

Scheduling

Si considerino n job da sottomettere ad un processore, il job i -esimo caratterizzato da una deadline positiva $D[i]$ e da un guadagno positiva $G[i]$, $1 \leq i \leq n$, entrambe interi. Tutti i job hanno durata standard 1. Se il job i è eseguito entro l'istante $D[i]$ produrrà un guadagno $G[i]$, altrimenti il guadagno è nullo. L'obiettivo è trovare una sequenza di esecuzione che massimizzi il guadagno.

```
maxgain(integer[] D, integer[] G, integer n )
```

```
{ ordina i vettori D, G per guadagno decrescente, ovvero
```

```
 $\forall i, j, 1 \leq i < j \leq n : G[i] < G[j] }$ 
```

```
integer t  $\leftarrow$  0
```

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

```
    if t + 1  $\leq$  D[i] then           % Job i può essere eseguito al tempo t
        print i
        t  $\leftarrow$  t + 1
```

- Provare che l'algoritmo proposto non è corretto
- Proporre un algoritmo greedy corretto.

Prima elementare

A mia figlia (prima elementare) è stato chiesto di disegnare tutte le possibili sequenze composte da tre pallini rossi e due pallini gialli. (i) Scrivere un programma che stampa tutte le possibili stringhe composte da n caratteri R e da m caratteri G , per un totale di $n + m$ caratteri. Discuterne la complessità. (ii) Scrivere un programma che conta tutte queste possibile stringhe – ovviamente senza generarle tutte e poi contandole.

Complessità correttezza blah blah

Stringhe primitive

Dato un insieme S contenente m stringhe dette *primitive* ed una stringa $X[1 \dots n]$, si vuole determinare in quanti modi diversi X è ottenibile dalla concatenazione di stringhe primitive. Ad esempio: dato l'insieme di primitive $\{01, 10, 011, 101\}$, per la stringa $X = 0111010101$ la risposta è 3 ($011 - 10 - 10 - 101$, $011 - 10 - 101 - 01$ e $011 - 101 - 01 - 01$) mentre per la stringa $X = 0110001$ la risposta è 0.

Descrivere in pseudo-codice un algoritmo che conta il numero di modi diversi in cui è possibile concatenare una stringa. Discuterne correttezza e complessità. Suggerimento: programmazione dinamica. Per comodità, supponete che la lunghezza di una stringa s sia $|s|$ e di avere a disposizione una primitiva $\text{check}(X, s, i)$ che ritorna vero se la stringa s è contenuta nella stringa X a partire dalla posizione i . Il costo della chiamata a $\text{check}()$ è $O(|s|)$, dove $|s|$ è la lunghezza di s . Ad esempio, se $X = 1001$ e $s = 00$, $\text{check}(X, s, 2)$ ritorna vero, per tutti gli altri indici i ritorna falso.

Vicini alla meta (o metà?)

Descrivere un algoritmo `nearest(integer[] V, integer n, integer k)` che, dato un vettore V di n interi distinti e un intero positivo $k \leq n$, stampi k numeri di V che sono più vicini alla mediana dei valori in V , dove la misura di distanza dalla mediana è data dalla distanza in valore assoluto da essa. Discutere complessità e correttezza.

Ad esempio, sia $n = 11$, $V = \{1, 6, 4, 3, 9, 12, 15, 2, 8, 10, 11\}$ e $k = 2$; la mediana di V è 8 e i due valori più vicini sono 6, 9 (che hanno distanza 2, 1) oppure 9, 10 (che hanno distanza 1, 2). In altre parole, se il k -esimo e il $k + 1$ -esimo numero sono equidistanti dalla mediana, se ne sceglie uno. Notate che la mediana stessa non viene contata fra i valori da restituire.

Doppio mediano

Siano $X[1 \dots n]$ e $Y[1 \dots n]$ due vettori, ciascuno contenente n interi già ordinati. Scrivere un algoritmo che trovi i valori mediani dei $2n$ elementi dei vettori X e Y presi insieme. Usiamo il plurale perchè essendo $2n$ pari, è possibile definire *due* valori mediani. Discutere correttezza e complessità.

Scheduling

- ① Si considerino due job, uno con guadagno 2 e deadline 2 e uno con guadagno 1 e deadline 1. Eseguendo prima il primo job, come da algoritmo, si può eseguire solo quello e il guadagno è 2; eseguendo invece prima il secondo e poi il primo, si ottiene un guadagno di 3. Questo dimostra che l'algoritmo greedy non è corretto.
- ② Ordinamento per deadline decrescente

Prima elementare

stampaCombinazioni(char[] V , integer i , integer n , integer m)

if $n = 0$ and $m = 0$ then

 | **print V**

if $n > 0$ then

 | $V[i] \leftarrow \text{"R"}$

 | **stampaCombinazioni($V, i + 1, n - 1, m$)**

if $m > 0$ then

 | $V[i] \leftarrow \text{"G"}$

 | **stampaCombinazioni($V, i + 1, n, m - 1$)**

Prima elementare

```
calcolaCombinazioniRic(integer  $n$ , integer  $m$ )
```

```
if  $n = 0$  or  $m = 0$  then
```

```
  | return 1
```

```
else
```

```
  | return
```

```
  | calcolaCombinazioniRic( $n - 1, m$ ) + calcolaCombinazioniRic( $n, m - 1$ )
```

Prima elementare

```
calcolaCombinazioni(integer  $n$ , integer  $m$ )
```

```
integer[][]  $M \leftarrow \text{newinteger}[1 \dots n][1 \dots m]$   
for  $i \leftarrow 1$  to  $n$  do  $M[i, 0] \leftarrow 1$   
for  $j \leftarrow 1$  to  $m$  do  $M[0, j] \leftarrow 1$   
for  $i \leftarrow 1$  to  $n$  do  
    for  $j \leftarrow 1$  to  $m$  do  
         $M[i, j] = M[i - 1, j] + M[i, j - 1]$   
return  $M[n, m]$ 
```

Doppio mediano

```
mediana(integer[] X, integer[] Y, integer bx, ex, by, ey)
```

```
if  $e_x - b_x = 1$  then return mediana4(X, Y, bx, ex, by, ey)
```

```
integer mx =  $\lfloor (b_x + e_x)/2 \rfloor$ 
```

```
integer my =  $\lceil (b_y + e_y)/2 \rceil$ 
```

```
if  $X[m_x] < Y[m_y]$  then return mediana(X, Y, mx, ex, by, my)
```

```
if  $Y[m_y] > X[m_x]$  then return mediana(X, Y, bx, mx, my, ey)
```

```
return (X[mx], Y[my])
```

Stringhe primitive

$$C[i] = \begin{cases} \sum_{s \in S \wedge \text{check}(X, s, i)} C[i + |s|] & 1 \leq i \leq n \\ 1 & i = n + 1 \end{cases}$$

count(integer[] X , integer n , SET S , integer[] C , integer i)

if $C[i] = n + 1$ **then**

└ **return** 1

if $C[i] = \perp$ **then**

┌ $C[i] \leftarrow 0$

foreach $s \in S$ **do**

 ┌ **if** $\text{check}(X, s, i)$ **then**

 └ $C[i] \leftarrow \text{count}(X, n, S, C, i + |s|)$

return $C[i]$

La chiamata iniziale è $\text{count}(X, n, S, C, 1)$. Detto $m = \sum_{s \in S} |s|$, la complessità è $O(mn)$.