

Esercizio Java n. 1: Trova la parola in diagonale

Esercizio estratto e adattato da – Compito III Appello – 12/07/2022

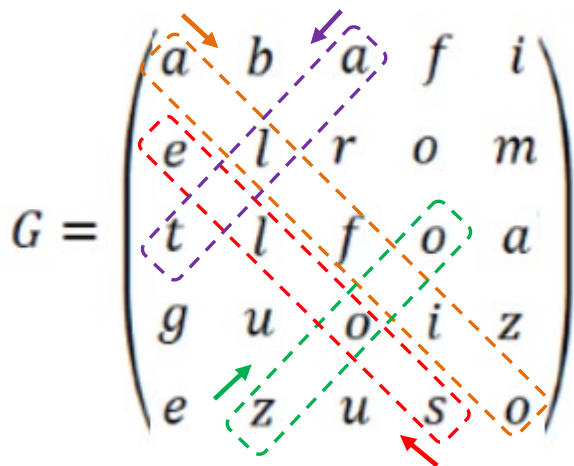
Il gioco del “trova la parola” della settimana enigmistica consiste nel trovare una certa parola (sequenza di caratteri) all’interno di una griglia contenente dei caratteri disposti a caso (di solito bisogna cerchiare con la matita la parola all’interno della griglia). Nel gioco “trova la parola in diagonale” la ricerca della parola avviene solo in diagonale (quindi non in orizzontale o in verticale), ovvero in tutte le diagonali della matrice nelle 4 possibili direzioni:

- Da Alto-Sinistra verso Basso-Destra (direzione ↘);
- Da Basso-Destra verso Alto-Sinistra (direzione ↗);
- Da Basso-Sinistra verso Alto-Destra (direzione ↗);
- Da Alto-Destra verso Basso-Sinistra (direzione ↘).

Sia G una matrice di caratteri di dimensione $m \times n$, con $m > 0$ e $n > 0$, che rappresenta la griglia nella quale ricercare la parola, e sia p un array di caratteri di dimensione k , con $k > 0$, che rappresenta la parola da ricercare nella griglia.

Scrivere un metodo Java-- chiamato `trovaParolaDiagonale` che, data una matrice G di caratteri di dimensione $m \times n$, con $m > 0$ e $n > 0$, e dato un array di caratteri p di dimensione k , con $k > 0$, restituisca `true` se esiste almeno una occorrenza della parola p in una diagonale della matrice G , `false` altrimenti.

Esempio: sia





- sia `sole` la parola da ricercare, allora il metodo deve restituire `true` poiché la parola `sole` è presente nella griglia scritta in diagonale. (esempio 1, direzione ↗)
- sia `alt` la parola da ricercare, allora il metodo deve restituire `true` poiché la parola `alt` è presente nella griglia scritta in diagonale. (esempio 2, direzione ↘)
- sia `zoo` la parola da ricercare, allora il metodo deve restituire `true` poiché la parola `zoo` è presente nella griglia scritta in diagonale. (esempio 3, direzione ↗)
- sia `alfio` la parola da ricercare, allora il metodo deve restituire `true` poiché la parola `alfio` è presente nella griglia scritta in diagonale. (esempio 4, direzione ↘)

- sia *zio* la parola da ricercare, allora il metodo deve restituire `false` poiché la parola *zio* NON è presente in nessuna diagonale della griglia. (esempio 5)

Consegna esercizio con difficoltà [MEDIA]:

E' possibile svolgere lo stesso esercizio assumendo di effettuare la ricerca della parola in tutte le diagonali della matrice considerando solo le due direzioni:

- Da Alto-Sinistra verso Basso-Destra (direzione ); ← si veda **esempio 4**.
- Da Basso-Destra verso Alto-Sinistra (direzione ); ← si veda **esempio 1**.

Consegna esercizio con difficoltà [RIDOTTA]:

E' possibile svolgere lo stesso esercizio assumendo di effettuare la ricerca della parola in tutte le diagonali della matrice considerando solo l'unica direzione:

- Da Alto-Sinistra verso Basso-Destra (direzione ); ← si veda **esempio 4**.

JUnit Test: I JUnit Test che devono essere superati sono i seguenti:

- per la consegna con difficoltà standard (tutte e 4 le direzioni): test della classe **TrovaParolaTest**.
- per la consegna con difficoltà "media" (solo 2 direzioni): test della classe **TrovaParolaMediaTest** (non considerare i test della classe TrovaParolaTest che ovviamente falliranno).
- per la consegna con difficoltà "ridotta" (solo 1 direzione): test della classe **TrovaParolaRidottaTest** (non considerare i test della classe TrovaParolaTest e TrovaParolaMediaTest che ovviamente falliranno).

Suggerimenti:

- Per la consegna con difficoltà "ridotta": a partire da una cella di indice (r,c), verificare che gli elementi nella diagonale corrispondano al contenuto di p (letto da sinistra verso destra).
- Per la consegna con difficoltà "media": a partire da una cella di indice (r,c), verificare che gli elementi nella diagonale corrispondano al contenuto di p (letto da sinistra verso destra) o al contenuto di p letto al contrario (da destra verso sinistra).
- Per la consegna con difficoltà "standard": seguire il suggerimento per la consegna con difficoltà "media" anche per trattare le due direzioni aggiuntive (da Basso-Sinistra verso Alto-Destra, e da Alto-Destra verso Basso-Sinistra).

NOTA BENE:

- Gli studenti dovranno consegnare per questo esercizio solo 1 sorgente relativo alla consegna standard, oppure relativo alla consegna con difficoltà "media" oppure relativo alla consegna con difficoltà "ridotta".

- 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: genera matrice da array

Esercizio estratto e adattato da – Compito IV Appello – 08/09/2022.

Sia arr un array di interi di dimensione k , con $k > 0$. La procedura di generazione di una matrice a partire dall'array arr consiste nel generare una matrice M di interi “concatenando” le k matrici quadrate costruite a partire dagli elementi dell'array arr .

La procedura di costruzione delle matrici quadrate può essere descritta come segue:

- Ciascun elemento dell'array arr definisce la dimensione della corrispondente matrice quadrata. Se arr è composto da k elementi, avrò quindi k matrici quadrate M_0, M_1, \dots, M_{k-1} di dimensione, rispettivamente, $arr[0] \cdot arr[0]$, $arr[1] \cdot arr[1]$, ..., $arr[k-1] \cdot arr[k-1]$.
- Ciascuna delle k matrici quadrate viene quindi riempita a valori interi consecutivi riempiendo le righe da sinistra verso destra e dall'alto verso il basso, come segue:
 - o La sequenza di interi della matrice M_0 inizia con il valore 1 e termina con il valore $arr[0] \cdot arr[0]$;
 - o La sequenza di interi della matrice M_1 inizia con il valore $arr[0] \cdot arr[0] + 1$ e termina con il valore $arr[0] \cdot arr[0] + arr[1] \cdot arr[1]$;
 - o ... e così via ...

In pratica, la sequenza di interi continua passando da una matrice quadrata all'altra, e quindi il primo elemento di una matrice M_{j+1} (ovvero $M_{j+1}[0][0]$) corrisponde al valore successivo dell'ultimo elemento inserito nella matrice quadrata precedente M_j (ovvero $M_j[arr[j]][arr[j]]$).

Ad esempio, sia $arr = [2, 4, 1, 3]$, le matrici quadrate M_0 (di dimensione 2×2), M_1 (di dimensione 4×4), M_2 (di dimensione 1×1) e M_3 (di dimensione 3×3) generate a partire dai quattro elementi dell'array sono le seguenti:

$$M_0 = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad M_1 = \begin{pmatrix} 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 \end{pmatrix} \quad M_2 = (21) \quad M_3 = \begin{pmatrix} 22 & 23 & 24 \\ 25 & 26 & 27 \\ 28 & 29 & 30 \end{pmatrix}$$

La matrice M generata è quindi data dalla “concatenazione” delle k matrici quadrate, allineandole alla prima riga e mettendo a zero gli altri elementi.

Nel precedente esempio, la matrice M risultante dalla “concatenazione” delle matrici M_0, M_1, M_2 e M_3 è quindi la seguente:

$$M = \begin{pmatrix} \begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} & \begin{matrix} 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 \end{matrix} & \begin{matrix} 21 \\ 0 \\ 0 \\ 0 \end{matrix} & \begin{matrix} 22 & 23 & 24 \\ 25 & 26 & 27 \\ 28 & 29 & 30 \\ 0 & 0 & 0 \end{matrix} \end{pmatrix}$$

Consegna obbligatoria:

Scrivere un metodo Java-- chiamato `generaMatriceDaArray` che, dato un array `arr` di interi di dimensione k , con $k > 0$, restituisca una matrice di interi M generata a partire dall'array `arr` come precedentemente descritto. **NOTA BENE:** l'algoritmo risolutivo deve essere di tipo ITERATIVO, NON ricorsivo.

Consegna aggiuntiva (FACOLTATIVA):

Scrivere un metodo Java-- chiamato `generaMatriceDaArrayRicorsivo` che risolve lo stesso esercizio implementando però una soluzione RICORSIVA che elimina l'iterazione sugli elementi dell'array di partenza `arr` (non eliminare le altre iterazioni necessarie per il calcolo della dimensione di M e per il riempimento della matrice).

Suggerimento:

Non è necessario creare effettivamente le k matrici quadrate, è possibile lavorare direttamente sulla matrice M .

JUnit Test: I JUnit Test che devono essere superati sono i seguenti:

- per la consegna obbligatoria (soluzione iterativa): test della classe **GeneraMatriceDaArrayTest**. Se non verrà svolta la consegna aggiuntiva (facoltativa), non considerare i test della classe **GeneraMatriceDaArrayRicorsivoTest** che ovviamente falliranno.
- per la consegna aggiuntiva facoltativa (soluzione ricorsiva): test della classe **GeneraMatriceDaArrayRicorsivoTest**.

NOTA BENE:

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.