Corso di Progettazione di algoritmi

Esercizi 6

Esercizio 1

Viene dato in input un intero positivo n. Scrivere lo pseudocodice di un algoritmo che in tempo $O(n)$ restituisce il numero di stringhe binarie lunghe n in cui non compaiono mai uni consecutivi.

Ad esempio per $n = 4$ deve essere restituito 8, le stringhe possibili sono infatti:

0000 1000 0100 0010 0001 1010 1001 0101

Esercizio 2

Viene dato in input un intero positivo n. Scrivere lo pseudocodice di un algoritmo che in tempo $O(n)$ restituisce il numero di stringhe binarie lunghe n in cui non compaiono mai tre uni consecutivi.

Ad esempio per $n = 4$ deve essere restituito 8, le stringhe possibili sono infatti:

0000 1000 0100 0010 0001 1010 1001 0101 0011 1011 0110 1100 1101

Esercizio 3

Si consideri una sequenza di interi X. La cancellazione da X di un certo numero di elementi determina una *sottosequenza*. La sottosequenza è crescente se il valore dei suoi elementi è crescente. La lunghezza della sottosequenza è il numero di elementi che la compongono. Il valore della sottosequenza è dato dalla somma dei valori degli elementi che la compongono.
Data la sequenza $X = (x_1, x_2, \ldots x_n)$ a valori distinti si considerino i seguenti tre problemi:

a. Trovare la sottosequenza crescente di X di lunghezza massima.
Ad esempio per $X = (50, 4, 1002, 48, 3, 34, 30)$ la sottosequenza crescente di lunghezza massima è $(2, 3, 34)$.

b. Trovare la sottosequenza crescente di X di valore massimo.
Ad esempio per $X = (50, 2, 100, 1, 20, 30)$ la sottosequenza crescente di valore massimo è $(50, 100)$.

c. Trovare il numero delle sottosequenze crescenti di X.
Ad esempio per $X = (5, 3, 7, 8, 6)$ il numero di sottosequenze crescenti è 14 (le sottosequenze sono: (5), $(5, 7)$, $(5, 7, 8)$, $(5, 8)$, $(5, 6)$, (3), $(3, 7)$, $(3, 7, 8)$, $(3, 8)$, $(3, 6)$, (7), $(7, 8)$, (8), (6).

c. Dato l'intero k, con $k \leq n$, trovare il numero delle sottosequenze crescenti di X.
Ad esempio per $k = 2$ e $X = (5, 3, 7, 8, 6)$ il numero di sottosequenze è 7 (le sottosequenze sono: $(5, 7)$, $(5, 8)$, $(5, 6)$, $(3, 7)$, $(3, 8)$, $(3, 6)$, $(7, 8)$).

Progettare quattro algoritmi che risolvono i quattro problemi. La complessità dei primi tre algoritmi deve essere $O(n^2)$, la complessità del quarto deve essere $O(n^2 k)$

Esercizio 4

Data una matrice quadrata binaria M di dimensione $n \times n$ si vuole sapere qual'è il massimo m per cui la matrice quadrata $m \times m$ di soli uni risulta sottomatrice di M.

- provare che il numero di sottomatrici quadrate presenti in una matrice quadrata$n \times n$ è $\Theta(n^3)$ calcolando esattamente il loro numero.
- Descrivere un algoritmo che, data la matrice M, risolve il problema in tempo $O(n^3)$.

Ad esempio: per la matrice $M =$

1	0	1	1	1
1	1	1	1	1
1	1	1	0	1
1	1	1	1	1
1	1	0	1	1

di dimensione 5×5 la risposta è 3 (e gli elementi della sottomatrice 3×3 sono in blu).

Esercizio 5

Data una matrice $n \times m$ di interi $M = [m_{i,j}]$, una M-lista è una sequenza $(m_{1,j_1}, m_{2,j_2} \ldots m_{n,j_m})$ tale che $1 \leq j_1 \leq \ldots j_m \leq m$. Il valore di una M-lista è la somma degli elementi che la compongono.
Progettare un algoritmo che trova un M-lista di valore minimo in $\mathrm{O}(n \cdot m)$.

Esercizio 6

Dato un insieme di m stringhe binarie dette *primitive* ed una stringa di n bit X, vogliamo sapere se la stringa X si può ottenere dalla concatenazione di stringhe primitive.
Ad esempio: dato l'insieme di primitive $\{01, 10, 011, 101\}$, per la stringa $X = 0111010101$ la risposta è sì (due possibili soluzioni sono $011 - 10 - 10 - 101$ e $011 - 101 - 01 - 01$) mentre per la stringa $X = 0110001$ la risposta è no.

a) Descrivere un algoritmo che, data la stringa X e le m primitive, risolve il problema in $O((m + n) \cdot l)$ dove l è la lunghezza massima per le stringhe primitive.

b) Modificare l'algoritmo proposto in modo che, nel caso X sia ottenibile dalla concatenazione di primitive, vengano prodotti in output gli indici delle primitive la cui concatenazione genera X.
Ad esempio: data la stringa $X = 0111010101$ e le primitive $Y_1 = 01, Y_2 = 10, Y_3 = 011$ e $Y_4 = 101$, l'algoritmo deve produrre in output la sequenza $3, 2, 2, 4$ o la sequenza $3, 4, 1, 1$.

Esercizio 7

Data una matrice binaria di dimensioni $n \times n$ vogliamo verificare se nella matrice è possibile raggiungere la cella in basso a destra partendo da quella in alto a sinistra senza mai toccare celle che contengono il numero 1.

Si tenga conto che dalla generica cella (i, j) ci si può spostare solo nella cella in basso (vale a dire la cella $(i + i, j)$) o nella cella a destra (vale a dire la cella $(i, j + 1)$).

Progettare un algoritmo che risolve il problema in tempo $O(n^2)$

Ad esempio: per la matrice A la risposta deve essere SI mentre per la matrice B la risposta deve essere NO.

$$A = \begin{array}{|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 1 & 1 & 0 & 1 & 0 & 0 \\ \hline \end{array} \qquad B = \begin{array}{|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 1 & 1 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \\ \hline \end{array}$$

Esercizio 8

Data una matrice di interi positivi e dimensione $n \times n$ vogliamo contare il numero di cammini di costo k che partono dalla cella in alto a sinistra e raggiungono la cella in basso a destra.

Il costo di un cammino è dato dalla somma dei valori delle celle toccate dal cammino inoltre, nel corso del cammino, dalla generica cella (i, j) ci si può spostare nella cella in basso (vale a dire la cella $(i + i, j)$) o nella cella a destra (vale a dire la cella $(i, j + 1)$).

Progettare un algoritmo che risolve il problema in tempo $O(n^2 k)$

Ad esempio: per la matrice

1	2	3
4	6	5
3	2	1

e $k = 12$ la risposta è 2. I cammini sono:

$$1 \to 2 \to 6 \to 2 \to 1$$

$$1 \to 2 \to 3 \to 5 \to 1$$

Esercizio 9

Data una matrice di dimensione $n \times n$ le cui celle sono numerate con numeri distinti che vanno da 1 a n^2, vogliamo trovare la massima lunghezza possibile per cammini che toccano celle con numerazione crescente e incremento di 1.

I cammini possono partire da una qualunque cella e, nel corso del cammino, dalla generica cella (i, j) ci si può spostare in una qualunque cella adiacente in orizzontale o verticale (vale a dire in una delle celle $(i, j+1), (i+1, j), (i, j-1), (i-1, j)$). La lunghezza di un cammino è data dal numero di nodi toccati dal cammino.

Progettare un algoritmo che risolve il problema in tempo $O(n^2)$

Ad esempio: per la matrice A la risposta è 1 mentre per la matrice B, grazie al cammino $2 \to 3 \to 4 \to 5 \to 6 \to 7$, la risposta è 6

$$A = \begin{array}{|c|c|c|} \hline 3 & 6 & 2 \\ \hline 7 & 1 & 9 \\ \hline 4 & 8 & 5 \\ \hline \end{array} \qquad B = \begin{array}{|c|c|c|} \hline 9 & 7 & 6 \\ \hline 8 & 2 & 5 \\ \hline 1 & 3 & 4 \\ \hline \end{array}$$

Esercizio 10

Data una sequenza di n interi positivi X e un intero positivo s vogliamo trovare la più lunga sottosequenza di X di somma s.
Ad esempio se $X = (5, 2, 2, 6, 1, 7, 3, 5, 11, 3, 6)$ e $s = 25$, la più lunga sottosequenza è lunga 7 $(5, 2, 2, , 1, 7, 3, 5, , ,)$; se $X = (3, 3, 5, 13, 3, 5)$ e $s = 28$, non ci sono sottosequenze di somma 28.

- Dare lo pseudo-codice di un algoritmo che dati X e s ritorna la lunghezza massima di una sottosequenza di somma s di X in tempo $O(ns)$ (se non ci sono sottosequenze di somma s, ritorna 0).
- Dare poi lo pseudo-codice di un algoritmo che ritorna una sottosequenza di lunghezza massima per X e s.

Esercizio 11

Abbiamo una sequenza $S = (s_1, s_2, \ldots, s_n)$ di interi positivi.
Una sottosequenza S' di S si definisce *valida* se per ogni coppia di elementi consecutivi di S almeno un elemento della coppia compare in S'.
Il valore di una sottosequenza valida è la somma dei suoi elementi.
Ad esempio: per $S = (1, 2, 3, 5, 4, 6, 7)$, la sottosequenza $S' = (1, 3, 6)$ non è valida, mentre la sottosequenza $S' = (2, 5, 4, 7)$ è valida ed ha valore valore 18 e la sottosequenza $S'' = (2, 3, 4, 6)$ è valida ed ha valore 15.

- Descrivere un algoritmo che, data la sequenza S, calcola il valore minimo di una sottosequenza valida in tempo $O(n)$.
- Descrivere poi un algoritmo che trova una sottosequenza valida di valore minimo.

Esercizio 12

Data una sequenza S di interi positivi si vuole trovare una sottosequenza di S senza elementi in posizioni consecutive e la somma dei cui elementi sia massima. Ad esempio, per $S = (1, 5, 4, 6, 10, 3, 2, 9)$ una soluzione è $(-, 5, -, -, 10, -, -, 9)$ e vale 24 (anche $(1, -, 4, -, 10, -, -, 9)$ è una soluzione), mentre $(1, -, -, 6, 10, -, -, 9)$ che vale 26 non è una sottosequenza ammissibile perché contiene gli elementi 6 e 10 in posizioni consecutive.

- Dare lo pseudo-codice di un algoritmo che risolve il problema in $O(n)$.
- Dare poi lo pseudo-codice di un algoritmo che trova una sottosequenza ottimale.

Esercizio 13

Si considerino 3 stringhe A, B e C, di lunghezza n, m ed $n+m$, rispettivamente. Diciamo che C è l'intreccio di A e B se contiene tutti i caratteri di A e tutti i caratteri di B e l'ordine di tutti i caratteri delle stringhe individuali è preservato.

Ad esempio se $A = aabxxz, B = abxy$ e $C = abaaxbxyxz$ allora C è l'intreccio di A e B infatti: $C = abaaxbxyxz$

- Assumendo che A e B non hanno caratteri in comune, dare lo pseudocodice di un algoritmo che, in tempo $O(n + m)$, determina se C è l'intreccio di A e B.
- Dare lo pseudocodice di un algoritmo che, in tempo $O(nm)$, determina se C è l'intreccio di A e B.

Esercizio 14

Si hanno n attività e per ogni attività, $1 \leq i \leq n$, l'intervallo temporale $[s_i, f_i)$ in cui l'attività dovrebbe svolgersi e il guadagno v_i che si ottiene dallo svolgimento dell'attività. Due attività i e j sono compatibili se gli intervalli temporali $[s_i, f_i)$ e$[s_j, f_j)$ sono disgiunti ed il valore di un sottoinsieme di attività è dato dalla somma dei valori delle attività del sottoinsieme. Vogliamo selezionare un sottoinsieme S di valore massimo di attività tra loro compatibili. Descrivere un algoritmo che risolve il problema in $\mathrm{O}(n^2)$.

Esercizio 15

Una banda di tre ladri deve spartirsi il frutto di una rapina di n oggetti ciascuno caratterizzato da un intero positivo che ne rappresenta il valore. 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 complessità $O(n \cdot M^2)$.

Esercizio 16

Vogliamo visitare in sequenza n città. Per la visita di ciascuna città è previsto il pagamento di una tassa a seguito del pagamento della tassa si riceve un bonus per l'esenzione dal pagamento di tasse successive. Ogni esenzione può essere usata una sola volta.

a. Descrivere un algoritmo basato sulla tecnica della programmazione dinamica che, dati i costi $t_1, t_2, \ldots, t_n$ delle n tasse determina la tassa totale minima che un viaggiatore deve pagare per visitare tutte le n città. La complessità dell'algoritmo deve essere $O(n^2)$.

b. Per minimizzare la tassa totale viene proposta la seguente strategia greedy: *all'arrivo nella generica città i la tassa va pagate se e solo se non si dispone di esenzioni o si dispone di $j \geq 1$ esenzioni e bisogna ancora visitare almeno j città che richiedono una tassa superiore a t_i.* Verificare se la strategia greedy minimizza effettivamente i costi.

Esercizio 17

Vengono dati in input tre interi positivi x_1, x_2 e x_3, con $x_1 < x_2 < x_3$, ed un intero positivo n.

a) Scrivere lo pseudocodice di un algoritmo che in tempo $O(n)$ restituisce il numero di sequenze sull'alfabeto $\{x_1, x_2, x_3\}$ la somma dei cui elementi è n.

Ad esempio per $x_1 = 2, x_2 = 4, x_3 = 8$ e $n = 10$ la risposta deve essere 10, le uniche sequenze possibili sono infatti:

$2,8 \quad 8,2 \quad 2,4,4 \quad 4,2,4 \quad 4,4,2 \quad 2,2,2,4 \quad 2,2,4,2 \quad 2,4,2,2 \quad 4,2,2,2 \quad 2,2,2,2$

b) Scrivere lo pseudocodice di un algoritmo che in tempo $O(n)$ restituisce il numero di multiinsiemi sull'alfabeto $\{x_1, x_2, x_3\}$ la somma dei cui elementi è n.

Ad esempio per $x_1 = 2, x_2 = 4, x_3 = 8$ e $n = 10$ la risposta deve essere 4, gli unici multiinsiemi possibili sono infatti:

$2,8 \quad 2,4,4 \quad 2,2,2,4 \quad 2,2,2,2$

Esercizio 18

Date due sequenze $X = (x_1, \ldots, x_n)$ e $Y = (y_1, \ldots, y_m)$ una supersequenza di X e Y è una qualsiasi sequenza Z tale che sia X che Y sono sottosequenze di Z.
Ad esempio, per le sequenze di lettere *alberi* e *libri* le seguenti sono supersequenze: *alberilibri, albelibri, lialberi, a liberi.*

- Dare lo pseudo-codice di un algoritmo che, date due sequenze X e Y, di lunghezze n ed m, calcola la lunghezza minima di una supersequenza di X e Y in $O(nm)$.
- Dare poi lo pseudo-codice di un algoritmo che ritorna una supersequenza di lunghezza minima di X e Y.

Esercizio 19

In una sequenza $S = (a_1, a_2 \dots a_n)$ di interi positivi, l'intero a_i rappresenta il prezzo di una certa merce fra i giorni, si vuole sapere qual'è il giorno i in cui conviene comprare la merce ed il giorno j, con $j \geq i$, in cui conviene rivenderla in modo da massimizzare il profitto. In altre parole siamo interessati a conoscere la coppia (i, j) con $i \leq j$ per cui risulta massimo il valore $a_j - a_i$.
Descrivere un algoritmo che risolve il problema in $\mathrm{O}(n)$ tempo.

Esercizio 20

Data una sequenza $S = (a_1, a_2, \ldots a_n)$ di interi, sia positivi che negativi ma non zero, una sottosequenza di indici $1 \leq i_1 < i_2 \ldots < i_m \leq n$ di S è detta alternante se per ogni $k = 1, \ldots, m-1$ il segno di a_{i_k} è diverso dal segno di $a_{i_{k+1}}$. Chiaramente ogni sottosequenza di lunghezza 1 è alternante. Data una sequenza S di interi (negativi e positivi ma non zero) vogliamo trovare una sottosequenza alternante di somma massima. Ad esempio, se $S = (2, -3, 4, 5, -4, 3, -3, 5, -2, 1)$ una sottosequenza alternante di somma massima è quella di indici $4, 7, 8$ con somma 7.

- Dare lo pseudo-codice di un algoritmo che calcola la somma massima di una sottosequenza alternante in $O(n^2)$
- Dare poi lo pseudo-codice di un algoritmo che trova una sottosequenza alternante di somma massima (come lista degli indici).

Esercizio 21

Sia $A = \{a_1, a_2, \ldots a_n\}$ un insieme di n esami, dove per ogni $i = 1, 2, \ldots n$, l'esame a_i vale c_i crediti e si supponga di avere per ogni esame a_i un coefficiente d_i che rappresenta il grado di difficoltà dell'esame. Ogni studente può redigere il proprio piano di studio individuale scegliendo nella lista degli esami attivati un insieme di esami tali che la somma dei crediti corrispondenti sia almeno P. Progettare un algoritmo che redige un piano di studi regolare di difficoltà minima in $\mathrm{O}(n \cdot P)$.

Esercizio 22

Nella versione classica abbiamo uno Zaino di capacità C ed n oggetti ciascuno con un suo valore v_i ed un suo peso p_i, $1 \leq i \leq n$. Possiamo mettere nello zaino un qualunque sottoinsieme degli oggetti il cui peso complessivo non superi la capacità C e vogliamo trovare il sottoinsieme di valore massimo. Nella versione rivisitata disponiamo di un numero arbitrario di copie di ciascuno degli n oggetti e quindi più copie di uno stesso oggetto possono essere inserite nello zaino allo scopo di ottenere una soluzione di valore massimo.
Ad esempio per uno zaino di capacità $C = 18$ e tre oggetti con valore e peso specificati dalle seguenti tabelle

	p_1	p_2	p_3
peso	9	5	4

	v_1	v_2	v_3
valore	6	4	3

la combinazione di oggetti che massimizza il valore dello zaino rispettandone la capacità è quella che prende due copie del secondo oggetto e due copie del terzo (questa combinazione ha peso 18 e valore 14).

- Dare lo pseudo-codice di un algoritmo che calcola il valore della soluzione ottima in $O(nC)$.
- Dare poi lo pseudo-codice di un algoritmo che produce una soluzione ottimale restituendo in output il numero di copie di ciascun oggetto da inserire nello zaino (per l'esempio l'algoritmo restituirà il vettore $(0, 2, 2)$).

Esercizio 23

Una stringa è palindroma se non cambia leggendola da sinistra a destra o viceversa.
Data una stringa $S = a_1 a_2 \ldots a_n$ di n caratteri si considerino i seguenti problemi:

a. Calcolare il minimo numero di caratteri che occorre inserire per rendere S palindroma.

b. Determinare la lunghezza della più lunga sottostringa palindroma di S.

Progettare due algoritmi che risolvono i due problemi.
La complessità degli algoritmi deve essere $\mathrm{O}(n^2)$.

Esercizio 24

Si ha una sequenza di n carte, ciascuna carta ha come valore un numero intero. Due giocatori a turno prendono una della carte da uno degli estremi della sequenza. Al termine del gioco il punteggio di ciascun giocatore è dato dalla somma dei valori delle carte da lui prese. Lo scopo del gioco è ottenere il punteggio massimo.
Dati gli n valori $v_1, v_2, \ldots v_n$ della sequenza all'inizio del gioco, progettare un algoritmo che in tempo $O(n^2)$ determini se il primo giocatore dispone di una *strategia vincente*. Un giocatore dispone di una strategia vincente se ha la possibilità di vincere qualunque sia la sequenza di mosse effettuata dal suo avversario. (per semplicità si può assumere che n sia un numero pari).

Esercizio 25

Abbiamo una scacchiera composta di tre righe ed n colonne. Ogni casella della scacchiera è contrassegnata da un intero positivo . In ciascuna casella della scacchiera è possibile piazzare una pedina. Un piazzamento di pedine è *regolare* se non prevede due caselle con pedina adiacenti in orizzontale o in verticale (l'adiacenza di caselle con pedine adiacenti in diagonale non crea invece alcun problema). Il valore del piazzamento è dato dalla somma dei contrassegni delle caselle in cui è stata posizionata una pedina.

- Progettare un algoritmo che, data la matrice M con il valore dei contrassegni delle $3 \times n$ caselle, determina in tempo $O(n)$ il valore del piazzamento regolare di valore massimo.

- Assumendo che i contrasegni delle caselle possano avere anche valore negativo. Progettare un algoritmo che, data la matrice M con il valore dei contrassegni delle $3 \times n$ caselle, determina in tempo $O(n)$ il piazzamento regolare di valore massimo.

Esercizio 26

Abbiamo una matrice M di interi (non necessariamente positivi) di dimensione $n \times n$. Il valore di una sua sottomatrice (non necessariamente quadrata) è dato dalla somma dei suoi elementi. Si vuole trovare una sottomatrice quadrata di M la somma dei cui elementi sia massima.

Ad esempio per la seguente matrice di dimensione 4×4

$$\begin{array}{rrrr} -20 & 10 & -10 & 30 \\ 15 & -6 & 20 & -10 \\ 20 & 10 & 5 & 0 \\ 1 & -5 & -10 & 20 \end{array}$$

la sottomatrice cercata ha valore 64 e dimensione 2×3, i suoi elementi sono sottolineati

a) Provare che il numero di sottomatrici (non necessariamente quadrate) presenti in una matrice $n \times n$ è $O(n^4)$ calcolando esattamente il loro numero.

b) Descrivere un algoritmo che, data la matrice M, risolve il problema in $O(n^3)$ tempo (la sottomatrice da restituire può essere rappresentata dando le coordinate della sua cella in basso a destra, la sua altezza e la sua larghezza).

Esercizio 27

Abbiamo una scacchiera M composta di 4 righe ed n colonne. Ogni casella della scacchiera è contrassegnata da un intero positivo. In ciascuna casella della scacchiera è possibile piazzare una pedina.
Un piazzamento di pedine è regolare se non prevede due caselle con pedine adiacenti in orizzontale o in verticale (l'adiacenza di pedine in diagonale non crea invece alcun problema).
Il valore del piazzamento è dato dalla somma dei contrassegni delle caselle in cui è stata posizionata una pedina.
Descrivere un algoritmo che, dati i valori delle $4 \times n$ caselle della scacchiera, in $O(n)$ tempo determina un piazzamento regolare di valore massimo.

Ad esempio: per la scacchiera

30	6	5	2	10
1	7	1	3	28
15	30	4	20	1
20	2	8	10	6

con $n = 5$ due possibili piazzamenti di pedine regolari sono

30	6	5	2	10
1	7	1	3	28
15	30	4	20	1
20	2	8	10	6

e

30	6	5	2	10
1	7	1	3	28
15	30	4	20	1
20	2	8	10	6

.

Il primo piazzamento vale 87, il secondo 90 ed esistono altri piazzamenti regolari di valore superiore.

Esercizio 28

Alice e Bob decidono di comunicare attraverso messaggi codificati. Viene concordato un codice binario comune. La lingua madre di Alice e Bob usa un alfabeto di sette lettere: A, B, C, D, E, F e G. Viene quindi deciso di assegnare un codice binario a ciascuna delle lettere suddette:

lettera	A	B	C	D	E	F	G
codifica	0	00	001	010	0010	0100	0110

Presto Alice e Bob realizzano che messaggi diversi sono codificati dalla stessa sequenza binaria.
Ad esempio, quando arriva la sequenza 00100, questa risulta ambigua in quanto può essere stata prodotta da ADA, AF, CAA, CB oppure EA. Con sequenze binarie più lunghe, il grado di ambiguità aumenta ulteriormente.
Inoltre, non tutte le sequenze binarie corrispondono a messaggi: per esempio, 00101 non codifica alcun messaggio.

Descrivere un algoritmo che, data una sequenza binaria $S = (a_1, a_2 \dots a_n)$, in tempo $O(n)$, calcola il numero di messaggi diversi che hanno codifica S.
Ad esempio, la sequenza $S = (0, 0, 0, 0, 0, 0)$ corrisponde a 13 messaggi distinti.