Mosse su scacchiera

- Supponete di avere (i) una matrice P di dimensione $n \times n$ che rappresenta una scacchiera e (ii) un pedone che dovete muovere dall'estremità inferiore a quella superiore.
- Il 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.
- Quando una cella (x, y) viene visitata, guadagnate un profitto reale $P[x][u]$.

Scrivere un algoritmo che restituisca il massimo profitto ottenibile partendo da una qualunque cella dell'estremità inferiore e raggiungendo una qualunque cella dell'estremità superiore, seguendo le regole appena descritte.

6	7	4	7	$\underline{8}$
7	6	1	1	$\underline{4}$
3	5	7	$\underline{8}$	2
2	6	$\underline{7}$	0	2
7	3	5	$\underline{6}$	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.
Scrivere un algoritmo minpal(ITEM[]s) che ritorna il numero minimo di caratteri da **inserire** in s 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".

Ottimizza la somma

Supponete di avere in input un vettore di n interi positivi distinti $V[1 \dots n]$ e un valore W. Scrivere un algoritmo che:

1. restituisca il massimo valore $X = \sum_{i=1}^{n} x[i]V[i]$ tale che $X \leq W$ e ogni $x[i]$ è un intero non negativo;
2. stampi il vettore x.

Ad esempio, per $V[\,] = \{18, 3, 21, 9, 12, 24\}$ e $W = 17$, una possibile soluzione ottima è $x = [0, 2, 0, 1, 0, 0]$ da cui deriva $X = 15$.
Discutere correttezza e complessità.

Costo partizione di un vettore

Costo $C(i,j) = \sum_{t=i}^{j} V[t]$ di un sottovettore $V[i \dots j]$ di V è pari alla somma dei suoi elementi

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 k-partizione è il costo massimo dei suoi sottovettori.

Scrivere un algoritmo che dato un vettore V di n interi e un intero k, con $2 \leq k \leq n$, restituisca il costo di 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: $O(kn^2)$).

Spoiler alert!

Mosse su scacchiera

Definiamo una matrice DP tale che $DP[r][c]$ rapppresenta il massimo profitto che si può ottenere partendo da una cella sull'ultima riga e arrivando alla cella (r, c), dove r e c rappresentano la riga e la colonna, rispettivamente.

$$DP[r][c] = \begin{cases} -\infty & c < 1 \textbf{ or } c > n \\ P[n][c] & r = n \textbf{ and } 1 \leq c \leq n \\ \max \begin{pmatrix} DP[r+1][c-1], \\ DP[r+1][c], \\ DP[r+1][c+1] \end{pmatrix} & 1 \leq r < n \textbf{ and } 1 \leq c \leq n \end{cases}$$

Mosse su scacchiera

search-path(**int**[ ][ ] P, **int** n)

int $DP = \textbf{new int}[1 \ldots n][1 \ldots n]$
% Copia l'ultima riga nel vettore DP
for $c = 1$ **to** n **do**
  $DP[n][c] = P[n][c]$
for $r = n - 1$ **downto** 1 **do**
  for $c = 1$ **to** n **do**
    $DP[r][c] = -\infty$
    foreach $d \in \{-1, 0, +1\}$ **do**
      int $newc = c + d$
      if $newc \geq 1$ **and** $newc \leq n$ **then**
        $DP[r][c] = \max(DP[r][c], DP[r+1][newc] + P[x][y])$
% Restituisce il massimo della prima riga
return $\max(DP[1])$

Palindroma

- Se la stringa $s = \mathrm{a}s'\mathrm{a}$ è composta da due identici caratteri "a" iniziale e finale, allora:

$$\mathsf{minpal}(s) = \mathsf{minpal}(s')$$

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

$$\mathsf{minpal}(s) = \min\{\mathsf{minpal}(\mathrm{a}s'), \mathsf{minpal}(s'\mathrm{b})\} + 1$$

Palindroma

---

minpal(ITEM[ ] s, **int** n)

% Crea un vettore DP inizializzato a -1
int[ ][ ] DP = **new int**$[1 \ldots n][1 \ldots n] = \{-1\}$
return minpalRec$(S, DP, 1, n)$

---

minpalRec(ITEM[ ] s, **int**[ ][ ] DP, **int** i, **int** j)

---

if $j \leq i$ **then**
  return 0
if $DP[i][j] < 0$ **then**
  if $s[i] == s[j]$ **then**
    $DP[i][j] = \text{minpalRec}(s, i+1, j-1)$
  else
    $DP[i][j] = \min(\text{minpalRec}(s, i, j-1), \text{minpalRec}(s, i+1, j)) + 1$
return $DP[i][j]$

Ottimizza la somma

Sia $DP[i][w]$ il massimo valore che posso ottenere seguendo le regole di cui sopra avendo a disposizione i primi i oggetti e un valore massimo w.

$$DP[i][w] = \begin{cases} -\infty & i \geq 0 \wedge w < 0 \\ 0 & i = 0 \vee w = 0 \\ \max\{DP[i-1][w], DP[i][w - V[i]] + V[i]\} & \text{altrimenti} \end{cases}$$

Ottimizza la somma

int sum(**int**[ ] V, **int** n, **int** W)

% Crea un vettore DP inizializzato a -1
int[ ][ ] DP = **new int**$[0 \ldots n][1 \ldots W] = \{-1\}$
return sum(V, n, W, DP)

int sumR(**int**[ ] V, **int** i, **int** w, **int**[ ][ ] DP)

if $w < 0$ **then**
  return $-\infty$
if $i == 0$ **or** $w == 0$ **then**
  return 0
if $DP[i][w] < 0$ **then**
  $DP[i][w] =$
    $\max(\text{sumR}(V, i-1, w, DP), \text{sumR}(V, i, w - V[i], DP) + V[i])$
return $DP[i][w]$

2-partizione

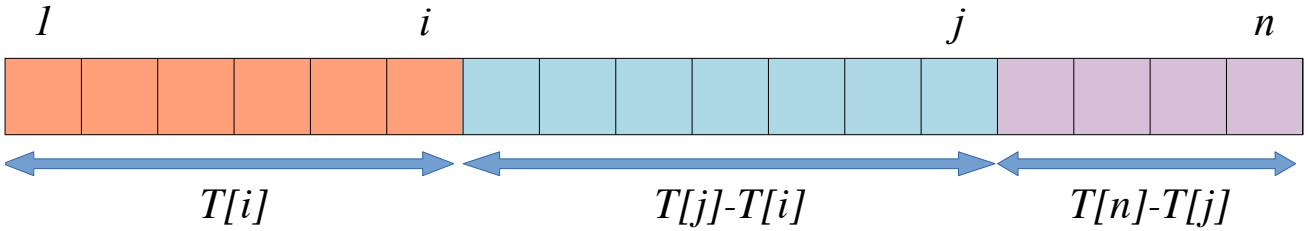
2-partition(**int**[ ] V, **int** n)

int $tot = 0$
for $i = 1$ **to** n **do**
  $tot = tot + V[i]$
int $totSoFar = 0$
int $minSoFar = +\infty$
for $i = 1$ **to** $n - 1$ **do**
  $totSoFar = totSoFar + V[i]$
  $minSoFar = \mathsf{min}(minSoFar, \mathsf{max}(totSoFar, tot - totSoFar))$
return $minSoFar$

3-partizione

int 3-partition(**int**[ ] V, **int** n)

int[ ][ ] T = **new int**$[1 \ldots n]$
$T[1] = V[1]$
for $i = 2$ **to** n **do**
  $T[i] = T[i-1] + V[i]$
int $minSoFar = +\infty$
for $i = 1$ **to** $n-2$ **do** % i: fine della prima partizione
  for $j = i+1$ **to** $n-1$ **do** % j: fine della seconda partizione
    $minSoFar = \min(minSoFar, \max(T[i], T[j]-T[i], T[n]-T[j]))$
return $minSoFar$



k-partizione

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

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

k-partizione

int partition(**int**[ ] V, **int** n, **int** k)

% Crea un vettore DP inizializzato a -1
int[ ][ ] DP = **new int**$[0 \ldots n][0 \ldots k] = \{+\infty\}$

% Crea il vettore delle somme parziali
int[ ][ ] T = **new int**$[1 \ldots n]$
$T[1] = V[1]$
for $i = 2$ **to** n **do**
  $T[i] = T[i-1] + V[i]$

return partitionRec(V, T, DP, n, k)

k-partizione

```
int partitionRec(int[] V, int[] T, int[][] DP, int i, int t)
if t > i then
    return +∞
if t == 1 then
    return T[i]
if DP[i][t] == +∞ then
    for j = 1 to i − 1 do
        int temp = max(partitionRec(V, T, DP, j, t − 1), T[i] − T[j])
        DP[i][t] = min(DP[i][t], temp)
return DP[i][t]
```