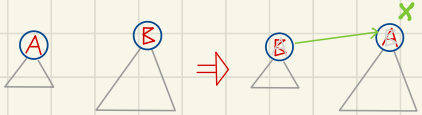
Gli alberi  sono stati ottenuti con meno di k union

$\Rightarrow$ ip. ind. $\#nodi(A) \geq 2^{rank(A)}$ $\#nodi(B) \geq 2^{rank(B)}$

Se $rank(A) > rank(B)$

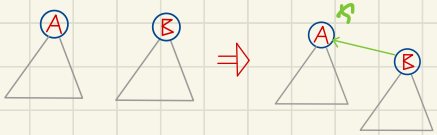
$rank(x) = rank(A)$

Se $rank(B) > rank(A)$



$rank(x) = rank(B)$

Se $rank(A) = rank(B)$



$rank(x) = rank(A) + 1$

$$\#nodi(x) = \#nodi(A) + \#nodi(B)$$
$$\geq \#nodi(A) \geq 2^{rank(A)} = 2^{rank(x)}$$
$$\Rightarrow \#nodi(x) \geq 2^{rank(x)}$$

come prima, mettendo B al posto di A

$$\#nodi(x) = \#nodi(A) + \#nodi(B)$$
$$\geq 2^{rank(A)} + 2^{rank(B)}$$
$$= 2 \cdot 2^{rank(A)} = 2^{rank(A)+1} = 2^{rank(x)}$$
$$\Rightarrow \#nodi(x) \geq 2^{rank(x)}$$

□

Lemma Ogni albero "QuickUnion" costruito effettuando operazioni di UNION bilanciate contiene almeno $2^{rank(x)}$ nodi, x radice

$$\#\text{nodi}(x) \geq 2^{rank(x)}$$

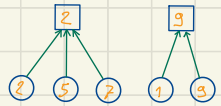
$$\Rightarrow \quad rank(x) \leq \log_2 \#\text{nodi}(x)$$

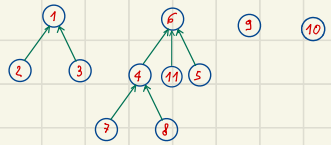
$\Rightarrow$ Se ci sono n elementi l'altezza è al max $\log_2 n$

Costo Find $T(n) = O(\log n)$

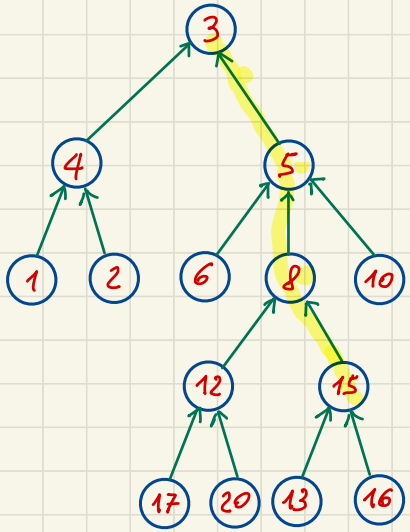
UNION-FIND riepilogo

	MAKESET	UNION	FIND
QuickFind	$O(1)$	$O(n)$	$O(1)$
QuickFind bilan.	$O(1)$	$O(\log n)$ amm	$O(1)$
QuickUnion	$O(1)$	$O(1)$	$O(n)$
QuickUnion bilan.	$O(1)$	$O(1)$	$O(\log n)$



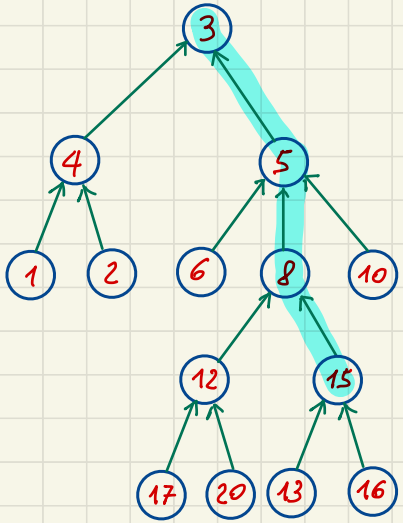


"QuickUnion" bilanciata con compressione di cammino

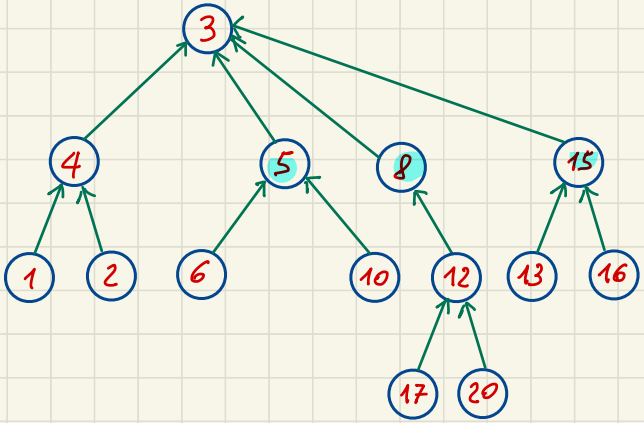


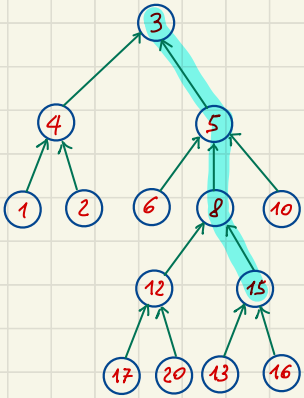
FIND(15)

"QuickUnion" bilanciata con compressione di cammino

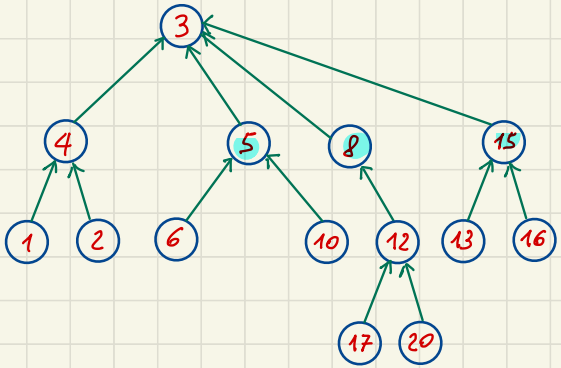


FIND(15)

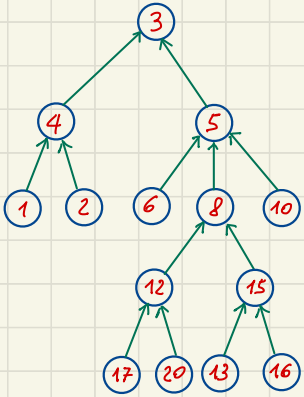




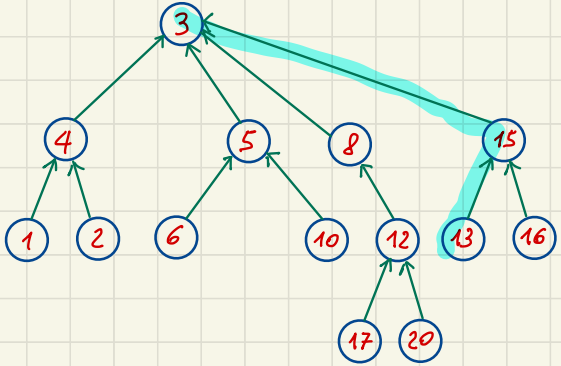
FIND(15)



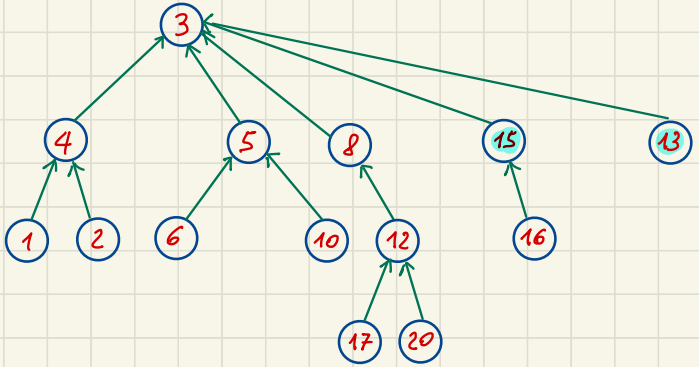
FIND(13)



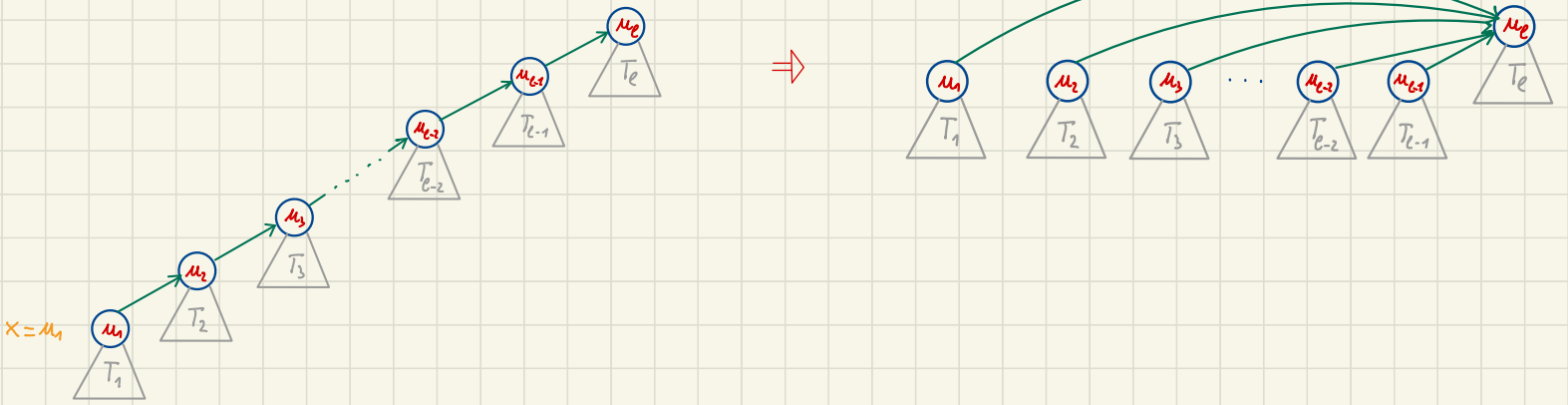
FIND(15)



FIND(13)



FIND(x) Compressione di cammino



"Quick Union" bilanciata con compressione di cammino

Si può dimostrare che effettuando una sequenza di

- n MAKESET
- $O(n)$ UNION e FIND

il tempo totale è $O(n \log^* n)$

$\Longrightarrow$ costo "ammortizzato" di FIND è $O(\log^* n)$

$$\log^{(i)} n = \underbrace{\log_2 \log_2 \dots \log_2}_{i \text{ volte}} n$$

$$\log^* n = \{ \min i \mid \log^{(i)} n \leq 1 \}$$

LOGARITMO ITERATO

$$\lg^{(i)} n = \underbrace{\lg_2 \lg_2 \dots \lg_2}_{i \text{ volte}} n$$

$$\lg^* n = \min \{ i \mid \lg^{(i)} n \leq 1 \}$$

$\lg^* 1 = 0$

$\lg^* 2 = 1$

$\lg^* 3 = 2$

$\lg^* 4 = 2$

$\lg^* 5 = 3$

$\vdots$

$\lg^* 16 = 3$

$\lg^* 17 = 4$

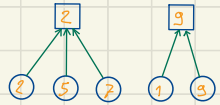
$\vdots$

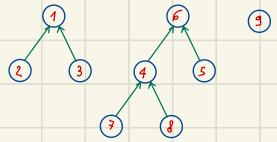
$\lg^* x = 4$ per $16 < x \leq 2^{16}$

$\lg^* x = 5$ per $2^{16} < x \leq 2^{2^{16}}$

UNION-FIND riepilogo

	MAKESET	UNION	FIND
QuickFind	$O(1)$	$O(n)$	$O(1)$
QuickFind bilan.	$O(1)$	$O(\lg n)$ amm	$O(1)$
QuickUnion	$O(1)$	$O(1)$	$O(n)$
QuickUnion bilan.	$O(1)$	$O(1)$	$O(\lg n)$
QuickUnion bilan. con compressione cammino	$O(1)$	$O(1)$	$O(\lg^* n)$





→ fn. che tende a infinito

↓

≤ 5 per valori usati nella realtà

Grafi

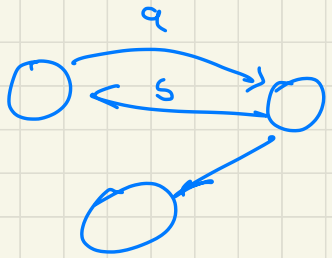
GRAFO $G = (V, E)$

V insieme finito NODI o VERTICI

$E \subseteq V \times V$ insieme ARCHI (LATI, SPIGOLI)

GRAFI — NON ORIENTATI — E simmetrica

\ ORIENTATI / DIRETTI



GRAFO $G = (V, E)$

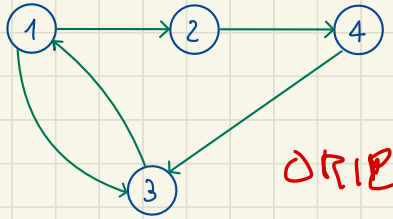
V insieme finito di NODI o VERTICI

$E \subseteq V \times V$ insieme di ARCHI (o LATI, SPIGOLI)

Esempi

$V = \{1, 2, 3, 4\}$

$E = \{(1,2), (1,3), (2,4), (4,3), (3,1)\}$

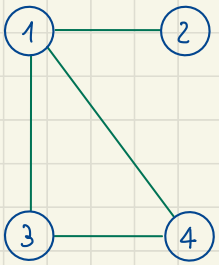


ORIENTATO

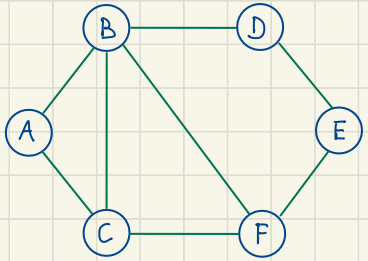
$V = \{1, 2, 3, 4\}$

$E = \{(1,2), (2,1), (1,3), (3,1), (3,4), (4,3), (1,4), (4,1)\}$

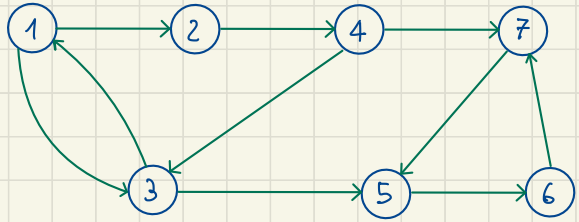
$\{1,2\}$ $\{1,3\}$ $\{3,4\}$ $\{1,4\}$



NON ORIENTATO



NON ORIENTATO



ORIENTATO

- $(x,y) \in E \Rightarrow (x,y)$ è INCIDENTE su x e su y

inoltre:

x e y sono ADIACENTI

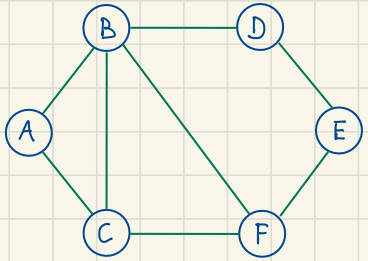
(x,y) esce da x, entra in y

y è ADIACENTE a x

- VICINI di $v \in V$: vertici adiacenti a v

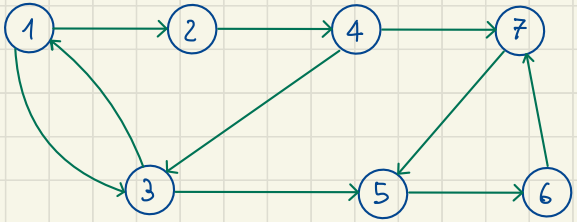
vicini B: A, C, D, F

vicini di 4: 7, 3



$\#V = n$

$\#E = m$



- GRADO $\delta(v)$ di un vertice v: # archi incidenti su v

grado di 1 : 3

Grado di B: 4

$\delta(B) = 4$

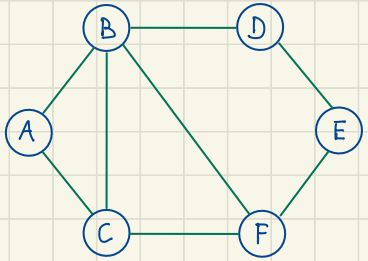
$\sum_v \delta(v) = 2m$

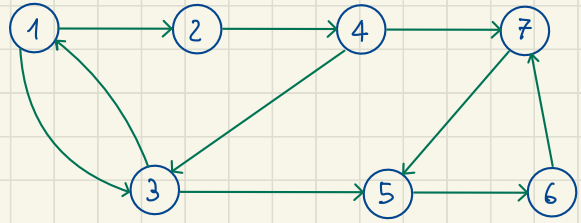
$\delta_{in}(v)$ grado in INGRESSO $\delta_{in}(1) = 1$

$\delta_{out}(v)$ grado in USCITA $\delta_{out}(1) = 2$

$\delta(v) = \delta_{in}(v) + \delta_{out}(v)$

$\sum_v \delta_{in}(v) = m = \sum_v \delta_{out}$





- **CAMMINO** da x a y $(x, y \in V)$:

  sequenza di vertici $v_0, v_1, \dots, v_k \in V$ t.c.

  $v_0 = x$, $v_k = y$ e $(v_{i-1}, v_i) \in E$ per $i = 1, \dots, k$

  1 2 4 3 5 6 7

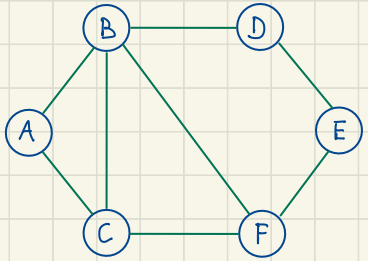
  cammino da 1 a 7

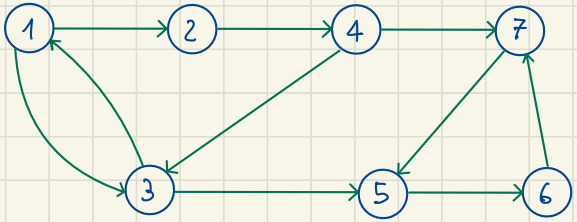
  - lunghezza del cammino = #archi (k)

  - cammino SEMPLICE: non contiene vertici ripetuti

- y è RAGGIUNGIBILE da x se $\exists$ cammino da x a y

  7 è raggiungibile da 1 / ma 1 non è raggiungibile da 7

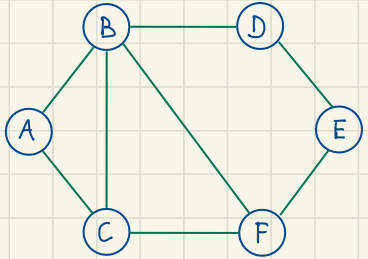


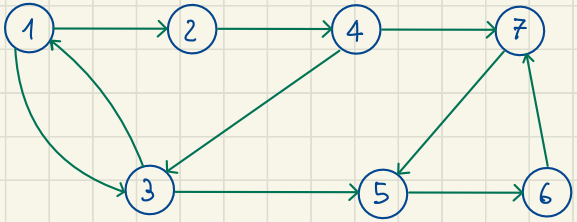


- CICLO: cammino da un vertice v a v stesso — 1, 3, 1 ; 1, 2, 4, 3, 1

  - ciclo SEMPLICE:

    solo il vertice iniziale è ripetuto (altrimenti) — 5, 6, 7, 5





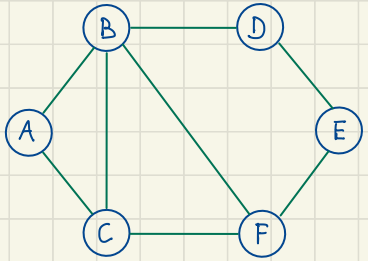
- CATENA tra x e y $(x, y \in V)$:

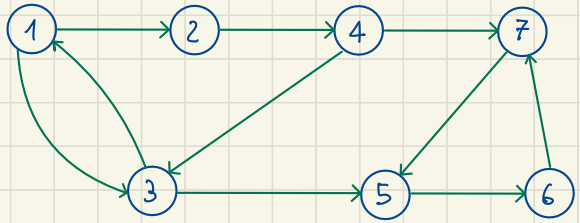
  sequenza di vertici $v_0, v_1, \dots, v_k \in V$ t.c.

  $v_0 = x$, $v_k = y$ e $\left((v_{i-1}, v_i) \in E \circ (v_i, v_{i-1}) \in E\right)$ per $i = 1, \dots, k$

  4, 2, 6 catena ma non cammino

- CIRCUITO: se $x = y$





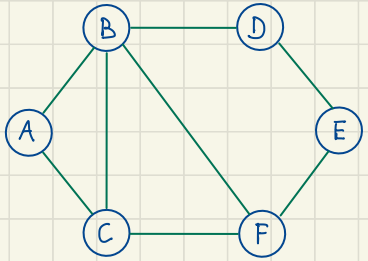
CONNESSO ma NON FORTEMENTE CONNESSO

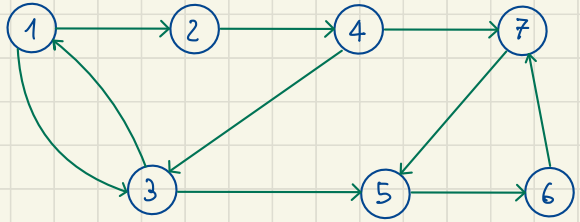
- GRAFO CONNESSO:

  tra ogni coppia di vertici esiste una catena.

- GRAFO FORTEMENTE CONNESSO:

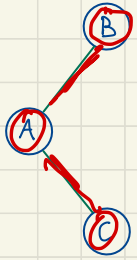
  tra ogni coppia di vertici esiste un cammino





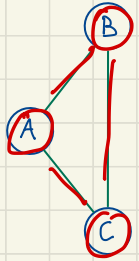
- $G' = (V', E')$ è un SOTTOGRAFO di $G = (V, E)$ quando

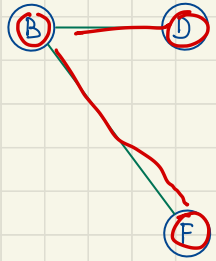
  $V' \subseteq V$ e $E' \subseteq E \cap (V' \times V')$

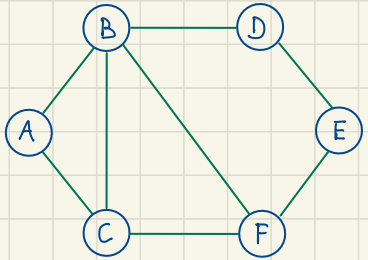


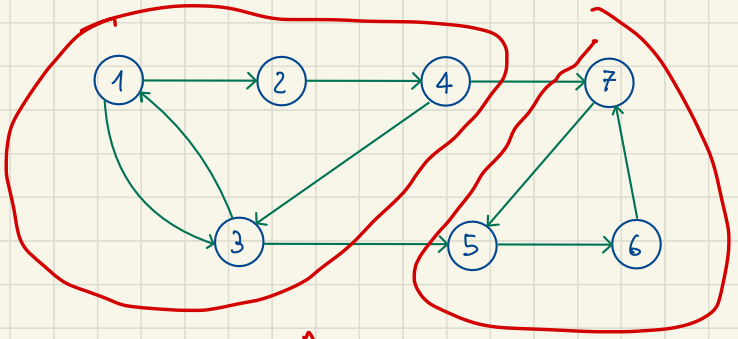
G' è il sottografo INDOTTO da V' quando

$E' = E \cap (V' \times V')$



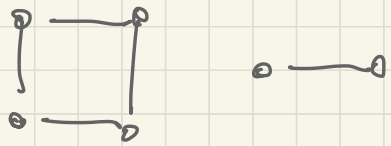


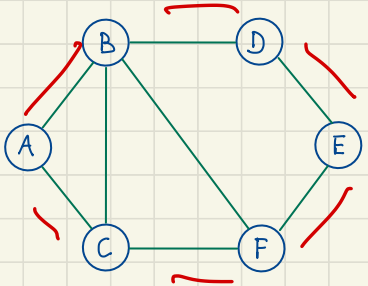




componenti fortemente connesse

- COMPONENTE FORTEMENTE CONNESSA:
  sottografo indotto fortemente connesso massimale

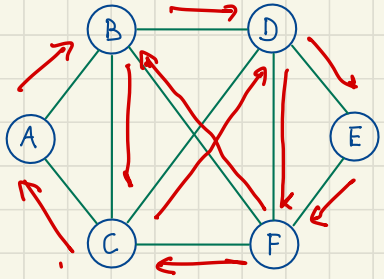




CIRCUITO HAMILTONIANO: (grafi non orientati)

circuito che passa per ogni vertice del grafo una e una sola volta

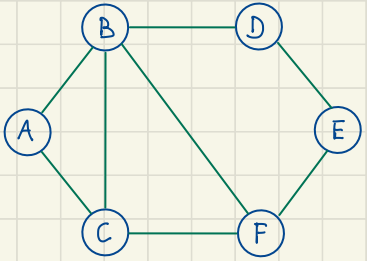
A, B, D, E, F, C



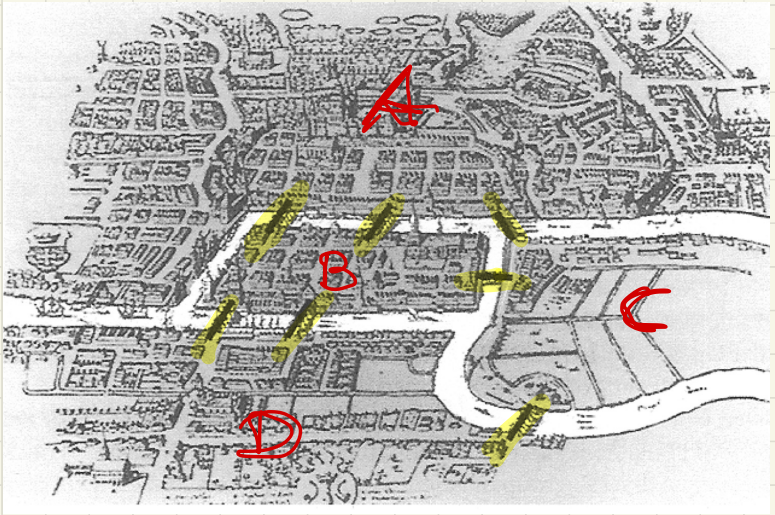
CIRCUITO EULERIANO

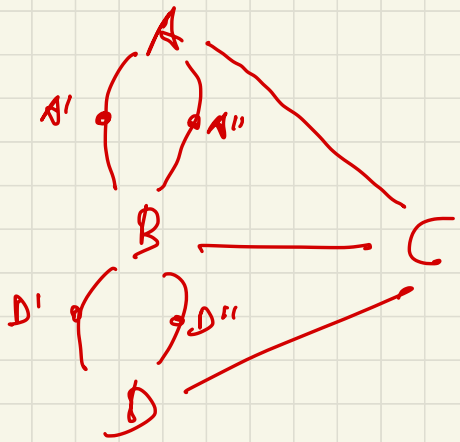
circuito che passa per ogni arco del grafo una e una sola volta

C, A, B, D, E, F, C, D, F, B, C



I PONTI DI KÖNIGSBERG





∃ circuito euleriano che attraversa tutti i ponti?

Teo ∃ un circuito euleriano sse $\forall v \; d(v)$ è pari