

11/01

Esercizio 2 (9 punti) Data una stringa di numeri interi $A = (a_1, a_2, \dots, a_n)$, si consideri la seguente ricorrenza $z(i, j)$ definita per ogni coppia di valori (i, j) con $1 \leq i, j \leq n$:

$$z(i, j) = \begin{cases} a_j & \text{if } i = 1, 1 \leq j \leq n, \\ a_{n+1-i} & \text{if } j = n, 1 < i \leq n, \\ z(i-1, j) \cdot z(i, j+1) \cdot z(i-1, j+1) & \text{altrimenti.} \end{cases}$$

1. Si fornisca il codice di un algoritmo iterativo bottom-up $Z(A)$ che, data in input la stringa A restituiscia in uscita il valore $z(n, 1)$.
2. Si valuti il numero esatto $T_Z(n)$ di moltiplicazioni tra interi eseguite dall'algoritmo sviluppato al punto (1).

1. Date le dipendenze tra gli indici nella ricorrenza, un modo corretto di riempire la tabella è attraverso una scansione "reverse column-major", in cui calcoliamo gli elementi della tabella in ordine decrescente di indice di colonna e, all'interno della stessa colonna, in ordine crescente di indice di riga. Il codice è il seguente.

```

Z(A)
n = length(A)
for i=1 to n do
    z[1,i] = a_i
    z[i,n] = a_{n+1-i}
for j=n-1 downto 1 do
    for i=2 to n do
        z[i,j] = z[i-1,j] * z[i,j+1] * z[i-1,j+1]
return z[n,1]

```

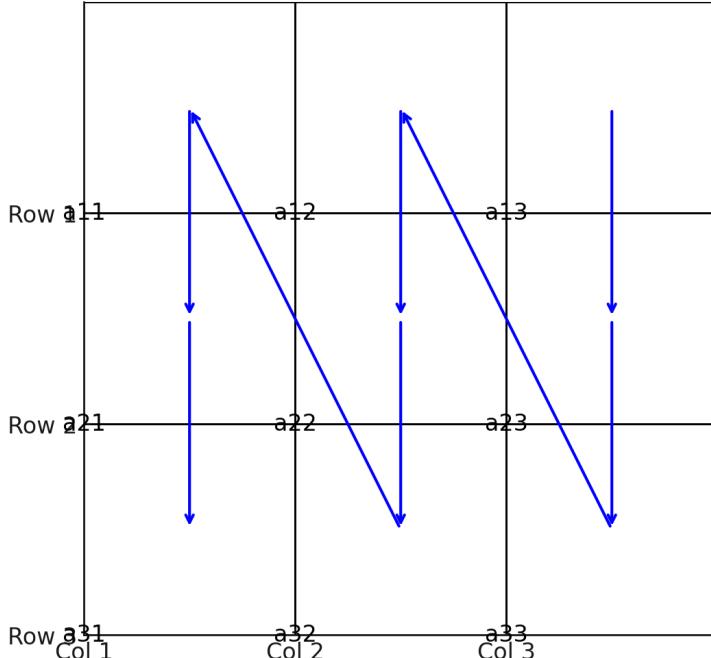
(ONS IS MSGL CHS RMO...)

1 solo indice = msglo
COMPUTAZIONE NAZIONALE

Si osservi che un altro modo corretto di riempire la tabella è attraverso una scansione "reverse diagonal", che scansiona per diagonali parallele alla diagonale principale partendo da quella contenente solo $z[1, n]$.

$$\sum_{j=1}^{n-1} \sum_{i=2}^m 2 \rightarrow 2(m+i)$$

Scansione reverse column-major

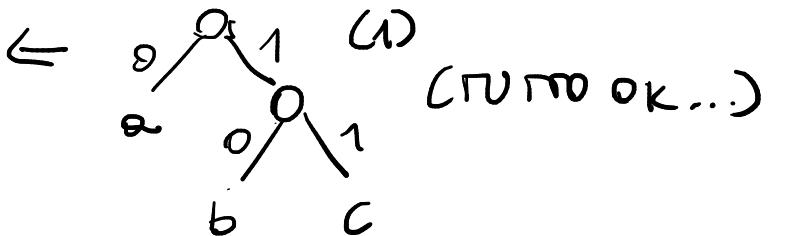


a ₁	a ₂	a ₃ (1)
? (6)	? (4)	a ₂ (2)
? (7)	? (5)	a ₁ (3)

Inizializzazione diretta (bo) Calcolato dalla formula

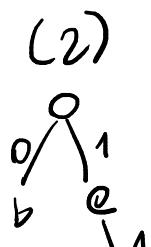
Esercizio 2 (9 punti) Si consideri un file definito sull'alfabeto $\Sigma = \{a, b, c\}$, con frequenze $f(a), f(b), f(c)$. Per ognuna delle seguenti codifiche si determini, se esiste, un opportuno assegnamento di valori alle 3 frequenze $f(a), f(b), f(c)$ per cui l'algoritmo di Huffman restituisce tale codifica, oppure si argomenti che tale codifica non è mai ottenibile.

$$1. e(a) = 0, e(b) = 10, e(c) = 11$$



$$\rightarrow 2. e(a) = 1, e(b) = 0, e(c) = 11$$

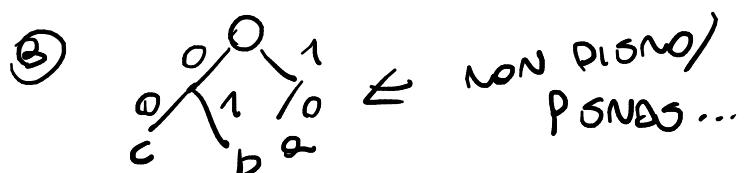
$$3. e(a) = 10, e(b) = 01, e(c) = 00$$



Soluzione:

- Questa codifica viene restituita dall'algoritmo di Huffman quando $f(b), f(c) < f(a)$: in questo caso i nodi associati a b e c vengono uniti creando un nuovo nodo interno, che poi viene unito al nodo associato ad a . Quindi, per esempio, $f(a) = 40, f(b) = 25, f(c) = 35$.
- La codeword $e(a) = 1$ è un prefisso della codeword $e(c) = 11$, cioè questa codifica non è libera da prefissi, quindi non è una codifica di Huffman.
- L'albero associato a questa codifica non è pieno perché c'è un nodo interno con un solo figlio (quello nel cammino che porta alla foglia associata al carattere a). Quindi questa codifica non è ottima e quindi non è una codifica di Huffman.

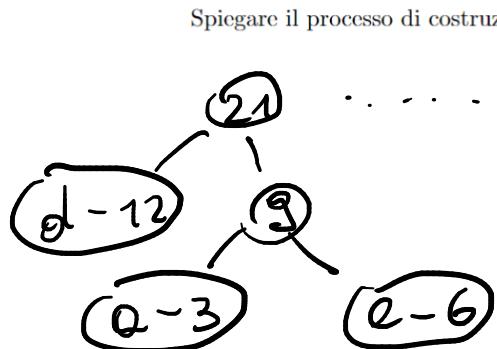
Preziosa...



Domanda 44 Indicare il codice prefisso ottenuto utilizzando l'algoritmo di Huffman per l'alfabeto $\{a, b, c, d, e, f, g\}$, supponendo che ogni simbolo appaia con le seguenti frequenze.

a	b	c	d	e	f	g
3	8	12	12	6	23	7

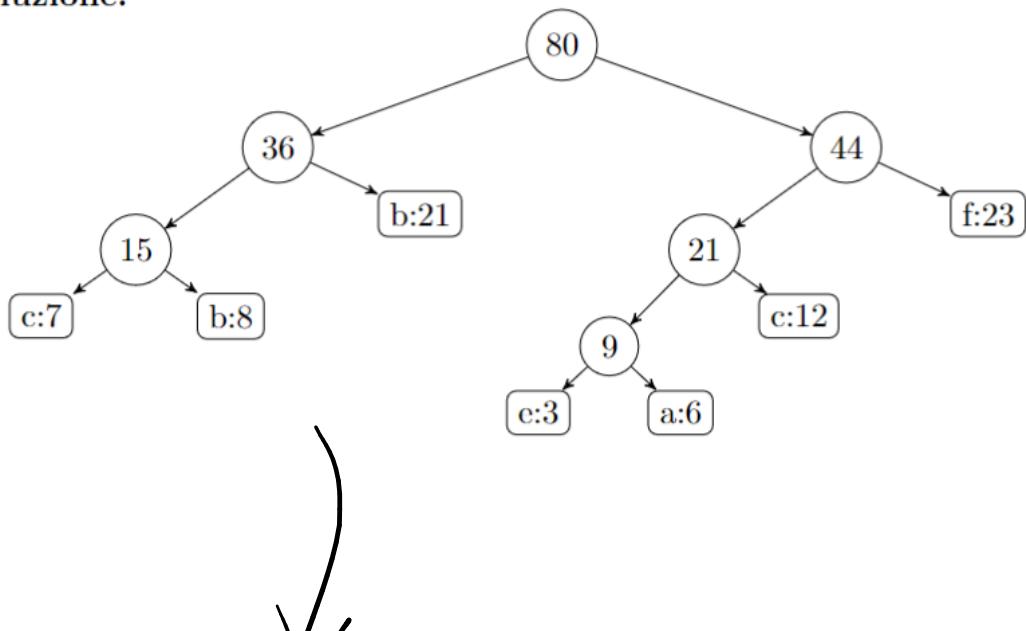
① FREQUENZE MINORI



② NODI COMPLETI SALVANDO

PENSANDO...

Soluzione:



Domanda B (6 punti) Si consideri una tabella hash di dimensione $m = 8$, e indirizzamento aperto con doppio hash basato sulle funzioni $h_1(k) = k \bmod m$ e $h_2(k) = 1 + 2(k \bmod (m - 2))$. Si descriva in dettaglio come avviene l'inserimento della sequenza di chiavi: 13, 29, 19, 27, 8.

$$\text{DOPPIO HASH: } f = [h_1(k) + i(h_2(k))] \bmod m$$

\emptyset quando no collisioni

- no collisions

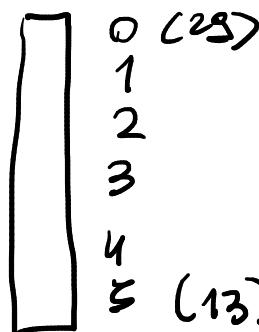
$$\rightarrow f = h_1 \bmod m$$

1 altrimenti si

- si collisions

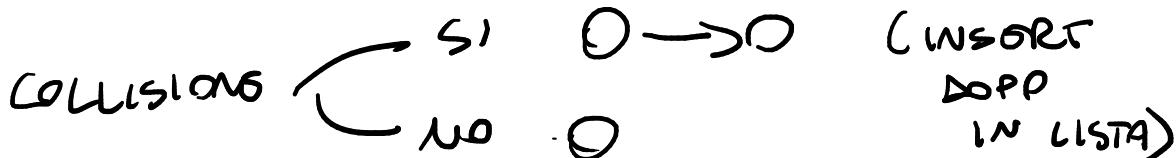
$$f = [h_1 + 1h_2] \bmod m$$

Domanda B (6 punti) Si consideri una tabella hash di dimensione $m = 8$, e indirizzamento aperto con doppio hash basato sulle funzioni $h_1(k) = k \bmod m$ e $h_2(k) = 1 + 2(k \bmod (m - 2))$. Si descriva in dettaglio come avviene l'inserimento della sequenza di chiavi: 13, 29, 19, 27, 8.



$$\begin{aligned} & [(29 \bmod 8) + 1(1 + 2(29 \bmod 6))] \bmod 8 \\ &= [5 + 1(1 + 2(5))] \bmod 8 \\ &= [5 + 11] \bmod 8 \\ &= 16 \bmod 8 = 0 \end{aligned}$$

HASHING \rightarrow USO DI TRABOCCHI



Domanda 34 Si consideri una tabella hash di dimensione $m = 8$, gestita mediante chaining (liste di trabocco) con funzione di hash $h(k) = k \bmod m$. Si descriva in dettaglio come avviene l'inserimento della sequenza di chiavi: 14, 10, 22, 18, 19.

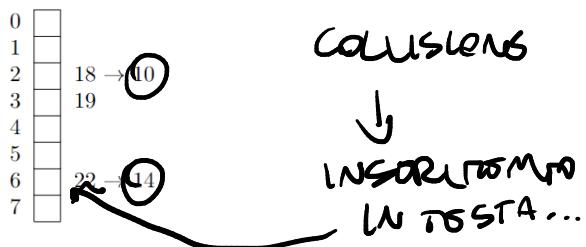


$$14 \bmod 8 = 6$$

$$10 \bmod 8 = 2$$

Soluzione: Si ottiene

$$22 \text{ mod } 8 = 6$$



Esercizio 5 Un array $A[1..n]$ di numeri si dice *alternante* se non ha elementi contigui identici (ovvero per ogni $i \leq n-1$ vale $A[i] \neq A[i+1]$) e inoltre per ogni $i \leq n-2$, vale che $[a_i < a_{i+1} > a_{i+2}]$ oppure $a_i > a_{i+1} < a_{i+2}$. Ad esempio gli array $[1, 2, -1, 3, 2]$ e $[5, 1, 2, -1, 3, 2]$ sono alternanti, mentre non lo sono $[1, 2, 3]$ e $[1, 1, 2]$. Scrivere una funzione ricorsiva $\text{alt}(A, n)$ che dato un array $A[1..n]$ di numeri verifica se è alternante. Valutarne la complessità.

$\text{alt}(A, n)$

1 2 1 3 2

return $\text{alt_rec}(A, 1, n)$

$\text{alt_rec}(A, i, n)$

if($\text{alt_rec}(A, i, i+2) == 0$ and $\text{alt_rec}(A, i, i+1) == 0$)
 return false

return true

$\text{altRec}(A, i, \text{dir})$
 # verifica se $A[1..i]$ è alternante.
 # usa un ulteriore parametro per indicare se la sequenza alternante deve

concludersi crescendo (0) o decrescendo (1).

```
if i==1  
    return True  
else  
    if dir==0  
        return  $\text{altRec}(A, i-1, 1)$  and ( $A[i-1] < A[i]$ )  
    else  
        return  $\text{altRec}(A, i-1, 0)$  and ( $A[i-1] > A[i]$ )
```

Dato che la parte non ricorsiva è di costo costante, la complessità si può esprimere come $T(n) = 1 + T(n-1)$ che porta a $T(n) = \Theta(n)$.

X CORRETTITÀ...

INVARIANTE = CONDIZ. DI

COSTANTITÀ
PER

CICLI
(For / While)

PRE / POST = CITTARANTE
RICOVERATO

Esercizio 2 (11 punti) Una longest common substring di due stringhe X e Y è una sottostringa di X e di Y di lunghezza massima. Si vuole progettare un algoritmo efficiente per calcolare la lunghezza di una longest common substring. Per semplicità si assume che entrambe le stringhe di input abbiano stessa lunghezza n .

↓ BRUTE FORCE (WORST CASE)

(n^2)

- (a) Qual è la complessità dell'algoritmo esauritivo che analizza tutte le possibili sottostringhe comuni?
- (b) Assumendo di conoscere un algoritmo che determina se una stringa di m caratteri è sottostringa di un'altra stringa di n caratteri in tempo $O(m+n)$, come si può modificare l'algoritmo del punto precedente per renderlo più efficiente?
MAX = MAX + SUBSTRING
- (c) Progettare un algoritmo di programmazione dinamica più efficiente di quello del punto precedente. Sono richiesti relazione di ricorrenza sulle lunghezze (senza dimostrazione) e algoritmo bottom-up. (Suggerimento: considerare la lunghezza della longest common substring dei prefissi $X_i = \langle x_1, \dots, x_i \rangle$ e $Y_j = \langle y_1, \dots, y_j \rangle$ che termina con x_i e y_j , rispettivamente.)

② $\sum X \subseteq Y$ $O(n^2)$

FOR $i=1$ TO X
 FOR $j=1$ TO Y

TUTTO DI TUTTE
 LE STRINGHE -- $[x_1 = x_5] \dots$ (n^2)

③

LCS(X, Y)

```

1  m = X.length
2  n = Y.length
3  for i = 0 to m
4      L[i, 0] = 0
5  for j = 0 to n
6      L[0, j] = 0
7  for i = 1 to m
8      for j = 1 to n
9          if  $x_i = y_j$ 
10             L[i, j] = L[i - 1, j - 1] + 1 (M = max(M, L[i-1,j-1]))
11             B[i, j] = '↖'
12         else if L[i - 1, j] ≥ L[i, j - 1]
13             L[i, j] = L[i - 1, j] (M = max(M, L[i-1,j]))
14             B[i, j] = '↑'
15         else
16             L[i, j] = L[i, j - 1] (M = max(M, L[i,j-1]))
17             B[i, j] = '←'
18 return (L[m, n], B)
    
```

↓ SALVANO MAX AD OGNI ITERAZIONE

$M + M$

④

$$l(i, j) = \begin{cases} 0 & \text{se } i = 0 \text{ o } j = 0 \quad (\text{caso 0}) \\ l(i - 1, j - 1) + 1 & \text{se } i, j > 0 \text{ e } x_i = x_j \quad (\text{caso 1}) \\ \max\{l(i, j - 1), l(i - 1, j)\} & \text{se } i, j > 0 \text{ e } x_i \neq x_j \quad (\text{caso 2}) \end{cases}$$

+

Alla fine ci interessa calcolare $l(m, n)$.

Per calcolare $l(i, j)$ mi possono servire tre valori:

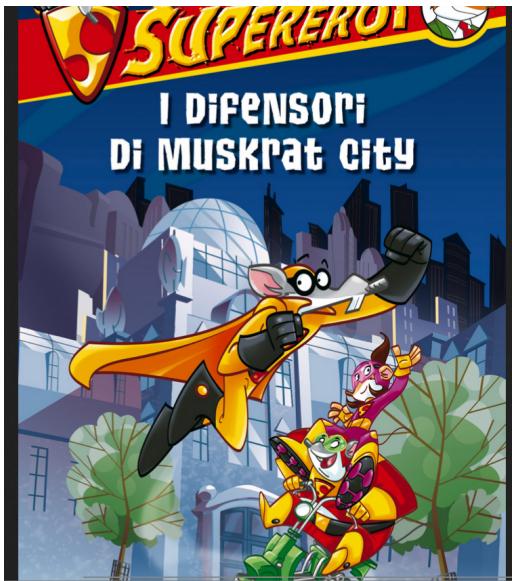
PSEUDOCODE...

$$L = \left[\begin{array}{cc} (i-1, j-1) & (i-1, j) \\ \nwarrow & \uparrow \\ (i, j-1) & \leftarrow (i, j) \end{array} \right]$$

Scansione "row-major": riempio la tabella per righe, da sinistra a destra.

Informazione addizionale per costruire la sequenza (vera e propria):

$$b(i, j) = \begin{cases} '↖' & \text{se } x_i = y_j \\ '←' & \text{se } x_i \neq y_j \text{ e } \max = LCS(i, j - 1) \\ '↑' & \text{se } x_i \neq y_j \text{ e } \max = LCS(i - 1, j) \end{cases}$$



Supersquizz !