

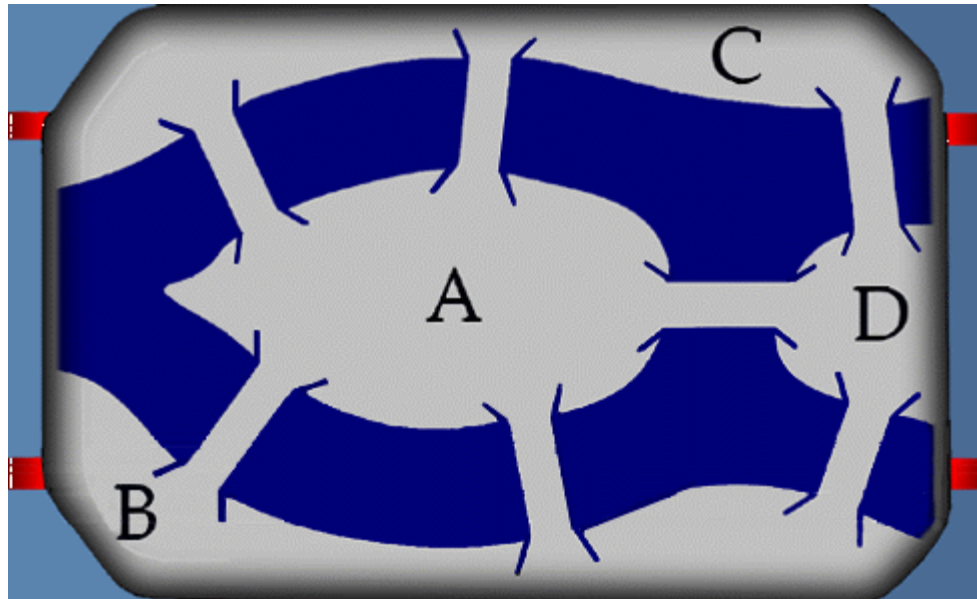
BISOFT-28

Estructuras de Datos 2

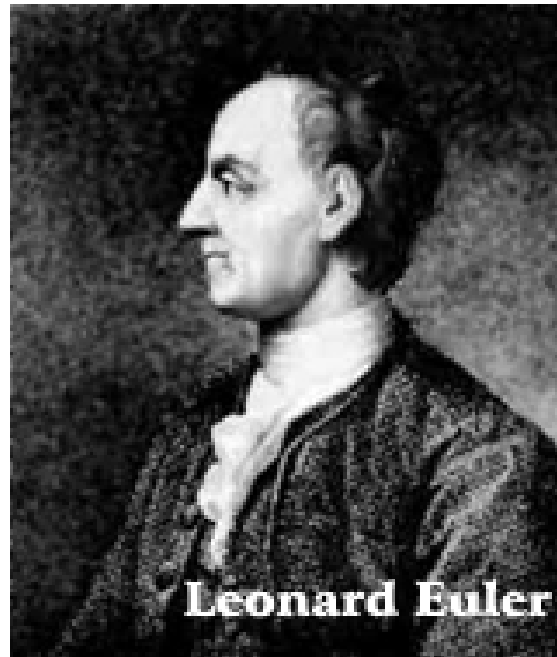
Grafos I: Introducción

Introducción

- Los grafos son la estructura de datos más **general**.
- **Componentes**: nodos y arcos.
- Historia:
 - Puentes de Königsberg (Prusia Oriental, 1736).
 - Leonhard **Euler**.



Leonhard Euler



*"Lisez Euler, Lisez Euler,
c'est notre maître à tous"*

Laplace

Definición

- **Grafo** $G = (V, A)$
- V : conjunto de **vértices** (nodos).
- A : conjunto de **arcos** (enlaces).
- Nodos **adyacentes** (vecinos): aquellos que están unidos por un arco.
- Arco **dirigido**:



- Arco **no-dirigido**:

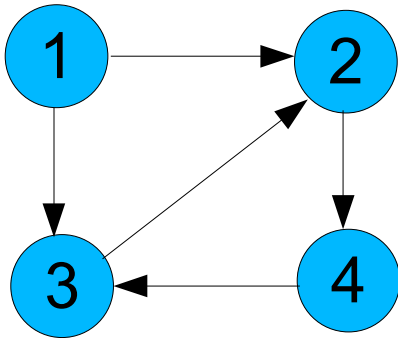


Definiciones

- **Grafos** dirigidos y no-dirigidos.
- **Camino**: secuencia de vértices $v_1, v_2, \dots, v_n$, tales que v_i y v_{i+1} son adyacentes, $i=1, 2, \dots, n-1$.
- **Longitud** del camino: número de arcos en el camino.
- Camino **simple**: todos los vértices con excepción del primero y el último son diferentes.
- **Ciclo**: camino en el que el primer y el último nodo son iguales.

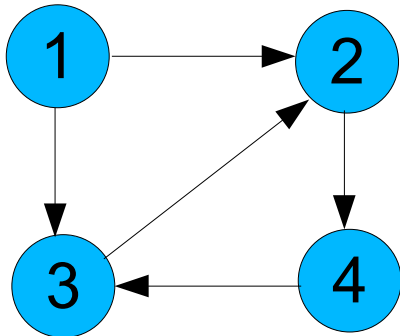
Representaciones de un grafo

- Matriz de adyacencia:



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

- Listas de adyacencia:



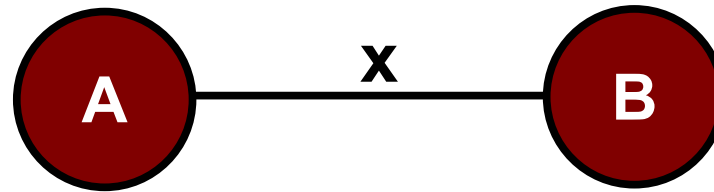
1 → 2 3
2 → 4
3 → 2
4 → 3

Recorridos

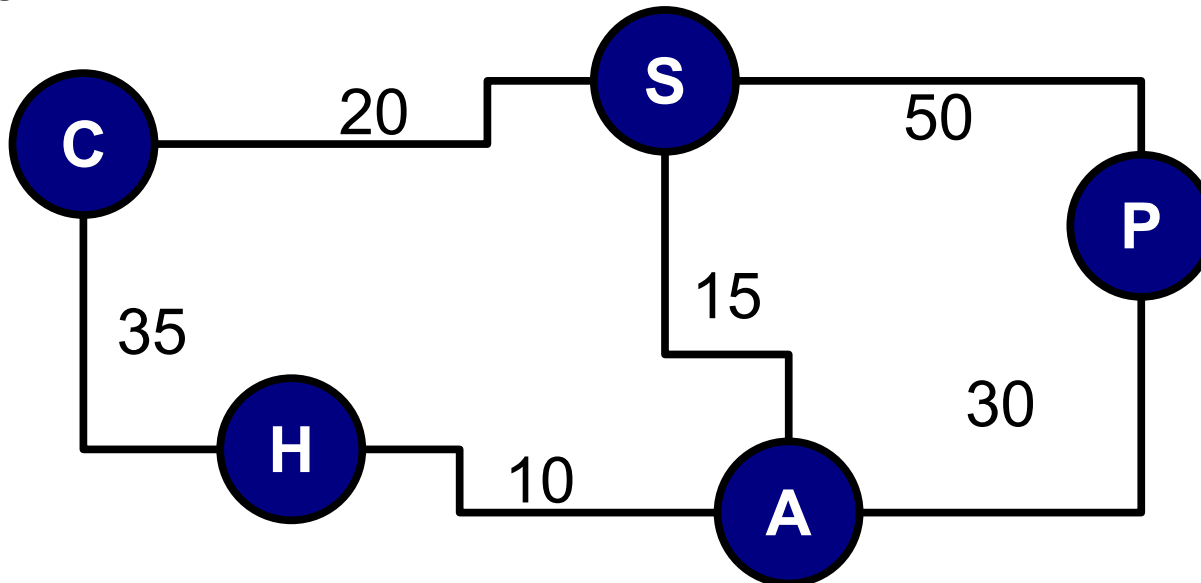
- *Profundidad:*
 - Regla 0: Escoja un vértice inicial y visítelo.
 - Regla 1: Si es posible elija un vértice adyacente que no haya sido visitado e introdúzcalo en la **pila**.
 - Regla 2: Si no se puede seguir la regla 1, saque un vértice de la **pila**.
- *Amplitud:*
 - Regla 0: Escoja un vértice inicial y visítelo.
 - Regla 1: Visite el próximo vértice no visitado adyacente, visítelo e insértelo en la **cola**.
 - Regla 2: Si no se puede seguir la regla 1, saque un vértice de la **cola**.

Grafos con pesos

- Cada arco está etiquetado con un **valor**:



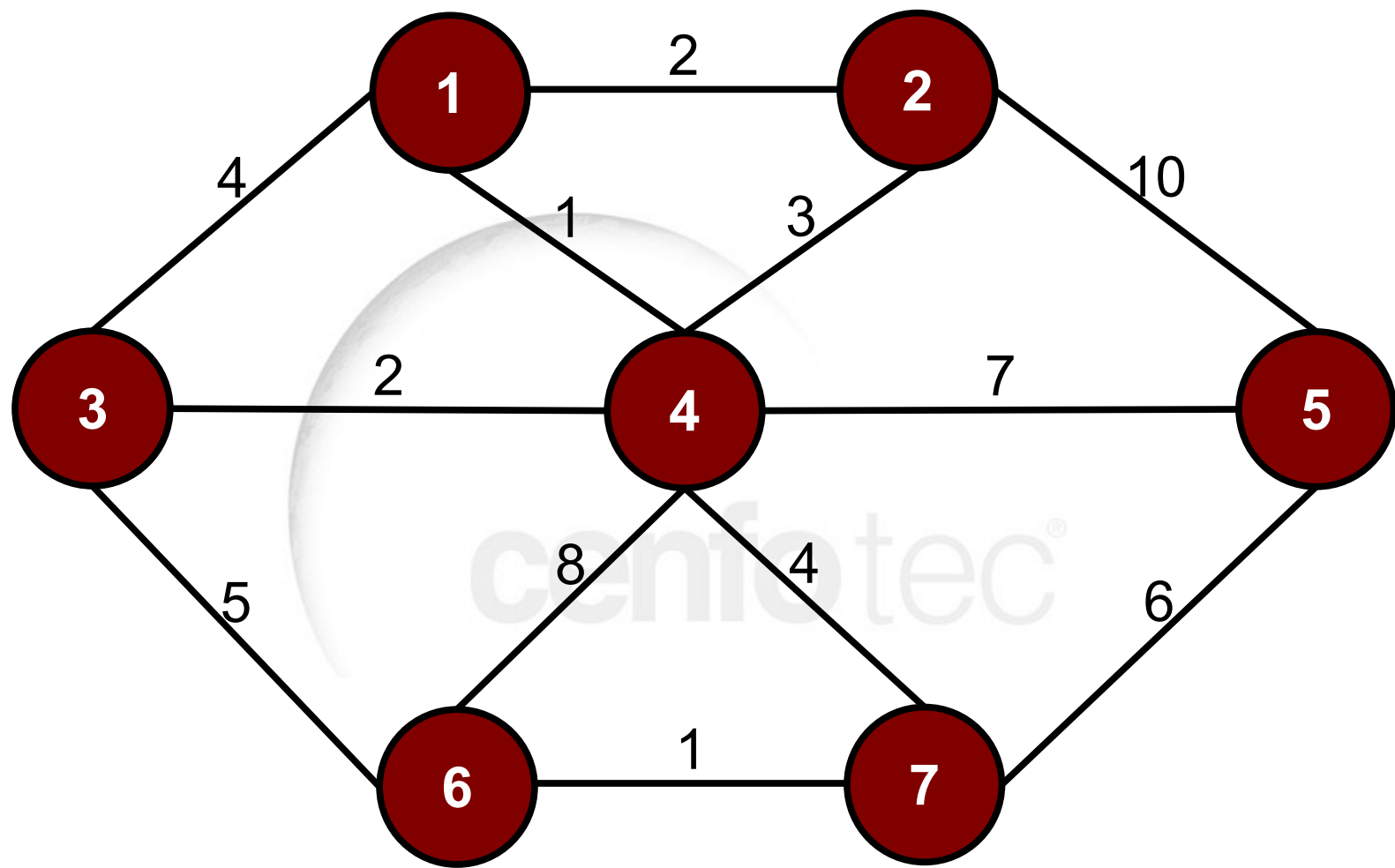
- Ejemplo:



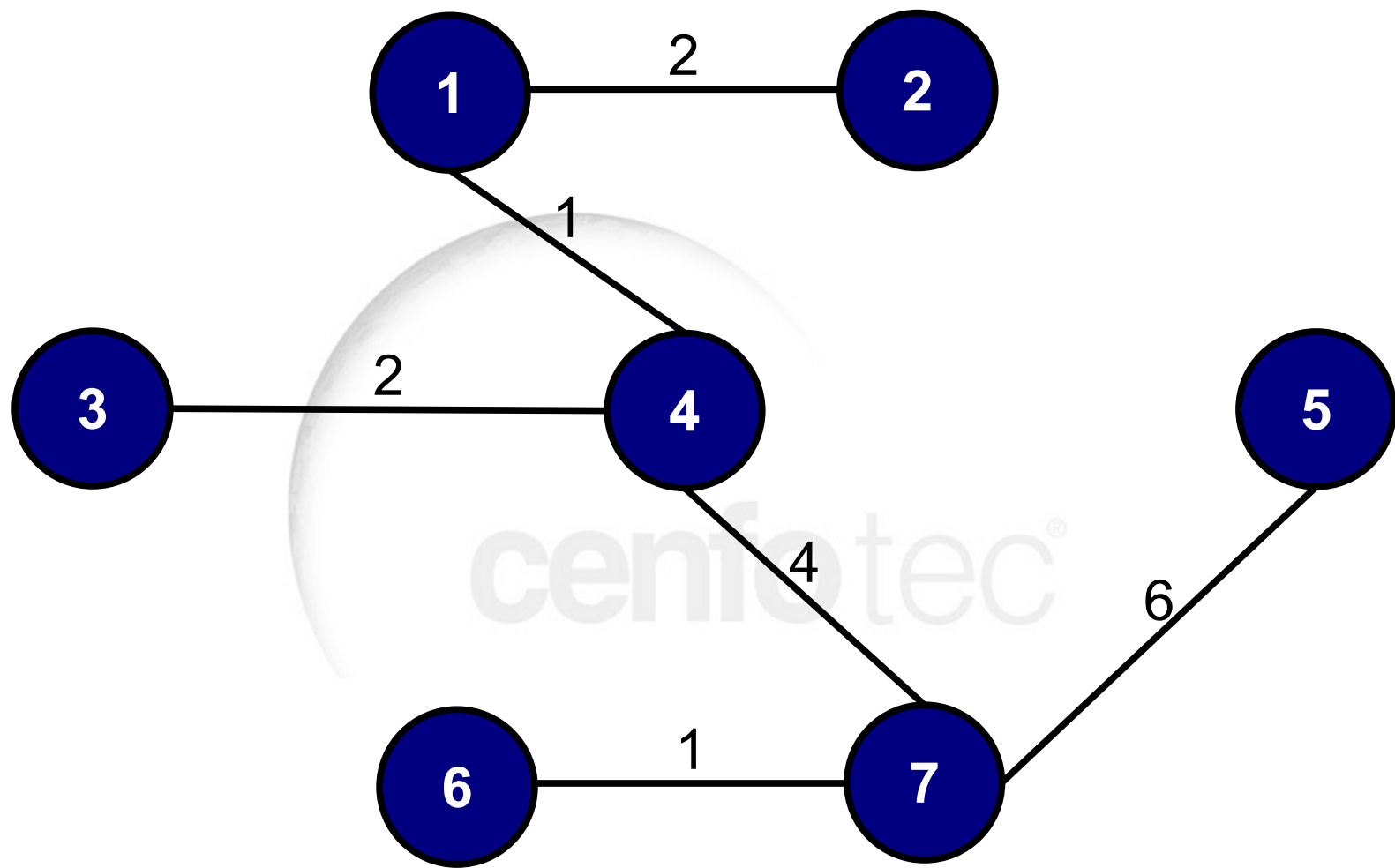
Árbol mínimo de expansión

- **Subgrafo** de $G=(V,A)$:
 - Es un grafo $G'=(V',A')$.
 - $V' \subseteq V$ y $A' \subseteq A$
- **Árbol libre** de $G=(V,A)$:
 - Subgrafo conexo.
 - Si el grafo tiene n nodos, el árbol libre contiene $n-1$ aristas.
- **Árbol mínimo** de expansión: árbol libre con la suma mínima del peso de sus arcos.

Árbol mínimo de expansión (cont.)



Árbol mínimo de expansión (cont.)



Algoritmo de Prim

- **Notación:**
 - T: conjunto de aristas del árbol mínimo de expansión.
 - U: conjunto de vértices.
- **Complejidad:**
 - $O(n^2)$, donde n es el número de nodos en el grafo.

Algoritmo de Prim (cont.)

$T \leftarrow \emptyset$

$U \leftarrow \{1\}$

while $U \neq V$

 <sea (u,v) la arista de costo mínimo tal que

$u \in U$ y $v \in V$ >

$T \leftarrow T \cup \{(u,v)\}$

$U \leftarrow U \cup \{v\}$

Algoritmo de Kruskal

- **Notación:**
 - T: conjunto de aristas del árbol mínimo de expansión.
- **Complejidad:**
 - $O(a \log(a))$, donde a es el número de aristas en el grafo.

Algoritmo de Kruskal (cont.)

$T \leftarrow \{\}$

aristas $\leftarrow \{\}$

comp $\leftarrow$ <número de nodos del grafo>

<crear tantos componentes como nodos>

while (comp > 1)

 <sea $a=(u,v)$ la arista mínima no considerada>

 <compU es el componente donde está u>

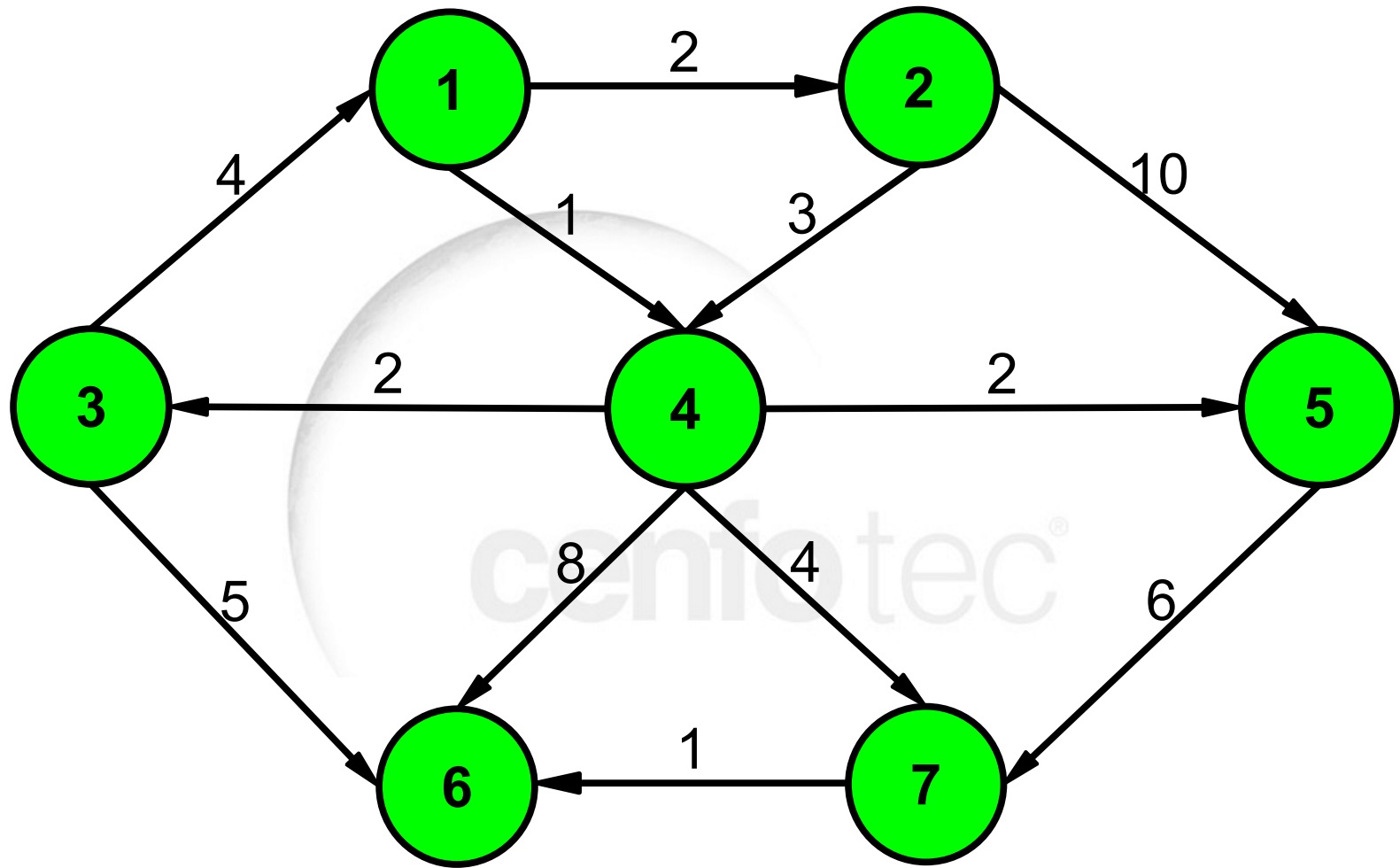
 <compV es el componente donde está v>

 if(compU $\neq$ compV)

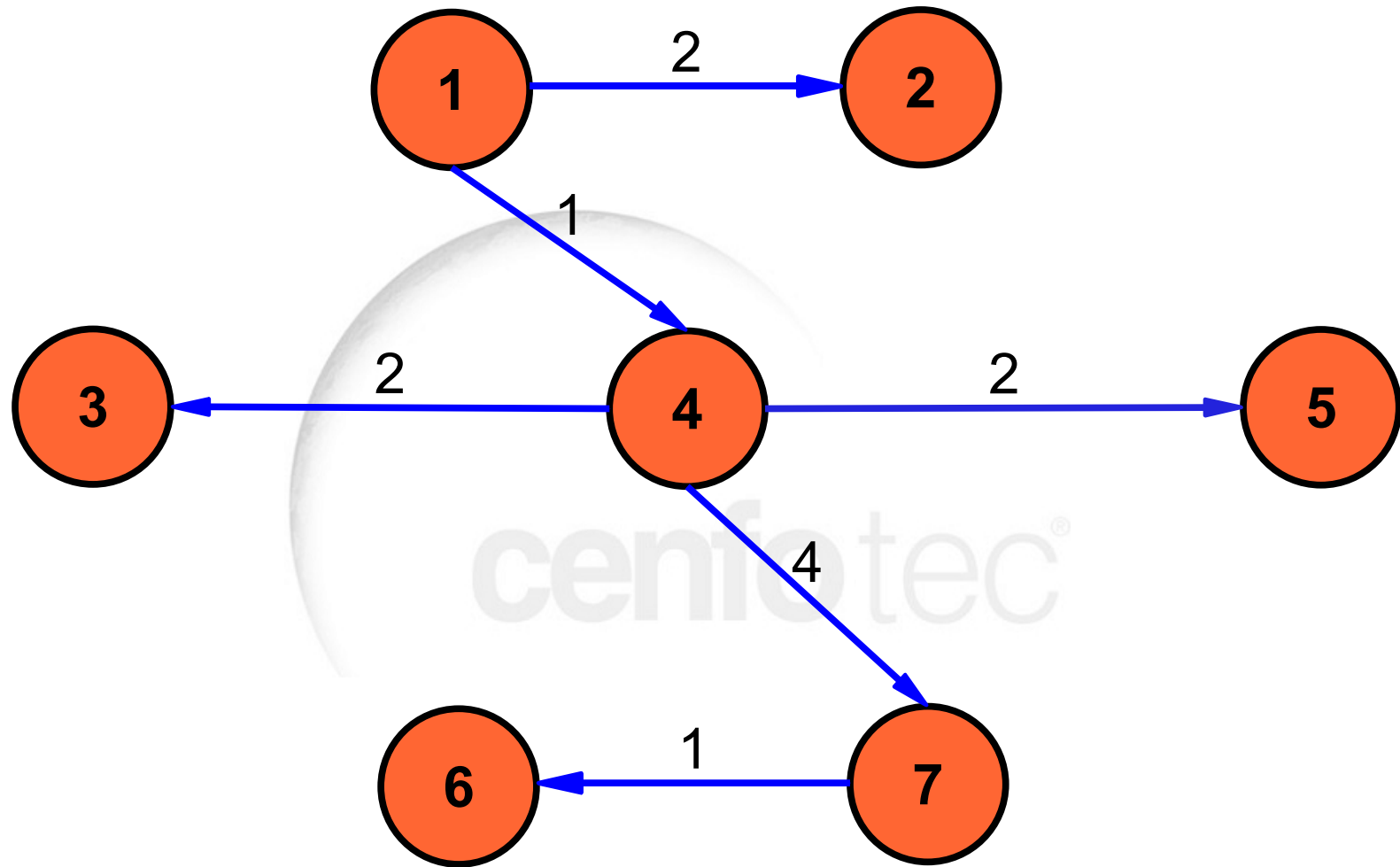
 <unir los componentes compU y compV>

$T \leftarrow T \cup \{a\}$

Ruta más corta



Ruta más corta (cont.)



Algoritmo de Dijkstra

- **Notación:**
 - D: vector que contiene las longitudes mínimas de los caminos desde el origen a cada nodo.
 - C: matriz de costos.
 - P: vector con el vértice anterior para cada nodo en el camino más corto.
- **Complejidad:**
 - $O(n^2)$ si se utiliza matriz de adyacencia. Donde n es el número de nodos en el grafo.
 - $O(a \log(n))$ si se utiliza listas de adyacencia. Donde a es el número de aristas en el grafo.

Algoritmo de Dijkstra (cont.)

$S \leftarrow \{1\}$

for $i=2$ to n

$D[i] \leftarrow C[1,i]$

for $i=1$ to $(n-1)$

<elegir un vértice en $V-S$ tal que $D[w]$ sea mínimo>

<agregar w a S >

for each vértice v en $V-S$

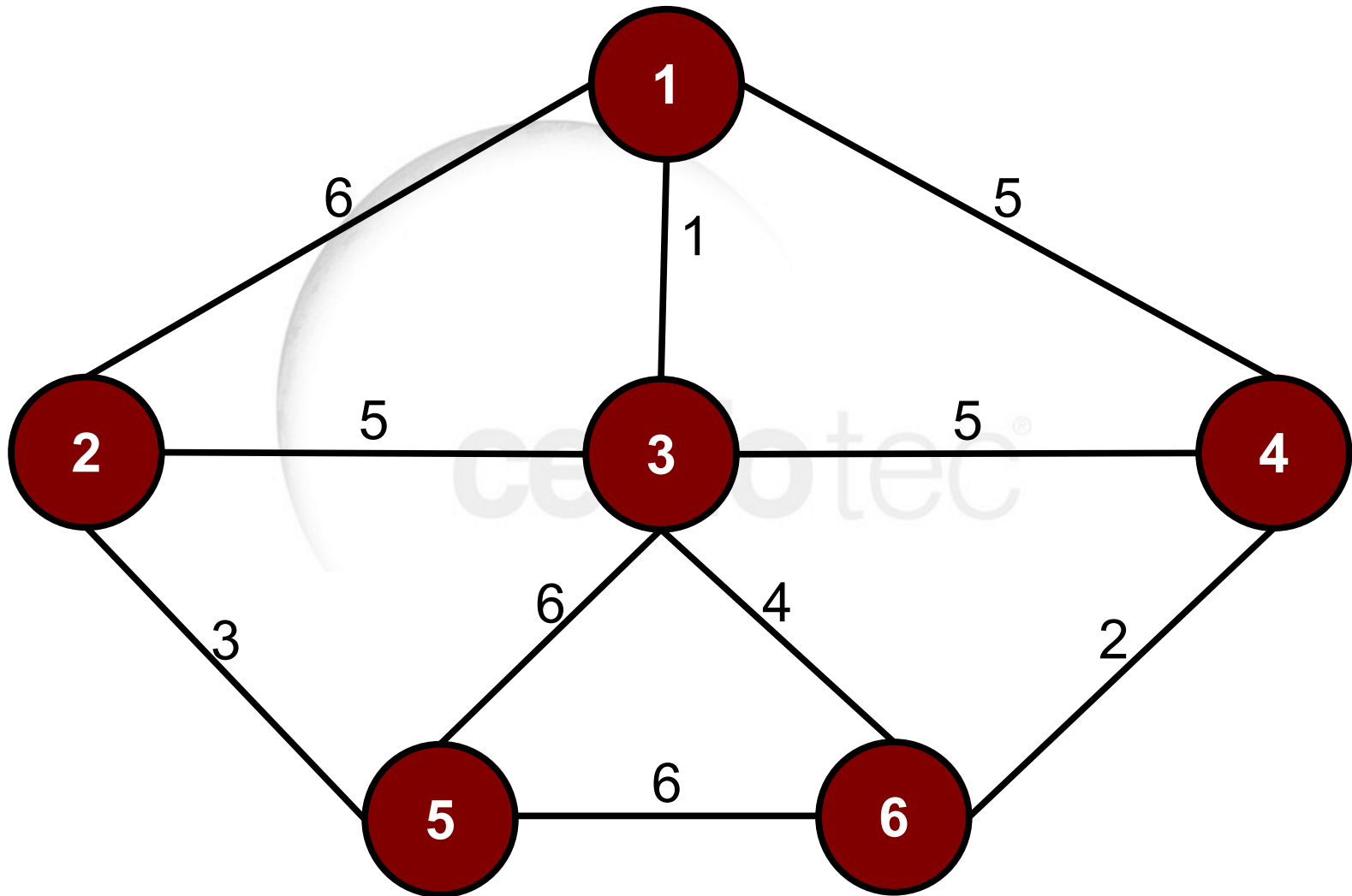
$D[v] \leftarrow \min(D[v], D[w]+[w,v])$

if($D[w]+C[w,v] < D[v]$)

$P[v] \leftarrow w$

Ejercicios

- **Ejercicio 1:** Aplique los algoritmos de Prim y Kruskal al siguiente grafo.



Ejercicios

- **Ejercicio 2:** Aplique el algoritmo de Dijkstra al siguiente grafo.

