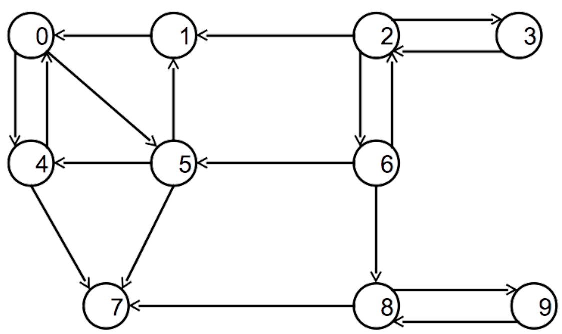
Analisi e progettazione di algoritmi

(III anno Laurea Triennale - a.a. 2021/22)

Prova scritta 6 settembre 2022

Esercizio 1 Si esegua, sul seguente grafo:

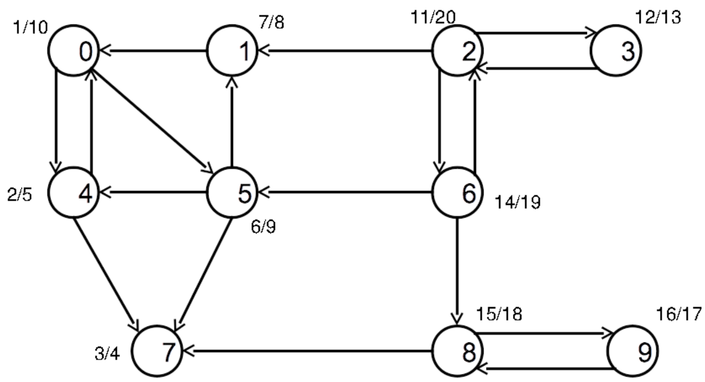


l'algoritmo per il calcolo delle componenti fortemente connesse. In particolare, si diano:

1. i tempi di inizio e fine visita ottenuti per ogni nodo in seguito alla visita in profondità (in tutti i casi in cui si deve scegliere un nodo, si consideri l'ordine numerico crescente)
2. la sequenza delle componenti fortemente connesse $\mathtt{Ord}^{\leftrightarrow}$ ottenuta
3. il grafo quoziente.

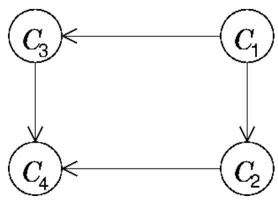
Soluzione

1. La visita in profondità assegna i seguenti tempi di inizio e fine visita:



2. La sequenza delle componenti fortemente connesse ottenuta è la seguente:
   $\mathcal{C}_1 = \{2,3,6\}, \mathcal{C}_2 = \{8,9\}, \mathcal{C}_3 = \{0,1,5,4\}, \mathcal{C}_4 = \{7\}$.

3. Il grafo quoziente è il seguente:



Esercizio 2 Siano $X[1..n]$ e $Y[1..n]$ due sequenze. Indichiamo con $\exists CS[i,j,k]$, con $0 \leq i,j,k \leq n$, il problema di decidere se esiste una sottosequenza comune di $X[1..i]$ e $Y[1..j]$ di lunghezza (almeno) k. Si dia una semplice variazione di quanto visto per LCS che risolve questi problemi. In particolare:

1. Si definisca induttivamente $\exists CS[i,j,k]$ giustificando la correttezza della definizione.

2. Si descriva un corrispondente algoritmo di programmazione dinamica ($\exists CS$ sarà quindi una matrice a valori booleani), indicandone la complessità.

3. Si descriva come ottenere anche una delle sottosequenze di di $X[1..i]$ e $Y[1..j]$ di lunghezza (almeno) k, se ne esistono.

1. Definizione induttiva di $\exists CS[i,j,k]$ per ogni $0 \leq i,j,k \leq n$:

   Base

   $\exists CS[i,j,0] = T$ per ogni $0 \leq i,j \leq n$
   (la sequenza vuota è sempre una sottosequenza comune)
   $\exists CS[0,j,k] = F$ per ogni $k > 0$ se $i = 0$ oppure $j = 0$
   (se una delle due stringhe è vuota non ci sono sottostringhe comuni di lunghezza > 0)

   Passo induttivo per ogni $0 < i,j,k \leq n$

   $$\exists CS[i,j,k] = \begin{cases} \exists CS[i-1,j-1,k-1] & \text{se } X[i] = Y[j] \\ \exists CS[i-1,j,k] \vee \exists CS[i,j-1,k] & \text{se } X[i] \neq Y[j] \end{cases}$$

   (analogamente a quanto visto per LCS)

   Analogamente a quanto visto per LCS la correttezza è basata sul principio di induzione forte (le chiamate ricorsive sono su triple (i,j,k) strettamente minori).

2. Un corrispondente algoritmo di programmazione dinamica è il seguente.

```
for (i = 0; i <= n; i++)
  for (j = 0; i <= n; i++) Exists[i,j,0] = true

for (k = 1; k <= n; k++)
  for (j = 1; j <= n; j++) Exists[0,j,k] = false
  for (i = 1; i <= n; j++) Exists[i,0,k] = false

for (k = 1; k <= n; k++)
  for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++)
      if (X[i] = Y[j]) Exists [i,j,k] = Exists[i-1,j-1,k-1]
      else Exists[i-1,j,k] || Exists[i,j-1,k]
```

   L'algoritmo costruisce una matrice tridimensionale quindi è $O(n^3)$.

3. Per ottenere anche una delle sottosequenze di di $X[1..i]$ e $Y[1..j]$ di lunghezza (almeno) k, se ne esistono, si può utilizzare una tecnica analoga a quella vista per LCS, ossia memorizzare anche un puntatore alla casella adiacente in diagonale (se $X[i] = Y[j]$) oppure a sinistra o in alto (se $X[i] \neq Y[j]$, solo se il valore della casella è `true`.

Esercizio 3 Dato il grafo G in figura, supponi che in un primo *run* siano campionati in sequenza gli archi

$$(1, 3), (5, 6), (13, 2), (132, 4)$$

e in un secondo

$$(5, 6), (4, 56), (1, 2), (12, 3)$$

Che cosa restituisce *MCMinCut(G)* nei due casi? Spiega il motivo per cui ti aspetti che *MCMinCut(G)* restituisca un taglio minimo con probabilità almeno pari a $1/15$ e quante volte devi lanciare *MCMinCut(G)* in modo tale da ottenere un taglio minimo con probabilità del 99.9%.