Algoritmi e Strutture Dati (Complementi)

Appello dell' 8/07/2003 – Prof. Giancarlo Mauri

Errata Corrige per l'Esercizio 1

La soluzione proposta per l'esercizio 1 è parzialmente errata, come è stato giustamente osservato da uno studente del corso, che ringraziamo per la segnalazione.

L'errore consiste in una svista nella scrittura dell'equazione di ricorrenza che esprime i valori $M[i, j]$ della matrice utilizzata per determinare la soluzione del problema. La conseguenza di questa svista è che l'algoritmo proposto dà la risposta corretta solo se il primo elemento dell'istanza (v_1) appartiene a un sottoinsieme $V' \subseteq \{v_1, v_2, \ldots, v_n\}$ tale che $\sum_{v \in V'} v = K$.

In particolare non basta porre $M[1, v_1] = 1$, ma occorre porre $M[i, v_i] = 1$ per ogni $i \in \{1, 2, \ldots, n\}$, dato che basta prendere l'elemento v_i per avere un sottoinsieme di $\{v_1, \ldots, v_i\}$ i cui elementi abbiamo somma uguale a v_i. In alternativa, si può aggiungere la colonna di indice 0 alla matrice M, ponendo $M[1, 0]$ e $M[1, v_1]$ uguali a 1; in tal caso, non è più necessario porre esplicitamente $M[i, v_i] = 1$ per ogni $i \in \{1, 2, \ldots, n\}$.

Segue il testo dell'esercizio con le due soluzioni corrette.

Esercizio 1.

Un ente pubblico ha ricevuto un finanziamento di K euro per effettuare degli acquisti. Gli oggetti che vorrebbe acquistare sono n, e hanno un valore commerciale pari a $v_1, v_2, \ldots, v_n$ euro. Purtroppo i soldi non bastano per acquistare tutti gli n oggetti; inoltre, i soldi che non vengono spesi vanno restituiti allo Stato. L'ente vorrebbe pertanto sapere se è possibile acquistare un sottoinsieme degli n oggetti in modo da spendere esattamente K euro, così da non dover restituire nulla allo Stato.

Progettare un algoritmo di programmazione dinamica che aiuti l'ente a decidere se ciò è possibile. In particolare, scrivere prima le equazioni di ricorrenza che esprimono una soluzione del problema e poi lo pseudo–codice dell'algoritmo. Infine, si dica qual è la complessità in tempo dell'algoritmo proposto.

Soluzione. Il problema può essere formalizzato come segue: dato un insieme $V = \{v_1, v_2, \ldots, v_n\}$ di interi positivi, e un intero positivo K, esiste un sottoinsieme $V' \subseteq V$ tale che $\sum_{v \in V'} v = K$?

Definiamo una matrice $M[1..n, 1..K]$ a valori in $\{0, 1\}$, dove per $i \in \{1, \ldots, n\}$ e $j \in \{1, \ldots, K\}$ vale $M[i, j] = 1$ se e solo se esiste un sottoinsieme A di $\{v_1, \ldots, v_i\}$ tale

che $\sum_{a \in A} a = j$. Calcolati i valori di M, risponderemo SI se $M[n, K] = 1$, e NO altrimenti. I valori di M verranno calcolati per righe, a partire dalla prima; all'interno di ogni riga, i valori verranno calcolati da sinistra verso destra (e quindi, in particolare, l'elemento $M[n, K]$ sarà l'ultimo ad essere calcolato).

Per calcolare $M[i, j]$, osserviamo che affinché sia $M[i, j] = 1$ ci sono due possibilità:

1. esiste un insieme $A \subseteq \{v_1, \ldots, v_i\}$ tale che $\sum_{a \in A} a = j$ e $v_i \in A$, oppure
2. v_i *non* appartiene a *nessun* insieme $A \subseteq \{v_1, \ldots, v_i\}$ tale che $\sum_{a \in A} a = j$.

Nel primo caso deve esistere un insieme $B \subseteq \{v_1, \ldots, v_{i-1}\}$ tale che $\sum_{b \in B} b = j - v_i$, e quindi deve valere $M[i-1, j-v_i] = 1$ (si noti che deve essere $i > 1$ e $j > v_i$, altrimenti l'elemento $M[i-1, j-v_i]$ non esiste). Nel secondo caso deve esistere un insieme $B \subseteq \{v_1, \ldots, v_{i-1}\}$ tale che $\sum_{b \in B} b = j$, e quindi deve valere $M[i-1, j] = 1$ (sempre con la richiesta che valga $i > 1$).

Pertanto, le equazioni di ricorrenza che consentono di calcolare il valore di $M[i, j]$ sono:

$$M[i, j] = \begin{cases} \max\{M[i-1, j-v_i], M[i-1, j]\} & \text{se } i > 1 \text{ e } j > v_i \\ M[i-1, j] & \text{se } i > 1 \text{ e } j < v_i \\ 1 & \text{se } j = v_i \\ 0 & \text{se } i = 1 \text{ e } v_1 \neq j \end{cases}$$

mentre lo pseudocodice dell'algoritmo che calcola i valori della matrice M è il seguente:

```
SUBSET SUM({v_1, v_2, ..., v_n})
for j ← 1 to K
   do M[1, j] ← 0
M[1, v_1] ← 1
for i ← 2 to n
    do for j ← 1 to K
           do if j = v_i
                 then M[i, j] ← 1
                 else M[i, j] ← M[i − 1, j]
                      if j > v_i and M[i − 1, j − v_i] > M[i, j]
                         then M[i, j] ← M[i − 1, j − v_i]
return M[n, K]
```

Il tempo di calcolo dell'algoritmo è $\Theta(nK)$.

Seconda Soluzione. L'analisi del problema è identica a quella fatta nella prima soluzione. Questa volta, però, la matrice M è definita come $M[1..n, 0..K]$, sempre a valori in

$\{0, 1\}$. Ovviamente, al termine dell'esecuzione la colonna di indice 0 conterrà un 1 in ogni posizione, dato che per ogni $i \in \{1, 2, \ldots, n\}$ esiste sempre un insieme $A \subseteq \{v_1, \ldots, v_i\}$ (basta prendere l'insieme vuoto) tale che $\sum_{a \in A} a = 0$.

Le equazioni di ricorrenza che consentono di calcolare il valore di $M[i, j]$ diventano:

$$M[i,j] = \begin{cases} \max\{M[i-1, j-v_i], M[i-1, j]\} & \text{se } i > 1 \text{ e } j \geq v_i \\ M[i-1, j] & \text{se } i > 1 \text{ e } j < v_i \\ 1 & \text{se } i = 1 \text{ e } j \in \{0, v_1\} \\ 0 & \text{se } i = 1 \text{ e } j \notin \{0, v_1\} \end{cases}$$

mentre lo pseudocodice dell'algoritmo che calcola i valori della matrice M diventa:

```
SUBSET SUM({v_1, v_2, ..., v_n})
for j ← 1 to K
   do M[1, j] ← 0
M[1, 0] ← M[1, v_1] ← 1
for i ← 2 to n
    do for j ← 0 to K
           do M[i, j] ← M[i − 1, j]
              if j ≥ v_i and M[i − 1, j − v_i] > M[i, j]
                 then M[i, j] ← M[i − 1, j − v_i]
return M[n, K]
```

Anche in questo caso, il tempo di calcolo dell'algoritmo è $\Theta(nK)$.