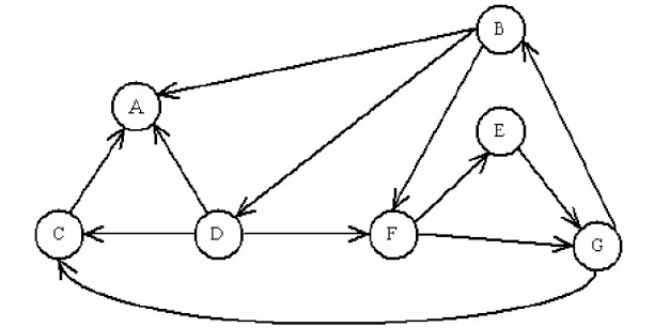
Analisi e progettazione di algoritmi

(III anno Laurea Triennale - a.a. 2024/25)

Prova scritta 16 giugno 2025

Esercizio 1 Si consideri il seguente grafo:



1. Si esegua la visita BFS iterativa, mostrando a ogni passo il nodo estratto, la coda restante e l'albero BFS parziale.
2. Si esegua la visita DFS iterativa, mostrando a ogni passo il nodo estratto, la pila restante e l'albero DFS parziale, evidenziandone la parte definitiva.
3. Si mostri il grafo quoziente.

In tutti i casi in cui si deve scegliere un nodo, si deve ottenere nella visita l'ordine alfabetico.

Soluzione (Data in formato non grafico per mia comodità.)

1.

nodo estratto	coda	albero parziale
	B	
B	~~B~~ A D F	(B,A), (B,D), (B,F)
A	~~B~~ ~~A~~ D F	(B,A), (B,D), (B,F)
D	~~B~~ ~~A~~ ~~D~~ F C	(B,A), (B,D), (B,F), (D,C)
F	~~B~~ ~~A~~ ~~D~~ ~~F~~ C E G	(B,A), (B,D), (B,F), (D,C), (F,E), (F,G)
C	~~B~~ ~~A~~ ~~D~~ ~~F~~ ~~C~~ E G	(B,A), (B,D), (B,F), (D,C), (F,E), (F,G)
E	~~B~~ ~~A~~ ~~D~~ ~~F~~ ~~C~~ ~~E~~ G	(B,A), (B,D), (B,F), (D,C), (F,E), (F,G)
G	~~B~~ ~~A~~ ~~D~~ ~~F~~ ~~C~~ ~~E~~ ~~G~~	(B,A), (B,D), (B,F), (D,C), (F,E), (F,G)

2.

nodo estratto	pila	albero parziale
	B	
B	~~B~~ A D F	(B,A), (B,D), (B,F)
A	~~B~~ ~~A~~ D F	**(B,A)**, (B,D), (B,F)
D	~~B~~ ~~A~~ ~~D~~ C F F	**(B,A)**, **(B,D)**, (D,C), (D,F)
C	~~B~~ ~~A~~ ~~D~~ ~~C~~ F F	**(B,A)**, **(B,D)**, **(D,C)**, (D,F)
F	~~B~~ ~~A~~ ~~D~~ ~~C~~ ~~F~~ E G F	**(B,A)**, **(B,D)**, **(D,C)**, **(D,F)**, (F,E), (F,G)
E	~~B~~ ~~A~~ ~~D~~ ~~C~~ ~~F~~ ~~E~~ G G F	**(B,A)**, **(B,D)**, **(D,C)**, **(D,F)**, **(F,E)**, (E,G)
G	~~B~~ ~~A~~ ~~D~~ ~~C~~ ~~F~~ ~~E~~ ~~G~~ G F	**(B,A)**, **(B,D)**, **(D,C)**, **(D,F)**, **(F,E)**, **(E,G)**
G	~~B~~ ~~A~~ ~~D~~ ~~C~~ ~~F~~ ~~E~~ ~~G~~ ~~G~~ F	**(B,A)**, **(B,D)**, **(D,C)**, **(D,F)**, **(F,E)**, **(E,G)**
F	~~B~~ ~~A~~ ~~D~~ ~~C~~ ~~F~~ ~~E~~ ~~G~~ ~~G~~ ~~F~~	**(B,A)**, **(B,D)**, **(D,C)**, **(D,F)**, **(F,E)**, **(E,G)**

3. Le componenti fortemente connesse sono A, $BDEFG$, C, e nel grafo quoziente ci sono gli archi $(BDEFG, A), (BDEFG, C), (C, A)$.

Esercizio 2 Un distributore di bevande contiene al suo interno n monete i cui valori sono dati da un array $c[1..n]$ di numeri naturali positivi, ed è possibile che più monete abbiano lo stesso valore. Il problema $P(n, R)$, con R numero naturale, consiste nel decidere se sia o meno possibile erogare un resto uguale a R utilizzando un opportuno sottoinsieme delle n monete.

1. Si descriva a parole un algoritmo brute-force che risolva il problema (quindi restituisca T o F) e se ne valuti la complessità.

2. Sia $P(k, r)$, con $1 \leq k \leq n$, $0 \leq r \leq R$, il sottoproblema che consiste nel decidere se sia o meno possibile erogare un resto uguale a r utilizzando un opportuno sottoinsieme delle prime k monete. Si definisca induttivamente $P(k, r)$. Suggerimento: il caso base è $k = 1$.

3. Considerando per semplicità solo il numero di monete come dimensione dell'input, si valuti la complessità di un algoritmo divide-et-impera basato sulla precedente definizione induttiva.

4. Si descriva un algoritmo di programmazione dinamica basato sulla precedente definizione induttiva.

5. Come si potrebbe ottenere anche la sequenza di monete (indici nell'array) che formano la soluzione trovata?

Soluzione

1. Un algoritmo brute-force risolve il problema provando a sommare tutti i possibili sottoinsiemi delle n monete, quindi ha complessità $n \cdot 2^n$.

2. $$P(1, r) = \begin{cases} T & \text{se } c[1] = r \text{ oppure } r = 0 \\ F & \text{altrimenti} \end{cases}$$
$$\text{per } k > 1,\ P(k, r) = \begin{cases} P(k-1, r) \vee P(k-1, r - c[k]) & \text{se } c[k] \leq r \\ P(k-1, r) & \text{altrimenti} \end{cases}$$

3. Considerando per semplicità solo il numero di monete come dimensione dell'input, un algoritmo divide-et-impera ha la seguente relazione di ricorrenza per il caso peggiore:

$$T(1) = 1$$
$$T(k) = 2\,T(k-1)$$

uguale a quella delle torri di Hanoi, quindi risulta esponenziale.

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

```
for (r = 0; r <= R; r++)
  P[1,r] = (c[1] = r || r = 0);

for (k = 2; k <= n; k++)
  for (r = 0; r <= R; r++)
    if (c[k] ≤ r) && P(k-1,r-c[k]) P[k,r] = P[k-1,r-c[k])
    else P[k,r]=P[k-1,r]
```

5. Per ottenere anche le monete utilizzate, possiamo costruire un'ulteriore matrice `U` di booleani, in modo che `U[k,r]` sia vero se e solo se possiamo ottenere il resto r utilizzando la moneta `k`.

```
for (r = 0; r <= R; r++)
  if (c[1] = r) P[1,r] = U[1,r] = true
  else if (r = 0) P[1,r] = true; U[1,r] = false
  else P[1,r] = U[1,r] = false

for (k = 2; k <= n; k++)
  for (r = 0; r <= R; r++)
    if (c[k] ≤ r) && P(k-1,r-c[k]) P[k,r] = P[k-1,r-c[k]); U[k,r] = true
    else P[k,r]=P[k-1,r]; U[k,r] = false
```

Esercizio 3 Descrivi il funzionamento ed esegui il test di Miller Rabin sui numeri 7 e 9 per tutte le basi (da 2 a 5 compresi per 7, da 2 a 7 compresi per 9). Quanti testimoni e quanti bugiardi ottieni nei due casi?

Guida alla correzione

1.1 gestione errata coda e/o albero max 4;

1.2 gestione errata pila e/o albero max 5; non distingue albero definitivo -1,5