

Non capirne un tubo....

Disponiamo di un tubo metallico di lunghezza L . Da questo tubo vogliamo ottenere al più n segmenti, aventi rispettivamente lunghezza $S[1], S[2], \dots, S[n]$. Il tubo viene segato sempre a partire da una delle estremità, quindi ogni taglio riduce la sua lunghezza della misura asportata. Ovviamente un tubo di lunghezza L non può dare origine a segmenti più lunghi di L .

- 1 Scrivere un algoritmo efficiente che, dati L e il vettore S , restituisca il numero massimo di segmenti che possono essere ottenuti dal tubo.
- 2 Dimostrare che tale algoritmo è corretto
- 3 Determinare il costo computazionale dell'algoritmo di cui al punto 1.

Palindroma

Una stringa si dice palindroma se è uguale alla sua trasposta, cioè se è identica se letta da sinistra e destra o da destra a sinistra.

Scopo dell'esercizio è scrivere una funzione $f(s)$ che ritorna il numero minimo di caratteri da **inserire** in s necessari per rendere s palindroma.

Per esempio, input: “casacca”:

- $n = 7$ caratteri: “casaccaACCASAC”
- $n = 6$ caratteri: “casaccaCCASAC”
- $n = 3$ caratteri: “casaccaSAC”
- $n = 2$ caratteri: “ACcasacca”

Notate che non necessariamente i caratteri si inseriscono in testa o in fondo; per esempio, “anta” \rightarrow “antNa”.

Fornire un algoritmo per calcolare $f()$. Discutere la complessità dell'algoritmo.

Massima copertura

Dati n segmenti della retta delle ascisse, dove l' i -esimo segmento inizia nella coordinata $a[i]$ e termina nella coordinata $b[i]$, si scriva un algoritmo che trovi il sottoinsieme di segmenti che coprono la maggior parte della retta e non sono sovrapposti. Valutare il costo computazionale.

Mosse su scacchiera

Supponete di avere una scacchiera $n \times n$ e un pedone che dovete muovere dall'estremità inferiore a quella superiore. Un pedone si può muovere (1) una casella in alto, oppure (2) una casella in diagonale alto-destra, oppure (3) una casella in diagonale alto-sinistra. Non può tornare indietro. Quando una cella (x, y) viene visitata, guadagnate un valore reale $p(x, y)$.

Calcolare un percorso da una qualunque casella dell'estremità inferiore ad una qualunque casella dell'estremità superiore, massimizzando il profitto.

6	7	4	7	<u>8</u>
7	6	1	1	<u>4</u>
3	5	7	<u>8</u>	2
2	6	<u>7</u>	0	2
7	3	5	<u>6</u>	1

Quadrato binario

Sia $A[1 \dots n, 1 \dots n]$ una matrice di valori booleani 0/1. Scrivere un algoritmo che restituisce la dimensione del più grande quadrato composto da valori 1. Ad esempio, nella matrice seguente, i quadrati di dimensione massima (ve ne sono due, di cui uno evidenziato in grassetto) hanno dimensione pari a 4.

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

Non capirne un tubo...

```
maxSections(integer  $L$ , integer[]  $S$ , integer  $n$ )
```

```
sort( $S$ )
```

```
integer  $i \leftarrow 1$ 
```

```
while  $i \leq n$  and  $L \geq S[i]$  do
```

```
     $L \leftarrow L - S[i]$ 
```

```
     $i \leftarrow i + 1$ 
```

```
return  $i - 1$ 
```

Palindroma

- Se la stringa $s = as'a$ è composta da due identici caratteri “a” iniziale e finale, allora:

$$f(s) = f(s')$$

- Se la stringa $s = as'b$ ha due caratteri iniziale e finale diversi, aggiungiamo o un carattere “b” in testa (e consideriamo il problema as' , oppure aggiungiamo un carattere “a” in coda (e consideriamo il problema $s'b$). Scegliamo fra le due possibilità quella con costo minore. In entrambi i casi, dobbiamo sommare 1 per il carattere aggiunto.

$$f(s) = \min\{f(as'), f(s'b)\} + 1$$

Palindroma

calcolaF(ITEM[] s , integer[][] F , integer i , integer j)

if $j \leq i$ **then**

\sqsubset **return** 0

else if $F[i, j] = \perp$ **then**

if $s[i] = s[j]$ **then**

$F[i, j] \leftarrow \text{calcolaF}(s, i + 1, j - 1)$

else

\sqsubset $F[i, j] \leftarrow \min(\text{calcolaF}(s, i, j - 1), \text{calcolaF}(s, i + 1, j)) + 1$

\sqsubset **return** $F[i, j]$

Massima copertura

Questo problema è molto simile al problema dell'insieme indipendente di intervalli pesati visto a lezione, dove il peso $w[i]$ è pari a $b[i] - a[i]$. Si risolve quindi con una singola chiamata a quella soluzione, in un tempo pari a $\Theta(n \log n)$.

```
SET segmentcover(integer[] a, integer[] b, integer n)
integer[] w = new integer[1 .. n]
for i ← 1 to n do
    w[i] ← b[i] - a[i]
return maxinterval(a, b, w, n)
```

Mosse su scacchiera

$$g[x, y] = \begin{cases} -\infty & x < 1 \vee x > n \\ p(x, y) & y = n \\ p(x, y) + \max_{d \in \{-1, 0, +1\}} \{ g[x + d, y + 1] \} & \text{otherwise} \end{cases}$$

Mosse su scacchiera

search-path(integer[][] p , integer n)

{ Calcola la tabella g }

for $x \leftarrow 1$ to n do $g[x, n] \leftarrow p[x, n]$

for $y \leftarrow n - 1$ downto 1 do

 for $x \leftarrow 1$ to n do

$g[x, y] \leftarrow -\infty$

 foreach $d \in \{-1, 0, +1\}$ do

 integer $x' \leftarrow x + d$

 if $x' \geq 1$ and $x' \leq n$ then

 integer $t \leftarrow g[x', y + 1] + p[x, y]$

 if $t > g[x, y]$ then

$g[x, y] \leftarrow t$

$m[x, y] \leftarrow d$

Quadrato binario

{ Cerca la casella iniziale con massimo guadagno }

integer x

for $i \leftarrow 1$ **to** n **do**

if $g[i, 1] > g[x, 1]$ **then**
 $x \leftarrow i$

{ Stampa il percorso }

for $y \leftarrow 1$ **to** $n - 1$ **do**

print “ $(x, y) \leftarrow (x + m[x, y], y + 1)$ ”
 $x \leftarrow x + m[x, y]$

Quadrato binario

$$M[i, j] = \begin{cases} 0 & A[i, j] = \mathbf{false} \\ 1 & A[i, j] = \mathbf{true} \wedge i = n \vee j = n \\ \min\{A[i + 1, j], \\ \quad A[i + 1, j + 1], \\ \quad A[i, j + 1]\} + 1 & \text{altrimenti} \end{cases}$$