Grafi

Def. Un grafo è una coppia (V, E) dove $E \subseteq V \times V$ e V è un insieme. In altre parole è equivalente a una relazione binaria.

→ convenzionalm. $n = |V|$ è il numero di nodi (el. di V) e $m = |E|$* il num. di archi (el. di E).

→ alberi e array sono casi speciali di grafi.

Rappresentazione di grafi

(supponiamo $V = \{1, \dots, n\}$ o $V = \{0, \dots, n-1\}$)

a) MATRICE DI ADIACENZA

$$A = (\underbrace{[(i, j) \in E]}_{\text{1 se vero, 0 se falso}})_{ij}$$

→ richiede $\underline{\underline{n^2}}$ spazio

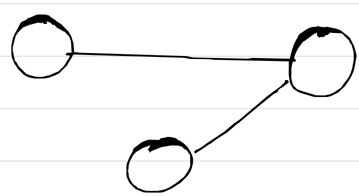
$$FS(x) = N(x) = \text{vicini}(x) = \{u \in V \mid (x, u) \in E\}$$

→ se $\text{grado}(x) = \deg(x) = |N(x)|$, allora $\text{grado}(x) =$ somma el. di A_x.

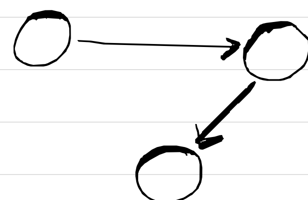
* nel caso di grafi orient. un arco si conta una sola volta.

Un grafo si dice non orientato se la sua matrice di adiac. è simmetrica, ossia sse $(i,j) \in E \iff (j,i) \in E$. In tal caso si può def. il grafo come coppia (V, E) con $E \subseteq \mathcal{P}(V)$ t.c. gli el. di E cont. al più due el.

Altrimenti si dice che il grafo è orientato.



non orientato



orientato

$\rightarrow$ si osserva che $E = \bigcup_{x \in V} \{x\} \times N(x) \Longrightarrow$

$\Longrightarrow |E| = \sum_{x \in V} \text{grado}(x)$ (se G è non or.

$2m = \sum_{x \in V} \text{grado}(x)$, altrim. $m = \sum_{x \in V} \text{gr}(x)$)

HANDSHAKING LEMMA

b) LISTE DI ADIACENZA

Possiamo rappr. G come una lista di vettori:

deg(0)	0	$\rightarrow$	N(0)
" (1)	1	$\rightarrow$	N(1)
" (2)	2	$\rightarrow$	N(2)
" (3)	3	$\rightarrow$	N(3)
" (4)	4	$\rightarrow$	N(4)
	⋮		

in qst. caso lo spazio occupato è

$$O(|V| + |E|)$$

$|V| \rightarrow$ 1ª lista, $|E| \rightarrow \sum_{x \in V} \underbrace{|N(x)|}_{deg(x)}$

in teoria dei grafi si dice LINEARE per intend. qst. complessità

$\rightsquigarrow$ in un grafo non orient. vale $0 \leq m \leq \binom{n}{2}$ (esclud. i "cappi")

$\rightarrow$ $d^+(x) = deg(x)$, mentre $d^-(x) =$
$= |BS(x)| = |\{u \in V \mid (u,x) \in E\}| =$
$=$ somma el. di A^x

$\rightarrow$ vale ancora $\sum_{x \in V} d^-(x) = |E|$.

⚠ Spesso a livello teorico si usano le matrici di adiac., ma nelle implement. si usano le liste di ad.

Cammini

Walk → cammino (enfasi sugli archi)
Path → cammino senza vertici ripetuti ("sui nodi")

Un **cammino** è una seq. finita di archi $(i_1, i_2), (i_2, i_3), \dots, (i_k, i_{k+1}), (i_{k+1}, i_{k+2})$ in cui la dest. dell'i-esimo arco è la sorg. dell'(i+1)-esimo. Si scrive più sempl. come $i_1 i_2 \dots i_{k+1} i_{k+2}$

→ la sorg. del camm. è la sorg. del primo arco, la dest. la dest. dell'ultimo arco.

Un **ciclo** è un camm. in cui sorg. e dest. coincidono.

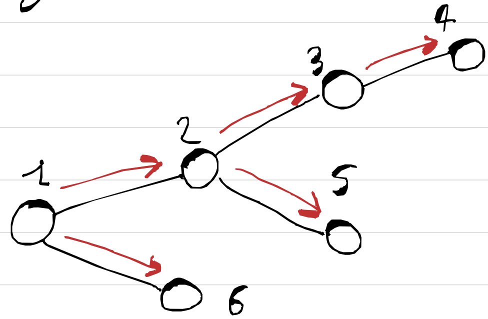
Il grafo si dice **CICLICO** se ammette un ciclo, aciclico altrimenti.

Il grafo si dice **CONNESSO** se $\forall a, b \in V$ esiste un cammino da a a b (ignorando l'orientam.).*

→ un albero è un grafo aciclico connesso (in un grafo aciclico deve esistere un nodo senza archi entranti).

* guardando anche all'orient. si parla di **FORTEMENTE CONNESSO**

Per studiare la connessione e la ciclicità si usa la DFS (depth-first search), che agisce usando gli stack.

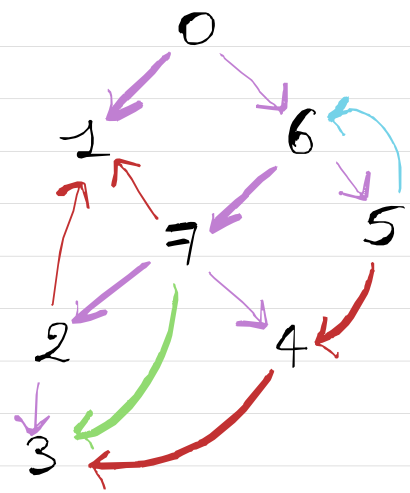


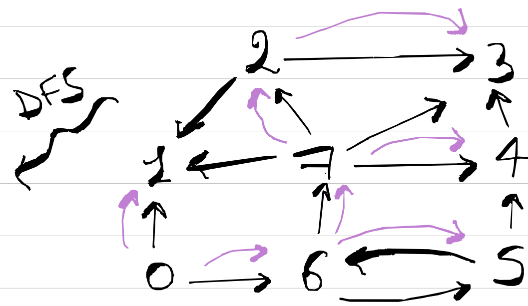
(si comporta come la visita anticipata)

→ se si inizia la DFS da x, la DFS trova tutti i nodi ragg da x. In un grafo non orient. trova la comp. fortemente connessa, in gen. si trova la comp. con. di x.

La DFS e' lineare ($O(|V|+|E|)$ e modella un albero detto DFS-tree, che permette di distinguere 4 tipi di archi sul grafo:

inizializz. dell'array booleano di raggiungibilità





Tipi di archi:

- archi del DFS-tree (•)
- archi forward, da antenato <u>non</u> genitore a discend. (•)
- archi backward, da disc. ad antenato (•)
- cross edge (trasversali), da archi visitati dopo ad archi visitati prima (•)

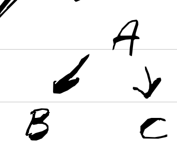
→ in grafi <u>non</u> orient. i forward si considerano backward e i cross o diventano forward o spariscono (<u>non</u> c'è più la dir. a bloccare la DFS).

→ per trovare le SCC (strongly-conn comp.) si usa una vers. più sofisticata della DFS.

Prop. G ciclico $\iff$ nella DFS esiste un arco backward.

($\Leftarrow$) ovvio

($\Rightarrow$) Sia $u_1 u_2 \dots u_k u_1$ un ciclo di G dove wlog u_1 è ragg. nella DFS (altrim. si trasla l'ordine). Allora (u_k, u_1) è backward dacché u_k sarà discend. di u_1. ■

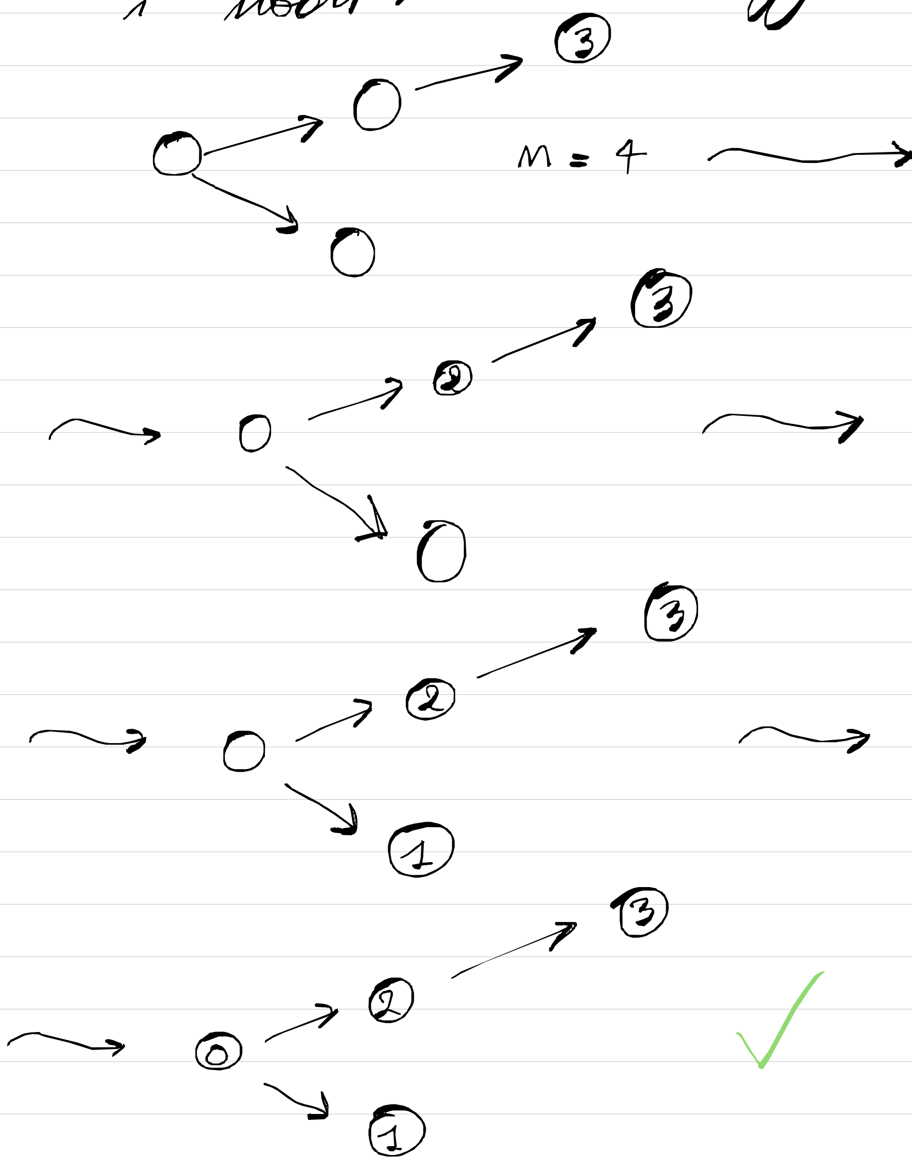
→ un albero può rappr. con delle parent. ben formate (e.g. $\underset{A\,B\,B\,C\,C\,A}{(()())}$ → )

DAG = directed acyclic graph

Sui DAG è possibile dare un ordinam. canonico sui nodi detto **TOPOLOGICO** t.c. $(i,j) \in E \Rightarrow i < j$, dove i e j sono le nuove etichette dei nodi. In altre parole $\exists \eta : V \to \{0, \dots, n-1\}$ t.c. $(i,j) \in E \Rightarrow \eta(i) < \eta(j)$.

È possibile trovare questo ordinamento sfruttando una DFS modificata:

- si arriva a un nodo senza archi uscenti con la DFS e gli si assegna η = contatore (cont parte da $n-1$) e lo si decrementa, lo si marca come ragg.
- si reitera fino a ragg. tutti i nodi



A livello pratico si può anche agire ponendo $\eta(r) = 0$ per un nodo qualsiasi senza archi entranti, considerando poi il sottogr. senza r e mettendo $\eta(r') = 1$ ad un'altra radice, ...

→ l'ord. top. sui DAG è utile per usare un velocissimo alg. SPT (shortest-path tree), ossia spt. acyclic