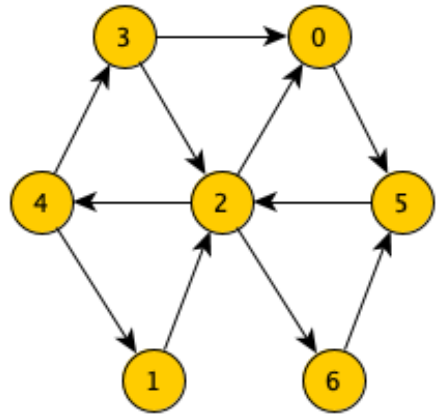
Esercizio appello straordinario aprile 2020

Si consideri il grafo diretto G rappresentato in figura:

$G =$ 

Assumiamo che G sia rappresentato tramite liste di adiacenza nelle quali i nodi compaiono in ordine crescente. Questo significa che in caso di visita del grafo nel caso un nodo abbia più nodi adiacenti da visitare viene sempre scelto quello di indice minimo.

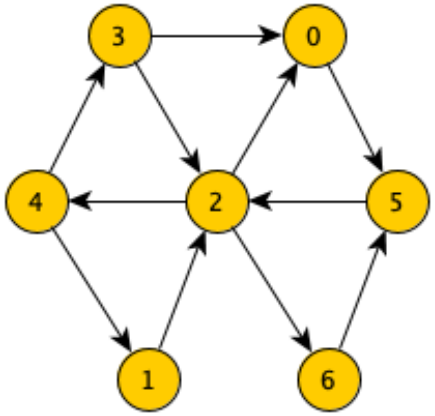
1. Considerate una visita in profondità che parte dal nodo 2 di G.
   - Riportate nell'ordine i nodi di G che vengono effettivamente visitati.
   - Individuate gli archi in avanti, gli archi all'indietro e gli archi di attraversamento che si incontrano durante la visita.
2. Considerate una visita in ampiezza che parte dal nodo 2 di G.
   - Riportate nell'ordine i nodi di G che vengono effettivamente visitati.
3. Qual è il numero minimo di archi da eliminare da G perché il grafo risulti avere ordinamenti topologici e quali sono questi archi? **Motivare bene la risposta.**
4. Eliminate da G gli archi ottenuti al punto precedente in modo che il grafo G' ottenuto risulti avere ordinamenti topologici.
   - quanti e quali sono gli ordinamenti topologici del grafo G'? **Motivare bene la risposta.**
   - Individuate l'ordinamento topologico che si ottiene applicando a G' l'algoritmo visto a lezione basato sulle sorgenti (in caso siano disponibili più sorgenti assumete che l'algoritmo scelga sempre quella di indice minimo).
   - Individuate l'ordinamento topologico che si ottiene applicando a G' l'algoritmo visto a lezione basato sulla visita in profondità (In caso siano disponibili più nodi da visitare assumete che l'algoritmo scelga sempre quello di indice minimo).
5. Eliminate le direzioni dagli archi del grafo G in modo da ottenere un grafo G" non diretto. Determinate i ponti del grafo G". **Motivare bene la vostra risposta.**

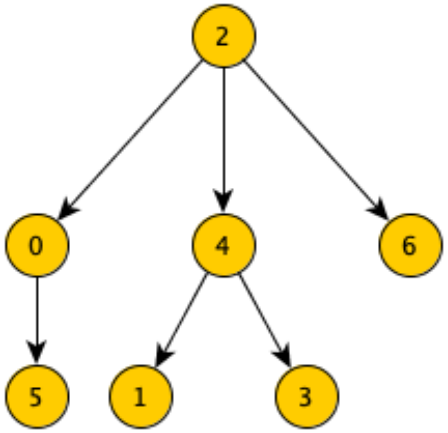
Assumiamo che G sia rappresentato tramite liste di adiacenza nelle quali i nodi compaiono in ordine crescente. Questo significa che in caso di visita del grafo nel caso un nodo abbia più nodi adiacenti da visitare viene sempre scelto quello di indice minimo.

1 Considerate una visita in profondità che parte dal nodo 2 di G.

- Riportate nell'ordine i nodi di G che vengono effettivamente visitati.
- Individuate gli archi in avanti, gli archi all'indietro e gli archi di attraversamento che si incontrano durante la visita.

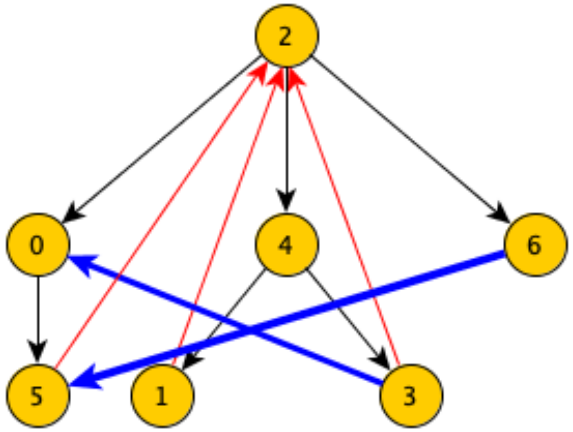
$G =$





Albero di visita DFS

-I nodi visitati sono nell'ordine : 2,0,5,4,1,3,6



In blu archi di attraversamento in rosso quelli all'indietro

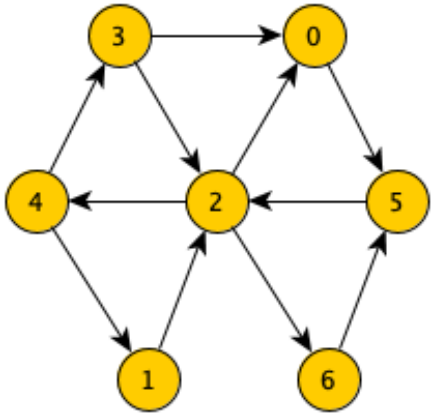
- *archi in avanti* : { }
- *archi all'indietro* : {(5,2), (1,2), (3,2)}
- *archi di attraversamento* : {(3,0), (6,5)}

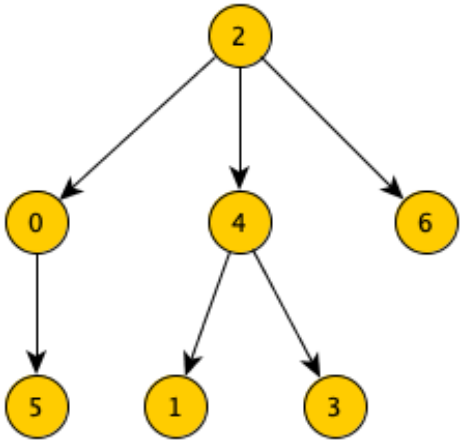
Assumiamo che G sia rappresentato tramite liste di adiacenza nelle quali i nodi compaiono in ordine crescente. Questo significa che in caso di visita del grafo nel caso un nodo abbia più nodi adiacenti da visitare viene sempre scelto quello di indice minimo.

2 Considerate una visita in ampiezza che parte dal nodo 2 di G.

- Riportate nell'ordine i nodi di G che vengono effettivamente visitati.

$G =$



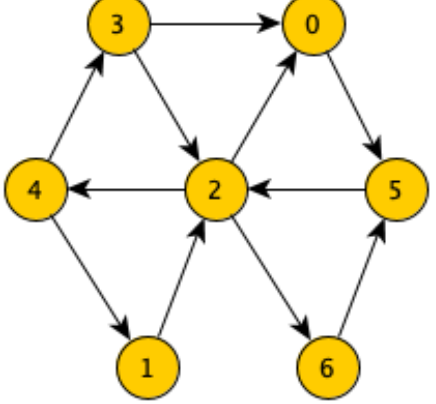


Albero di visita BFS

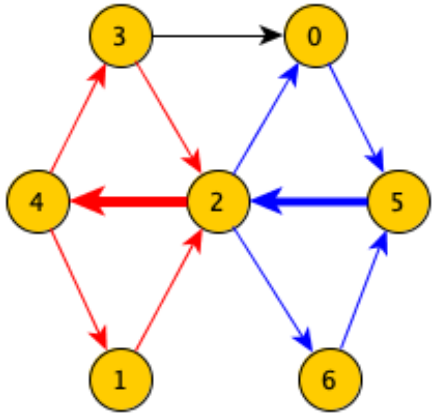
-I nodi visitati sono nell'ordine : 2,0,4,6,5,1,3

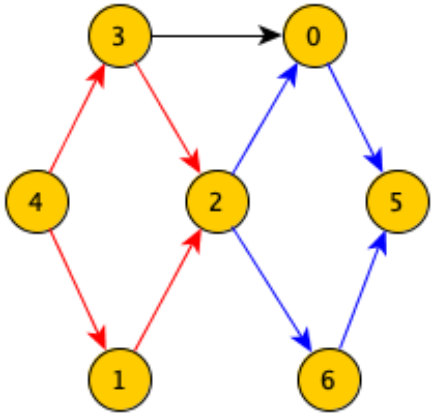
3 Qual è il numero minimo di archi da eliminare da G perché il grafo risulti avere ordinamenti topologici e quali sono questi archi? **Motivare bene la risposta.**

$G =$



- il grafo per avere sort topologici deve essere aciclico devo dunque rompere tutti i cicli presenti in G.
- ho due insiemi disgiunti di archi (in figura archi rossi e archi blu) al cui interno sono presenti cicli
- devo togliere almeno un arco rosso ed almeno un arco blu. (bisogna togliere quindi almeno due archi)
- se tolgo l'arco rosso $(2, 4)$ rompo tutti i cicli nella zona rossa e se tolgo l'arco $(5, 2)$ rompo tutti i cicli della zona blu)
- gli archi da eliminare sono dunque $(2, 4)$ e $(5, 2)$).

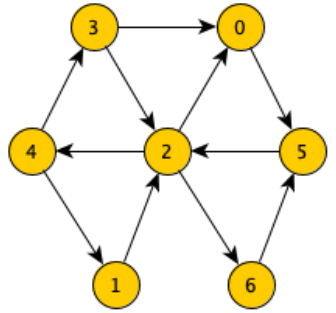




4a Eliminate da G gli archi ottenuti al punto precedente in modo che il grafo G' ottenuto risulti avere ordinamenti topologici.

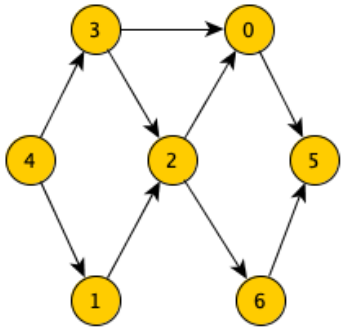
- quanti e quali sono gli ordinamenti topologici del grafo G'? **Motivare bene la risposta.**

$G =$



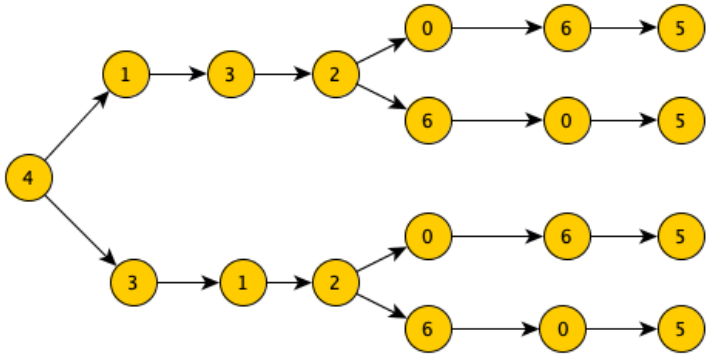
Eliminando gli archi $(2,4)$ **e** $(5,2)$ **da G si ottiene il grafo:**

$G' =$



Nota che:

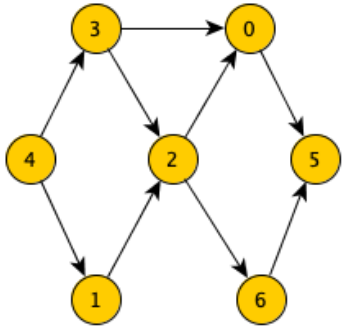
- i sort topologici iniziano tutti con il nodo 4 (è l'unico senza archi entranti)
- al secondo posto deve esserci uno dei due nodi 1 e 3 (sono gli unici senza archi entranti tolto il 4)
- solo dopo aver sistemato l'1 e il 3 c'è un nuovo nodo senza archi entranti (il 2) e solo dopo aver sistemato il 2 ho due nodi tra cui scegliere: lo 0 e il 6
- i possibili ordinamenti topologici sono dunque 4

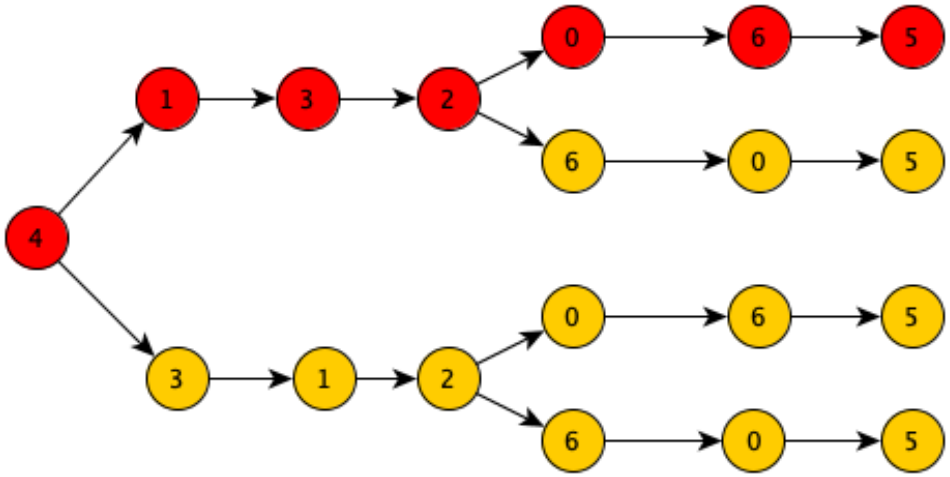


4b Eliminate da G gli archi ottenuti al punto precedente in modo che il grafo G' ottenuto risulti avere ordinamenti topologici.

- Individuate l'ordinamento topologico che si ottiene applicando a G' l'algoritmo visto a lezione basato sulle sorgenti (in caso siano disponibili più sorgenti assumete che l'algoritmo scelga sempre quella di indice minimo).

$G' =$





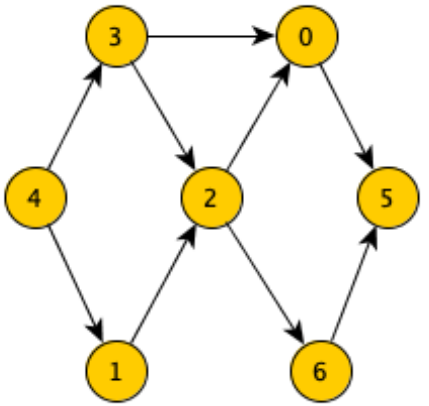
L'ordinamento topologico prodotto dall'algoritmo delle sorgenti è 4,1,3,2,0,6,5

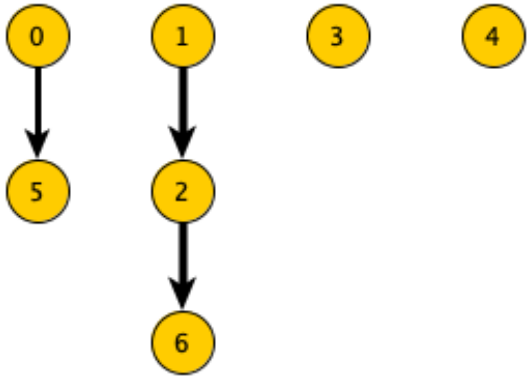
Assumiamo che G sia rappresentato tramite liste di adiacenza nelle quali i nodi compaiono in ordine crescente. Questo significa che in caso di visita del grafo nel caso un nodo abbia più nodi adiacenti da visitare viene sempre scelto quello di indice minimo.

4b Eliminate da G gli archi ottenuti al punto precedente in modo che il grafo G' ottenuto risulti avere ordinamenti topologici.

- Individuate l'ordinamento topologico che si ottiene applicando a G' l'algoritmo visto a lezione basato sulla visita in profondità (In caso siano disponibili più nodi da visitare assumete che l'algoritmo scelga sempre quello di indice minimo).

$G' =$



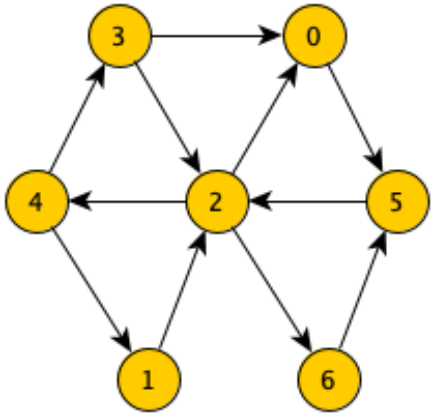


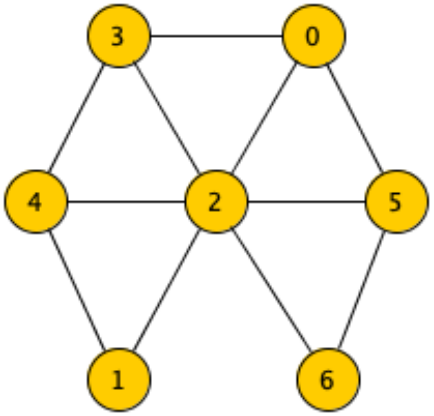
- La visita dell'intero grafo richiederà 4 diverse DFS:

- inserendo in coda ad una lista i nodi man mano che termina la loro visita si ha: [5,0,6,2,1,3,4]

- L'ordinamento topologico è dato dal riverse dei nodi in lista: 4,3,1,2,6,0,5

5 Eliminate le direzioni dagli archi del grafo G in modo da ottenere un grafo G" non diretto. Determinate i ponti del grafo G". **Motivare bene la vostra risposta.**

$G =$ 

$G'' =$ 

- un ponte è un arco la cui rimozione sconnette il grafo.
- tutti gli archi di G'' sono all'interno di cicli
- eliminare un arco di un ciclo non sconnette il grafo (esiste sempre almeno un altro cammino che unisce gli estremi dell'arco eliminato)
- nel grafo G'' non ci sono ponti.

ESERCIZIO 2: Vogliamo il numero di sequenze decimali di lunghezza n in cui non appaiono cifre pari adiacenti. Progettare un algoritmo che prende come parametro l'intero n e, in tempo $O(n)$, restituisce il numero delle sequenze cui siamo interessati.

Ad esempio:

- per $n = 1$ la risposta dell'algoritmo deve essere 10
- per $n = 2$ la risposta dell'algoritmo deve essere 75

Definisco la tabella T di $n + 1$ locazioni.

- $T[i]$ = il numero di sequenza decimali composte da i cifre senza cifre pari adiacenti.
- la soluzione al problema la troveremo in $T[n]$

Serve ora una regola che permetta di calcolare efficientemente le varie celle della tabella

La regola ricorsiva per calcolare il valore della cella $T[i]$ in funzione delle precedenti celle già calcolate è

- $T[i] = \begin{cases} 10 & \text{se i=1} \\ 75 & \text{se i=2} \\ 5 \cdot T[i-1] + 25 \cdot T[i-2] & \text{se } i \geq 3 \end{cases}$

Spiegazione:

- se $i = 1$ tutte le 10 cifre decimali vanno bene
- se $i = 2$ dalle 100 sequenze decimali di 2 cifre devo togliere le $5 \cdot 5 = 25$ sequenze decimali composte da 2 cifre pari
- se $i \geq 3$ posso vedere le sequenze che ci interessano come la somma di quelle che terminano con un numero dispari e quelle che terminano con un numero pari.
  - quelle che terminano con un numero dispari si ottengono concatenando il numero dispari a quelle che vanno bene e che hanno $i-1$ cifre, sono dunque $5 \cdot T[i-1]$
  - quelle che terminano con un numero pari devono avere al penultimo posto un numero dispari ed essere poi precedute da una qualunque di quelle che vanno bene di lunghezza $i-2$, sono dunque $25 \cdot T[i-2]$

```
def es2(n):
    T=[0 for _ in range(n+1)]
    T[1],T[2]=10,75
    for i in range(3,n+1):
        T[i]=5*T[i-1]+25*T[i-2]
    return T[n]

#esempio
>>> es2(3)
625
```

La complessità è $O(n)$

ESERCIZIO 3: Vogliamo trovare le sequenze di lunghezza n sull'alfabeto $\{0, 1, 2\}$ in cui non appaiono cifre pari adiacenti. Progettare un algoritmo che prende come parametro l'intero n e stampa tutte le sequenze cui siamo interessati.

La complessità dell'algoritmo proposto deve essere $O(nS(n))$ dove $S(n)$ è il numero di sequenze da stampare.
Motivare bene la complessità del vostro algoritmo.

Ad esempio:

- per $n = 1$ bisogna stampare le sequenze $'0'$,$'1'$ e $'2'$ (non necessariamente in quest'ordine).
- per $n = 2$ bisogna stampare le sequenze $'01'$,$'10'$, $'11'$, $'12'$ e $'21'$ (non necessariamente in quest'ordine).

```python
def es3(n):
    sol=[0]*n
    es3a(0,n,sol)

def es3a(i,n,sol):
    if i==n:
        print(''.join(sol))
        return
    if i>0 and sol[i-1] !='1':
            sol[i]='1'
            es3a(i+1,n,sol)
    else:
        for x in ['0','1','2']:
            sol[i]=x
            es3a(i+1,n,sol)

>>> es3(2)
01
10
11
12
21
```

Spiegazione

- nell'algoritmo si ottiene la sequenza da stampare aggiungendo un simbolo per volta e facendo attenzione di non aggiungere una cifra pari se la cifra che precede è pari.
- in questo modo siamo sicuri che le stringhe lunghe n che si ottengono sono stringhe effettivamente da stampare.
- l'albero di ricorsione avrà altezza n ed ogni nodo interno avrà almeno un cammino verso una delle $S(n)$ foglie da stampare.
    1. Gli $S(n)$ nodi foglia dell'albero lavorano ciascuno per tempo $O(n)$. I nodi foglia richederanno in totale tempo $O(nS(n))$
    2. Il numero di nodi interni dell'albero sarà limitato da $nS(n)$ e ciascuno di questi lavorerà $O(1)$. I nodi interni richiederanno in totale tempo $O(nS(n))$.
- Il tempo d'esecuzione dell'algoritmo sarà $O(nS(n))$