Esercizio 1 Un programma può essere eseguito indifferentemente su k supercomputer $S_1 ... S_k$ e per le prossime $n > 1$ unità di tempo si conosce il numero $x_{i,j}$, con $1 \leq i \leq k$ e $1 \leq j \leq n$, di cicli macchina che il supercomputer S_i può dedicare all'esecuzione del programma nell'unità di tempo j. L'esecuzione può partire da uno qualunque dei k supercomputer ed è possibile poi trasferirla da una macchina all'altra ma lo spostamento richiede un'unità di tempo. Bisogna pianificare quale delle k macchine utilizzare nelle prossime n unità di tempo in modo da massimizzare il numero di cicli macchina dedicati al programma.

Ad esempio: per $n = 6$, $k = 2$ ed $X =$

10	1	1	10	10	30
5	1	20	20	5	10

- La pianificazione $[1, 1, 1, 1, 1, 1]$ che prevede l'utilizzo della sola macchina S_1 ha costo $10 + 1 + 1 + 10 + 10 + 30 = 62$.
- La pianificazione $[2, 2, 2, *, 1, 1]$ che prevede di far partire la simulazione sulla macchina S_2 e cambiare poi macchina all'unità di tempo 4 ha valore $5 + 1 + 20 + 10 + 30 = 66$.
- La pianificazione che massimizza il numero di cicli macchina dedicati al programma è la $[1, *, 2, 2, *, 1]$ che vale $10 + 20 + 20 + 30 = 80$ e prevede di far partire la simulazione sulla macchina S_1 e cambiare macchina nelle unità di tempo 2 e 5.

Per risolvere il problema si propone il seguente algoritmo greedy:

```
INPUT: n, k, la tabella unità di tempo X = {x_{i,j} | 1 ≤ i ≤ k, 1 ≤ j ≤ n}
    SOL[1] ← 1
    FOR i = 2 TO k DO
        IF x_{i,1} + x_{i,2} > x_{SOL[1],1} + x_{SOL[1],2} THEN SOL[1] ← i
    j ← 2
    WHILE j < n DO
        max ← 1
        FOR i = 1 TO k DO
            IF x_{i,j+1} > x_{max,j+1} THEN max ← i
        m ← SOL[j − 1]
        IF x_{m,j} + x_{m,j+1} ≥ x_{max,j+1} THEN
            SOL[j] ← m
            j ← j + 1
        ELSE
            SOL[j] ← *
            SOL[j + 1] ← max
            j ← j + 2
        ENDIF
    ENDWHILE
    IF j = n THEN SOL[n] = SOL[n − 1]
OUTPUT SOL
```

1. *(max 10 punti)* Provare che l'algoritmo proposto non è corretto fornendo un'istanza per cui la pianificazione SOL prodotta non è quella di valore massimo.
2. *(max 10 punti)* Proporre un algoritmo che in tempo $O(n\,k^2)$ calcola il costo della pianificazione di valore massimo (*Suggerimento: utilizzare la programmazione dinamica calcolando i $k \cdot n$ valori $M[i, j]$, con $1 \leq i \leq k$ e $1 \leq j \leq n$ dove $M[i, j]$ è il valore della pianificazione di valore massimo per le prime j unità di tempo quando nella j-esima unità si utilizza la macchina S_i.*

3. *(max 10 punti)* Modificare l'algoritmo proposto al punto 2 in modo da avere in output una pianificazione di valore massimo anzicché il suo valore. La modifica deve avere un costo additivo $O(n\,k)$.

Soluzione Esercizio 1

1. Si consideri l'istanza: $n = 4$, $k = 2$ ed $X =$

1	1	1	100
1	1	3	1

   L'algoritmo produce la pianificazione $[1, *, 2, 2]$ di valore $1 + 3 + 1 = 5$ mentre la pianificazione di valore massimo è $[1, 1, 1, 1]$ ed ha valore $1 + 1 + 1 + 100 = 103$.

2. Per $j = 1$ ovviamente si ha $M[i, 1] = x_{i,1}$ e per $j = 2$ si ha $M[i, 2] = x_{i,1} + x_{i,2}$ in quanto nella prima unità di tempo non è possibile effettuare cambi di macchina. Per $j > 2$ nella pianificazione di valore massimo possono accadere due cose, o nell'unità di tempo precedente $j - 1$ non c'è stato cambio di macchina (ed allora il valore non può che essere $M[i, j - 1] + x_{i,j}$) o il cambio di macchina c'è stato (e allora il valore non può che essere $\max_{1 \le a \le k,\, a \ne i}\{M[a, j - 2]\} + x_{i,j}$). Si ha quindi:

$$M[i,j] = \begin{cases} x_{i,1} & \text{se } j = 1 \\ x_{i,1} + x_{i,2} & \text{se } j = 2 \\ \max\{M[i, j-1], \max_{1 \le a \le k,\, a \ne i}\{M[a, j-2]\}\} + x_{i,j} & \text{altrimenti} \end{cases}$$

   Poiché il valore della pianificazione ottima è $\max_{1 \le a \le k}\{M[a, n]\}$ l'algoritmo per calcolarlo è:

```
INPUT: n, k, la tabella delle unità di tempo X = {x_{i,j} | 1 ≤ i ≤ k, 1 ≤ j ≤ n}
FOR i ← 1 TO k DO
    M[i, 1] ← x_{1,1}
    M[i, 2] ← x_{i,1} + x_{i,2}
ENDFOR
FOR j ← 3 TO n DO
    a ← 1
    FOR i ← 2 TO k DO
        IF M[i, j − 2] > M[a, j − 2] THEN a ← i
    ENDFOR
    IF M[i, j − 1] > M[a, j − 2] THEN M[i, j] ← M[i, j − 1] + x_{i,j}
        ELSE M[i, j] ← M[a, j − 2] + x_{i,j}
ENDFOR
SOL ← M[1, n]
FOR i ← 2 TO k DO
    IF M[i, n] > SOL THEN SOL ← M[i, n]
OUTPUT SOL
```

3. Per ottenere la pianificazione di valore massimo basta ripercorrere all'indietro le scelte che hanno portato a calcolarne il valore sostituendo quindi le ultime quattro righe di codice dell'algoritmo precedente con quelle che seguono:

```
SOL[n] ← 1
FOR i ← 2 TO k DO
    IF M[i, n] > SOL[n] THEN SOL[n] ← i
m ← SOL[n]
j ← n − 1
WHILE j > 0 DO
    IF M[m, j + 1] = M[m, j] + x_{m,j+1} THEN
        SOL[j] ← m
        j ← j − 1
    ELSE
        SOL[j] ← *
        FOR i ← 1 TO k DO
            IF i ≠ m AND M[m, j + 1] = M[i, j − 1] + x_{m,j+1} THEN
                m ← i
                SOL[j − 1] ← m
                BREAK
        j ← j − 2
ENDWHILE
OUTPUT SOL
```