24/11

Esercizio 1 (10 punti) Realizzare una funzione `Prod(A,k)` che dato un array `A` di interi ≥ 0 ordinato in senso crescente e un valore intero $k \geq 0$ verifica se esistono due indici `i` e `j` tali che `k = A[i]*A[j]`. Valutarne la complessità. Adattare la soluzione al caso in cui i valori nell'array possono essere negativi (assumendo ancora $k \geq 0$).

RIT. → K = A[i] * A[j]

A = [1, $\underbrace{2}_{i}$, $\underbrace{3}_{j}$, 4, 5, 6]   K = 6

PROD(A, K)

CONDIZIONE
// K = A[i] * A[j]

RETURN
PROD_REC(A, 1, n) → [PROD_REC(A, p, m)

(MODO PER CREARE DIVIDE ET IMPERA

q = ⌊p + m⌋ / 2

p → ... q ... ← m

A[p] * A[m]

p + 1 < q   m − 1 > q

PROD_REC(A, p, m, k)
q = ⌊p + m⌋ / 2

K = A[i] * A[j]
=
p * m

WHILE (A[p] * A[R] <> K)
IF (A[p] * A[R] < K)
M = M − 1

p $\boxed{q}$ m

1 2 $\underset{k}{\boxed{3}}$ 9 9 * 1 = 9

IF (A[P] * A[R] == K)

RETURN K; ↑ FOUND (CASO BASE)

IF (A[P] * A[R] < K)

RETURN PROD(A, P+1, M, K)

// ELSE (*)

A[P] * A[R] > K (*)

RETURN PROD_REC (A, P, M-1, K)

↓

DIM. CORRETT. → $\underline{P < Q < M}$

A[P-1] < A[P]

Esercizio: metric matching on the line
Sia $S = \{s_1, s_2, ..., s_n\}$ un insieme di punti ordinati sulla retta reale, rappresentanti dei server. Sia $C = \{c_1, c_2, ..., c_n\}$ un insieme di punti ordinati sulla retta reale, rappresentanti dei client. Il costo di assegnare un client c_i ad un server s_j è $|c_i - s_j|$. Si fornisca un algoritmo greedy che assegna ogni client ad un server distinto e che minimizzi il costo totale dell'assegnamento.

$S = \{S_1, S_2 ...\}$ = ORDINATO

$C = \{C_1, C_2 ... C_n\}$ $|C_i - S_j|$

↑ MAKESPAN

← DECIMALI ∈ ℝ

S = {0,2 / 0,3 / 0.1} ..
C = {0,8 / 0.7 / 0.9} ..] DISORDINATI

GREEDY_MATCHING (S, C)

(LINEA) | 6 4 8 |

(DIAG.) | 5 6 7 |

0,8 − 0,2 = 0.6

0.7 − 0.4 = 0.3

0.9 − 0.1 = 0.8

[O (n log n) → 2 ARRAY]

GREEDY_MATCHING (S, C)

① DECREASE_SORT (S)

② SORT (C)

[C_1 − S_3]
min

FOR (I = 1 TO N)

ASSUMENDO STESSA LUNGHEZZA

IF (C[I] − S[I] != NIL)

COST += C[I] − S[I]

END FOR

GOOD :)

Esercizio 2 (9 punti) Data una stringa $X = x_1, x_2, \ldots, x_n$, si consideri la seguente quantità $\ell(i,j)$, definita per $1 \le i \le j \le n$:

$$\ell(i,j) = \begin{cases} 1 & \text{se } i = j \\ 2 & \text{se } i = j-1 \\ 2 + \ell(i+1, j-1) & \text{se } (i < j-1) \text{ e } (x_i = x_j) \\ \sum_{k=i}^{j-1} (\ell(i,k) + \ell(k+1,j)) & \text{se } (i < j-1) \text{ e } (x_i \neq x_j). \end{cases}$$

1. Scrivere una coppia di algoritmi INIT_L(X) e REC_L(X, i, j) per il calcolo memoizzato di $\ell(1, n)$.
2. Si determini la complessità *al caso migliore* $T_{\text{best}}(n)$, supponendo che le uniche operazioni di costo unitario e non nullo siano i confronti tra caratteri.

PROG. DINAMICA → BOTTOM-UP / TOP-DOWN] ITER.

→ MEMOIZZ. → REC.

INIT → CASI BASE (1, 2) + RIEMPIMENTO DI ZERI PER DOPO

REC → CALCOLO RICORSIVO DELLA SOLUZIONE IN BASE AD INIT

$$\ell(i,j) = \begin{cases} 1 & \text{se } i = j \\ 2 & \text{se } i = j-1 \\ 2 + \ell(i+1, j-1) & \text{se } (i < j-1) \text{ e } (x_i = x_j) \\ \sum_{k=i}^{j-1} (\ell(i,k) + \ell(k+1,j)) & \text{se } (i < j-1) \text{ e } (x_i \neq x_j). \end{cases}$$

INIT_L(X)

N = LENGTH(X)

IF N = 1 RETURN 1

IF N = 2 RETURN 2

] LEGGO CASI BASE SE ESISTONO SOLO "i" e "j"

FOR I = 1 TO N

L[I, 0] = 1

FOR J = 1 TO N

L[0, J] = 2

→ INIT. PER IL RESTO DELLA MATRICE

```
INIT_L(X)
 n <- length(X)
 if n = 1 then return 1
 if n = 2 then return 2
 for i=1 to n-1 do
   L[i,i] <- 1
   L[i,i+1] <- 2
 L[n,n] <- 1
```

1 se i=j →

2 → i = j+1 →

CASI BASE

$$\ell(i,j) = \begin{cases} 1 & \text{se } i = j \\ 2 & \text{se } i = j-1 \\ 2 + \ell(i+1, j-1) & \text{se } (i < j-1) \text{ e } (x_i = x_j) \\ \sum_{k=i}^{j-1} (\ell(i,k) + \ell(k+1,j)) & \text{se } (i < j-1) \text{ e } (x_i \neq x_j). \end{cases}$$

```
FOR I=1 TO N-2        (N-1)
   FOR J=I+2 TO N        (I+1)
      L[I,J] = 0
```

↑ RIEMPIAMO PER ZERI DI REC

(1) → $\begin{cases} 2 + \ell(i+1, j-1) & \text{se } (i < j-1) \text{ e } (x_i = x_j) \\ \sum_{k=i}^{j-1} (\ell(i,k) + \ell(k+1,j)) & \text{se } (i < j-1) \text{ e } (x_i \neq x_j). \end{cases}$

↑ LCS

```
REC_L(X, I, J)
 IF (L[I,J] = 0)
 IF (X_I = X_J)
 [L[I,J] = 2 + REC_L(I+1, J-1)   (1)
 L[n,n]]
```

$$\begin{array}{c|ccccc} & 1 & 2 & 1 & 2 & 1 \\ \hline 2 & 0 & & & & \\ 1 & & 0 & & & \\ 2 & & & 0 & & \\ 1 & & & & 0 & \end{array}$$

↑ RIEMPIAMO

[$\sum_{k=i}^{j-1}(\ell(i,k)+\ell(k+1,j))$ se $(i<j-1)$ e $(x_i \neq x_j)$.

IF ($X_I \neq X_J$)

ORDINATO PER COSTRUZIONE

FOR K=1 TO J-1

L[I,J] = L[I,J] + REC_L(X,I,K)

+ REC_L(X,K+1,J)

RETURN L[I,J]

```
REC_L(X,i,j)
 if L[i,j] = 0 then
   if x_i = x_j then L[i,j] <- 2 + REC_L(X,i+1,j-1)
   else for k=i to j-1 do
             L[i,j] <- L[i,j] + REC_L(X,i,k) + REC_L(X,k+1,j)
 return L[i,j]
```

2. Si determini la complessità *al caso migliore* $T_{\text{best}}(n)$, supponendo che le uniche operazioni di costo unitario e non nullo siano i confronti tra caratteri.

```
for i=1 to n-2 do
   for j=i+2 to n do
    [L[i,j] <- 0 ]
```

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

SEMPLIFICA

$A \neq B \rightarrow \Sigma$ ①

$A \neq B \neq C \rightarrow \Sigma$ ②

② CALCOLA IL N. ESATTO DI OPERAZIONI (SOMMATORIE)

```
INIT_L(X)
 n <- length(X)
 if n = 1 then return 1
 if n = 2 then return 2
 for i=1 to n-1 do
   L[i,i] <- 1
   L[i,i+1] <- 2
 L[n,n] <- 1
 for i=1 to n-2 do
   for j=i+2 to n do
     L[i,j] <- 0
 return REC_L(X,1,n)
```

$O(n)$

```
REC_L(X,i,j)
 if L[i,j] = 0 then
   if x_i = x_j then L[i,j] <- 2 + REC_L(X,i+1,j-1)
   else for k=i to j-1 do
           L[i,j] <- L[i,j] + REC_L(X,i,k) + REC_L(X,k+1,j)
 return L[i,j]
```

$O(n/2)$ $\quad$ $< O(n/2)$

ORDINATO

Esercizio 2 (8 punti) Per $n > 0$, siano dati due vettori a componenti intere $\mathbf{a}, \mathbf{b} \in \mathbf{Z}^n$. Si consideri la quantità $c(i,j)$, con $0 \leq i \leq j \leq n-1$, definita come segue:

$$c(i,j) = \begin{cases} a_i & \text{se } 0 < i \leq n-1 \text{ e } j = n-1, \\ b_j & \text{se } i = 0 \text{ e } 0 \leq j \leq n-1, \\ c(i-1,j) \cdot c(i,j+1) & 0 < i \leq j < n-1. \end{cases}$$

J ↑ (BOTTOM UP

Si vuole calcolare la quantità $M = \max\{c(i,j) : 0 \leq i \leq j \leq n-1\}$.

1. Si fornisca il codice di un algoritmo iterativo bottom-up per il calcolo di M.

1. Date le dipendenze tra gli indici nella ricorrenza, un modo corretto di riempire la tabella è attraverso una scansione in cui calcoliamo gli elementi in ordine crescente di indice di riga e, per ogni riga, in ordine decrescente di indice di colonna. Il codice è il seguente.

A[i]

```
COMPUTE(a,b)
n <- length(a)
M = -infinito
for i=1 to n-1 do
   C[i,n-1] <- a_i
   M <- MAX(M,C[i,n-1])
for j=0 to n-1 do
   C[0,j] <- b_j
   M <- MAX(M,C[0,j])
for i=1 to n-2 do
   for j=n-2 downto i do
      C[i,j] <- C[i-1,j] * C[i,j+1]
   M <- MAX(M,C[i,j])
return M
```

M = sol (MAX(M,C[i,n-1])

M = sol(MAX(B[J])

BOTTOM UP

sol. (C[i,J]

2. Si valuti la complessità esatta dell'algoritmo, associando costo unitario ai prodotti tra numeri interi e costo nullo a tutte le altre operazioni.

```
for i=1 to n-2 do
   for j=n-2 downto i do
      C[i,j] <- C[i-1,j] * C[i,j+1]
      M <- MAX(M,C[i,j])
return M
```

$$\left[\sum_{i=1}^{n-2}\right] \sum_{j=i}^{n-2} 1 \to 1 \text{ PROD}$$

$$= \left[\sum_{i=1}^{n-2}\right] [n-1-i] \leftarrow (n-3)$$

ACCORCIAMO IL NOSTRO DISEGNO

$$\sum_{k=1}^{n\,(n-2)} k = \frac{(n-2)(n-1)}{2}$$

$$\Leftarrow \sum_{k=1}^{n} k$$

GAUSS. $\to \left[\sum_{i=1}^{n} k = \frac{n(n-1)}{2}\right]$

$$\approx O(n^2)$$

$$T(n) = \sum_{i=1}^{n-2}\sum_{j=i}^{n-2} 1 = \sum_{i=1}^{n-2} n-1-i = \sum_{k=1}^{n-2} k = (n-1)(n-2)/2.$$