

Esercitazione:
divide et impera
relazioni di ricorrenza

13 aprile 2021



Nelle prossime pagine, gli esercizi svolti

Alcuni esercizi sulla tecnica del Divide et Impera.

1. a) Descrivere gli aspetti essenziali della tecnica Divide et Impera, utilizzando lo spazio designato.
b) Descrivere ed analizzare un algoritmo **basato sulla tecnica Divide et Impera** che dato un array $A[1, \dots, n]$ di interi ne restituisca il massimo.
2. Sia $V[1..n]$ un vettore ordinato di 0 e 1.
Descrivere ed analizzare un algoritmo per determinare il numero di 0 presenti in $V[1..n]$ in tempo $O(\log n)$.

? Relazioni di ricorrenza 1

Se $T(n) = 3 T(n - 2) + 2$, con $T(0) = T(1) = 6$, allora

A. $T(6) = 62$

B. $T(6) = 188$

C. $T(6) = 162$

D. Nessuna delle risposte precedenti

? Relazioni di ricorrenza 2

Nella risoluzione della relazione di ricorrenza

$$T(n) = 2 T(n/2) + n, \text{ con } T(1) = c,$$

col metodo di iterazione, qual è il valore di $T(n)$ alla i -esima iterazione?

A. $T(n) = 2^i T(n/2^i) + n$

B. $T(n) = 2^i T(n/2^i) + 2^i n$

C. $T(n) = 2^i T(n/2^i) + c n$

D. Nessuna delle risposte precedenti

? Soluzione relazione di ricorrenza 3

La soluzione della relazione di ricorrenza

$$T(n) = 2T(n/2) + c, \text{ con } T(1) = c, \text{ è:}$$

- A. $T(n) = \Theta(\log n)$
- B. $T(n) = \Theta(n)$
- C. $T(n) = \Theta(n \log n)$
- D. Nessuna delle risposte precedenti.

? Soluzione relazione di ricorrenza 4

La soluzione della relazione di ricorrenza

$$T(n) = 4T(n/2) + n, \text{ con } T(1) = 1 \text{ è:}$$

- A. $T(n) = \Theta(n^{\log_4 2})$
- B. $T(n) = \Theta(n)$
- C. $T(n) = \Theta(n^2)$
- D. Nessuna delle risposte precedenti.

? Tempo Divide et Impera

Se un algoritmo **Divide et Impera** divide il problema di taglia n in a sotto-problemi di taglia n/b in tempo $D(n)$ e combina le soluzioni ai sotto-problemi in tempo $C(n)$, allora il suo tempo di esecuzione $T(n)$ soddisfa

- A. $T(n) = \Theta(n \log n)$
- B. $T(n) = a T(n/b) + \Theta(n)$
- C. $T(n) = a T(n/b) + D(n) + C(n)$
- D. Nessuna delle risposte precedenti.

? Partition

Una chiamata alla procedura PARTITION (come studiata) su [6, 1, 3, 9, 8] restituisce

- A. 2
- B. 3
- C. 6
- D. Nessuna delle risposte precedenti

```
Partition (A, p, r)
x = A[p]
i = p-1
j = r+1
while True
    do repeat j=j-1 until A[j] ≤ x
    repeat i=i+1 until A[i] ≥ x
    if i < j
        then scambia A[i] ↔ A[j]
    else return j
```

? Merge

Quanti confronti effettua l'algoritmo MERGE per la fusione dei due array ordinati [1,5,6,7] e [2,3,4]?

A. 4

B. 6

C. 12

D. Nessuna delle risposte precedenti

```
i = 1, j = 1
while (both lists are nonempty) {
    if ( $a_i \leq b_j$ ) append  $a_i$  to output list and increment i
    else ( $a_i > b_j$ ) append  $b_j$  to output list and increment j
}
append remainder of nonempty list to output list
```

? Tempo di esecuzione 4

Il tempo di esecuzione del seguente frammento di pseudocodice è

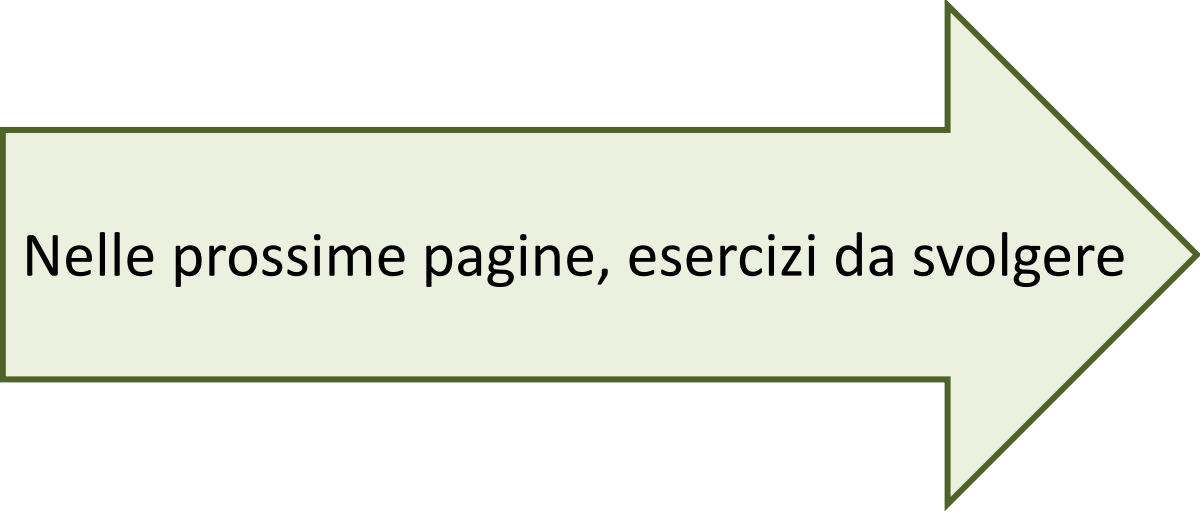
```
for i=1 to n/2  
    PARTITION (A, i, n)
```

A. $\Theta(\log n)$

B. $\Theta(n \log n)$

C. $\Theta(n)$

D. Nessuna delle risposte precedenti



Nelle prossime pagine, esercizi da svolgere

Esercizio (Fibonacci con 2 celle)

Fornire una variante dell'algoritmo **Fibonacci3-iter(n)**, mostrato nelle slide della lezione 12, che utilizzi soltanto **2** celle di memoria (anziché n).

Appello 9 luglio 2015

Quesito 2 (24 punti)

Da quando ti sei registrato su Facebook ad oggi, i tuoi amici sono aumentati in maniera vertiginosa. Il primo anno avevi solo 10 amici; il secondo 35; il terzo 100 e nessuno ti elimina mai dagli amici. Hai poi notato che ogni tuo amico, 3 anni dopo averti dato la sua amicizia, ti porta un nuovo amico (spesso è un collega di università/lavoro, fidanzato/a, fratello/a, cugino/a). E tutti i tuoi nuovi amici si aggiungono sempre e solo in questo modo.

Sapresti calcolare quanti diventeranno i tuoi amici nei prossimi anni?

- a) Descrivere un algoritmo efficiente per il calcolo del numero dei tuoi amici dopo n anni dalla tua registrazione su Facebook, supponendo che aumentino sempre rispettando la regola sopra descritta. E' necessario analizzare la complessità di tempo e di spazio dell'algoritmo proposto.
- b) Valutare la crescita del numero di amici rispetto ad n (in notazione asintotica).

Appello 9 febbraio 2011

I numeri di Tribonacci sono così definiti:

$$R(0) = 0$$

$$R(1) = 0$$

$$R(2) = 1$$

$$R(n) = R(n-1) + R(n-2) + R(n-3) \text{ se } n \geq 3.$$

- a) Scrivere lo pseudocodice di un algoritmo di programmazione dinamica per il calcolo dell' n -esimo numero di Tribonacci $R(n)$.
- b) Analizzare la complessità di tempo e di spazio dell'algoritmo proposto.
- c) E' possibile realizzare l'algoritmo con spazio $O(1)$? Giustificare la risposta.

Appello 12 settembre 2016

Quesito 1 (24 punti) (*Cappanacci*)

La sequenza dei numeri di Fibonacci k-generalizzati, per un intero k, è definita come segue

$$F_{n,k} = 0 \text{ per } n = 0, 1, \dots, k-2$$

$$F_{k-1,k} = 1$$

$$F_{n,k} = F_{n-1,k} + F_{n-2,k} + \dots + F_{n-k,k} \text{ per ogni } n \geq k.$$

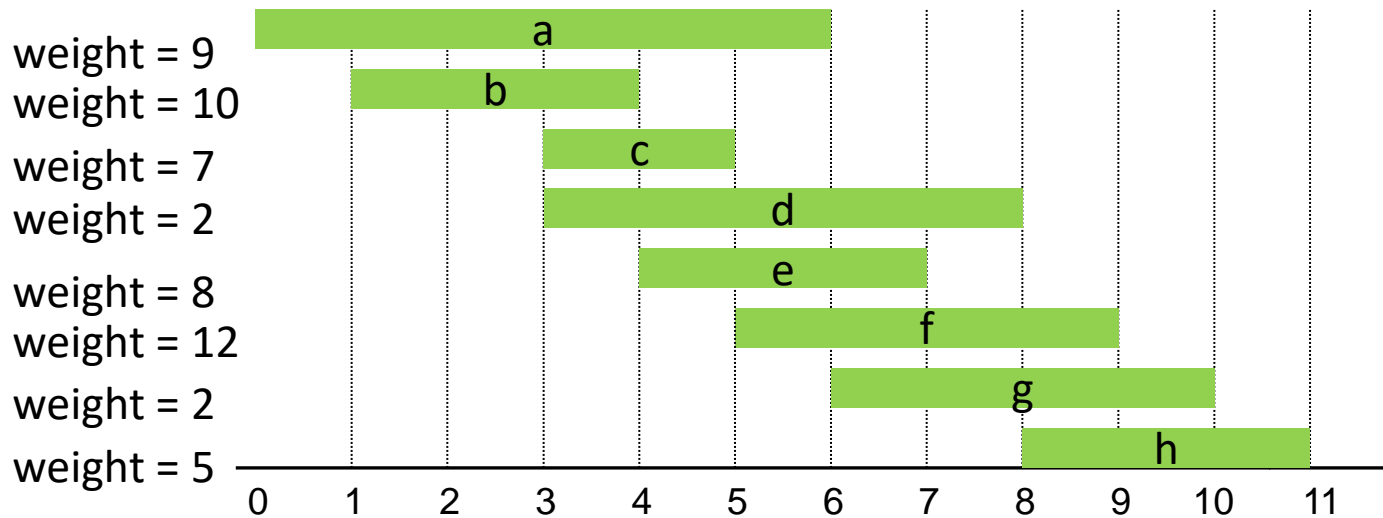
Descrivere ed analizzare un algoritmo di programmazione dinamica che dati due interi, n e k, calcola il numero $F_{n,k}$.

Esempio. Per $k=3$, i primi numeri di Fibonacci 3-generalizzati sono:

$$F_{0,3} = F_{1,3} = 0, F_{2,3} = 1, F_{3,3} = 1, F_{4,3} = 2, F_{5,3} = 4, \dots$$

Esercizi

- Eseguire l'algoritmo (Bottom-up dynamic programming) della slide successiva sui seguenti dati :



- Eseguire l'algoritmo **Find-Solution (8)** sugli stessi dati.

Weighted Interval Scheduling: Bottom-Up

Bottom-up dynamic programming. Unwind recursion.

Input: $n, s_1, \dots, s_n, f_1, \dots, f_n, v_1, \dots, v_n$

Sort jobs by finish times so that $f_1 \leq f_2 \leq \dots \leq f_n$.

Compute $p(1), p(2), \dots, p(n)$

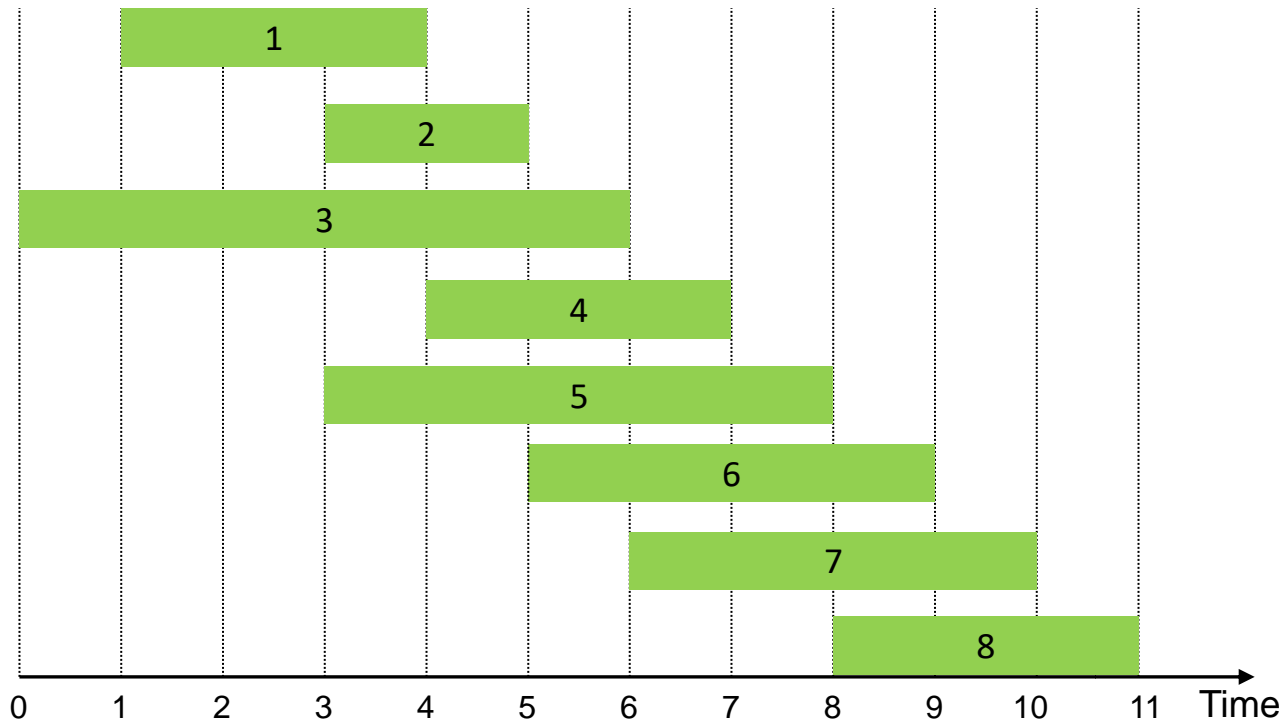
```
Iterative-Compute-Opt {  
     $M[0] = 0$   
    for  $j = 1$  to  $n$   
         $M[j] = \max(v_j + M[p(j)], M[j-1])$   
}
```

Weighted Interval Scheduling

Notation. Label jobs by finishing time: $f_1 \leq f_2 \leq \dots \leq f_n$.

Def. $p(j)$ = largest index $i < j$ such that job i is compatible with j .

Ex: (independently from weights) $p(8) = 5$, $p(7) = 3$, $p(2) = 0$.



Appello del 4 aprile 2018

Quesito 2 (24 punti) (*Programmazione dinamica*)

Si supponga che la soluzione ad un certo problema (a noi ignoto) sia data, per un certo intero n positivo, dal massimo fra i valori $\text{OPT}(n, R)$ e $\text{OPT}(n, B)$ definiti ricorsivamente come segue (R sta per Rosso e B sta per Blu):

$$\text{OPT}(1, R) = 2$$

$$\text{OPT}(1, B) = 1$$

$$\text{OPT}(i, R) = \text{OPT}(i-1, B) + 1, \text{ se } i > 1$$

$$\text{OPT}(i, B) = \max \{ \text{OPT}(i, R) - 1, \text{OPT}(i-1, R) \}, \text{ se } i > 1$$

- a) Calcolare i valori di $\text{OPT}(i, R)$ e $\text{OPT}(i, B)$ per ogni $i=1, 2, \dots, 5$, organizzandoli in una tabella.
- b) Scrivere lo pseudocodice di un algoritmo **ricorsivo** per il calcolo della soluzione al problema.
- c) Scrivere lo pseudocodice di un algoritmo di **programmazione dinamica** per il calcolo della soluzione al problema. Analizzarne la complessità di tempo e di spazio, giustificando la risposta.

Appello del 21 marzo 2019

Quesito 1 (26 punti) (*Giornata di seminari*)

Vi state occupando di organizzare una giornata di seminari nell'Aula Magna della vostra università. Avete avuto la disponibilità di vari relatori a tenere un loro intervento; ognuno ha specificato da che ora a che ora si terrebbe il suo seminario. Purtroppo, non riuscite ad organizzare la giornata in modo da inserire tutti i relatori. Dovete perciò scegliere alcuni fra i relatori in modo che i loro seminari possano essere svolti nell'aula senza sovrapposizioni di orario. Volete inoltre fare in modo che sia massimo il **tempo** totale di utilizzo effettivo dell'aula.

- a) Definire il problema computazionale, specificandone i dati in ingresso e in uscita.
- b) Indicare se si tratta di un problema studiato e, se sì, quale.
- c) Si consideri un algoritmo *greedy* basato sul criterio di scelta del seminario che utilizza l'aula per più tempo. Mostrare, con un contro-esempio, che tale algoritmo non sempre porta ad una soluzione ottimale.
- d) Progettare un algoritmo di **programmazione dinamica** che risolve il problema e valutarne la complessità. Si potrà ottenere il massimo della votazione solo se l'algoritmo è descritto tramite pseudo-codice ed è discussa la sua correttezza.