Ingegneria Informatica *A.A. 2021-2022*
Programmazione Avanzata [38090 – MOD2] *Gianluca Maffioletti [1059062]*

RELAZIONE PROGETTO HASKELL

1 – INTRODUZIONE

Il programma realizzato consente di calcolare il determinante di una matrice fornita in input dall'utente.
Il determinante è un valore numerico associato a ciascuna matrice quadrata, ossia il cui numero di righe è equivalente al numero di colonne.

Per il calcolo del determinante di una matrice di ordine n qualsiasi si ricorre al teorema di Laplace, che sfrutta delle formule ricorsive, dette sviluppi di Laplace, applicabili per righe o per colonne.
Considerando una matrice quadrata di ordine n

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

Denotiamo con A_{ij} la matrice che si ottiene eliminando la riga i e la colonna j dalla matrice A.
Fissato poi un elemento $a_{ij} \in A$, definiamo il complemento algebrico di a_{ij} come

$$C_{ij} = (-1)^{i+j} \cdot \det(A_{ij})$$

Consideriamo anche che il determinante di una matrice di ordine 1, ossia composta da un solo elemento, è uguale all'elemento stesso.

Sviluppo di Laplace per righe: il determinante di una matrice A è pari alla sommatoria dei prodotti degli elementi della riga scelta per i rispettivi complementi algebrici:

$$\det(A) = \sum_{j=1}^{n} \left[a_{ij} \cdot C_{ij}\right] = \sum_{j=1}^{n} \left[a_{ij} \cdot (-1)^{i+j} \cdot \det(A_{ij})\right]$$

Sviluppo di Laplace per colonne: il determinante di una matrice A è pari alla sommatoria dei prodotti degli elementi della colonna scelta per i rispettivi complementi algebrici:

$$\det(A) = \sum_{i=1}^{n} \left[a_{ij} \cdot C_{ij}\right] = \sum_{i=1}^{n} \left[a_{ij} \cdot (-1)^{i+j} \cdot \det(A_{ij})\right]$$

Ci sono due casi particolari, ai quali si possono applicare delle formule semplificate.
Considerando una matrice quadrata di ordine 2

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Il determinante è dato dal prodotto degli elementi della diagonale principale meno il prodotto degli elementi dell'antidiagonale:

$$\det(A) = (a \cdot d) - (b \cdot c)$$

Considerando invece una matrice quadrata di ordine 3

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Il determinante viene calcolato applicando la regola di Sarrus:

$$\begin{aligned}\det(A) = a_{11} \cdot a_{22} \cdot a_{33} + a_{12} \cdot a_{23} \cdot a_{31} + a_{13} \cdot a_{21} \cdot a_{32} \\ - (a_{13} \cdot a_{22} \cdot a_{31} + a_{12} \cdot a_{21} + a_{33} + a_{11} \cdot a_{23} \cdot a_{32})\end{aligned}$$

Per ricavare la formula in maniera visuale, è possibile accostare le prime due colonne della matrice a destra della matrice stessa e calcolare la differenza tra la somma dei prodotti delle tre diagonali complete da sinistra verso destra e la somma delle tre antidiagonali complete da destra verso sinistra:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{matrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{matrix} \quad - \quad \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{matrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{matrix}$$

Infine, si definisce triangolare superiore (inferiore) una matrice quadrata di ordine n qualsiasi i cui elementi sopra (sotto) la diagonale principale sono nulli.

$$A = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \qquad A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix}$$

In particolare, una matrice che è sua triangolare superiore sia triangolare inferiore viene detta matrice diagonale

$$A = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix}$$

Considerando una matrice triangolare, è possibile calcolare il determinante come prodotto degli elementi sulla diagonale principale:

$$\det(A) = a_{11} \cdot a_{22} \cdot \ldots \cdot a_{nn}$$

2 – FUNZIONAMENTO

Innanzitutto, il programma chiede in input all'utente l'inserimento di una matrice di numeri in forma di lista.

Ad esempio

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \Rightarrow [1, 2, 3, 4]$$

Successivamente, si controlla che la matrice inserita sia quadrata (condizione necessaria per il calcolo del determinante). In caso negativo, viene stampato a video il messaggio "la matrice inserita non è quadrata".

Procedendo, viene effettuato un controllo sulla dimensione della matrice, in base al quale si distinguono i seguenti casi:

- Se la matrice è quadrata di ordine 1, viene restituito l'elemento della matrice;
- Se la matrice è quadrata di ordine 2, viene calcolato il determinante tramite il metodo semplificato $(a \cdot d - b \cdot c)$ e restituito;
- Se la matrice è quadrata di ordine 3, viene calcolato il determinante tramite il metodo di Sarrus e restituito;
- Se la matrice è quadrata di ordine superiore al 3 ed è triangolare, viene calcolato il determinante tramite il prodotto degli elementi sulla diagonale;
- Altrimenti, viene calcolato il determinante tramite il teorema di Laplace e restituito.

Il valore ottenuto viene poi stampato a video.

3 – FUNZIONI HASKELL

Innanzitutto, nel main viene richiesta in input la matrice, che viene mappata come lista di numeri in virgola mobile *Float*.

```haskell
main = do

  putStrLn "\nInserisci una matrice (in forma di lista): "
  input <- getLine
  let matrix = read input :: [Float]
  if isSquare matrix
  then putStrLn ( toString matrix (length matrix) ++
                  "\nIl determinante della matrice inserita è: "
                  ++ show(det matrix) )
  else putStrLn "La matrice inserita non è quadrata"
```

Per verificare se la matrice fornita è quadrata è definita la funzione *isSquare*, che prende in input la matrice e restituisce un valore booleano che indica se la radice quadrata della lunghezza della lista rappresentante la matrice è un intero. Infatti, ad esempio, se la lunghezza della lista è 4, allora la matrice è quadrata di ordine $\sqrt{4} = 2$, mentre se la lunghezza della lista è 5, il numero $\sqrt{5}$ non è intero e la matrice quindi non è quadrata.

```haskell
isInt :: Float -> Bool
isInt x = x == fromInteger ( round x )

isSquare :: [Float] -> Bool
isSquare m = isInt ( sqrt ( fromIntegral ( length m ) ) )
```

La funzione *toString* serve per stampare la matrice in una forma più leggibile. Prende in input la matrice e la lunghezza della stessa e stampa gli elementi separati per riga.

```haskell
toString :: [Float] -> Int -> String
toString [] l = "   |  "
toString m l | length m `mod` n == 0 = "   |   " ++ show (head m) ++ toString (tail m) l
             | otherwise             = "   " ++ show (head m) ++ toString (tail m) l
             where n = round ( sqrt ( fromIntegral l ) )
```

Il calcolo del determinante inizia valutando l'ordine della matrice.
Le diverse condizioni vengono elencate con il costrutto delle guardie di Haskell.

```haskell
det :: [Float] -> Float
det m | length m == 1  = head m
      | length m == 4  = detM2 m
      | length m == 9  = detM3 m
      | isTriangular m = detMT m
      | otherwise      = detMN m
```

Se la lunghezza della lista è uguale a 1, viene restituita la testa della lista, che corrisponde al suo unico elemento.
Per il calcolo del determinante di una matrice quadrata di ordine 2 oppure 3, ossia quando la lunghezza della lista è rispettivamente uguale a 4 e 9, vengono definite le funzioni *detM2* e *detM3*, che sfruttano il pattern matching per riconoscere gli elementi all'interno della matrice e calcolano il determinante ricorrendo alla formula delle diagonali e alla regola di Sarrus.

```haskell
detM2 :: [Float] -> Float
detM2 [a,b,c,d] = a * d - b * c

detM3 :: [Float] -> Float
detM3 [m11,m12,m13,m21,m22,m23,m31,m32,m33] = m11*m22*m33 + m12*m23*m31 + m13*m21*m32 -
                                              (m13*m22*m31 + m11*m23*m32 + m12*m21*m33)
```

Se l'ordine della matrice è maggiore di 3, innanzitutto si verifica se la matrice è triangolare, in modo da applicare il calcolo semplificato se possibile, e altrimenti procedere con la formula generica.

Prima di definire i rispettivi algoritmi, occorre introdurre delle funzioni ausiliari che hanno lo scopo di ottimizzare la struttura della lista-matrice per ricavare e utilizzare gli indici di riga e di colonna di ciascun elemento.

In particolare, la lista di numeri viene trasformata in una lista di tuple contenenti tre valori: il primo è l'elemento della matrice, il secondo e il terzo sono due interi rappresentanti rispettivamente l'indice di riga e l'indice di colonna dell'elemento.

Ciò viene fatto ricorrendo alla funzione *zip3*, che crea una lista di tuple costruite associando gli elementi di tre liste passate come parametri.

Le liste degli indici di riga e colonna vengono generate dalle funzioni *rowIndexes* e *columnIndexes*, le quali iterano n volte l'accodamento di una lista.

Ad esempio, per una matrice quadrata di ordine 3 le liste di indici sono generate in questo modo:

$$righe: \quad [1,1,1] + [2,2,2] + [3,3,3] \quad \Rightarrow \quad [1,1,1,2,2,2,3,3,3]$$

$$colonne: \quad [1,2,3] + [1,2,3] + [1,2,3] \quad \Rightarrow \quad [1,2,3,1,2,3,1,2,3]$$

Per entrambe le funzioni, sono scritte due versioni: una ricorsiva e una tail recursive.

La lista così costruita viene restituita tramite la funzione *getMatrixWithIndexes*.

```haskell
rowIndexes :: Int -> Int -> [Int]
rowIndexes n 0 = []
rowIndexes n x = rowIndexes n (x-1) ++ replicate n x

rowIndexesTail :: [Int] -> Int -> Int -> [Int]
rowIndexesTail l n 0 = l
rowIndexesTail l n x = rowIndexesTail (replicate n x ++ l) n (x-1)

columnIndexes :: Int -> Int -> [Int]
columnIndexes n 0 = []
columnIndexes n x = [1..n] ++ columnIndexes n (x-1)

columnIndexesTail :: [Int] -> Int -> Int -> [Int]
columnIndexesTail l n 0 = l
columnIndexesTail l n x = columnIndexesTail (l ++ [1..n]) n (x-1)

getMatrixWithIndexes :: [Float] -> [(Float,Int,Int)]
getMatrixWithIndexes m = zip3 m (rowIndexesTail [] n n) (columnIndexesTail [] n n)
                         where n = round ( sqrt ( fromIntegral ( length m ) ) )
```

Successivamente, si è definita la funzione *filterMatrix*, funzione di ordine superiore, che prende in input la lista di tuple e una funzione che rappresenta un filtro, e restituisce gli elementi della matrice (senza indici) che soddisfano il filtro. Per accedere al primo elemento della tupla si è definita una lambda function.

```haskell
filterMatrix :: ((Float, Int, Int) -> Bool) -> [(Float, Int, Int)] -> [Float]
filterMatrix p mt = [(\(x,y,z) -> x) e | e <- mt, p e]
```

Tornando al caso della matrice triangolare, per prima cosa occorre verificare se la matrice sia effettivamente triangolare. Questo avviene tramite la funzione *isTriangular*, che prende in input la lista di elementi, effettua un bound nella clausola where per trasformare la lista semplice nella lista con gli indici, e poi verifica se la somma degli elementi sopra la diagonale o degli elementi sotto la diagonale è nulla. Questi elementi vengono ricavati dalla funzione *filterMatrix*, passando come criterio una lambda function che data una tupla restituisce true se l'indice di riga è minore dell'indice di colonna nel primo caso e se l'indice di riga è maggiore dell'indice di colonna nel secondo caso.
Se la condizione è riscontrata, si calcola il determinante tramite la funzione *detMT*, che restituisce il prodotto degli elementi della diagonale, ottenuti applicando un nuovo filtro sull'uguaglianza degli indici di riga e di colonna.

```
isTriangular :: [Float] -> Bool
isTriangular m = sum (filterMatrix (\(x,y,z) -> y<z) mt) == 0.0 || sum (filterMatrix (\(x,y,z) -> y>z) mt) == 0.0
                 where mt = getMatrixWithIndexes m

detMT :: [Float] -> Float
detMT m = product ( filterMatrix (\(x,y,z) -> y==z) mt )
          where mt = getMatrixWithIndexes m
```

Nell'ultimo caso, ovvero quando la matrice è quadrata di ordine $n > 3$, si applica il teorema di Laplace.
Essendo gli sviluppi di Laplace degli algoritmi ricorsivi, anche la funzione *detMN* lo è.
In questo caso abbiamo:

- Case base: la matrice è quadrata di ordine 1, perciò il determinante è uguale al singolo elemento della stessa;
- Ipotesi ricorsiva: si suppone di conoscere il determinante della matrice quadrata di ordine inferiore, ottenuta rimuovendo una riga e una colonna dalla matrice del passo ricorsivo precedente;
- Passo ricorsivo: si calcola la sommatoria degli elementi di una riga o di una colonna (in questo caso si è stabilito di default di considerare la riga 1) moltiplicati per i rispettivi complementi algebrici.

La funzione *detMN*, quindi, prevede due guardie: la prima in cui si verifica se la lunghezza della lista è uguale a 1 e si restituisce la testa della lista; la seconda in cui si restituisce la somma degli elementi di una lista costruita filtrando gli elementi della matrice per indice di riga uguale a quello specificato nella condizione, moltiplicati per i rispettivi complementi algebrici.
Nel passo ricorsivo, viene scandita la lista di tuple contenenti gli elementi della matrice e gli indici di riga e colonna. A questi elementi si accede direttamente, e non tramite funzioni, stabilendo il pattern *(x,y,z)* per gli elementi ricavati dalla lista.
Per ricavare la matrice con una specifica riga e colonna rimosse, si è definita la funzione *removeRowAndColumn*, che riceve la matrice di tuple e gli indici di riga e colonna da rimuovere, richiama la funzione *filterMatrix* passando un filtro che impone agli indici di riga e di colonna di essere diversi da quelli specificati.

```haskell
removeRowAndColumn :: [(Float,Int,Int)] -> Int -> Int -> [Float]
removeRowAndColumn mt r c = filterMatrix (\(x,y,z) -> y/=r && z/=c) mt
```

```haskell
detMN :: [Float] -> Float
detMN m | length m == 1 = head m
        | otherwise     = sum [x * (-1)^(y+z) * detMN (removeRowAndColumn mt y z) | (x,y,z) <- mt, y == 1]
                          where mt = getMatrixWithIndexes m
```

Questo programma consente così di calcolare il determinante di una matrice quadrata di ordine n qualsiasi, applicando diversi metodi in relazione alla matrice in input.