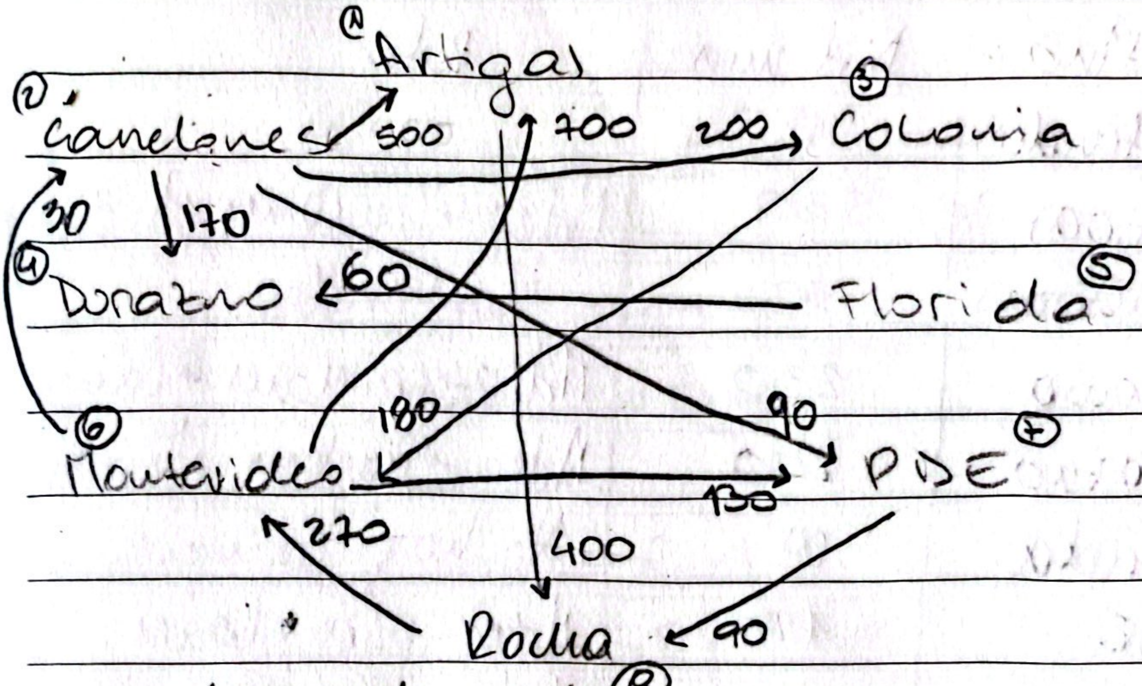
Mo Tu We Th Fr Sa Su

UT7 TA1 -EJ1

Memo No. ________
Date / /



matriz de adyacencia

	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	400
2	500	0	200	170	0	0	90	0
3	0	0	0	0	0	180	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	60	0	0	0	0
6	700	30	0	0	0	0	130	0
7	0	0	0	0	0	0	0	90
8	0	0	0	0	0	270	0	0

Mo | Tu | We | Th | Fr | Sa | Su

Memo No.

Date / /

lista de adyacencia

1) 8(400) ⏚

2) 1(500) → 3(200) → 4(170) → 7(90) ⏚

3) 6(180)

4) ⏚

5) 4(60)

6) 1(700) → 2(30) → 7(130)....

7) 8(90)

8) 6(270)

Mo | Tu | We | Th | Fr | Sa | Su

EJ2

Memo No. ______

Date / /

Destino	Dist. min.	Ruta
Montevideo	0	-
Artigas	530	Mdeo → Canelones → Artigas
Canelones	30	Montevideo
Colonia	230	Mdeo → Canelones → Colonia
Durazno	200	Mdeo → Canelones → Durazno
Florida	∞	No hay camino
PDE	120	Mdeo → Canelones
Rocha	1210	Mdeo → Canelones → PDE → Rocha

EJ3

operaciones que debería tener un grafo dirigido:

→ Agregar Vertice (V v)
→ Eliminar Vertice (V v)
→ AgregarArista (V origen, V destino, E etiqueta)
→ ExisteArista (V origen, V destino)
→ Obtener adyacentes (V v)

Se representa a traves de una lista de adyacencia, se puede usar Map o Set de la API de Java

Operación	Orden	
AgregarVertice	O(1)	
EliminarVertice	O(V)	hay que eliminar la relación entre los otros vertice
AgregarArista	O(1)	
ExisteArista	O(1)	
ObtenerAdyacentes	O(1)	