Algoritmi e Strutture di Dati (2^a Ed.)
Errata corrige

Alan Bertossi, Alberto Montresor

Gli autori saranno riconoscenti a chiunque segnali errori presenti nel libro di testo, scrivendo ad `alberto.montresor@unitn.it`.

- Pag. 15, manca n nella lista dei parametri in ingresso di iterativeBinarySearch().
- pag. 22: non considerando la parte intera, la ricorrenza per $T(n)$ non funziona per $n = 1$, che non è pari. Per essere precisi, bisognerebbe introdurre il caso $T(1) = T(0) + d$ (vengono eseguite tutte le operazioni per trovare il mediano e verificare se è il valore cercato, e poi si cerca su un sottovettore vuoto)
- Pag. 29, seconda riga: (Fig. 2.2) $\rightarrow$ (Figg. 2.3-2.4)
- Pag. 31, nella relazione di ricorrenza, secondo caso: $n \geq 2$, e non $n \leq 2$.
- Pag. 35, algoritmo countingSort(): il parametro di input A è in realtà un vettore di interi, non di ITEM.
- Pag. 35, algoritmo countingSort(), quarta riga: il ciclo dovrebbe essere:

$$\textbf{for } i \leftarrow 1 \textbf{ to } n \textbf{ do } B[A[i]] \leftarrow B[A[i]] + 1$$

- Pag. 47, Es. 2.12, codice follia(): nell'assegnamento $i \leftarrow \lfloor n/2 \rfloor$ si modifica l'indice i del for: sostituirlo con **integer** $k \leftarrow \lfloor n/2 \rfloor$
- Pag. 49, soluzione Esercizio 2.1: $n^{\log n}$ precede 3^{n-2}.
- Pag. 50, algoritmo ORDINABANDIERA() Le prime due righe devono essere sostituite dalle seguenti:

$$k \leftarrow 1j \qquad\qquad \leftarrow n$$

- Pag. 59, funzione insert(): "Ritorna la nuova posizione" $\rightarrow$ "Ritorna la posizione del nuovo elemento"
- Pag. 59, funzione remove(): "Ritorna il successore di p" $\rightarrow$ "Ritorna la posizione del successore di p"
- Pag. 70, codice DonGiovanni: L.insert(L.next(L.tail()), v) al posto di L.insert(L.tail(L.next()), v).
- Pag. 72, procedura insert(): la riga t.*pred* $\leftarrow$ t.*pred* va sostituita con t.*pred* $\leftarrow$ p.*pred*.
- Pag. 74, la formula va sostituita con la seguente:

$$C(n) \leq n + 2^{\lfloor \log_2 n \rfloor + 1} - 1 \leq n + 2n$$

- Pag. 81, manca la parentesi di chiusura della lista degli argomenti per la procedura hanoi-iterativa()
- Pag. 81,83, nelle procedure viene dichiarata la variabile $temp$ che non viene mai utilizzata
- Pag. 92, sesta riga: parent() $\neq$ **nil** $\Rightarrow$ parent() $=$ **nil**
- Pag. 94, algoritmi invisita() e visitaAmpiezza():
"precontion" $\rightarrow$ "precondition"
- Pag. 96, procedura insertSibling(TREE t). La prima riga deve essere sostituita t.*parent* $\leftarrow$ *parent*.
- Pag. 98. "Essendo $n! = n(n-1) \cdot 3 \cdot 2 \cdot 1$" $\rightarrow$ "Essendo $n! = n \cdot (n-1) \cdots 3 \cdot 2 \cdot 1$"
- Capitolo 8. Negli algoritmi, le variabili C mancano della dichiarazione del tipo SET, mentre le variabili p, q ed r mancano della dichiarazione di tipo POS.
- Pag. 105. Funzione DELETELEAF(TREE T); la quarta riga (t.DELETELEAF()) deve essere sostituita con t.DELETELEFT().
- Pag. 108: Nel testo dell'Esempio 6.2, l'Esempio 1.2 è in realtà l'Esempio 1.3.
- Pag. 109. "Possono sorgere tre casi, illustrati nella Fig. 6.1" $\rightarrow$ "6.2"
- Pag. 109. Nella funzione insertNode(), "**if** $p = x$ **nil then return** n" $\rightarrow$ "**if** $p =$ **nil then return** n"
- Pag. 110, ultime due righe di testo. "Se sia u che il suo figlio destro sono **nil** " $\rightarrow$ "Se entrambi il padre e il figlio u sono **nil**"

- Pag. 111, algoritmo removeNode(), blocco **if** del caso (3). In fondo, aggiungere la seguente linea di codice:

$$x \leftarrow s.key$$

- Pag. 117, Figura 6.5. Caso (5a), parte sinistra: scambiare p e t.

- Pag. 120, Figura 6.6(1). Entrambi i figli di f sono etichettati ns quando dovrebbero essere ns ed nd.

- Pag. 121. Codice balanceDelete(), quartultima riga: $t \leftarrow T$, invece di $t \leftarrow$ **nil**.

- Pag 144 e seguenti. Nel algoritmi del Capitolo 8, le variabili C mancano della dichiarazione del tipo SET, mentre le variabili p, q ed r mancano della dichiarazione di tipo POS.

- Pag. 146, in due occasioni, *dim* () deve essere sostituita con size().

- Pag. 147, procedura difference(). L'istruzione $p \leftarrow A.next(p)$ non va inserita nel blocco if, ma a livello del ciclo **while**.

- Pag. 153. La soluzione dell'esercizio 8.8 è in realtà quella dell'8.10.

- Pag. 156. La didascalia della Fig. 9.3 deve essere sostituita con "Un grafo non orientato con tre componenti connesse"

- Pag. 164, Soluzione di Holmes: Il ciclo (2) è "A,G,H,B,A" e non "A,G,A,B,A"

- Pag. 166. Nella procedura stampaCammino, il parametro GRAPH G non è necessario.

- Pag. 167. Nella descrizione della dfs(), "L'algoritmo viene richiamato su un grafo G a partire da un nodo r" $\rightarrow$ "nodo u"

- Pag. 167. L'algoritmo dfs() contiene diversi errori, va riscritto nel modo seguente:

```
dfs(GRAPH G, NODE u, boolean[] visitato)
    visitato[u] ← true
(1) { esamina il nodo u (caso previsita) }
    foreach v ∈ G.adj(u) do
        { esamina l'arco (u, v)
        if not visitato[v] then
            dfs(G, v, visitato)
(2) { esamina il nodo u (caso postvisita) }
```

- Pag. 171. Nella procedura scc() viene definito uno Stack senza nome. "STACK $\leftarrow$ Stack()" $\rightarrow$ "STACK $S \leftarrow$ Stack()"

- Pag. 171. La quarta riga dell'algoritmo scc() deve essere sostituita da:

```
foreach u ∈ G.V() do
    if not visitato[u] then
        dfsStack(G, visitato, S, u)
```

- Pag. 177, es. 9.4: sostituire il codice della soluzione con il seguente

```
boolean bipartito(GRAPH G, NODE r)
    QUEUE S ← Queue( )
    S.enqueue(r)
    integer[ ] color ← new integer[1 . . . G.n]
    foreach u ∈ G.V() − {r} do color[u] ← senzacolore
    color[r] ← rosso
    while not S.isEmpty() do
        NODE u ← S.dequeue()
        foreach v ∈ G.adj(u) do
            if color[v] = senzacolore then
                color[v] ← 1 − color[u]
                S.enqueue(v)
            else
                if color[v] ≠ color[u] then return false
    return true
```

- Pag. 178, es. 9.7: sostituire le righe:

integer $j \leftarrow ordine[i]$
$partenza[j] \leftarrow partenza[j] + durata[j]$

con la riga:

$partenza[ordine[i]] \leftarrow partenza[ordine[i-1]] + durata[ordine[i]]$

- Pag. 187: la dichiarazione della funzione deleteMin è deleteMin() e non deleteMin(ITEM x).
- Pag. 191: nell'occorrenza di MFSET(n) (terza riga sezione 10.2.1), il font deve essere helvetica (Mfset(n)) invece che maiuscoletto (MFSET(n)). Nell'esempio 10.7 i termini "MFSET restituisce" devono essere sostituiti con "Mfset(4) restituisce".
- Pag. 193, Esempio 10.9. La sequenza di operazioni per produrre l'albero di altezza massima deve includere anche l'operazione merge$(6, 7)$ (in testa).
- Pag. 191, Fig. 10.5, l'ultima sottofigura in basso a destra deve essere etichettata (g), non d
- Pag. 193, pseudocodice merge(): i parametri x e y vanno scritti in corsivo
- Pag. 195, Fig. 10.6 l'etichetta della freccia va sostituita con "find(8)"
- Pag. 198, Esercizio 10.2: la procedura sort() va sostituita con la procedura heapsort().
- Pag. 198, Esercizio 10.3: la complessità $O(1 + \log(dim/i))$ va sostituita con $O(1 + \log(dim, i))$.
- Pag. 209, funzione camminiMinimi(). $u \leftarrow S$.estrai() manca della dichiarazione di tipo, dovrebbe essere **integer** $u \leftarrow S$.estrai()
- pag. 212: sezione 11.5, conserva un costo di $O(\log n)$ per l'operazione deleteMin(), ma richiede un costo ammortizzato di $O(1)$ per le operazioni insert() e decrease().
- Pag. 223, Esempio 12.2: "al vettore dell'Esempio 1.2" → "Esempio 1.3"
- Pag. 246. La figura 13.3 in realtà dovrebbe essere numerata 13.2.
- Pag. 249. Nella procedura stampaPar() il parametro S dovrebbe essere di tipo matrice e non vettore.
- Pag. 250. "Durante l'esecuzione per calcolare una riga ... Fig. 13.3" → "... Fig 13.1"

- Pag. 252. Nella formula per la definizione delle condizioni iniziali di D^l_{uv}, "se $i = j$" $\rightarrow$ "se $u = v$".
- Pag. 253. Nella figura 13.3 (e), tabella $P^{(5)}_{ij}$, l'elemento $[1, 4] = 2$ deve essere sostituito dal valore 3.
- Pag. 254. Nella procedura floydWarshall() si sostituisca la riga $d[u, v] \leftarrow +\infty$ con $d[u, v] \leftarrow$ iif$(u = v, 0, +\infty)$.
- Pag. 254. Nella procedura floydWarshall(), "**for** $k \leftarrow 1$**to** n **do**" $\rightarrow$ "**to** $G.n$"
- Pag. 255, soluzione del problema dello zaino. La definizione ricorsiva della tabella D e il codice sono errati quando la capacità è negativa; dovrebbero essere rispettivamente:

$$D[i, c] = \begin{cases} 0 & \text{se } i = 0 \vee c = 0 \\ -\infty & \text{se } c < 0 \\ \max\{D[i-1, c], D[i-1, c-v_i] + p[i]\} & \text{altrimenti} \end{cases}$$

integer zaino(**integer**[ ] p, **integer**[ ] v, **integer** i, **integer** c, **integer**[ ][ ] D)

```
if i = 0 or c = 0 then
  return 0
if c < 0 then
  return −∞
if D[i, c] = ⊥ then
  D[i, c] ← max(zaino(p, v, i − 1, c, D), zaino(p, v, i − 1, c − v[i], D) + p[i])
return D[i, c]
```

- Pag. 257. Nella relazione $D[i, j]$, sostituire $i = 0 \vee i = 0$ con $i = 0 \vee j = 0$
- Pag. 257, algoritmo lcs(), il comando **return** deve stare fuori dall'**if**.
- Pag. 259, esercizio 13.3

$$d^{k+1}_v = \min\{d^k_v, \min_{h \neq v}\{d^k_v + w(h, v)\}\}.$$

deve essere sostituito con

$$d^{k+1}_v = \min\{d^k_v, \min_{h \neq v}\{d^k_h + w(h, v)\}\}.$$

- Pag. 255, definizione ricorsiva di $D[i, c]$ (e conseguentemente, algoritmo zaino()): l'ordine delle condizioni per $c < 0$ e per $i = 0 \vee c = 0$ va scambiato.

$$D[i, c] = \begin{cases} -\infty & \text{se } c < 0 \\ 0 & \text{se } i = 0 \vee c = 0 \\ \max\{D[i-1, c], D[i-1, c-v_i] + p[i]\} & \text{altrimenti} \end{cases}$$

- Pag. 266, algoritmo zaino(): l'ordinamento del profitto specifico deve essere decrescente, ovvero: { ordina p e v in modo che $p[1]/v[1] \geq p[2]/v[2] \geq \cdots \geq p[n]/v[n]$}
- Pag. 266, Essendo un vettore di valori compresi nel range di numeri reali $[0, 1]$, il vettore x di ZAINO() deve essere dichiarato come **real**[ ]. I vettori p, v e la capacità C *possono* essere valori reali, diversamente dallo Zaino 0-1.
- Pag. 270, algoritmo kruskal(): aggiungere in fondo l'operazione di ritorno **return** T.
- Pag. 271, algoritmo prim(): la variabile u va dichiarata di tipo **integer**.
- Pag. 273. La figura 14.1 è in realtà la 14.4.
- Pag. 273, Esempio 14.8. Perchè $t_1 = 5$ è massimo $\rightarrow$ "$t_1 = 4$".
- Pag. 274, algoritmo moore(): le variabili i e j vanno dichiarate **integer**.

- Pag. 278, Esercizio 14.7: rimuovere "di lunghezza k" e "con $k_i \in [0, k]$" dal testo dell'esercizio, in quanto non usato nella soluzione.
- Pag. 283, fig. 15.1(c). Il quinto grafo non fa parte dell'intorno di SOL.
- Pag. 287, figura 15.4: gli ultimi due valori della seconda riga devono essere 15 e 14 (e non 2 e 1).
- Pag. 288, formula dell'ultima riga: La prima somma è per $v \in V$, mentre la terza somma è per $u \in S - s$.
- Pag. 289, "La rete di flusso residua $r = (V, E_r, s, t, r)$ ha ..." → "$R = (V, E_r, s, p, r)$"
- Pag. 290, nella dimostrazione del teorema 15.1 P deve essere definito come $V - S$ (e non come $N - S$).
- Pag. 290, ultima riga della dimostrazione del Teorema: sostituire $|f| = c(S, P)$ (al posto di $|f| \leq c(S, P)$).
- Pag. 294, la funzione satura() non deve ritornare alcun valore; va quindi eliminato il valore di ritorno **integer** e l'istruzione **return** g.
- Pag. 295, algoritmo instrada(), le variabili v e M vanno dichiarate come **integer**.
- Pag. 308, algoritmo graham(). La variabile *min* va dichiarata come **integer**, mentre la variabile **integer** n deve essere inserita tra i parametri.
- Pag. 317, schema iniziale del sudoku: La quarta riga ".891.526." dovrebbe essere sostituita da ".894.526."
- Pag. 318, algoritmo sudoku(). Riga 9: **if**(check(A, x, y, c)) deve essere sostituto da **if**(check(S, x, y, c)). Riga 11: $S[i]$ deve essere sostituito con $S[x, y]$.
- Pag. 321, la soluzione dell'esercizio 16.1 è in realtà la soluzione per l'esercizio 16.4
- Pag. 325, prima dell'algoritmo selezione(). "è riapplicata ad $A[q + 1...$*ultimo*$]$" → "$A[j + 1...$*ultimo*$]$"
- Pag. 325, algoritmo selezione(): nell'ultimo **if** manca un caso base:

```
if k = q then
    return A[j]corollary
else if k < q then
    return selezione(A, primo, j − 1, k)
else
    return selezione(A, j + 1, ultimo, k − q)
```

- Pag. 326, sostituire la formula

$$A[\text{random}(primo, ultimo)] \leftrightarrow A[1]$$

con

$$A[\text{random}(primo, ultimo)] \leftrightarrow A[primo]$$

- Pag. 337, Fig 18.2. "tra province delila Toscana" → "della"
- Pag. 340, testo esempio 18.11. della soluzione parziale se "**choice**(**true**, **false**) = **true**" → "**choice**($\{$**true**, **false**$\}$) = **true**"
- Pag. 345, Fig. 18.5. Su lato destro della figura, "Risposta No" è in realtà "Risposta Sì".
- Pag. 358, Teorema 14: $\mathbb{P}$ e' propriamente contenuta in $\mathbb{EXP}$ (e non in $\mathbb{NEXP}$).
- Pag. 375, ultima riga
  "genera dapprima nel ciclo (2)" deve essere sostituito con "genera dapprima nel ciclo (1)"
- Pag. 376, modificare l'intestazione dell'algoritmo apSubsetSum():

```
apSubsetSum(integer[] A, integer n, integer k, integer h)
    [...]
```

- Pag. 383, la variabile *in* deve essere dichiarata come un vettore di interi.
- Pag. 384: nella penultima riga, sostituire “Anche in questo caso” con “In questo caso”