Estructura de Datos y Algoritmos 1

Tema 6 - Grafos

Nodos + Relaciones:
- 1:1 (Secuencias)
- 1:N (Antecesores)
- N:N (Red)

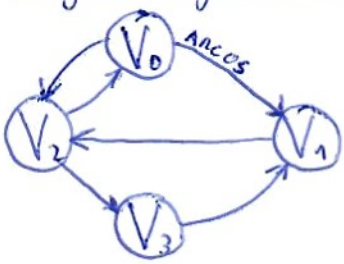
$G = (V, E)$ (VERTEX, EDGE)

$E : V \rightarrow V$

$(V_1, V_2) \rightarrow V_1 \Rightarrow V_2$

$G(V, E, W)$ $\quad W(V_1, V_2) \rightarrow V_1 \xrightarrow{w} V_2$ $\quad W \in \mathbb{R}^+$ $\quad W = [w_0, w_1, w_2, \ldots, w_{n-1}]$

Grafo dirigido (Digraf) $(V_1, V_2) \neq (V_2, V_1)$ ↳ VALORADO / NO VALORADO



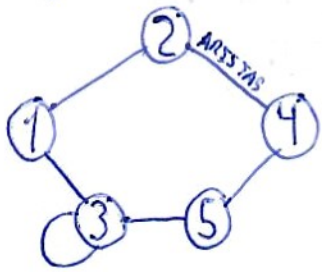
$V = \{V_0, V_1, V_2, V_3\}$

$E = \{(V_0, V_1), (V_0, V_2), (V_1, V_2), (V_2, V_0), (V_2, V_3), (V_3, V_1)\}$

Camino $P = \langle V_1, V_2, \ldots, V_n \rangle \rightarrow \boxed{(V_i, V_{i+1}) \in E}$

$G = G\dot{E}(v) + GS(v)$ (GRADO ENTRADA + GRADO SALIDA)

Grafo bidimensional (Grafo NO Dirigido) $(V_1, V_2) = (V_2, V_1)$ ↳ VALORADO / NO VALORADO



$V = \{1, 2, 3, 4, 5\}$

$E = \{(1,2), (1,3), (1,4), (2,1), (2,4), (3,1), (3,5), (4,1), (4,2), (4,5), (5,3), (5,4)\}$

Grado (v) → Nº de aristas que intervienen en él.

Grado (1) = 3

$(V_1 \longrightarrow V_2$ Adyacente$)$

Representación

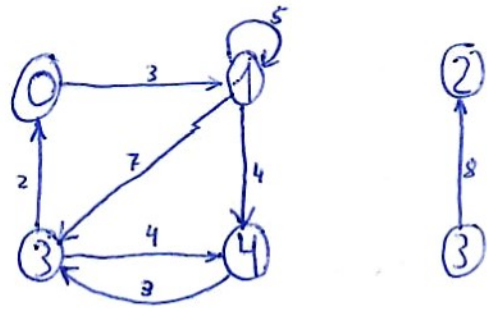
- Matriz de adyacencia
- Lista de adyacencia

$M_{n \times n} / M(i,j) = M[i,j] = M_{i,j}$ { TRUE SS $\exists i \to j$; FALSE OTHERWISE

$G(V, E, W)$

$$M_{i,j} = \begin{cases} W_{ij} & \text{si } i \to j \\ 0 & \text{otherwise} \end{cases}$$

G)



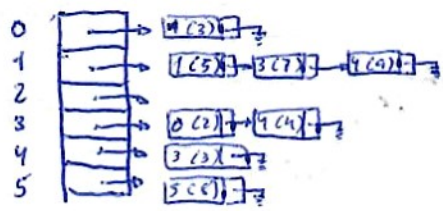
· Creamos la matriz de G

$M_{6,6}$

	0	1	2	3	4	5
0	0	3	0	0	0	0
1	0	5	0	7	9	0
2	0	0	0	0	0	0
3	2	0	0	0	4	0
4	0	0	0	3	0	0
5	0	0	8	0	0	0

Eficiencia especial

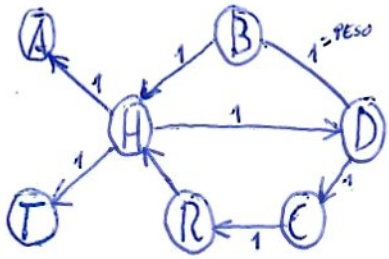
Lista de adyacencia →



{Vertices} 1 vértice → 1 conjunto de vértices adyacentes + el peso de la relación

TreeMap<Vertex, TreeMap<Vertex, Double>> o HashMap<Vertex, HashMap<Vertex, Double>>

Ejercicio:



TreeMap<Vertex, Integer>

A	0
B	1
C	2
D	3
H	4
R	5
T	6

	0	1	2	3	4	5	6
A = 0	0	0	0	0	0	0	0
B = 1	0	0	0	0	1	0	0
C = 2	0	0	0	0	0	1	0
D = 3	0	1	1	0	0	0	0
H = 4	1	0	0	1	0	0	1
R = 5	0	0	0	0	1	0	0
T = 6	0	0	0	0	0	0	0

Grado (G) = GE(H) + GD(H)

= 2 + 3 = 5

Recorrido en profundidad

Stack

Visitados TreeMap < Vertex, Boolean >

A	F
B	F
C	F
D	F
H	F
R	F
T	F

STACK

D → PROCESO Y PONGO F
BC
BR
BH
BAT
BA
B
Ø

LIFO

SALIDA

D C R H T A B

RECORRIDO EN PROFUNDIDAD

Cambiamos pila (Stack) por cola (Queue)

QUEUE

D
BC
CH
HR
RAT
AT
T
Ø

FIFO

SALIDA

D B C H R A T

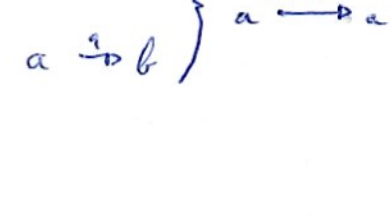
```
TreeMap < Vertex, TreeMap < Vertex, Double >> adyacenList;

public TreeMap < Vertex, TreeMap < Vertex, Double >> transpose () {
  TreeMap < Vertex, TreeMap < Vertex, Double >> result = new TreeMap <> ();
  for (Entry < Vertex, TreeMap < Vertex, Double >> parMain : adyacencyList.entrySet()){
    for (Entry < Vertex, Double > parSecond : parMain.getValue().entrySet()) {
      TreeMap < Vertex, Double > value = result.get (parSecond.getKey());
      if (value == null) result.get ( parSecond, value = new TreeMap <> ());
      value.put ( parMain.getKey(), parSecond.getValue());
    }
  }
  return result;
}
```

Ejercicio de examen:

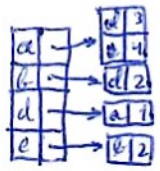
```
public void combine (TreeMap<Vertex, TreeMap<Vertex, Double>> otro) { }
```

this.combine(otro)

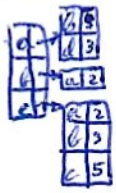
this = this + otro



this



otro



```
public void combine (TreeMap<Vertex, TreeMap<Vertex, Double>> otro) {
  for (Entry<Vertex, TreeMap<Vertex, Double>> par : otro.entrySet()) {
    TreeMap<Vertex, Double> aux = this.get(par.getKey());
    if (aux == null) this.put(par.getKey(), aux = new TreeMap<>(par.getValue());
    for (Entry<Vertex, Double> par2 : otro.getValue().entrySet()) {
      Double aux2 = aux.get(par2.getKey());
      aux.put(par2.getKey(), aux2 == null ? (par2.getValue + aux)/2);
    }
  }
}
```

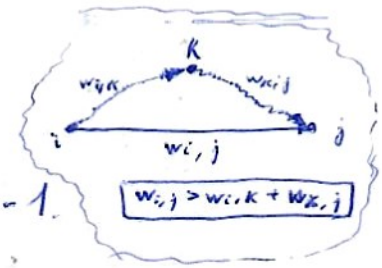
Caminos mínimos

- Dijkstra: $V_{START} \rightarrow$ Resto
- Floid: Caminos mínimos entre todos los pares de vértices.

$G = (V, E, W)$

V = Vértices $\quad E \cdot V \rightarrow V \quad (v_i, v_j) \quad v_i \xrightarrow{\text{ADYACENTE}} v_j$

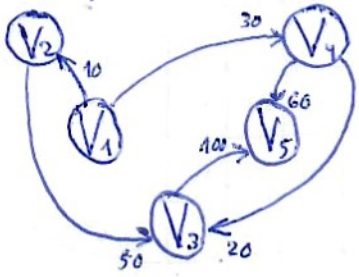
PESO COSTE $w \in W \quad w > 0 \quad w(w_i, w_j) = w_{i,j}$



$P = \langle v_1, v_2, \ldots, v_K \rangle$ longitud $(P) = n$ vértices $- 1$.

$\text{coste}(P) = \sum_{i,K}^{1K} \text{coste}(v_i, v_{i+1}) = \sum w_{i,i+1}$

Ejemplo 1:



V_1	0
V_2	1
V_3	2
V_4	3
V_5	4

	0	1	2	3	4
0	∞	10	∞	30	100
1	∞	∞	50	∞	∞
2	∞	∞	∞	∞	10
3	∞	∞	20	∞	60
4	∞	∞	∞	∞	∞

$V_{START} = V_1$

Desde V_1 a	V_2		V_3		V_4		V_5	
ITERACIÓN	VER	COSTE	DESDE	COSTE	DESDE	COSTE	DESDE	COSTE	DESDE
Ø	–	10	V_1	∞	V_1	30	V_1	100	V_1
1	V_2			60 (50+10)	V_2	30	V_1	100	V_1
2	V_4			50 (30+20)	V_4			90 (60+30)	V_3
3	V_3								
4		10	V_1	50	V_4	30	V_1	60	V_5

V_1	0	V_1
V_2	10	V_1
V_3	50	V_4
V_4	30	V_1
V_5	60	V_3

$V_1 \rightarrow V_2$

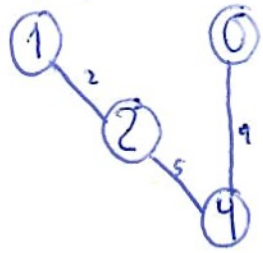
$V_1 \rightarrow V_4 \rightarrow V_3$

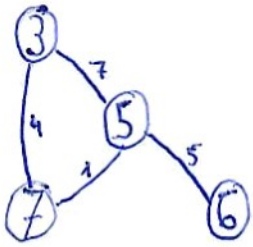
$V_4 \rightarrow V_1$

$V_1 \rightarrow V_4 \rightarrow V_4 \rightarrow V_5$

TreeMap<Vertex, Pair<Vertex, Double>> Arraylist de <Vertex>

Ejemplo 2:





$C =$

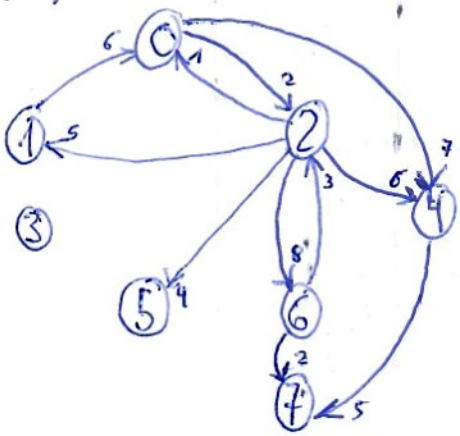
	0	1	2	3	4	5	6	7
0	∞	∞	∞	∞	9	∞	∞	∞
1	∞	∞	2	∞	∞	∞	∞	∞
2	∞	2	∞	∞	5	∞	∞	∞
3	∞	∞	∞	∞	∞	7	∞	4
4	9	∞	5	∞	∞	∞	∞	∞
5	∞	∞	∞	7	∞	∞	5	∞
6	∞	∞	∞	∞	∞	5	∞	∞
7	∞	∞	∞	4	∞	1	∞	∞

Desde V1 a		1		2		3		4		5		6		7	
ITERACIÓN	VCM	COSTE	DESDE	COSTE	DESDE	COSTE	DESDE	COSTE	DESDE	COSTE	DESDE	COSTE	DESDE	COSTE	DESDE
∅	-	∞	V_0	∞	V_0	∞	V_0	9	V_0	~~∞~~	V_0	∞	V_0	∞	V_0
1	V_4	∞	V_0	14	V_0	∞	V_0			∞	V_0	∞	V_0	∞	V_0
2	V_2	~~16~~	V_2			∞	V_0			∞	V_0	∞	V_0	∞	V_0
3	V_1					∞	V_0			∞	V_0	∞	V_0	∞	V_0
4		16	V_2	14	V_0	∞	V_0	9	V_0	∞	V_0	∞	V_0	∞	V_0

V_0	0	V_0
V_1	16	V_2
V_2	14	V_4
V_3	∞	V_0
V_4	9	V_0
V_5	∞	V_0
V_6	∞	V_0
V_7	∞	V_0

DESDE V_0 A ...

Ejemplo 3:



V_0	V_0	6
V_1	V_0	0
V_2	V_1	8
V_3	V_0	∞
V_4	V_2	13
V_5	V_2	12
V_6	V_4	16
V_7	V_4	18

$V_0 \rightarrow V_1$

—

$V_1 \rightarrow V_0 \rightarrow V_2$

NO PDH

$V_1 \rightarrow V_0 \rightarrow V_4$

$V_1 \rightarrow V_0 \rightarrow V_2 \rightarrow V_5$

$V_1 \rightarrow V_0 \rightarrow V_0$

$V_1 \rightarrow V_8 \Rightarrow V_4 \Rightarrow V_7$

$V_{START} = V_1$