Gráficas

O Grafos

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

5 de agosto de 2021

Definición

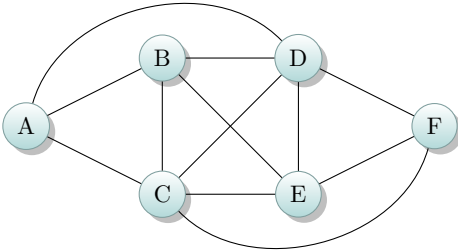
Definición

Definición

Una *gráfica* es una tupla $G = (V, E)$ con:

V un conjunto de vértices.

E un conjunto de aristas que conectan pares de vértices en V.

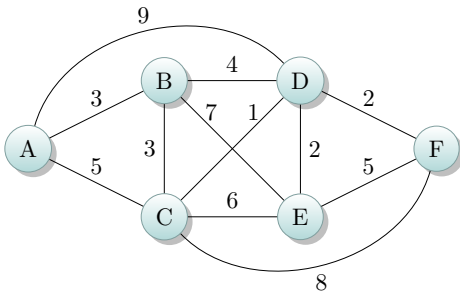


Definición

Definición

Una gráfica *pesada* también tiene:

- Una función $\omega : E \to \mathcal{R}$



Definición

Definición

- En una gráfica *dirigida* $e = v_i \to v_j$, v_i es vecino de v_j.
- En una gráfica *no dirigida* $e = v_i - v_j$, v_i es vecino de v_j y v_j es vecino de v_i.

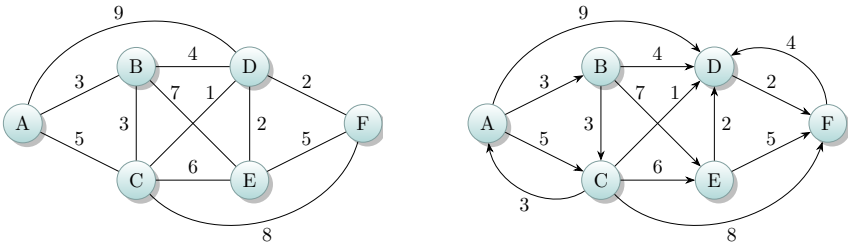


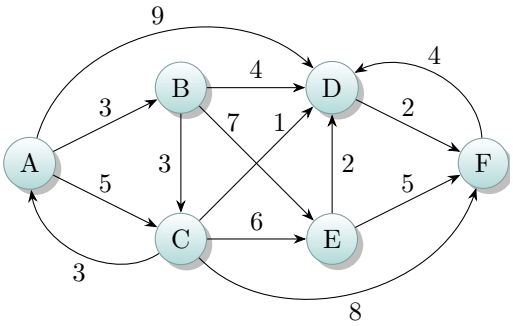
Figura: **Izquierda:** Gráfica no dirigida. **Derecha:** Gráfica dirigida.

Representación en la computadora

Representaciones

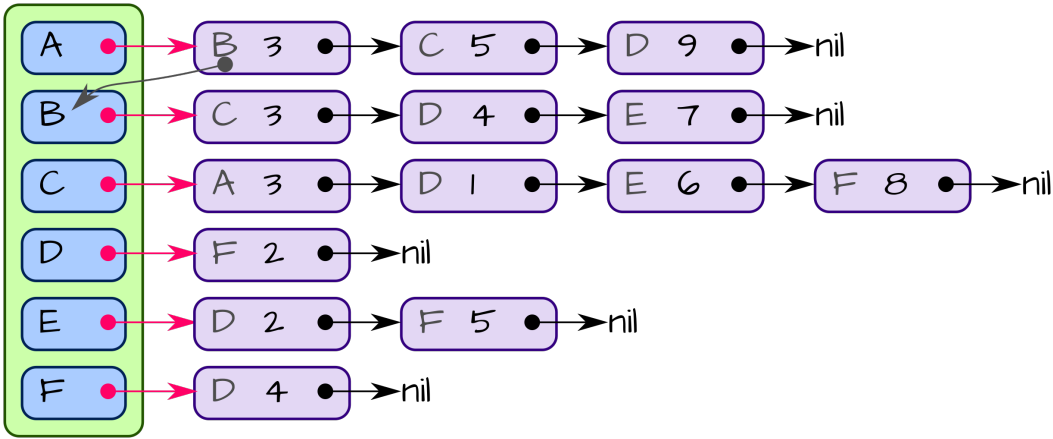
- Hay varias formas de implementar gráficas en la computadora:
  - Matriz de adyacencia.
  - Listas de adyacencia.
- Se pueden utilizar estructuras auxiliares como `ArrayList` o `HashMap` para almacenar a todos los nodos de una gráfica con listas de adyacencia y accederlos más rápido cuando sea necesario.
- Si los vértices tienen identificadores únicos, las listas de adyacencia se pueden suplir por diccionarios.

Matriz de adyacencia



	A	B	C	D	E	F
A	0	3	5	9	∞	∞
B	∞	0	3	4	7	∞
C	3	∞	0	1	6	8
D	∞	∞	∞	0	∞	2
E	∞	∞	∞	2	0	5
F	∞	∞	∞	4	∞	0

Listas de adyacencia



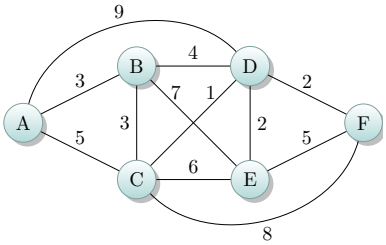
Recorridos

Temas

Primero en profundidad

DFS : Primero en profundidad.

- Utiliza una pila o se programa recursivamente.
- Usa un arreglo de `boolean` por vértice para indicar si ya fue **visitado**.



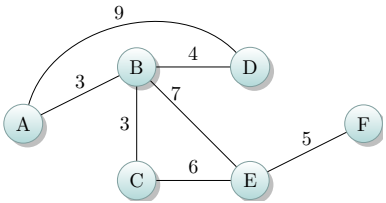
Visitado

A	B	C	D	E	F

Pila

0	1	2	3	4	5

Ejemplo 2



Visitado					
A	B	C	D	E	F

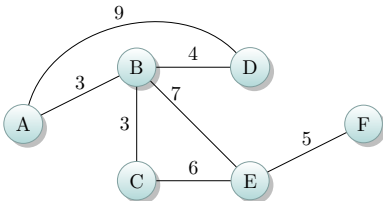
Pila					
0	1	2	3	4	5

Temas

Primero en amplitud

BFS : Primero en amplitud.

- Utiliza una cola.
- Un arreglo de `boolean` por vértice para indicar si ya está **formado**.



Formado					
A	B	C	D	E	F

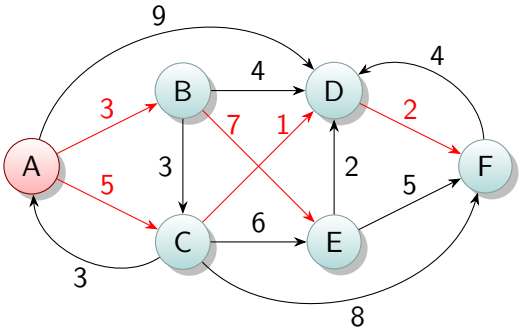
Cola					
0	1	2	3	4	5

Caminos de peso mínimo

Temas

Dijkstra

- **Objetivo:** Encontrar el camino de peso mínimo entre un nodo y sus vecinos.



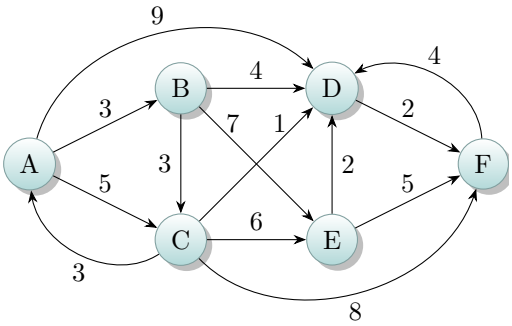
- Para cada nodo mantener:
  - $k(v)$ Terminado, ya se conoce la distancia mínima.
  - $d(v)$ Distancia.
  - $P(v)$ Predecesor.
- Q cola de prioridades **mínima** según $d(v)$, con $k(v) = False$

Algoritmo 1 Dijkstra

1: $k(v) \leftarrow False$, $d(v) \leftarrow \infty$, $P(v) \leftarrow \emptyset$
2: $Q \leftarrow V$
3: **for** $|V|$ veces **do**
4: Elegir de Q al vértice v con menor $d(v)$.
5: $k(v) \leftarrow True$
6: **for all** vecino u de v con $k(u) = False$ **do**
7: **if** $d(u) > d(v) + \omega(v, u)$ **then**
8: $d(u) \leftarrow d(v) + \omega(v, u)$
9: $P(u) \leftarrow v$

Dijkstra (Ejercicio)

- **Objetivo:** Encontrar el camino de peso mínimo entre un nodo y sus vecinos.



	k(v)	d(v)	P(v)	Prioridades
A				
B				
C				
D				
E				
F				

Temas

4. Caminos de peso mínimo
   - Dijkstra
   - Floyd-Warshall

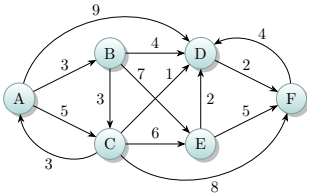
Floyd-Warshall

- **Objetivo:** Dada $G = (V, E)$, para cada par de vértices en V encontrar el camino pesado más corto entre ellos.
- **Principio:**
  1. Encontrar los caminos más cortos entre vértices del subgrafo $V_k = \{v_1, v_2, ..., v_k\}$ para $0 \leqslant k \leqslant |V|$.
  2. Agregar v_{k+1} y verificar si cada ruta se puede acortar pasando por v_{k+1}.
- **Complejidad:** $\mathcal{O}(n^3)$ con $n = |V|$
- **Utiliza:** para subgrafos de tamaño k:
  - Matrices D^k para almacenar el peso de la mejor ruta del nodo v_i al nodo v_k conocida.
  - Matrices Π^k para almacenar los predecesores de cada nodo.

Algoritmo 2 Floyd-Warshall

1: $D^k \leftarrow matriz(n \times n)$, $\Pi^k \leftarrow matriz(n \times n)$ ▷ Costos, Padres
2: $D^0 \leftarrow W$ ▷ Matriz de pesos (adyacencia)
3: **for** $i \in [1, n]$ **do**
4: **for** $j \in [1, n]$ **do**
5: $$\Pi^0_{i,j} \begin{cases} \text{NIL} & \text{si } i = j \text{ ó } \omega(i,j) = \infty \\ i & \text{si } i \neq j \text{ y } \omega(i,j) < \infty \end{cases}$$
6: **for** $k \in [1, n]$ **do** ▷ Agrega vértices
7: **for** $i \in [1, n]$ **do**
8: **for** $j \in [1, n]$ **do**
9: $d_{ikj} \leftarrow d^{k-1}_{i,k} + d^{k-1}_{k,j}$
10: $D^k_{i,j} \leftarrow \text{mín}(d^{k-1}_{i,j}, d_{ikj})$
11: $$\Pi^k_{i,j} \leftarrow \begin{cases} \pi^{k-1}_{ij} & \text{si } d^{k-1}_{ij} \leqslant d_{ikj} \quad \triangleright \text{No pasa por } v_k \\ \pi^{k-1}_{kj} & \text{si } d^{k-1}_{ij} > d_{ikj} \quad \triangleright \text{Pasa por } v_k \end{cases}$$

Floyd-Warshall (Ejercicio)



$D_0 =$

	A	B	C	D	E	F
A	0	3	5	9	∞	∞
B	∞	0	3	4	7	∞
C	3	∞	0	1	6	8
D	∞	∞	∞	0	∞	2
E	∞	∞	∞	2	0	5
F	∞	∞	∞	4	∞	0

$D_1 =$

	A	B	C	D	E	F
A						
B						
C						
D						
E						
F						

$\Pi_0 =$

	A	B	C	D	E	F
A	0	A	A	A	NIL	NIL
B	NIL	0	B	B	B	NIL
C	C	NIL	0	C	C	C
D	NIL	NIL	NIL	0	NIL	D
E	NIL	NIL	NIL	E	0	E
F	NIL	NIL	NIL	F	NIL	0

$\Pi_1 =$

	A	B	C	D	E	F
A						
B						
C						
D						
E						
F						

Árbol generador de peso mínimo

Árbol de expansión mínima

Definición

Un *árbol de expansión mínima* es un grafo acíclico, cuyas aristas $T \subseteq E$ conectan a todos los vértices V y cuyo peso total

$$w(T) = \sum_{(u,v)\in T} w(u,v) \tag{1}$$

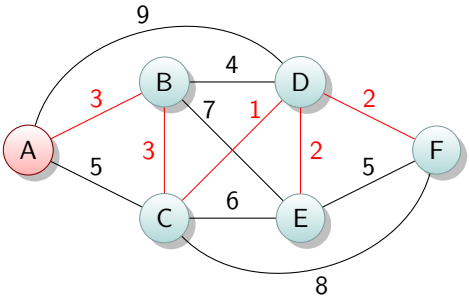
es mínimo.

- **Objetivo:** Encontrar un árbol de expansión mínima en el grafo conexo, **no dirigido** $G = (V, E)$.
- **Ejemplo de aplicación:** Se desea diseñar una placa electrónica, con la menor cantidad de silicio, sin que haga corto circuito.

Temas

Prim

- Sean:
  - $\mathcal{A}$ el árbol de peso mínimo en construcción.
  - $\mathcal{Q}$ los vértices que no están aún en el árbol.
- $\mathcal{A}$ y $\mathcal{Q}$ definen una partición de la gráfica.
- En cada paso se agrega una arista de la frontera.

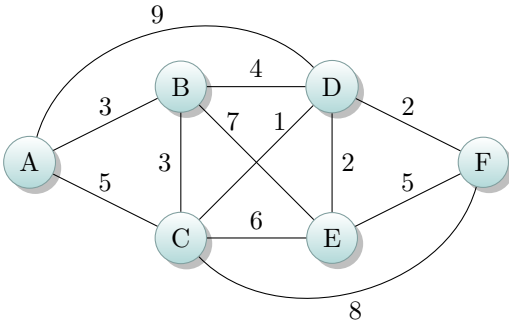


- Para cada nodo mantener:
  - $llave(v)$ Peso de la arista de menor peso que conecta a v con el árbol.
  - $\pi(v)$ Padre en el árbol de peso mínimo.
- Q cola de prioridades **mínima** según $llave(v)$, con vértices.

Algoritmo 3 Prim

```
1: llave(v) ← ∞, π(v) ← ∅, Q ← V, A ← ∅
2: llave(r) ← 0                                   ▷ Un nodo cualquiera r
3: while Q ≠ ∅ do
4:     Elegir de Q al vértice v con menor llave(v).
5:     agrega(A, v)
6:     for all vecino μ de v con μ ∈ Q do          ▷ No forman ciclos
7:         if ω(v, μ) < llave(μ) then
8:             llave(μ) ← ω(v, μ)
9:             π(μ) ← v
```

Prim (Ejercicio)



	$\text{llave}(v)$	$\pi(v)$	Prioridades
A			
B			
C			
D			
E			
D			

Temas

5. Árbol generador de peso mínimo
   - Prim
   - Kruskal

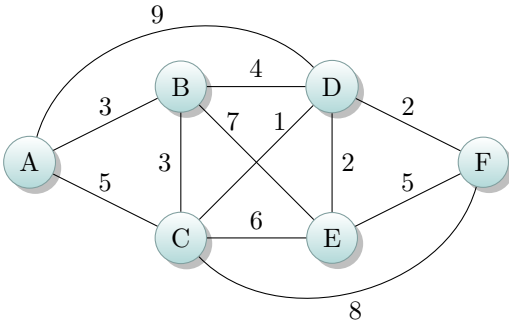
Kruskal

- Q cola de prioridades **mínima** según $\omega(e), e \in E$
- A bosque (conjunto de subárboles disjuntos).
  - Cada árbol tiene un vértice representativo.
  - Dado un vértice se encuentra el representante en $\mathcal{O}(1)$.

Algoritmo 4 Kruskal

```
1: A ← ∅
2: for all v ∈ V do
3:     ConstruyeConjunto(v)
4: Q ← E
5: while Q ≠ ∅ do
6:     Elegir de Q al arista e(μ, v) con menor ω(e).
7:     if Conjunto(μ) ≠ Conjunto(v) then
8:         A ← A ∪ (μ, v)
9:         Une(Conjunto(μ), Conjunto(v))
```

Kruskal (Ejercicio)



Peso	Arista

Bibliografía I

 Cormen, Thomas H. y col. (2009). *Introduction to Algorithms*. 3rd. The MIT Press.

Licencia