

Algoritmi 2: esercizi sulla programmazione dinamica *

1 Furto

Una banda di tre ladri deve spartirsi il frutto di una rapina di n oggetti $\{a_1, \dots, a_n\}$ ciascuno caratterizzato da un proprio valore intero positivo v_i . Sapendo che l'ammontare totale del bottino M è divisibile per tre, descrivere un algoritmo che verifichi se è possibile spartire gli n oggetti in parti di ugual valore e, in caso affermativo, produca la spartizione. L'algoritmo deve avere tempo di esecuzione $O(n \cdot M^2)$.

Consideriamo innanzitutto il problema di verificare se è possibile spartire gli n oggetti in tre parti ciascuna di valore $M/3$. Osserviamo che, se i ladri sono professionisti, nessun oggetto sarà scartato né tanto meno restituito al legittimo proprietario! La parte assegnata al terzo ladro sarà quindi univocamente determinata dalle parti assegnate al primo e al secondo. Sarà pertanto sufficiente verificare se sia possibile spartire gli oggetti in modo tale che i bottini del primo e del secondo ladro siano entrambi pari a $M/3$.

Per trovare una soluzione a questo problema, risolveremo il seguente sottoproblema $\mathcal{P}(i, j, k)$:

$\mathcal{P}(i, j, k)$: esiste una spartizione degli oggetti $\{a_1, \dots, a_k\}$ che assegna al primo ladro un valore i e al secondo ladro un valore j ?

Tali sottoproblemi sono definiti per k tale che $0 \leq k \leq n$ e per i, j tali che $0 \leq i, j \leq M/3$. Memorizzeremo le soluzioni per i sottoproblemi $\mathcal{P}(i, j, k)$ in una matrice Booleana tridimensionale M , di dimensioni $[0, M/3] \times [0, M/3] \times [0, n]$. La soluzione del problema originario sarà in $M[M/3, M/3, n]$.

E' facile verificare che $M[0, 0, 0] = true$ e che $M[i, j, 0] = false$ se $i \neq 0$ o $j \neq 0$. Questi casi rappresentano le condizioni al contorno. Per calcolare il valore $M[i, j, k]$ a partire dai valori $M[-, -, k-1]$ consideriamo i tre casi in cui l'oggetto a_k venga assegnato al primo, al secondo o al terzo ladro:

- Assegnando l'oggetto a_k al terzo ladro, $M[i, j, k]$ risulterebbe vera se e solo se $M[i, j, k-1] = true$.

*Irene Finocchi, Dipartimento di Informatica, Università degli Studi di Roma "La Sapienza", Via Salaria 113, 00198 Rome, Italy. E-mail: finocchi@di.uniroma1.it.

Algoritmo VerificaSpartizione

1. **Input** n oggetti $\{a_1, \dots, a_n\}$ con valori positivi $\{v_1, \dots, v_n\}$, valore totale M
2. **Output** Valore Booleano
3. **begin**
4. **for** $i = 0$ **to** $M/3$ **do**
5. **for** $j = 0$ **to** $M/3$ **do**
6. $M[i, j, 0] \leftarrow false$
7. $M[0, 0, 0] \leftarrow true$
8. **for** $k = 1$ **to** n **do**
9. **for** $i = 0$ **to** $M/3$ **do**
10. **for** $j = 0$ **to** $M/3$ **do**
11. **case:**
12. $i \geq v_k$ e $j \geq v_k$:
13. $M[i, j, k] \leftarrow M[i, j, k - 1] \text{ OR } M[i - v_k, j, k - 1] \text{ OR } M[i, j - v_k, k - 1]$
14. $i \geq v_k$ e $j < v_k$:
15. $M[i, j, k] \leftarrow M[i, j, k - 1] \text{ OR } M[i - v_k, j, k - 1]$
16. $i < v_k$ e $j \geq v_k$:
17. $M[i, j, k] \leftarrow M[i, j, k - 1] \text{ OR } M[i, j - v_k, k - 1]$
18. $i < v_k$ e $j < v_k$:
19. $M[i, j, k] \leftarrow M[i, j, k - 1]$
20. **endcase**
21. **return** $M[M/3, M/3, n]$
22. **end**

Figure 1: Algoritmo per verificare se esiste una spartizione in tre parti uguali.

- Assegnando l'oggetto a_k al primo ladro, $M[i, j, k]$ risulterebbe vera se e solo se $M[i - v_k, j, k - 1] = true$. Osserviamo che deve essere $i \geq v_k$ affinché $M[i - v_k, j, k - 1]$ sia propriamente definita.
- Assegnando l'oggetto a_k al secondo ladro, $M[i, j, k]$ risulterebbe vera se e solo se $M[i, j - v_k, k - 1] = true$. Osserviamo che deve essere $j \geq v_k$ affinché $M[i, j - v_k, k - 1]$ sia propriamente definita.

La ricorrenza che descrive i passi base e il passo di avanzamento nella costruzione della matrice M è pertanto:

$$M[i, j, k] = \begin{cases} true & \text{se } i = j = k = 0 \\ false & \text{se } k = 0 \text{ e } i, j \neq 0 \\ M[i, j, k - 1] \text{ OR } M[i - v_k, j, k - 1] \text{ OR } M[i, j - v_k, k - 1] & \text{se } i \geq v_k \text{ e } j \geq v_k \\ M[i, j, k - 1] \text{ OR } M[i - v_k, j, k - 1] & \text{se } i \geq v_k \text{ e } j < v_k \\ M[i, j, k - 1] \text{ OR } M[i, j - v_k, k - 1] & \text{se } i < v_k \text{ e } j \geq v_k \\ M[i, j, k - 1] & \text{se } i < v_k \text{ e } j < v_k \end{cases}$$

La matrice M può essere calcolata in tempo $O(n \cdot M^2)$ come mostrato dallo pseudocodice in Figura 1.

Nel caso in cui $M[M/3, M/3, n] = true$, vogliamo calcolare la spartizione vera e propria degli oggetti. Per fare ciò, possiamo mantenere una matrice ausiliaria S che prende valori

Algoritmo Spartisci

1. **Input** matrici M e S
2. **Output** partizione degli oggetti tra i tre ladri
3. **begin**
4. **if** $M[M/3, M/3, n] = true$
5. $i, j \leftarrow M/3$
6. **for** $k = n$ **downto** 1 **do**
7. **if** $S[i, j, k] = 1$
8. assegna oggetto a_k al primo ladro
9. $i \leftarrow i - v_k$
10. **else if** $S[i, j, k] = 2$
11. assegna oggetto a_k al secondo ladro
12. $j \leftarrow j - v_k$
13. **else** assegna oggetto a_k al terzo ladro
14. **return** partizione degli oggetti
15. **end**

Figure 2: Algoritmo per calcolare la partizione degli oggetti.

in $[0, 3]$. In particolare, $S[i, j, k] = 0$ se $M[i, j, k] = false$, altrimenti $S[i, j, k]$ indica il ladro a cui è possibile assegnare l'oggetto a_k per far sì che il primo ladro abbia un bottino di valore pari a i ed il secondo ladro abbia un bottino di valore pari a j . In breve:

$$S[i, j, k] = \begin{cases} 0 & \text{se } M[i, j, k] = false \\ 1 & \text{se } M[i, j, k] = true = M[i - v_k, j, k - 1] \\ 2 & \text{se } M[i, j, k] = true = M[i, j - v_k, k - 1] \\ 3 & \text{se } M[i, j, k] = true = M[i, j, k - 1] \end{cases}$$

La matrice S può essere facilmente calcolata in tempo $O(n \cdot M^2)$ durante il calcolo della matrice M . A partire da S , possiamo ottenere la partizione degli oggetti come mostrato dallo pseudocodice in Figura 2.