

Per quanto riguarda la complessità, la ricorrenza è del tipo $T(n) = T(n/2) + c$, dato che ad ogni chiamata si dimezza la dimensione dell'array e il costo della parte non ricorsiva è costante. Per il Master Theorem (con $a = 1$, $b = 2$, $f(n) = c = \Theta(n^{\log_b a}) = \Theta(n^0) = \Theta(1)$) si conclude $T(n) = \Theta(n^{\log_b a} \log n) = \Theta(\log n)$.

Esercizio 2 (9 punti) Sia n un intero positivo. Si consideri la seguente ricorrenza $M(i, j)$ definita su tutte le coppie (i, j) con $1 \leq i \leq j \leq n$:

$$M(i, j) = \begin{cases} 2 & \text{se } i = j, \\ 3 & \text{se } j = i + 1, \\ M(i + 1, j - 1) \cdot M(i + 1, j) + M(i, j - 1) & \text{se } j > i + 1. \end{cases}$$

(vedo (1, n) e so che la scansione è per diagonale)

- (a) Si scriva una coppia di algoritmi INIT_M(n) e REC_M(i, j) per il calcolo memoizzato di $M(1, n)$.
 (b) Si calcoli il numero esatto $T(n)$ di moltiplicazioni tra interi eseguite per il calcolo di $M(1, n)$.

Soluzione:

```
(a) INIT_M(n)
    if n=1 then return 2
    if n=2 then return 3
    for i=1 to n-1 do
        M[i, i] = 2
        M[i, i+1] = 3
        M[n, n] = 2
    for i=1 to n-2 do
        for j=i+2 to n do
            M[i, j] = 0
    return REC_M(1, n)
```

Se gli indici sono uno solo o due soli, allora, seguendo i casi base si ritorna 2 e 3.

Ricordandosi che:

- "i" va da 1 ad (n-1), quindi diventa perché abbiamo già inizializzato, (n-2) la scansione
 - "j" va da (n-1) a 0 compresi, quindi dato che abbiamo inizializzato, parte da (n-2)

Una volta inizializzato secondo i casi base, tutto il resto della matrice viene riempita di zeri. Poi, si considera, essendo una scansione per diagonale, noi partiamo dal basso e riempiamo solo la parte sopra delle diagonali principali, ovviamente riempita anche questa di zeri.

```
REC_M(i, j)
    if M[i, j] = 0 then
        M[i, j] = REC_M(i+1, j-1) * REC_M(i+1, j) + REC_M(i, j-1)
    return M[i, j]
```

(b)

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

La sommatoria interna è costituita da soli 1, in quanto richiede una moltiplicazione tra tre numeri interi, nello specifico (n-2), (n-1), n.

Quanti uno?

Beh, da $j=i+2$ a n ci sono esattamente $n-(i+2)+1=n-i-1$ termini (sostituisco j in i , perché la serie è basata su i e il +1 si ha per il fatto che $i=1$). Poi sostituire con k accorgendoti che sono esattamente gli stessi termini della sommatoria, se provi a svilupparli, e l'ultima sommatoria la puoi riscrivere in quel modo, ricordandoti che la somma di $1 \dots n$ in generale è $n(n+1)/2$. Non avendo il termine 2 per linearità della sommatoria, non viene moltiplicato con il "fratto 2" di $(n-2)(n-1)$ e quindi il risultato è proprio $(n-2)(n-1)/2$.

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

Si capisce che si scansione in ordine decrescente di colonna perché quando "i" vale 0, allora "j" vale qualcosa

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

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

1. Si fornisca il codice di un algoritmo iterativo bottom-up per il calcolo di m .
2. Si valuti la complessità esatta dell'algoritmo, associando costo unitario ai prodotti tra numeri interi e costo nullo a tutte le altre operazioni.

Soluzione:

- (a) 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.

```

COMPUTE(a,b)
n <- length(a)
m = +infinito
for i=1 to n-1 do
  C[i,n-1] <- a_i
  m <- MIN(m,C[i,n-1])
for j=0 to n-1 do
  C[0,j] <- b_j
  m <- MIN(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-1] * C[i,j+1]
    m <- MIN(m,C[i,j])
return m

```

(ci prendiamo la lunghezza dell'array e inizializziamo il minimo ad infinito (così, qualsiasi prima quantità minima sarà più piccola))

Siccome devo salvare il minimo, ogni volta si fa il confronto tra il minimo attuale e l'indice di C[i, j] appena salvato.

Ricordandosi che:

- "i" va da i ad (n-1), quindi diventa perché abbiamo già inizializzato, (n-2) la scansione
- "j" va da (n-1) a 0 compresi, quindi dato che abbiamo inizializzato, parte da (n-2)

Siccome nuovamente abbiamo un ciclo in cui "i" va ad (n-2) e in quello di inizializzazione andava ad (n-1), la sommatoria è data da (n-2)(n-1)/2

(b)

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

La sommatoria interna è costituita da soli 1, in quanto richiede una moltiplicazione tra tre numeri interi, nello specifico (n-2), (n-1), n.

Quanti uno?

Beh, da $j=i+2$ a n ci sono esattamente $n-(i+2)+1=n-i-1$ termini (sostituisco j in i , perché la serie è basata su i e il $+1$ si ha per il fatto che $i=1$). Poi sostituire con k accorgendoti che sono esattamente gli stessi termini della sommatoria, se provi a svilupparli, e l'ultima sommatoria la puoi riscrivere in quel modo, ricordandoti che la somma di $1 \dots n$ in generale è $n(n+1)/2$.

Non avendo il termine 2 per linearità della sommatoria, non viene moltiplicato con il "fratto 2" di (n-2)(n-1) e quindi il risultato è proprio (n-2)(n-1)/2

Uguale a quello precedente, ma abbiamo a che fare con MAX piuttosto che MIN

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:

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

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

1. Si fornisca il codice di un algoritmo iterativo bottom-up per il calcolo di M .
2. Si valuti la complessità esatta dell'algoritmo, associando costo unitario ai prodotti tra numeri interi e costo nullo a tutte le altre operazioni.

Soluzione:

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.

```
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
```

(ci prendiamo la lunghezza dell'array e inizializziamo il minimo a meno infinito (così, qualsiasi prima quantità massima sarà più grande)

Ricordandosi che:
- "i" va da 1 ad (n-1), quindi diventa perché abbiamo già inizializzato, (n-2) la scansione
- "j" va da (n-1) a 0 compresi, quindi dato che abbiamo inizializzato, parte da (n-2)

Si come devo salvare il massimo, ogni volta si fa il confronto tra il massimo attuale e l'indice di C[i, j] appena salvato.

2.

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

La sommatoria interna è costituita da soli 1, in quanto richiede una moltiplicazione tra due numeri interi, nello specifico (n-2), (n-2) (apici della serie).

Quanti uno?

Beh, da $j=n-2$ a i ci sono esattamente $n-i-1 = n-1-1$ termini (sostituisco j in i , perché la serie è basata su i e sottraggo 1 in quanto $i=1$ ed i sarebbe il valore di j ($j=i$ è il pedice della seconda serie)).

Poi sostituire con k accorgendoti che sono esattamente gli stessi termini della sommatoria, se provi a svilupparli, e l'ultima sommatoria la puoi riscrivere in quel modo, ricordandoti che la somma di $1 \dots n$ in generale è $n(n+1)/2$ quando si abbia come qui la serie con termine generale k (serie di Gauss).

Non avendo il termine 2 per linearità della sommatoria, non viene moltiplicato con il "fratto 2" di $(n-2)(n-1)$ e quindi il risultato è proprio $(n-2)(n-1)/2$

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

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

1. Scrivere una coppia di algoritmi $\text{INIT_L}(X)$ e $\text{REC_L}(X, i, j)$ 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.

(vedo $(1, n)$ e so che la scansione è per diagonale)

Soluzione:

1. Pseudocodice:

```
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)
```

Prendo la lunghezza della stringa iniziale

Similmente, anche $n=1$ e $n=2$ vengono dai due casi base (array di 1/2 elementi max), quindi o ha un elemento oppure ne ha due soli

(ultimo elemento diagonale inizializzato)

Una volta inizializzato secondo i casi base, tutto il resto della matrice viene riempita di zeri. Poi, si considera, essendo una scansione per diagonale, noi partiamo dal basso e riempiamo solo la parte sopra delle diagonali principali, ovviamente riempita anche questa di zeri.

```
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. Il caso migliore è chiaramente quello in cui tutti i caratteri sono uguali, poiché l'albero della ricorsione in quel caso è unario e i suoi nodi interni corrispondono alle chiamate con indici

2. Il caso migliore è chiaramente quello in cui tutti i caratteri sono uguali, poiché l'albero della ricorsione in quel caso è unario e i suoi nodi interni corrispondono alle chiamate con indici

$(1, n), (2, n - 1), \dots, (k, n - k + 1)$, ognuno associato a un costo unitario. Ci sono al più $\lfloor n/2 \rfloor$ di queste chiamate, e quindi $T_{\text{best}}(n) = O(n)$.

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 restituisca in uscita il valore $z(n, 1)$. (vedo $(n, 1)$ e la scansione è per colonna)
2. Si valuti il numero esatto $T_Z(n)$ di moltiplicazioni tra interi eseguite dall'algoritmo sviluppato al punto (1).

Reverse column major:

a11	a12	a13
a21	a22	a23
a31	a32	a33

Soluzione:

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]
```

Piuttosto che usare per l'inizializzazione due indici, se ne usa solo uno.
 Quando si ha "j", si sa che "i" vale 1, da cui i=j e quindi [1, i]
 Quando invece si ha "n+1-i" come indice, si vede che "j" è =n e quindi
 avremo che per [i, n] avremo [n+1-i].

Vedendo che "j" parte da n e invece "i" parte da 1,
 per effettuare una scansione giusta per colonne (avendo che
 la prima colonna/riga è stata inizializzata dal caso base), allora
 "j" parte da (n-1) piuttosto che da (n) e "i" parte da 2 piuttosto che da (1)

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

2. Ogni iterazione del doppio ciclo dell'algoritmo esegue due moltiplicazioni tra interi, e quindi

$$\begin{aligned} T_Z(n) &= \sum_{j=1}^{n-1} \sum_{i=2}^n 2 \\ &= \sum_{j=1}^{n-1} 2(n-1) \\ &= 2(n-1)^2. \end{aligned}$$

Equivalentemente, basta osservare che l'algoritmo esegue due moltiplicazioni per ogni elemento di una tabella $(n-1) \times (n-1)$.

La sommatoria interna è costituita da soli 2, in quanto richiede due moltiplicazioni tra due numeri interi, nello specifico $(n-1)$ ed n (apici della serie). Beh, da $i=2$ ad n . Quindi, dovendo sostituire i in j (essendo la serie basata sull'indice più esterno), si ha che $n-1$ semplicemente (in quanto, l'indice j è pari ad 1 e il 2 verrebbe portato poi fuori per linearità).
 Ora, siccome abbiamo una serie che ha come apice $(n-1)$, non possiamo esprimere la soluzione in questo modo, dato che $(n-1)$ compare sia a pedice che dentro la serie. Dobbiamo quindi esprimere la serie in termini di j . Ciò comporta che io vada a "portare dentro" $\leftarrow (n-1) \rightarrow$ nella serie, in quanto moltiplicheremmo implicitamente $2(n-1)$ per altre $(n-1)$ volte e quindi è come se andassi a fare $2(n-1)(n-2) = 2(n-1)^2$

Esercizio 2 (9 punti) Sia $n > 0$ un intero. Si consideri la seguente ricorrenza $M(i, j)$ definita su tutte le coppie (i, j) con $1 \leq i \leq j \leq n$:

$$M(i, j) = \begin{cases} 1 & \text{se } i = j, \\ 2 & \text{se } j = i + 1, \\ \underline{M(i+1, j-1) \cdot M(i+1, j) \cdot M(i, j-1)} & \text{se } j > i + 1. \end{cases}$$

1. Scrivere una coppia di algoritmi INIT_M(n) e REC_M(i, j) per il calcolo memoizzato di $M(1, n)$.
2. Calcolare il numero esatto $T(n)$ di moltiplicazioni tra interi eseguite per il calcolo di $M(1, n)$.

Soluzione:

1. Pseudocodice:

```
INIT_M(n)
if n=1 then return 1
if n=2 then return 2
for i=1 to n-1 do
  M[i,i] = 1
  M[i,i+1] = 2
M[n,n] = 1      (ultimo elemento diagonale inizializzato)
for i=1 to n-2 do
  for j=i+2 to n do
    M[i,j] = 0
return REC_M(1,n)
```

Siccome viene già passata la lunghezza, non serve salvarla.
Similmente, anche $n=1$ e $n=2$ vengono dai due casi base (array di 1/2 elementi max), quindi o ha un elemento oppure ne ha due soli

Ricordandosi che:
- "i" va da 1 ad $(n-1)$, quindi diventa perché abbiamo già inizializzato, $(n-2)$ la scansione,
- "j" va da $(n-1)$ a 0 compresi, quindi dato che abbiamo inizializzato, parte da $(n-2)$

Una volta inizializzato secondo i casi base, tutto il resto della matrice viene riempita di zeri. Poi, si considera, essendo una scansione per diagonale, noi partiamo dal basso e riempiamo solo la parte sopra delle diagonali principali, ovviamente riempita anche questa di zeri.

```
REC_M(i,j)
if M[i,j] = 0 then
  M[i,j] = REC_M(i+1,j-1) * REC_M(i+1,j) * REC_M(i,j-1)
return M[i,j]
```

2.

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

Esercizio 2 (9 punti) Data una stringa di numeri interi $A = (a_1, a_2, \dots, a_n)$, si consideri la seguente

ricorrenza $c(i, j)$ definita per ogni coppia di valori (i, j) con $1 \leq i, j \leq n$:

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

1. Si fornisca il codice di un algoritmo iterativo bottom-up COMPUTE_C(A) che, data in input la stringa A restituisca in uscita il valore $c(n, 1)$. (vedo $(1, n)$ e so che la scansione è per diagonale)
2. Si valuti il numero esatto $T_{CC}(n)$ di moltiplicazioni tra interi eseguite dall'algoritmo sviluppato al punto (1).

La sommatoria interna è costituita da soli 2, in quanto richiede due moltiplicazioni tra tre numeri interi, nello specifico $(n-2)$, $(n-1)$, n . Quanti due? Beh, da $j=i+2$ a n ci sono esattamente $n-(i+2)+1=n-i-1$ termini (sostituisco j in i , perché la serie è basata su i).

Quindi sarebbe sommatoria di $2(n-i-1)$, poi si può portare fuori il 2 per linearità.

Poi sostituire con k accorgendosi che sono esattamente gli stessi termini della sommatoria, se provi a svilupparli, e l'ultima sommatoria la puoi riscrivere in quel modo, ricordandoti che la somma di $1 \dots n$ in generale è $n(n+1)/2$ quando si abbia come qui la serie con termine generale k (serie di Gauss).

Infatti, la sommatoria da 1 a $n-2$ è $(n-2)(n-2+1)/2$, semplificando diventa quello che vedi

Il 2 come argomento della serie deriva dal fatto che ogni termine $M[i, j]$ richiede due moltiplicazioni tra 3 numeri interi.

Soluzione:

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.

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

Dato che non viene passata la lunghezza dell'array, conviene salvarla nella variabile "n" per riuscire ad iterare.

Ricordandosi che:
- "j" va da n ad 1, quindi diventa perché abbiamo già inizializzato, (n-1) la scansione
- "j" va da 1 a n compresi, quindi dato che abbiamo inizializzato, parte da 2

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

2. Ogni iterazione del doppio ciclo dell'algoritmo esegue due operazioni tra interi, e quindi

$$\begin{aligned} T_{CC}(n) &= \sum_{j=1}^{n-1} \sum_{i=2}^n 2 \\ &= \sum_{j=1}^{n-1} 2(n-1) \\ &= 2(n-1)^2. \end{aligned}$$

Equivalentemente, basta osservare che l'algoritmo esegue due moltiplicazioni per ogni elemento di una tabella $(n-1) \times (n-1)$.

La sommatoria interna è costituita da soli 2, in quanto richiede due moltiplicazioni tra due numeri interi, nello specifico (n-1) ed n (apici della serie). Beh, da $i=2$ ad n. Quindi, dovendo sostituire i in j (essendo la serie basata sull'indice più esterno), si ha che $n-1$ semplicemente (in quanto, l'indice j è pari ad 1 e il 2 verrebbe portato poi fuori per linearità). Ora, siccome abbiamo una serie che ha come apice (n-1), non possiamo esprimere la soluzione in questo modo, dato che (n-1) compare sia a pedice che dentro la serie. Dobbiamo quindi esprimere la serie in termini di j. Ciò comporta che io vada a “portare dentro” $\leftarrow (n-1) \rightarrow$ nella serie, in quanto moltiplicheremmo implicitamente $2(n-1)$ per altre (n-1) volte e quindi è come se andassi a fare $2(n-1)(n-2) = 2(n-1)^2$.