6.8 Pseudo-polinomialità e programmazione dinamica

Concludiamo il capitolo sulla programmazione dinamica commentando la complessità computazionale in tempo derivata con tale paradigma. Ricordiamo che gli algoritmi presentati per le sottosequenze nel Paragrafo 6.3 e per gli alberi nei Paragrafi 6.6 e 6.7 hanno complessità polinomiale nel numero di elementi in ingresso.

Gli altri problemi hanno invece algoritmi il cui costo dipende anche da alcuni dei *valori* degli elementi, non solo dal loro *numero*. Per il problema del resto e per il problema della partizione di n interi di somma totale $2s$, abbiamo un costo pari a $O(mR)$ per il primo e $O(ns)$ per il secondo, che sono polinomiali in m, n, s e R ma *non* lo sono necessariamente nella dimensione dei dati di ingresso. Ciò deriva dal fatto che s e R sono valori che possono essere esponenzialmente più grandi del numero di elementi coinvolti.

Prendendo come riferimento il problema della partizione, pur avendo n interi da partizionare, ciascuno di essi richiede $k = O(\log s)$ bit di rappresentazione. Quindi la dimensione dei dati è nk mentre il costo dell'algoritmo è $O(ns) = O(n2^k)$: tale costo non è polinomiale rispetto alla dimensione dei dati e, per questo motivo, l'algoritmo viene detto **pseudo-polinomiale**, in quanto il suo costo è polinomiale solo se si usano interi piccoli rispetto a n (per esempio, quando $s = O(n^c)$ per una costante $c > 0$). Anche l'algoritmo discusso per il problema della bisaccia è pseudo-polinomiale in quanto richiede $O(n \times \text{possanza}) = O(n2^k)$ tempo, mentre la dimensione dei dati in ingresso richiede $O(nk)$ bit dove $k = O(\log(\text{possanza}))$. Anche in questo caso, il costo dell'algoritmo non è polinomiale, ma lo diviene nel momento in cui il valore della possanza è polinomiale rispetto al numero di oggetti. Notare che anche la soluzione immediata di costo $O(2^n nk)$ è pseudo-polinomiale, visto che k può essere molto grande rispetto a n, mentre il rilassamento della bisaccia con elementi frazionabili è polinomiale. In generale, per tutti i problemi che coinvolgono quantità numeriche che possono crescere velocemente rispetto al numero n dei dati in ingresso, è opportuno adottare il costo non-uniforme per il modello RAM, in cui ciascuna operazione su interi di k bit richiede un tempo di esecuzione che aumenta al crescere di k.

Da ciò consegue che, mentre gli algoritmi sulle sotto-sequenze e sugli alberi visti in questo capitolo sono polinomiali a tutti gli effetti, gli altri algoritmi sono solo apparentemente polinomiali: l'anomalia deriva dal fatto che i rispettivi problemi sono NP-completi, come vedremo nel seguito del libro (da notare, però, che non è vero che ogni problema NP-completo ammetta un algoritmo pseudo-polinomiale).