Vicini-vicini – difficile

Siano dati due vettori $x[1 \ldots n]$ e $y[1 \ldots n]$ che rappresentano le coordinate cartesiane di n punti nel piano; ovvero il punto i-esimo è in posizione $(x[i], y[i])$. Scrivere un algoritmo che restituisca, fra tutte le coppie, quella con minima distanza.

Note:

- Esiste ovviamente una soluzione banale $O(n^2)$; si può fare di meglio?
- Questo problema ha applicazioni per esempio nei GIS software

I Promessi Sposi

"Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte; e il ponte, che ivi congiunge le due rive, par che renda ancor più sensibile all'occhio questa trasformazione, e segni il punto in cui il lago cessa, e l'Adda rincomincia, per ripigliar poi nome di lago dove le rive, allontanandosi di nuovo, lascian l'acqua distendersi e rallentarsi in nuovi golfi e in nuovi seni."

Quante volte questo testo contiene la sottosequenza "lucia"?

I Promessi Sposi

"Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte; e il ponte, che ivi congiunge le due rive, par che renda ancor più sensibile all'occhio questa trasformazione, e segni il punto in cui il lago cessa, e l'Adda rincomincia, per ripigliar poi nome di lago dove le rive, allontanandosi di nuovo, lascian l'acqua distendersi e rallentarsi in nuovi golfi e in nuovi seni."

Quante volte questo testo contiene la sottosequenza "lucia"?
Alcune considerazioni:

- Due sottosequenze sono diverse (e quindi vanno contate separatamente) se esiste almeno una differenza negli insiemi di caratteri utilizzati.
- Esempio: "did you go" contiene due volte la sottosequenza "dog"....

Vito's Family

The famous gangster Vito Deadstone is moving to New York. He has a very big family there, all of them living on Lamafia Avenue. Since he will visit all his relatives very often, he wants to find a house close to them. Indeed, Vito wants to minimize the total distance to all of his relatives and has blackmailed you to write a program that solves his problem.

Input For each test case you will be given the integer number of relatives r ($0 < r < 500$) and the street numbers (also integers) $s_1, s_2, ..., s_i, ..., s_r$ where they live ($0 < s_i < 30000$). Note that several relatives might live at the same street number.

Output For each test case, your program must write the minimal sum of distances from the optimal Vito's house to each one of his relatives. The distance between two street numbers s_i and s_j is $d_{i,j} = |s_i - s_j|$.

Costo partizione vettore $V[1 \dots mn]$

Costo $C(i,j)$ di un sottovettore $V[i \dots j]$ di V $C(i,j) = \sum_{t=i}^{j} V[t]$
k-partizione di V è una divisione di V in k sottovettori
$V[1, j_1], V[j_1 + 1, j_2], V[j_2 + 1, j_3], \dots, V[j_{k-1} + 1, j_k]$ con $j_k = n$ e
$j_t < j_{t+1}, \forall 1 \leq t < k$, ovvero tale per cui i sottovettori coprono totalmente il vettore e non si sovrappongono.
Costo della partizione è il costo massimo dei suoi sottovettori.
Dato un vettore V di n interi e un intero k,con $2 \leq k \leq n$, trovare una k-partizione di V di costo minimo
Esempio: $V = \{2, 3, 7, -7, 15, 2\}$, $k = 3$
1: $\{2, 3, 7\}, \{-7, 15\}, \{2\}$, costo $2 + 3 + 7 = 12$
2: $\{2, 3\}, \{7\}\{-7, 15, 2\}$, costo $-7 + 15 + 2 = 10$

1. Soluzione per $k = 2$ (suggerimento: $O(n)$).
2. Soluzione per $k = 3$ (suggerimento: $O(n^2)$).
3. Soluzione generale (suggerimento: $O(kn^2)$).

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: $1 + 1 + 4$, $1 + 2 + 3$, $2 + 2 + 2$.

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.

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.

2-partizione

2-partition(**integer**[ ] V, **integer** n)

integer $tot \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
  $tot \leftarrow tot + V[i]$
integer $sofar \leftarrow 0$
integer $min \leftarrow +\infty$
for $i \leftarrow 1$ **to** $n-1$ **do**
  $sofar \leftarrow sofar + V[i]$
  $min \leftarrow \mathsf{min}(min, \mathsf{max}(sofar, tot - sofar))$
return min

3-partizione

```
integer 3-partition(integer[] V, integer n)
integer[][] T ← new integer[0...n]
T[0] ← 0
for i ← 1 to n do
    T[i] ← T[i − 1] + V[i]
integer min ← +∞
for i ← 1 to n − 2 do
    for j ← i + 1 to n − 1 do
        integer temp ← max(T[i], T[j] − T[i], T[n] − T[j])
        min ← min(min, temp)
return min
```

k-partizione

Sia $M[i,t]$ il minimo costo associato al sottoproblema di trovare la migliore t-partizione nel vettore $V[1 \ldots i]$. Il problema iniziale corrisponde a $M[n,k]$ – ovvero trovare la migliore k-partizione in $V[1 \ldots n]$. Sfruttiamo un vettore di appoggio T definito come nel caso $k = 3$.

$$M[i,t] = \begin{cases} T[i] & t = 1 \\ +\infty & t > i \\ \min_{1 \le j < i} \max(M[j, t-1], T[i] - T[j]) & \text{altrimenti} \end{cases}$$

k-partizione

```
integer partition-rec(integer[] V, integer[] T, integer[][] M, integer i, integer t)
if t > i then return +∞
if t = 1 then return T[i]
if M[i,t] = ⊥ then
    integer M[i,t] ← +∞
    for j ← 1 to i − 1 do
        integer temp ← max(partition-rec(V,T,M,j,t − 1), T[i] − T[j])
        if temp < M[i,t] then M[i,t] ← temp
return M[i,t]
```

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:

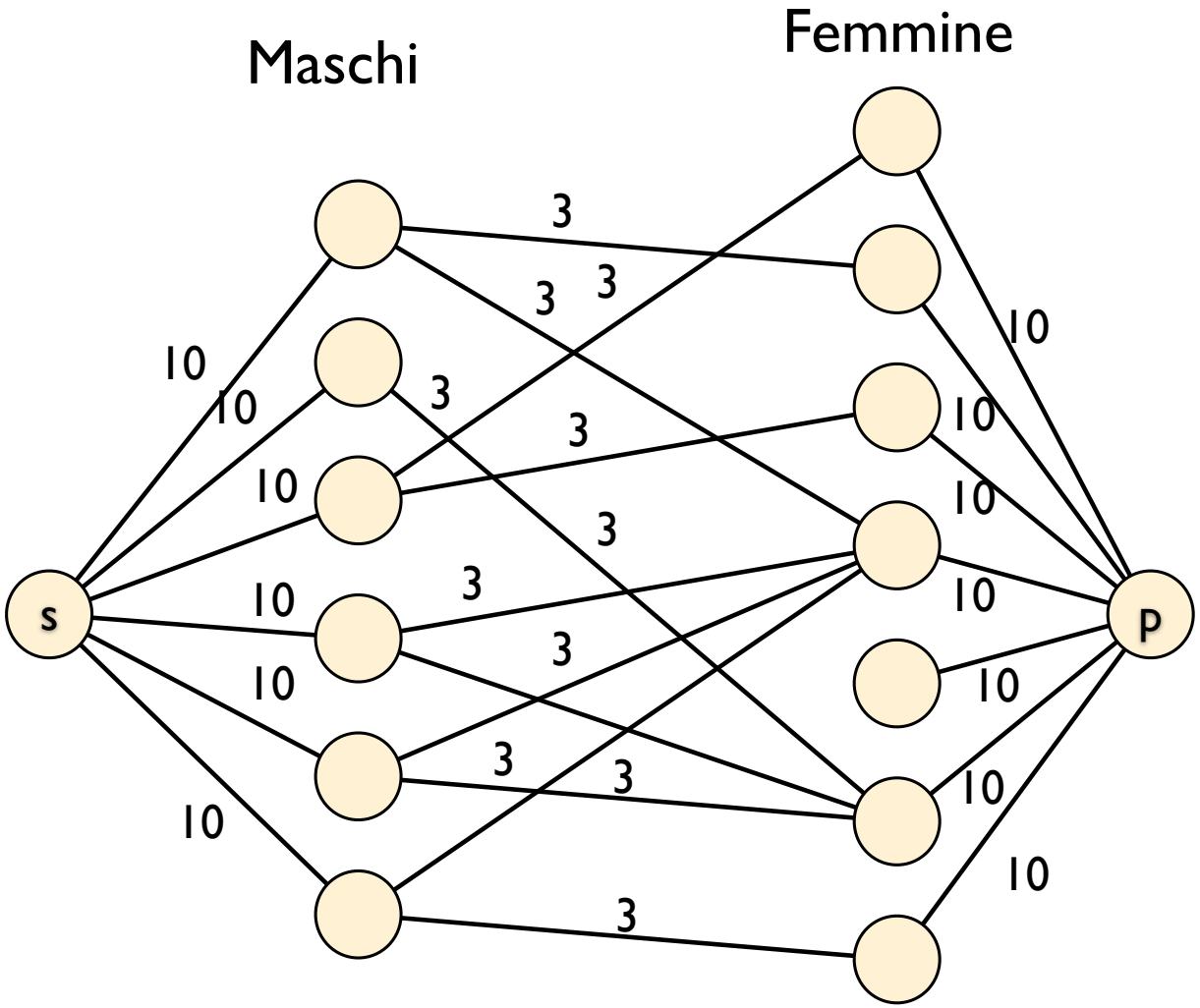
$$T[x,i,m] = \begin{cases} \sum_{j=m}^{F[i]} T[x-j,i-1,j] & i>0 \wedge x>0 \\ 1 & x=0 \wedge i=0 \\ 0 & \text{altrimenti} \end{cases}$$

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 $O(M)$.

Ballo di fine anno



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 $(X[m_x], Y[m_y])$
