Grafi:

I grafi sono delle strutture dati date da un insieme di nodi V connessi da archi raggruppati in un insieme.

Un grafo si definisce come:

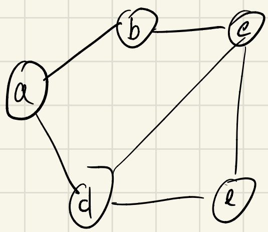
$$G = (V, E);$$

Tipi di Grafi:

Grafo Orientato = gli archi hanno una direzione;

Grafo Non Orientato = gli archi NON hanno una direzione.

Grafo sparso = Grafo con il numero di archi MOLTO MINORE rispetto al quadrato dei vertici (gli archi si collegano solo ad ALCUNI nodi) $|E| < |V^2|$



Grafo denso: grafo con il numero di Archi MOLTO ELEVATO, circa pari al quadrato dei vertici (quindi gli archi collegano molti vertici).

Adiacenza dei Nodi

Un nodo si dice Adiacente se esiste un Arco che li collega.

La presenza (o meno) di un orientamento è FONDAMENTALE per controllare L'ADIACENZA DEI NODI.

- Se l'arco NON è ORIENTATO allora i due archi sono vicendevolmente adiacenti.

Come rappresentare un grafo:

- lista di adiacenza = contenente una lista di VERTICI ADIACENTI al vertice preso in considerazione, può essere infinita.
- matrice di adiacenza = contenente 0 e 1 a seconda della adiacenza (1) o non adiacenza (0).

In caso di grafo denso (MATRICE ADIACENZA)

In caso di grafo sparso (LISTA DI ADIACENZA)

- la scelta si base sull'occupazione della memoria (per la matrice l'occupazione della memoria è fissa, non conviene utilizzarla per un grafo con pochi ARCHI)

- La scelta ricade anche sulla dimensione del grafo:
  - se è di dimensione ridotta è preferibile utilizzare matrice di adiacenza.
  - La lista di adiacenza richiede più tempo per l'accesso ad un elemento perché occorre scorrere tutta la lista.

Se il grafo NON È ORIENTATO la struttura dati non è simmetrica.

Lista di adiacenza:

Elementi indispensabili per la lista:

- il vertice del grafo
- vertice adiacenti

// Complessità $\Theta(|V|+|E|)$

Matrice di Adiacenza:

Gli elementi indispensabili per la matrice di adiacenza sono:

- Vertex <T> con attributo:
  - T Key
- Grafo con matrice di adiacenza Graph <T> che ha gli attributi:
  - maxSize;
  - Array di puntatori a vertici (Vertex <T>** vertices);
  - matrice di adiacenza (bool** adj);
  - numero di vertici attuali (nVertices);

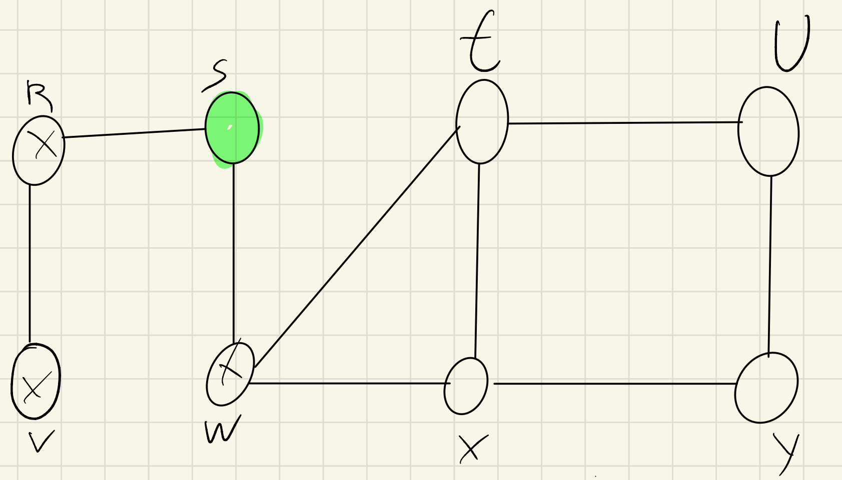
// Complessità $\Theta(|V|)^2$

BFS (visita Ampiezza)

Questo tipo di visita è definita IN AMPIEZZA perché procede per DISTANZA DAL NODO SORGENTE. Quindi al passo K avró scoperto i nodi a distanza K dalla sorgente.
È un algoritmo utilizzato per visitare tutti i nodi di un grafo, partendo da un nodo sorgente e visitando prima tutti i nodi a distanza 1 dal nodo sorgente, poi quelli a distanza, e cosí via.

Puó essere implementato utilizzando una coda per tenere traccia dei nodi da visitare.
Inizialmente vi è un puntatore che punta al (nella coda) nodo sorgente, il nodo sorgente viene inserito nella coda, verifichiamo se vi sono nodi adiacenti, aggiungo i nodi adiacenti alla coda
Se il nodo non ha piú nodi adiacenti lo elimino e faccio avanzare il mio puntatore effettuó il controllo di nodi adiacenti con il nodo successore e aggiungo i nodi alla coda.

esempio



NIL	NIL	NIL	NIL	NIL	NIL	NIL

Scelgo S nodo Sorgente, lo salvo nella CODA

S	//	//	//	//	//	//

↑

Quali sono distanti 1? R-W, li salvo nella coda
li coloro di grigio

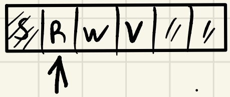
S	R	W	//	//	//	//

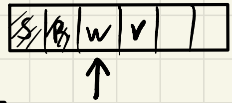
↑

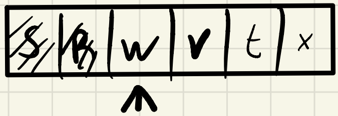
S non ha piú nodi adiacenti perció lo elimino dalla coda e faccio avanzare il puntatore

~~S~~	R	W	//	//	/

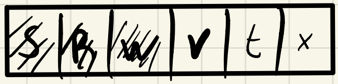
↑

R che nodi ha adiacenti? solo v, inserisco 

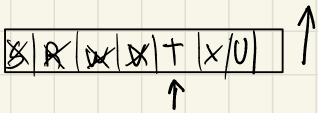
R non ha più nodi adiacenti, lo elimino e incremento il puntatore 



Ora siamo a w che nodi ha adiacenti? T e X li inserisco

W non ha altri adiacenti lo elimino dalla coda e avanza il puntatore 

V non ha adiacenti lo elimino e avanza il puntatore 

t ha nodi adiacenti? si U lo aggiungo 

t non ha altri nodi adiacenti lo elimino e avanza (Così via per tutti gli elementi del grafo)

Quando il valore risulterà NULL avremmo finito.

Complessità INIZIALIZZAZIONE: $O(|V|)$

Complessità VISITA: $O(|E|)$

Complessità Tot: $O(|E|) + O(|V|)$

DFS ➡ Vista dei Grafi in profondità

Questo tipo di visita è definita dalla lista (o ARRAY) di nodi ADIACENTI.
Cioé prima si visitano tutti i nodi adiacenti e poi si passa ai vertici successivi della stessa DISTANZA.

Nella visita in profondità non si ha un NODO SORGENTE, È semplicemente considerato tutto il GRAFO.

Procedimento:
La visita è da intendere come una FUNZIONE RICORSIVA poiché
se il nodo ha nodi adiacenti BIANCHI allora procedo con la visita,
altrimenti la visita termina e il nodo diventa tutto nero.

Significato dei colori:
BIANCO: COLORE DEI NODI NON ANCORA SCOPERTI (All'inizio tutti i nodi sono BIANCHI)

GRIGIO: Colore dei NODI PARZIALMENTE SCOPERTI.

NERO: È il colore dei nodi scoperti

Ordinamento topologico:
L'ordinamento topologico su un grafo ORIENTATO Aciclico lo si ha quando si fa un ordinamento in modo decrescente rispetto al tempo finale della visita di ogni vertice.

Cosa ci Serve?

Array Discovery d: Contiene il tempo di scoperta di un nodo) con solo valori infiniti.

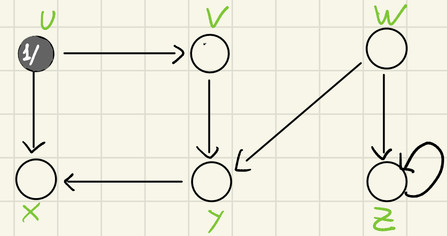
Array Completamento F: Contenente il tempo di completamento della visita) cioé il passaggio a NERO.

Coda dei predecessori: (Contenente i predecessori dei nodi considerati in quell'istante)

VARIABILE time: incrementata di volta in volta a dovere, (grazie a questo si definisce F)

tutti i nodi di colore BIANCO

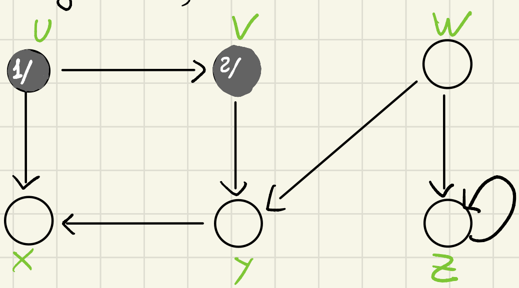
Svolgimento:



Time
0
1

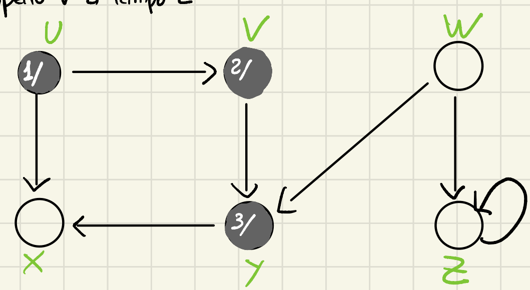
Considero u e lo coloro di grigio
Incremento time a 1
Discovery di u è 1

Considero gli adiacenti, essendo orientato è v



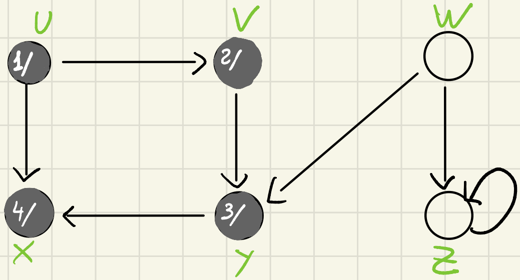
Time
0
1
2

v lo coloro di grigio
Incremento Time:
Scoperto v a tempo 2



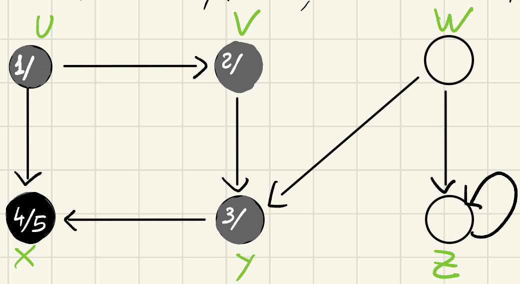
Time
0
1
2
3

Considero y lo coloro, incremento time
Y visitato a tempo 3



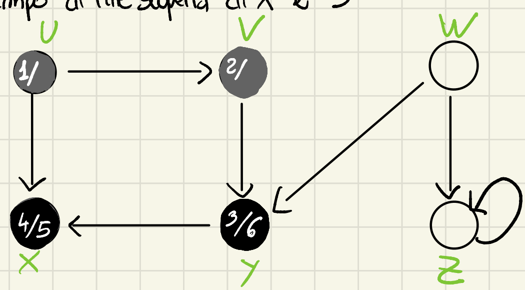
Time
0 4
1
2
3

Passo a x lo coloro, time++, avvenuto il Discovery in tempo 4



Time
0 4
1 5
2
3

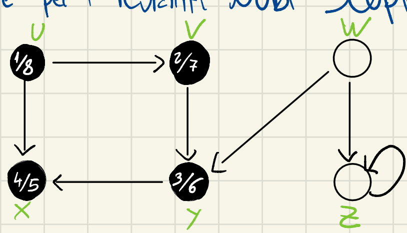
X ha nodi bianchi? No!
Incremento time e coloro X di nero
il tempo di fine scoperta di X è 5



Time
0 4
1 5
2 6
3

Torno indietro mediante i predecessori
Gli Adiacenti di y sono BIANCHI? No
time++ coloro di NERO tempo fine Scoperta avvenuto in 6

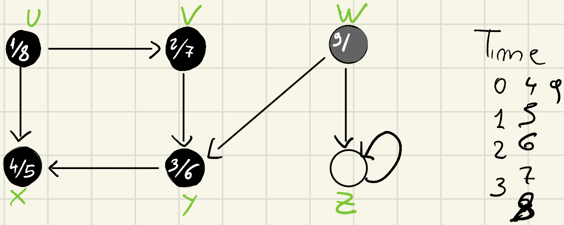
Analogamente per i restanti NODI scopro così che: V = 2/7, U = 1/8



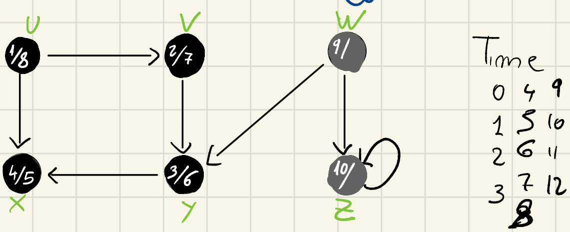
Time
0 4
1 5
2 6
3 7
8

Tornando indietro con i predecessori ho NULL, ho finito il Grafo? NO

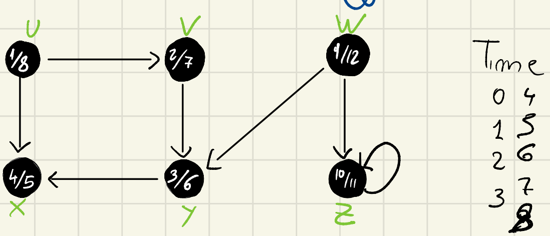
Procedo con l'altro lato del grafo:



Controllo w, incremento time++ e
lo coloro di grigio, ha Adiacenti Bianchi? Si



Controllo z lo coloro di grigio, time++, Discovered 10
Ha adiacenti BIANCHI? NO!
time++ e lo coloro di NERO! diventa 11
Con i predecessori torno indietro
Controllo w non ha bianchi, time++, Discovered 12



Conclusione Ho Finito i nodi del Grafo?
Si, procedura terminata.

Complessità $O(|V|) + O(|E|)$