Analisi e progettazione di algoritmi

(III anno Laurea Triennale - a.a. 2018/19)

Prova scritta 17 luglio 2019

NB: I punteggi sono indicativi.

Esercizio 1 – (Punti 6)

Esercizio 2 – (Punti 6)

Esercizio 3 - Design e analisi di algoritmi (Punti 7) Si consideri una funzione che calcola l'n-simo numero di "Tribonacci" nel modo seguente:

```
trib(n)
  if (n = 0) return 0
  if (n = 1 || n = 2) return 1
  else return trib(n-1) + trib(n-2) + trib(n-3)
```

1. Si valuti la complessità di questo algoritmo ricorsivo.
2. Si dia un algoritmo che calcola `trib(n)` in tempo $\Theta(n)$ e spazio costante.
3. Si dia un algoritmo *ricorsivo* che che calcola `trib(n)` in tempo $\Theta(n)$.

Soluzione

1. L'algoritmo ricorsivo per calcolarli ha relazione di ricorrenza:

   $$T(n) = T(n-1) + T(n-2) + T(n-3) + 1$$

   Quindi, dato che ovviamente $T(n+1) > T(n)$ per $n > 2$,

   $$T(n) > 3T(n-3) + 1$$

   Risolviamo:

   $T(n) > 3(3T(n-6)+1)+1 = 3^2 T(n-6) + 3^1 + 3^0 > \ldots > 3^i T(n-3i) + 3^{i-1} + \ldots + 3^1 + 3^0$
   (ultimo termine per $i = k$ se $n = 3k$)
   $> 3^k + \ldots + 3^0 = 3^{k+1} - 1 = 3^{\frac{n}{3}+1} - 1.$

   Si ha quindi un algoritmo esponenziale.

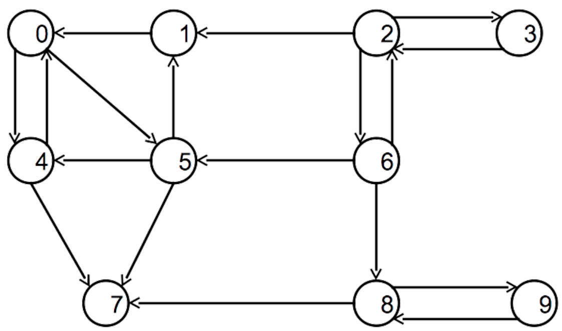
2. Supponiamo di avere la struttura dati "triple" (oppure si può usare un array `trib[0..2]`).

```
trib(n)
  if (n = 0) return 0
  if (n = 1 || n = 2) return 1
    (terzultimo, penultim, ultimo) = (0,1,1)
    for (i=3; i ≤ n; i++)
      (terzultimo, penultimo, ultimo) =
        (penultimo, ultimo, terzultimo + penultimo + ultimo)
  return ultimo
```

3.

```
trib(0) = (0,1,1)
trib(n) =
  let (terzultimo,penultimo,ultimo) = trib(n-1)
  in return (penultimo,ultimo,terzultimo+penultimo+ultimo)
```

Esercizio 4 - Grafi (Punti 7) Si esegua, sul seguente grafo:



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

- i tempi di inizio e fine visita ottenuti per ogni nodo in seguito alla visita in profondità (si effettui la visita a partire dal nodo 0 e si considerino gli adiacenti di un nodo in ordine numerico crescente)
- la sequenza delle componenti fortemente connesse $\mathtt{Ord}^{\leftrightarrow}$ ottenuta
- il grafo quoziente.

Soluzione

Esercizio 5 - NP-completezza (Punti 6) Consideriamo problemi di decisione concreti, quindi sottoinsiemi di $\{0,1\}^\star$, e sia $\mathcal{P}$ un problema NP-completo tale che $01 \in \mathcal{P}$. Assumendo $\mathsf{P} \neq \mathsf{NP}$, provare che il problema $\mathcal{P} \setminus \{01\}$ non può essere risolvibile in tempo polinomiale.

Soluzione Infatti, se esistesse un algoritmo polinomiale A che decide $\mathcal{P} \setminus \{01, 10\}$, potremmo costruire un algoritmo polinomiale per $\mathcal{P}$ nel modo seguente:

```
input u
if (u = 01) return true
return algo(u)
```