Fiumi e porti

Lungo un fiume ci sono n porti. A ciascuno di questi porti è possibile affittare una barca che può essere restituita ad un altro porto. E' praticamente impossibile andare controcorrente. Il costo dell'affitto di una barca da un punto di partenza i ad un punto di arrivo j, con $i < j$, è denotato con $C[i, j]$. E' possibile che per andare da i a j sia più economico effettuare alcune soste e cambiare la barca piuttosto che affittare un'unica barca. Se si affitta una barca in $k_1, k_2, k_3, \ldots, k_l$ (con $k_1 = 1, k_1 < k_2 < k_3 < k_l$) allora il costo totale è
$C[k_1, k_2] + C[k_2, k_3] + \ldots + C[k_{l-1}, k_l] + C[k_l, n]$.
Scrivere un algoritmo che dato in input i costi $C[i, j]$, determini il costo minimo per recarsi da 1 ad n. Analizzare la complessità computazionale dell'algoritmo proposto. Per un punteggio aggiuntivo, si stampino i porti in cui devono essere noleggiate le barche.

D20

Dati n dadi, con il dado i-esimo dotato di $F[i]$ facce numerate da 1 a $F[i]$, trovare il numero di modi diversi con cui è possibile ottenere una certa somma X sommando i valori di tutti i dadi. Ad esempio, avendo due dadi a quattro facce numerati da 1 a 4, il valore 7 è ottenibile in un solo modo non contando le possibili permutazioni: $3 + 4$. Avendo tre dadi sempre a 4 facce, il valore 6 è ottenibile in tre modi diversi non contando le possibili permutazioni: $1 + 1 + 4$, $1 + 2 + 3$, $2 + 2 + 2$.

Sottosequenze palindrome

Sia data una stringa $s[1 \ldots n]$ di n caratteri. Scrivere un algoritmo che restituisce la lunghezza della *sottosequenza* palindroma massimale contenuta all'interno di s. Ricordiamo che una sottosequenza è un sottoinsieme ordinato dei caratteri di s, anche non contigui.
Ricordiamo che una stringa palindroma si legge allo stesso modo da sinistra a destra e da destra a sinistra. Per massimale, si intende che non esistono sottosequenze palindrome più lunghe (ma possono esisterne altre della stessa lunghezza).
Ad esempio, se l'input è "BBABCBCAB", allora l'output deve essere 7 in quanto "BABCBAB" è la più lunga sottosequenza palindroma contenuta in essa. "BBBBB" e "BBCBB" sono anch'esse sottosequenze palindrome, ma non sono massimali.

Ballo di fine anno

Una scuola vuole organizzare un ballo di fine anno. Ci sono n maschi e m femmine. Ogni coppia di studenti (composta da un ragazzo ed una ragazza che intendono danzare insieme) ha dovuto registrarsi (altrimenti non avrebbero potuto danzare insieme). I regolamenti della scuola impongono che ogni data coppia non possa danzare insieme più di 3 volte. In più, ogni studente non può danzare più di 10 volte in totale. Potete assumere che il ballo duri abbastanza a lungo da permettere a tutti di completare le proprie danze, se le registrazioni lo permettono.

Trovare un algoritmo che, dato in input l'insieme dei maschi e delle femmine e l'insieme delle registrazioni, massimizzi il numero di danze in totale.

Discutere la complessità dell'algoritmo proposto.

Fiumi e porti (1)

$$best[i,j] = \begin{cases} C[i,j] & j = i+1 \\ \min\{C[i,j], \min_{i<k<j}\{best[i,k] + best[k,j]\}\} \end{cases}$$

```
boat(integer[][] C, integer n, integer[][] best, integer i, j)
if j = i + 1 then
    return C[i, j]
if best[i, j] = ⊥ then
    best[i, j] ← C[i, j]
    for k = i + 1 to j − 1 do
        if best[i, k] + best[k, j] < C[i, j] then
            best[i, j] ← best[i, k] + best[k, j]
return best[i, j]
```

Fiumi e porti (2)

$$best[i] = \begin{cases} 0 & i = n \\ \min_{i<j\leq n}\{C[i,j] + best[j]\} \end{cases}$$

integer boat(**integer**[ ][ ] C, **integer** n)

---

integer[ ] $best \leftarrow$ **new integer**$[1\ldots n]$
$best[n] \leftarrow 0$
for $i \leftarrow n-1$ **downto** 1 **do**
    $best[i] \leftarrow +\infty$
    for $j \leftarrow i+1$ **to** n **do**
        if $C[i,j] + best[j] < best[i]$ **then**
            $best[i] \leftarrow C[i,j] + best[j]$
return $best[1]$

D20

Utilizziamo la programmazione dinamica per il calcolo e calcoliamo il numero di modi $T[x, i]$ con cui è possibile ottenere un valore x con i primi i dadi:

$$T[x,i] = \begin{cases} \sum_{j=1}^{F[i]} T[x-j, i-1] & i > 0 \wedge x > 0 \\ 1 & x = 0 \wedge i = 0 \\ 0 & \text{altrimenti} \end{cases}$$

D20

Il problema di questa versione è che conta tutte le possibili permutazioni; per ovviare a questo, è possibile aggiungere un terzo parametro m che indica il valore minimo del dado che può essere considerato, che deve essere più alto o uguale dei valori già ottenuti:

$$T[x, i, m] = \begin{cases} \sum_{j=m}^{F[i]} T[x-j, i-1, j] & i > 0 \land x > 0 \\ 1 & x = 0 \land i = 0 \\ 0 & \text{altrimenti} \end{cases}$$

D20

```
dice(integer[] F, integer i, integer x, integer m, integer[][][] T)
if x = 0 and i = 0 then return 1
if x = 0 or i = 0 then return 0
if T[x, i, m] = ⊥ then
    T[x, i, m] ← 0
    for j ← m to F[i] do
        T[x, i, m] ← T[x, i, m] + dice(F, i − 1, X − j, j, T)
return T[x, i, m]
```

Il costo è pari a $O(nXM^2)$, dove M è il dado con il maggior numero di facce. Questo perchè ci sono nXM celle da riempire, ognuna delle quali viene riempita con costo $O(M)$.

Sottosequenza palindroma

Definiamo con $L[i, j]$ la lunghezza della più lunga sottosequenza palindroma contenuta nella sottostringa $s[i \ldots j]$.

$$S[i,j] = \begin{cases} 0 & j < i \\ 1 & j = i \\ L[i+1, j-1] + 2 & j > i \wedge s[i] = s[j] \\ \max\{L[i+1, j], L[i, j-1]\} & j > i \wedge s[i] \neq s[j] \end{cases}$$

Sottosequenza palindroma

```
longestPalindrome(integer[] s, integer n)
integer[][] L ← new integer[1...n][1...n]
for i ← 1 to n do
    for i ← 1 to n do
        L[i][j] ← ⊥
longRec(s, 1, n, L)
```

```
integer longRec(integer[] s, integer i, integer j, integer[][] L)
if j > i then return 0
if j = i then return 1
if L[i, j] = ⊥ then
    if s[i] = s[j] then
        L[i, j] = longRec(s, i + 1, j − 1, L) + 2
    else
        L[i, j] = max(longRec(s, i + 1, j, L), longRec(s, i, j − 1, L), )
return L[i, j]
```

Ballo di fine anno