UNIVERSIDAD AUTONOMA GABRIEL RENE MORENO

FACULTAD DE INGENIERIA Y CIENCIAS DE LA COMPUTACION Y TELECOMUNICACIONES

INTELIGENCIA ARTIFICIAL

CONTENIDO:

LAB-7. DETERMINANTE DE UNA MATRIZ..

PORCENTAJE TERMINADO: 100%

INTEGRANTES	DT	HG	HI	EVAL
Garcia Taboada Brayan Albaro	1	1	1	100

Fecha de presentación : Jueves, 11 de Abril de 2024

Fecha Presentada : : Jueves, 11 de Abril de 2024

Días de Atraso : 0

DETERMINANTE DE UNA MATRIZ

ACTIVIDAD INDIVIDUAL.

El algoritmo para encontrar el determinando de una matriz de n x n, utiliza la estructura de código de llamada recursiva dentro de un ciclo:

Para implementar, realizar:

1. Definir una representación de clase Matriz, con atributos de: elem[][], cantFil, cantCol, maxFil, maxCol, . . . Implementar constructor copia y otros métodos estándares para manipular una matriz.

2. M1.toString() : Método standard que devuelve en cadena, los elementos de la matriz M1.

3. M1.eliminarFil(k) : Método que elimina la fila k, de la matriz M1. Se deben desplazar las filas posteriores a la fila k,para cubrir las filas eliminadas y decrementar la cantidad de filas.

4. M1.eliminarCol(k): Método que elimina la columna k, de la matriz M1. Se deben desplazar las columnas posteriores a la columna k, para cubrir las columnas eliminadas y decrementar la cantidad de columnas.

5. M1.generarRandom(a, b) : Método que genera n x m elementos aleatorios entre a y b, inclusive en la matriz M1.

6. det(M1) : Función que devuelve la determinando de la matriz cuadrada M1, de elementos enteros.

7. encontrarMenores(M1, L1) : Procedimiento que encuentra en la Lista de Matrices L1, todas las matrices menores derivados de la matriz M1.

8. consulta(L1): Hacer algunas consultas a la Lista de Matrices Menores, encontrados previamente.

```
public class Pruebas {
   public static void main(String[] args) {
  Matriz M1 =new Matriz (10,10,3,3);
  M1.generar(1,9);
  System.out.println(M1);
  System.out.println(det(M1));
   }
```

```
    public static int signo(int i,int j){
    if((i+j)%2==0){
       return +1;
    }
    return -1;
  }
    public static int det(Matriz M1){
       if(M1.cantFil==1) return M1.elem[0][0];
       int sum=0,i=0,j=0;
       while (i<M1.cantFil){
          sum=sum+signo(i,j)*M1.elem[i][j]*det(M1.menor(i,j));
          i++;
       }
       return sum;
    }
}


import java.util.LinkedList;

/**
 *
 * @author braya
 */
public class Matriz {

   public int maxFil;
   public int maxCol;
   public int cantFil;
   public int cantCol;
   public int elem[][];
```

```
public Matriz(int maxFil, int maxCol, int cantFil, int cantCol) {
   this.maxFil = maxFil;
   this.maxCol = maxCol;
   this.cantFil = cantFil;
   this.cantCol = cantCol;
   this.elem = new int[maxFil][maxCol];
}

public Matriz(Matriz M1) {
   this.maxFil = M1.maxFil;
   this.maxCol = M1.maxCol;
   this.cantFil = M1.cantFil;
   this.cantCol = M1.cantCol;
   this.elem = new int[maxFil][maxCol];
   for (int i = 0; i < cantCol; i++) {
      for (int j = 0; j < cantFil; j++) {
         {
            this.elem[i][j] = M1.elem[i][j];
         }
      }
   }

}

public String toString() {
   String S1 = "";
   for (int i = 0; i < cantFil; i++) {
      for (int j = 0; j < cantCol; j++) {
         S1 = S1 + elem[i][j] + "\t";
      }
```

```
        S1 += "\n";
    }
    return S1 + "----------------------";
}

public static void mostrarSubMatriz(Matriz M) {
    for (int i = 0; i < M.cantFil; i++) {
        for (int j = 0; j < M.cantCol; j++) {
            for (int k = i; k < M.cantFil; k++) {
                for (int l = j; l < M.cantCol; l++) {
                    Matriz M2 = M.subMatriz(i, j, k, l);
                    System.out.println(M2);
                }
            }
        }
    }
}

public Matriz subMatriz(int x, int y, int a, int b) {
    Matriz M = new Matriz(a - x + 1, b - y + 1, a - x + 1, b - y + 1);
    for (int i = 0; i < M.maxFil; i++) {
        for (int j = 0; j < M.maxCol; j++) {
            M.elem[i][j]=elem[x+1][y+j];
        }
    }
    return M;
}

public static void encontrarSubMatriz(Matriz M1, LinkedList<Matriz> L1) {
    for (int i = 0; i < M1.cantFil; i++) {
        for (int j = 0; j < M1.cantCol; j++) {
```

```
            for (int k = i; k < M1.cantFil; k++) {
               for (int l = j; l < M1.cantCol; l++) {
                  Matriz M2 = M1.subMatriz(i, j, k, l);
                  L1.add(M2);
               }
            }
         }
      }
      for (Matriz mat : L1) {
         System.out.println(mat);
      }
   }
```

//4. M1.generarElem(a, b) : Método que genera valores enteros aleatorios entre a y b respectivamente. Los valores aleatorios serán los elementos de la matriz M1.

```
   public void generar(int a, int b) {
      for (int i = 0; i < cantFil; i++) {
         for (int j = 0; j < cantCol; j++) {
            this.elem[i][j] = (int) (Math.random() * b) + a;
         }
      }
   }
```

//5. mostrarMatrices(n. m, L1) : Procedimiento que muestra las matrices de dimensiones n x m, de la Lista de Matrices L1.

```
   public void mostrarMatrices(int n, int m, LinkedList<Matriz> L1) {
      for (Matriz M : L1) {
         if (M.cantFil == n && M.cantCol == m) {
            System.out.println(M);
         }
      }
   }
```

//6. mostrarCuadrados(L1) : Procedimiento que muestra las matrices cuadradas de la Lista de Matrices L1.

```
public void mostrarCuadrados(LinkedList<Matriz> L1) {
    for (Matriz M : L1) {
        if (M.cantFil == M.cantCol) {
            System.out.println(M);
        }
    }
}
```

//7. mostrarFila(L1) : Procedimiento que muestra las matrices fila de la Lista de Matrices L1.

```
public void mostrarFila(LinkedList<Matriz> L1) {
    for (Matriz M : L1) {
        if (M.cantFil == 1) {
            System.out.println(M);
        }
    }
}

public Matriz menor(int i, int j) {
    Matriz M2 = new Matriz(this);
    M2.eliminarFil(i);
    M2.eliminarCol(j);
    return M2;
}

public void eliminarFil(int k) {
    for (int i = k + 1; i < this.cantFil; i++) {
        for (int j = 0; j < this.cantCol; j++) {
            elem[i - 1][j] = elem[i][j];
```

```
            }
        }
        this.cantFil--;
    }

    public void eliminarCol(int k) {
        for (int i=k+1;i<this.cantCol;i++){
            for(int j=0;j<this.cantFil;j++){
                elem[j][i - 1] = elem[j][i];
            }
        }
        this.cantCol--;
    }

}
```