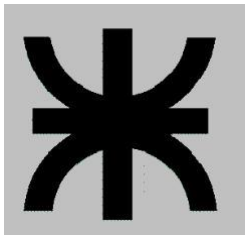

# UNIVERSIDAD TECNOLOGICA NACIONAL
# FACULTAD REGIONAL GENERAL PACHECO

## TÉCNICO SUPERIOR EN PROGRAMACIÓN

### LABORATORIO V

### TP 3

**¿Qué es una matriz de adyacencia?**

La matriz de adyacencia es una matriz cuadrada, que sea cuadrada significa que tiene el mismo número de filas y columnas, ejemplo: matrices de 2x2 (dos filas, dos columnas), 3x3 (tres filas, tres columnas), etc.

Es una matriz binaria, es decir que solo contiene números
1 y 0 en su interior.

Es una matriz simétrica porque cumple con la siguiente funcionalidad:

- Si agregamos un elemento en la posición [fila 2, columna 3], se agrega también en la posición [fila 3, columna 2]. Si agregamos un elemento, se agrega el valor 1 en las posiciones mencionadas.

- Si eliminamos un elemento de la posición [fila 2, columna 3], se elimina también en la posición contraria [fila 3, columna 2]. Para eliminar un elemento se coloca el valor 0, en las posiciones mencionadas.

- Si agregamos un elemento el [2,3], también se agrega el [3,2] y esto es una relación.

**Ejercicio**

1. Importar dentro Eclipse, el proyecto subido al aula virtual llamado “Matriz de adyacencia sin testeo”.

2. Cambiarle el nombre al proyecto, para esto hacer segundo click sobre el proyecto -> Refactor -> Rename -> Colocar el nombre -> Tarea2_GrupoX

3. Realizar los siguientes test en la clase **MatrizAdyacenciaTest** (respetar los nombres de los métodos):

    a. Crear un método llamado **`agregarElementoTest`**, que verifique que luego de agregar un elemento este elemento exista dentro de la matriz

    b. Crear un método llamado **`agregarElementoSimetriaTest`**, que verifique que luego de agregar un elemento, ese exista en su posición opuesta/simétrica. Ejemplo, si agrego un elemento en la posición [2,3], verificar que se haya agregado el elemento [3,2]

c. Crear un método llamado **`eliminarElementoTest`**, que verifique que luego de eliminar un elemento este elemento no exista dentro de la matriz

d. Crear un método llamado **`eliminarElementoSimetricoTest`**, que verifique que luego de eliminar un elemento también elimine su simétrico. Ejemplo, si elimino el elemento de la posición [2,3], verificar que se haya eliminado el elemento [3,2]

e. Crear un método llamado **`contarRelacionesTest`** que verifique que el método getCantidadRelaciones de la clase MatrizAdyacencia. Ejemplo: Si agregamos tres elementos [2,3] [1,4] y [1,2] … hay un total de tres relaciones.

f. Crear un método llamado **`existenTodosLosElementoTest`** Verificar que si se completan todos las posiciones de la matriz, todos estos elementos se hayan guardado correctamente en su posición original y en su simetrico.

g. Crear un método llamado **`agregarElementoFilaNegativaTest`** que verifique que, si uno quiere agregar un elemento en una fila negativa, éste arroje una excepción.

h. Crear un método llamado **`agregarElementoColumnaNegativaTest`** que verifique que, si uno quiere agregar un elemento en una columna negativa, éste arroje una excepción.

i. Crear un método llamado **`agregarElementoFueraRangoTest`** que verifique que, si uno quiere agregar un elemento en una columna fuera del rango, éste arroje una excepción. Ejemplo: si tenemos una matriz de 2x2,(dos filas, dos columnas) probar que si uno quiere agregar en la columna 3 o fila 3, se arroje una excepción.

4. Crear dos TestSuite:

   El primer TestSuite se debe llamar **AllPackageTest** que debe correr todos los test que se encuentran en el paquete llamado test

   El segundo TestSuite se debe llamar **AllClassTest** que debe correr todos los test de la clase MatrizAdyacenciaTest