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 positivo $G[i]$, $1 \leq i \leq n$, 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.

Il gioco delle coppie

Scrivere un algoritmo che, dato un vettore A di n interi distinti (n pari), ritorna **true** se è possibile partizionare A in coppie di elementi che hanno tutte la stessa somma (intesa come la somma degli elementi della coppia), **false** altrimenti. Ad esempio:

$$7, 4, 5, 2, 3, 6$$

può essere partizionato in $7 + 2 = 4 + 5 = 3 + 6$.
Discutere la complessità e la correttezza – per questo esercizio, la dimostrazione di correttezza è importante e va scritta bene.

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

Descrivere in pseudo–codice un algoritmo che conta il numero di modi diversi in cui è possibile concatenere 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 $\mathsf{check}(X, s, i)$ che ritorna vero se la stringa s è contenuta nella stringa X a partire dalla posizione i. Il costo della chiamata a $\mathsf{check}()$ è $O(|s|)$, dove $|s|$ è la lunghezza di s. Ad esempio, se $X = \mathtt{1001}$ e $s = \mathtt{00}$, $\mathsf{check}(X, s, 2)$ ritorna vero, per tutti gli altri indici i ritorna falso.

Doppio mediano

Siano $X[1 \ldots n]$ e $Y[1 \ldots n]$ 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 (in caso di parità, guadagno decrescente)

Il gioco delle coppie - $O(n \log n)$

checkPairs(**integer**[ ] A, **integer** n)

sort(A, n)
integer $s \leftarrow A[1] + A[n]$
for $i \leftarrow 2$ **to** $n/2 - 1$ **do**
  if $A[i] + A[n - i + 1] \neq s$ **then**
    return false
return true

Dimostrazione: supponiamo per assurdo che esista un insieme di coppie che rispetti le condizioni per restituire **true**, in cui l'elemento maggiore M sia associato ad un elemento M' diverso dal minore m ($m < M'$). Quindi il minore m è associato ad un elemento m' diverso dal massimo M ($m' < M$). Allora $m + m' < M + M'$, il che contraddice l'ipotesi che tale insieme di coppie rispetti le condizioni per restituire **true**.

Il gioco delle coppie - $O(n)$, tabella hash

checkPairs(**integer**[ ] A, **integer** n)

integer $tot \leftarrow 0$
for $i \leftarrow 1$ **to** n **do** $tot \leftarrow tot + A[i]$
integer $S \leftarrow tot/(n/2)$
HASH $H \leftarrow$ Hash()
for $i \leftarrow 1$ **to** n **do**
  H.insert($A[i], A[i]$)
for $i \leftarrow 1$ **to** n **do**
  if H.lookup($S - A[i]$) = **nil then**
    return false
return true

Il gioco delle coppie - $O(n)$

checkPairs(**integer**[ ] A, **integer** n)

integer $S \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
  $S \leftarrow S + A[i]$
return $\min(A, n) + \max(A, n) \cdot (n/2) = S$

Prima elementare

---

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

---

if $n = 0$ **and** $m = 0$ **then**
  print V

if $n > 0$ **then**
  $V[i] \leftarrow$ "R"
  stampaCombinazioni$(V, i + 1, n - 1, m)$

if $m > 0$ **then**
  $V[i] \leftarrow$ "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[ ][ ] $M \leftarrow$ **new integer**$[1 \ldots n][1 \ldots m]$
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** n **do**
    $M[i, j] = M[i - 1, j] + M[i, j - 1]$
return $M[n, m]$

Stringhe primitive

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

---

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

```
if C[i] = n + 1 then
    return 1
if C[i] = ⊥ then
    C[i] ← 0
    foreach s ∈ S do
        if check(X, s, i) then
            C[i] ← count(X, n, S, C, i + |s|)
return C[i]
```

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

Doppio mediano

---

mediana(**integer**[ ] X, **integer**[ ] Y, **integer** b_x, e_x, b_y, e_y)

---

if $e_x - b_x = 1$ **then** **return** mediana4$(X, Y, b_x, e_x, b_y, e_y)$
integer $m_x = \lfloor (b_x + e_x)/2 \rfloor$
integer $m_y = \lceil (b_y + e_y)/2 \rceil$
if $X[m_x] < Y[m_y]$ **then** **return** mediana$(X, Y, m_x, e_x, b_y, m_y)$
if $Y[m_y] > X[m_x]$ **then** **return** mediana$(X, Y, b_x, m_x, m_y, e_y)$
return $(X[m_x], Y[m_y])$
