

1.- (Uso de una estructura de datos de tipo Grafo)

Daré el resultado null si el  
grafo no es dirigido.

b)

```
public static GrafoMA grafoTransverso(GrafoMA grafo){
```

```
GrafoMA res = null;
```

```
if(grafo.getDirigido()) {
```

```
res = new GrafoMA(grafo.getNumVertices(), true);
```

```
for(
```

```
) {
```

```
for(
```

```
) {
```

```
if((grafo.existeArista(i, j))) res = res.insertarArista(j, i);
```

```
}  
}
```

```
return res;
```

```
}
```

2.- (Nuevos métodos en grafo MA)

a) `public int verticeDesconectado()`

```
int res = -1;  
boolean conectado = false;
```

```
for(int i=0; i<numVertices && res == -1; i++)  
    for(int j=0; j<numVertices && conectado == false; j++)  
        if(i != j && mat[i][j] || mat[j][i])
```

```
            conectado = true;
```

}

```
if(!conectado == false) res = i;
```

}

```
return res;
```

}

-1 si no hay vértice desconectado.

que este conectado consigo mismo  
no significa que este conectado  
con el grafo.

	a	b	c	d	e
a	T				
b			x	T	T
c			T		x
d		T	T	x	T
e		T	T		x

(a)

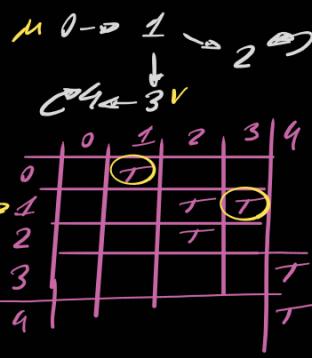


Debe tener filas y columnas

sin "T" y sin contar la diagonal  
que es estar conectado consigo mismo  
bucle

Dentro de grafo MA

```
public boolean existeCaminoCorto(int u, int v){  
    boolean res = false;  
    for(int i=0; i < numVertices && !res; i++) {  
        if(matrizAdy[u][v]) {  
            res = true;  
        } else if(matrizAdy[u][i] && matrizAdy[i][v]) {  
            res = true;  
        }  
    }  
    return res;  
}
```



Camino de  
longitud 1  
Camino de  
longitud 2

Dentro de grafo Dinámico

```

public void desconectarVertice(int v) {
    if(v >= 0 && v < numVertices) {
        for(int i = 0; i < numVertices; i++) {
            Iterator<Integer> ite = adyacencias[i].iterator();
            while(ite.hasNext()) {
                int temp = ite.next();
                if(temp == v && i != v) {
                    adyacencias[i].remove(v);
                }
            }
        }
        for(int j = 0; j < numVertices; j++) {
            if(j != v) {
                adyacencias[v].remove(j);
            }
        }
    }
}

```

Desconectar de  $v$  las aristas entrantes y salientes a otros vértices.

TreeSet

0	1
1	2, 3
2	2
3	4
4	2, 4

contains?  $\Rightarrow$   $\text{adyacencias}[x].contains$

iterator? cómo recorre conjuntos

$\text{adyacencias}[v].iterator$

Ejercicio 3c



Cambia la dirección de todos los aristas dirigido

```
public void trasponer(){  
    if(dirigido){  
        for(int i=0; i<numVertices; i++){  
            Iterator<Integer> ite = adyacentes[i].iterator();  
            while(ite.hasNext()){  
                int temp= ite.next();  
                adyacentes[i].remove(temp);  
            }  
        }  
    }  
}
```

→

0	1	0   2,3
1		1   0,3,4
2	0,3	2
3	0,1	3
4	2,3	4

⇒

i	temp	0   2,3
0	X	1   0,3,4
1		2
2	X	3   2,4
2	X	4
3	X	
3	X	
4	X	
4	X	

Ejemplo de Grafo visto en una estructura de datos tipo Grafo

```

boolean[] visitados = lab.getVisitados(5);    // Laberinto
if(visitados[14]) {
    System.out.println("Se ha visitado");
}
  
```

X	0	1	2	3	4
5	6	X	7	X	8
X	9	X	10	X	X
11	12	13	X	X	14
15	X	16	X	X	17
18	X	19	20	21	22

23 nodos

arco entre vértices con los que están cerca

En memoria estática  
esto es muy ineficiente  
matrices muy grandes.

b) public static boolean alcanzaResto(GrafoMA g, int v)

```

if(g.verticeEnGrafo(v)) {
    boolean[] visitados = g.getVisitados(v);
    res = true;
    for(int i=0; i < visitados.length && res; i++) {
        for(int j=0; j < visitados.length; j++) {
            if(visitados[i][j] == false) {
                res = false;
            }
        }
    }
}
  
```

Recorre todos los vértices

Este retorna una matriz de por cada fila pasada

$$\begin{bmatrix} T & F & T \\ T & T & T \\ \text{no psga} & \text{por el vrtice} & \text{ese.} \end{bmatrix}$$

public static int tamañoComponenteConexa(GrafoMA g, int v)

El resultado es ① yo solo

```

res = 0;
if(!g.esDirigido() && g.verticeEnGrafo(v)) {
    boolean[] visitados = g.getVisitados(v);
    for(int i = 0; i < visitados.length; i++) {
        for(int j = 0; j < visitados.length; j++) {
            if(visitados[i][j] == true) {
                res++;
            }
        }
    }
}
  
```

	0	1	2	3	4
0	T				
1		T	T		
2			T		
3				T	
4					T

### Ejercicios de software 11

Buscar caminos de longitud 1 ó 2 entre  $u$  y  $v$

Caminos de longitud 1 si están en la matriz

ya está matriz adyacente

if (mat[u][v]) res = true;

Caminos de longitud 2, búsqueda de vecinos

```
for (int i; i < nVertices && res == false; i++) {
```

```
    if (mat[u][i]) {
```

```
        res = mat[i][v];
```

```
}
```

Revisar para grafos como funciona iterador { -hasNext  
-next }

Hacer uso de iteradores (mirar página 38 teoría)

### Ejercicio 3 Grafo Dinámico

b)

Devuelve conjunto con los vértices que cumplen que sus aristas se dirijan a vértices pares

```
public TreeSet<Integer> verticesConAdyacentesPares() {
    TreeSet<Integer> resultado = new TreeSet<>();
    for(int i=0; i<numVertices; i++) {
        if(adyacencias[i].size() > 0) {
            Iterator<Integer> ite = adyacencias[i].iterador();
            while(ite.hasNext() && !esPar) {
                int temp = ite.next();
                if(temp % 2 != 0) {
                    esPar = false;
                } else if(esPar) resultado.add(i);
            }
        }
    }
    return resultado;
}
```