Progettazione di algoritmi

Programmazione Dinamica

ESERCIZIO 1

Dato un intero n vogliamo contare quante sono le stringhe binarie lunghe n in cui non compaiono 2 zeri consecutivi.
L'algoritmo proposto deve avere complessità $O(n)$.

Ad esempio:

- per $n = 1$ sono 2 (0, 1)
- per $n = 3$ sono 5 (010, 011, 101, 110, 111)

ESERCIZIO 2

Dato un intero n vogliamo contare quante sono le stringhe binarie lunghe n in cui non compaiono 3 zeri consecutivi.
L'algoritmo proposto deve avere complessità $O(n)$.

Ad esempio:

- per $n = 1$ sono 2 (0, 1)
- per $n = 4$ sono 13 (0100, 0010, 0011, 0101, 0110, 1010, 1001, 1100, 0111, 1011, 1101, 1110, 1111)

ESERCIZIO 3

Dato un intero n vogliamo contare quanti modi diversi ci sono di sistemare n persone in un albergo che dispone di camere singole e camere doppie.
L'algoritmo proposto deve avere complessità $O(n)$.

Ad esempio:

- per $n = 2$ sono 2:
  - $\{[1], [2]\}$,
  - $\{[1, 2]\}$
- per $n = 4$ sono 10:
  - $\{[1], [2], [3], [4]\}$,
  - $\{[1, 2], [3], [4]\}$,
  - $\{[1, 3], [2], [4]\}$,
  - $\{[1, 4], [2], [3]\}$,
  - $\{[2, 3], [1], [4]\}$,
  - $\{[2, 4], [1], [3]\}$,
  - $\{[3, 4], [1], [2]\}$,
  - $\{[1, 2], [3, 4]\}$,
  - $\{[1, 3], [2, 4]\}$,
  - $\{[1, 4], [2, 3]\}$

ESERCIZIO 4

Il valore di una lista di interi è dato dalla somma degli interi contenuti dalla lista.
Data una lista di n interi vogliamo trovare la sua sottolista di valore massimo.
L'algoritmo proposto deve avere complessità $O(n)$.

Ad esempio:

- se $lista = [3, -1, 2, 1]$ allora la risposta è $[3, -2, 2, 1]$
  di valore 4 che possiamo denotare con la tupla $(0, 3)$
- $lista = [-4, 1, 2, -5, 4, -3, 2, 2, -4, 1]$ allora la risposta è $[-4, 1, 2, -5, 4, -3, 2, 2, -4, 1]$
  di valore 5 che possiamo denotare con la tupla $(4, 7)$

ESERCIZIO 5

Data una sequenza di n interi vogliamo trovare una sottosequenza strettamente crescente di lunghezza massima.
L'algoritmo proposto deve avere complessità $O(n^2)$.

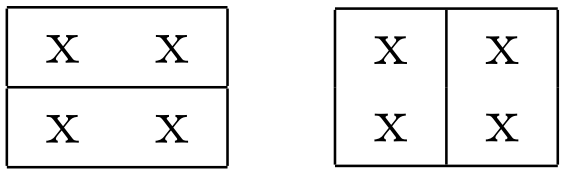
Ad esempio:

- per $[5, 2, 1, -1]$ le sottosequenze di lunghezza massima hanno lunghezza 1 (ad esempio $[5, \_, \_, \_, \_]$ oppure $[\_, 2, \_, \_, \_]$)
- per $[1, 3, 8, 9]$ esiste un'unica sottosequenza di lunghezza massima, la sua lunghezza è 4 ed è la sequenza stessa.
- per $[5, 1, 8, 4, 1, 8, 6, 3, 7, 3, 2]$ esiste un'unica sottosequenza di lunghezza massima e la sua lunghezza è 4 ($[\_, 1, \_, 4, \_, \_, 6, \_, 7, \_, \_]$)

ESERCIZIO 6

Dato l'intero n vogliamo contare il numero di differenti tassellamenti di una superficie di dimensione $2 \times n$ tramite tessere di domino di dimensione 1×2. L'algoritmo deve avere complessità $O(n)$.

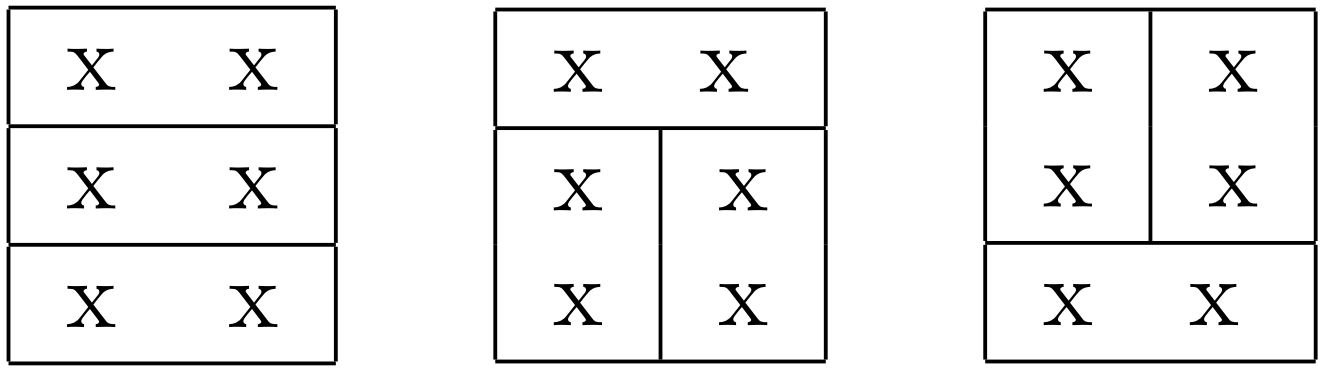
Ad esempio: per $n = 1$ la risposta dell'algoritmo deve ovviamente essere 1 mentre per $n = 2$ la risposta deve essere 2 perché sono possibili i soli due seguenti tassellamenti:



ESERCIZIO 7

Dato l'intero n vogliamo contare il numero di differenti tassellamenti di una superficie di dimensione $3 \times n$ tramite tessere di domino di dimensione 1×2. L'algoritmo deve avere complessità $O(n)$.

Ad esempio: per $n = 1$ la risposta dell'algoritmo deve ovviamente essere 0 mentre per $n = 2$ la risposta deve essere 3 perché sono possibili i soli tre seguenti tassellamenti:



ESERCIZIO 8

in un gioco a due un giocatore ha una *strategia vincente* se può vincere indipendentemente dalle mosse effettuate dall'avversario.

Su di un tavolo vi sono n gettoni. Due giocatori, a turno, possono prelevare dal tavolo 1, 3 o 4 gettoni. Vince il gioco il primo giocatore che si ritrova senza possibilità di prelevare gettoni (i.e. al suo turno sul tavolo non ci sono più gettoni). Bisogna determinare se il giocatore che effettua la prima mossa mossa ha una strategia vincente.
L'algoritmo deve avere avere complessità $O(n)$.

Ad esempio:

- Per $n = 5$ il primo giocatore ha una strategia vincente (preleva 4 gettoni, il suo avversario sarà costretto poi a prelevare l'unico gettone rimasto facendolo vincere).
- Per $n = 3$ il primo giocatore non ha una strategia vincente. Può infatti prelevare 3 o 1 gettoni nel primo caso fa vincere l'avversario, nel secondo lascia all'avversario 2 gettoni, l'avversario ne preleva 1 e vince in quanto nella mossa successiva il primo giocatore non può che prelevare l'ultimo gettone rimasto.

ESERCIZIO 9

Un numero intero può sempre essere rappresentato come somma di quadrati di altri numeri interi. Infatti, usando il quadrato 1^2, il generico numero x possiamo sempre scomporlo come somma di n addendi tutti uguali a 1^2(vale a dire $x = 1^2 + \ldots + 1^2$).

Dato un intero n bisogna scoprire qual è il numero minimo di quadrati necessari ad ottenere n.
La complessità dell'algoritmo deve essere $O\left(n^{\frac{3}{2}}\right)$.

Ad esempio:

- per $n = 0$ la risposta è 0.
- per $n = 100$ la risposta è 1 infatti $100 = 10^2$. Nota che vale anche $100 = 5^2 + 5^2 + 5^2 + 5^2$ ma questa scomposizione usa 4 quadrati e non è minimale.
- per $n = 6$ la risposta è 3 infatti $6 = 2^2 + 1^2 + 1^2$ e non è difficile vedere che 6 non può esprimersi come somma di due soli quadrati.