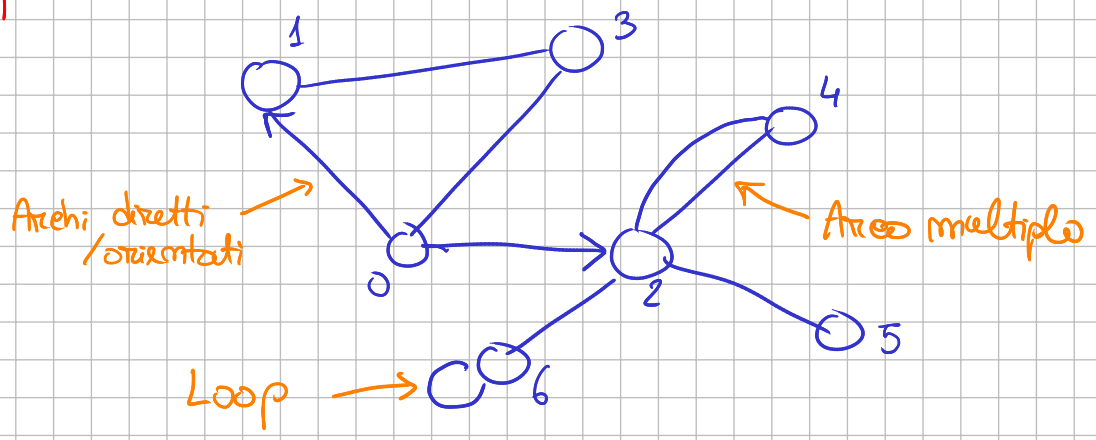
GRAFI



Definizione formale: Un grafo è $G = (V, E)$

V: insieme dei nodi (finito); E: insieme degli archi

$$E = \{ (u_i, v_i) : u_i, v_i \in V \}$$

(u_i, v_i): ordinate

No loop $\iff u_i \neq v_i \ \forall i$

Non orientato $\iff$ se c'è (u_i, v_i) c'è anche (v_i, u_i)

Rappresentazioni

Una rappresentazione è un modo di "codificare" il grafo.

Tre modi sensati:

(i) Lista di archi $\rightsquigarrow$ $\#$archi, elenco di coppie (vector<pair<int, int>>)

(ii) Matrice di adiacenza $\rightsquigarrow$ è una matrice M $|V| \times |V|$ dove

$$M_{ij} = \begin{cases} 0 & \text{se } i \not\to j \ \ ((i,j) \notin E) \\ 1 & \text{se } i \to j \ \ ((i,j) \in E) \end{cases}$$

M del pretest:

	0	1	2	3	4	5
0	0	0	1	1	0	1
1	0					
2	1					
3	1					
4	0					
5	1					

(simmetrica)

(iii) Liste di adiacenza $\rightsquigarrow$ per ogni nodo, manteniamo l'elenco dei suoi "vicini"

vector<vector<int>> Memoria $O(|E|)$

Esempio: Dato un grafo non orientato, gestire velocemente la seguente operazione:
Dato $v \in V$, rimuovere tutti gli archi che escono da v.

Osservazione: non basta cancellare la lista di adiacenza di v!
Se c'è $(v,u) \in E$, c'è anche (u,v)

Con vector<vector<...>> è lineare $\rightsquigarrow$ Usiamo dei vector<set<...>>

Esempio: Sia adj la struttura "liste di adiacenza" di G.
Dato $v \in V$, i, trovare j t.c. se adj[v][i] = u, allora adj[u][j] = v.

Si usano contemporaneamente lista di archi e liste di adiacenza.

per (u,v) tengo i e j vector<vector<pair<int, int>>>
(u ↗, idx ↑)

idx è l'indice di (u,v) nella lista di archi.

Esempio per casa: Dato un grafo G, etichettare i nodi da 1 a $N = |V|$ (permutazione) in modo da minimizzare
$\max_{v \in V} \{ \#\, u \text{ vicino di } v \text{ t.c. } label(u) > label(v) \}$

- in $O(M \log^2 N)$?
- in $O(M \log N)$?
- in $O(M)$?

Visite

Una visita è un modo per strutturare informazione relativa al grafo iterando sui nodi.

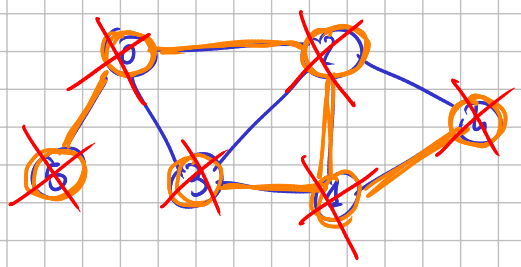
Localmente, è come disegnare il grafo.

Due tipi: in profondità (DFS), in ampiezza (BFS)

DFS: ricorsiva, greedy, FILO

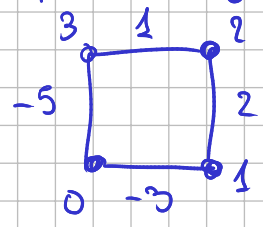
BFS: usa un approccio FIFO

DFS: apertura e chiusura.



Grafi pesati e cammini minimi

Un grafo è pesato quando ogni arco ha un numero (peso).



Peso totale = -5

← cicli negativi

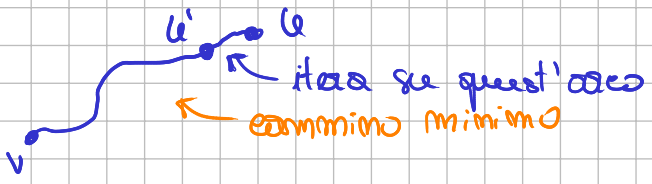
Come si calcola la distanza tra due nodi? Algoritmo di Dijkstra.

Dimostrazione di correttezza: per induzione

- Induzione su cosa? Sui nodi
- Cosa dimostro? Che Dijkstra calcola la distanza corretta.
- In che ordine itero sui nodi? In ordine di distanza

Passo base: dist(v) = 0 (è il primo) e Dijkstra calcola 0.

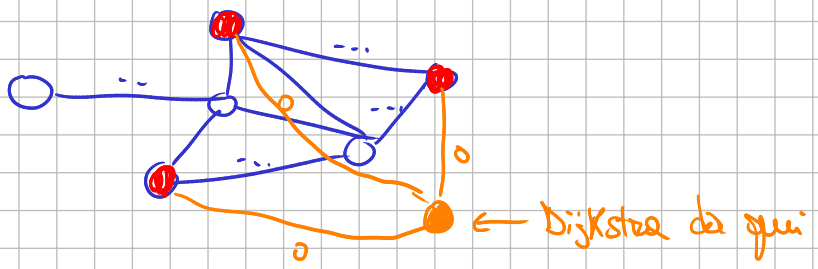
Passo induttivo: per ipotesi induttiva, Dijkstra calcola correttamente dist(u') per ogni u' tale che dist(u') < dist(u).



Fatto: Una BFS è Dijkstra con pesi = 1.

Fatto: Dijkstra è:
- un algoritmo su grafi
- greedy in natura
- una DP (stati?)

Sorgenti multiple:



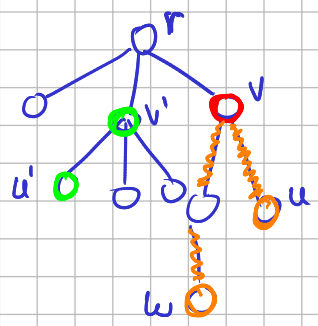
Di fatto, stiamo facendo Dijkstra inserendo all'inizio tutte le sorgenti.

Alberi & affini

Un albero è un grafo non orientato tale che:

- |E| = |V| - 1 ed è connesso; oppure
- esiste esattamente un cammino semplice tra ogni coppia di nodi; oppure
- |E| = |V| - 1 e non ha cicli; oppure
- è connesso e non ha cicli.

Un albero si può radicare.
Cioè si può scegliere un nodo e chiamarlo "radice".

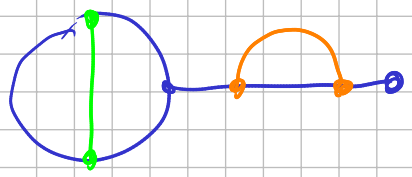


u è figlio di v
v è padre di u

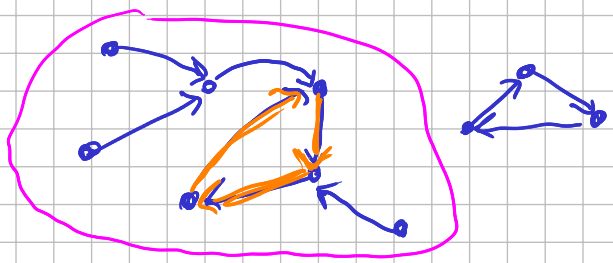
Discendenti di v = { v, figli di v, figli di figli di v, ... }
Antenati di v = { v, padre di v, padre di padre di v, ... }
LCA (u, v) = unico nodo x che è antenato sia di u sia di v e ha profondità minima
Lowest common ancestor

Cosa succede aggiungendo un arco a un albero?
- Non è più un albero perché $|E| = |V| \neq |V| - 1$
- È connesso $\Rightarrow$ ha un ciclo ed è unico

⚠ Aggiungendo un arco a un grafo con $K \geq 1$ cicli si forma almeno un ciclo, ma possibilmente anche di più

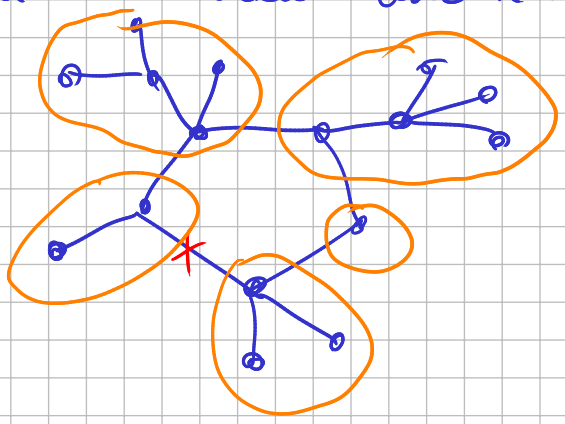


Grafo funzionale: È un grafo diretto in cui ogni nodo ha esattamente un arco uscente.



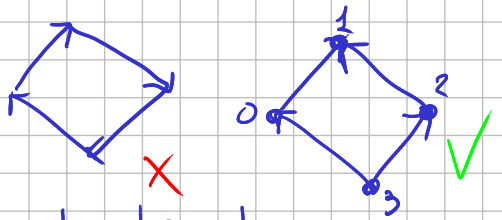
Osservazione: ogni componente connessa ha esattamente un ciclo, e partendo da un qualunque nodo e "camminando" sul grafo si arriva sempre a un ciclo.

Prendiamo una componente connessa e togliamo le orientazioni.
Cosa otteniamo? Un albero + archi!
Visione alternativa: un albero + archi è un "cielo di alberi":



DAG e ordine topologico

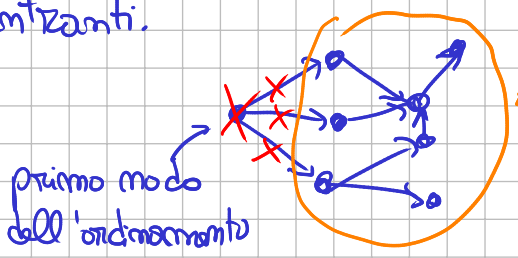
Un DAG è un grafo diretto aciclico.



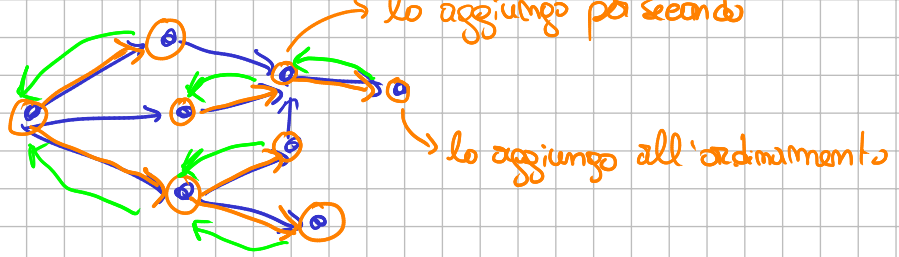
(topsort)
Un ordinamento topologico è un ordinamento dei nodi $V_0, V_1, \dots, V_{N-1}$ tale che non ci sono archi $V_i \to V_j$ per $i > j$.

(1) Esiste sempre in un DAG?
(2) Se sì, come si trova?
(3) È unico? NO!

(1) Sì! Fatto cruciale: esiste sempre una sorgente, cioè un nodo senza archi entranti.



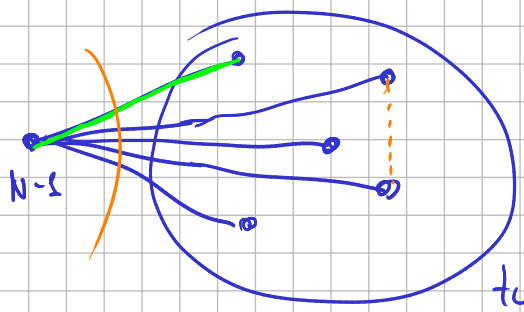
(2)



Problema "Game"

Costruisco una strategia per induzione.

- Passo base: 1 nodo $\rightarrow$ nessuna query
- Passo induttivo: N-1 nodi $\rightarrow$ N nodi



Due casi:

- dentro il grafo $\{0, \dots, N-1\}$, uso la strategia precedente
- per gli archi $(N-1) \rightarrow v$, aggiungo solo l'ultimo che viene chiesto

Di fatto: Mantengo contatori c_i per ogni nodo.

All'inizio $c_i = i$.

Quando arriva la query (u, v) con $u < v$:

- aggiungo l'arco se e solo se $c_v = 1$
- decremento c_v di 1.