

Esercizio Java n. 1: Espandi Array

Esercizio estratto e adattato da – Compito I Appello – del 26/05/2021.

Sia V un array contenente n numeri interi positivi (strettamente maggiori di zero), con almeno un elemento ($n > 0$). L'espansione dell'array V consiste nel generare una matrice M in cui ciascuna colonna k contiene $V[k]$ ripetizioni dell'elemento dell'array $V[k]$, a partire dalla prima riga della matrice e lasciando a zero gli eventuali elementi rimanenti della colonna.

Ad esempio, sia $V = [2, 4, 1, 2]$, l'espansione dell'array V genera la seguente matrice

$$M = \begin{pmatrix} 2 & 4 & 1 & 2 \\ 2 & 4 & 0 & 2 \\ 0 & 4 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}, \text{ infatti:}$$

- nella prima colonna della matrice, il numero 2 ($V[0]$) è ripetuto per 2 volte (gli elementi rimanenti della colonna sono lasciati a zero);
- nella seconda colonna della matrice, il numero 4 ($V[1]$) è ripetuto per 4 volte;
- nella terza colonna della matrice, il numero 1 ($V[2]$) è ripetuto per 1 volta (gli elementi rimanenti della colonna sono lasciati a zero);
- nella quarta colonna della matrice, il numero 2 ($V[3]$) è ripetuto per 2 volte (gli elementi rimanenti della colonna sono lasciati a zero).

Altri esempi: se $V = [3]$, la sua espansione è la matrice $M = \begin{pmatrix} 3 \\ 3 \\ 3 \end{pmatrix}$; se $V = [1]$, la sua espansione è la matrice $M = (1)$.

Scrivere un metodo Java-- chiamato `espandiArray` che, dato in input un array V di n numeri interi positivi, con $n > 0$, restituisca la matrice generata dall'espansione dell'array.

SUGGERIMENTO PER SOLUZIONE “STANDARD”: Una soluzione semplice (ed efficiente) consiste nel costruire la matrice M lavorando su una colonna alla volta, ovvero iniziando a inserire i valori corretti nella prima colonna della matrice, poi inserendo i valori corretti nella seconda colonna, e così via fino all'ultima colonna della matrice.

DIFFICOLTA' EXTRA (FACOLTATIVA): Gli studenti che vogliono aumentare la difficoltà e la complessità dell'esercizio (e quindi anche ottenere una valutazione migliore in caso di soluzione corretta), possono assumere che ci sia il seguente vincolo aggiuntivo: la matrice M **deve** essere costruita lavorando su una riga alla volta, ovvero iniziando a inserire i valori corretti nella prima riga della matrice, poi inserendo i valori corretti nella seconda riga, e così via fino all'ultima riga della matrice.

NOTA BENE:

- Gli studenti dovranno consegnare per questo esercizio solo 1 sorgente relativo alla soluzione “standard” oppure adottando la soluzione con “difficoltà extra”.
- Nello svolgere l'esercizio NON devono essere utilizzati i metodi `clone`, o `arraycopy`, o metodi della classe `Arrays`. L'utilizzo di tali metodi renderà l'esercizio automaticamente insufficiente.

Esercizio Java n. 2: Visita a Serpentina

Esercizio estratto e adattato da – *Compito II Appello* – del 29/06/2021.

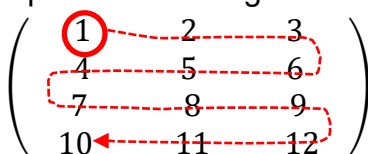
Sia M una matrice di numeri interi di dimensione $m \times n$ (con $m > 0$, $n > 0$ e m numero pari). Partendo dall'elemento della matrice di posizione $(riga, colonna)$, con $0 \leq riga < m$ e $0 \leq colonna < n$, la visita a serpentina della matrice M genera un array di interi i cui elementi corrispondono a quelli della matrice visitati seguendo queste regole:

- Si parte dalla cella di posizione $(riga, colonna)$ e si visita la parte di riga rimanente da sinistra verso destra.
- Poi si passa alla riga successiva e si visita tutta la riga da destra verso sinistra.
- Poi si passa alla riga successiva e si visita tutta la riga da sinistra verso destra.
- ...
- E così via, fino a visitare tutti gli elementi della matrice.
- Nota bene: la visita è da intendersi come circolare, ovvero la riga successiva all'ultima riga corrisponde alla prima riga della matrice.

Ad esempio, sia $M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}$, con $m=4$ (pari) e $n=3$.

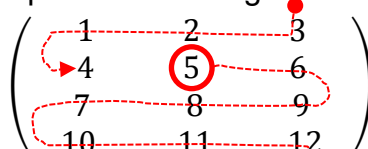
- Se $(riga, colonna) = (0, 0)$, la visita a serpentina di M genera il seguente array:

[1,2,3,6,5,4,7,8,9,12,11,10]. Infatti:



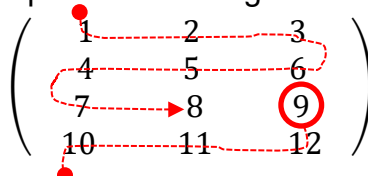
- Se $(riga, colonna) = (1, 1)$, la visita a serpentina di M genera il seguente array:

[5,6,9,8,7,10,11,12,3,2,1,4]. Infatti:



- Se $(riga, colonna) = (2, 2)$, la visita a serpentina di M genera il seguente array:

[9,12,11,10,1,2,3,6,5,4,7,8]. Infatti:



Scrivere un metodo Java-- chiamato `visitaSerpentina` che, data una matrice M di numeri interi di dimensione $m \times n$ (con $m > 0$, $n > 0$ e m numero pari), e dati due interi $riga$ e $colonna$ (con $0 \leq riga < m$ e $0 \leq colonna < n$), restituisca l'array di interi generato dalla visita a serpentina della matrice M a partire dalla posizione $(riga, colonna)$.

DIFFICOLTA' RIDOTTA: E' possibile svolgere l'esercizio assumendo che la cella di partenza sia sempre quella di posizione (0,0), ovvero assumendo che il metodo possa essere invocato soltanto con $riga=0$ e $colonna=0$ (è la situazione corrispondente al primo

dei tre esempi precedenti). In questo caso i **JUnit Test** che devono essere superati sono solo quelli della classe **VisitaSerpentinaRidottaTest**.

DIFFICOLTA' STANDARD (FACOLTATIVA): Gli studenti che vogliono aumentare la difficoltà e la complessità dell'esercizio (e quindi anche ottenere una valutazione migliore in caso di soluzione corretta), possono svolgere l'esercizio nella sua forma originale, ovvero considerando che la visita a serpentina può cominciare da una qualsiasi cella di posizione (*riga,colonna*), con $0 \leq riga < m$ e $0 \leq colonna < n$. In questo caso i **JUnit Test** che devono essere superati sono solo quelli della classe **VisitaSerpentinaStandardTest**.

NOTA BENE:

- Ricordarsi del vincolo relativo al numero di righe m , che deve essere pari!
- Gli studenti dovranno consegnare per questo esercizio **solo 1 sorgente** relativo alla soluzione con “difficoltà ridotta” oppure relativo alla soluzione con “difficoltà standard”.
- Ci sono due classi di test per l'Esercizio 1 ma solo una tra le due classi di test deve essere superata, ovvero:
 - Per la soluzione con “**difficoltà ridotta**”: i JUnit Test che devono essere superati sono solo quelli della classe **VisitaSerpentinaRidottaTest** (non considerare quelli della classe `VisitaSerpentinaStandardTest` che ovviamente falliranno);
 - Per la soluzione con “**difficoltà standard**”: i JUnit Test che devono essere superati sono solo quelli della classe **VisitaSerpentinaStandardTest** (se questi test saranno superati, lo saranno comunque anche quelli dell'altra classe `VisitaSerpentinaRidottaTest` essendo questi ultimi un sottoinsieme dei test standard).
- Nello svolgere l'esercizio **NON** devono essere utilizzati i metodi `clone`, o `arraycopy`, o metodi della classe `Arrays`. L'utilizzo di tali metodi renderà l'esercizio automaticamente insufficiente.