UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Investigación de Operaciones - Problemas sobre redes

Víctor Viana

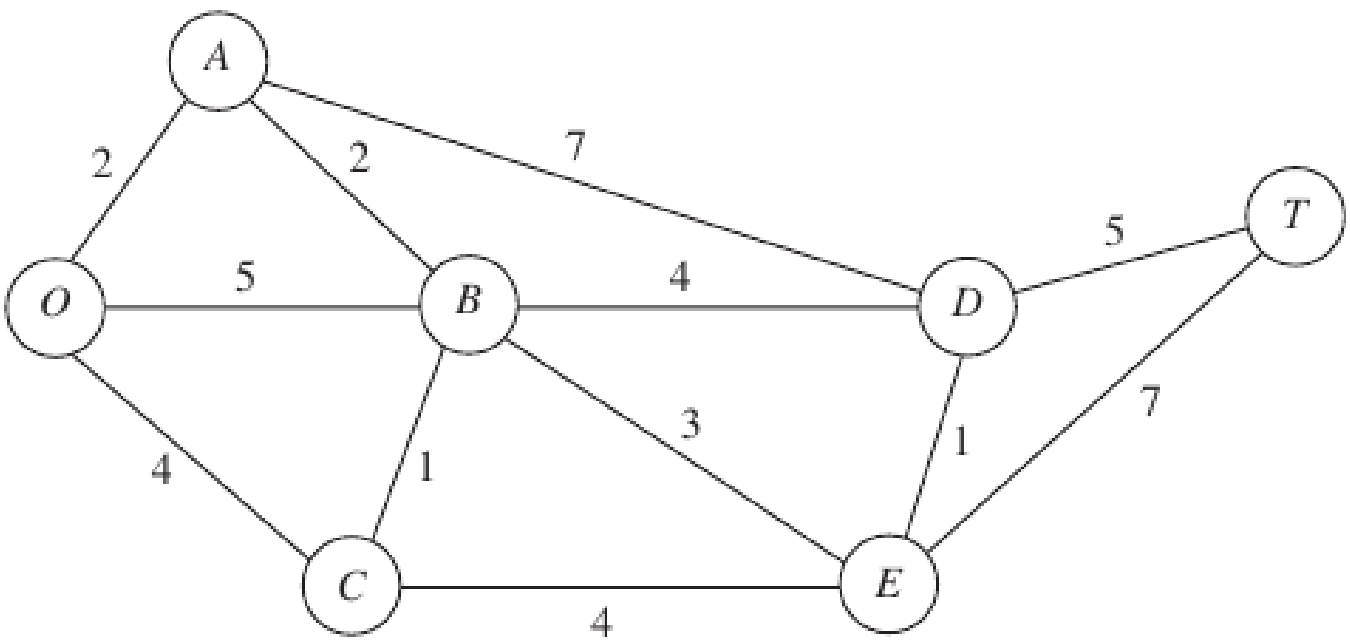
victor.viana@noreste.udelar.edu.uy

5/6/2026

Contenido

El Problema de la Ruta Mas Corta

- Considere una red (grafo) conexa y no dirigida con dos nodos especiales llamados **origen** y **destino**.
- A cada arista se le asocia una distancia no negativa.
- El objetivo es encontrar la ruta más corta (la trayectoria con la mínima distancia total) del origen al destino.



Algoritmo de la Ruta Mas Corta (Dijkstra)

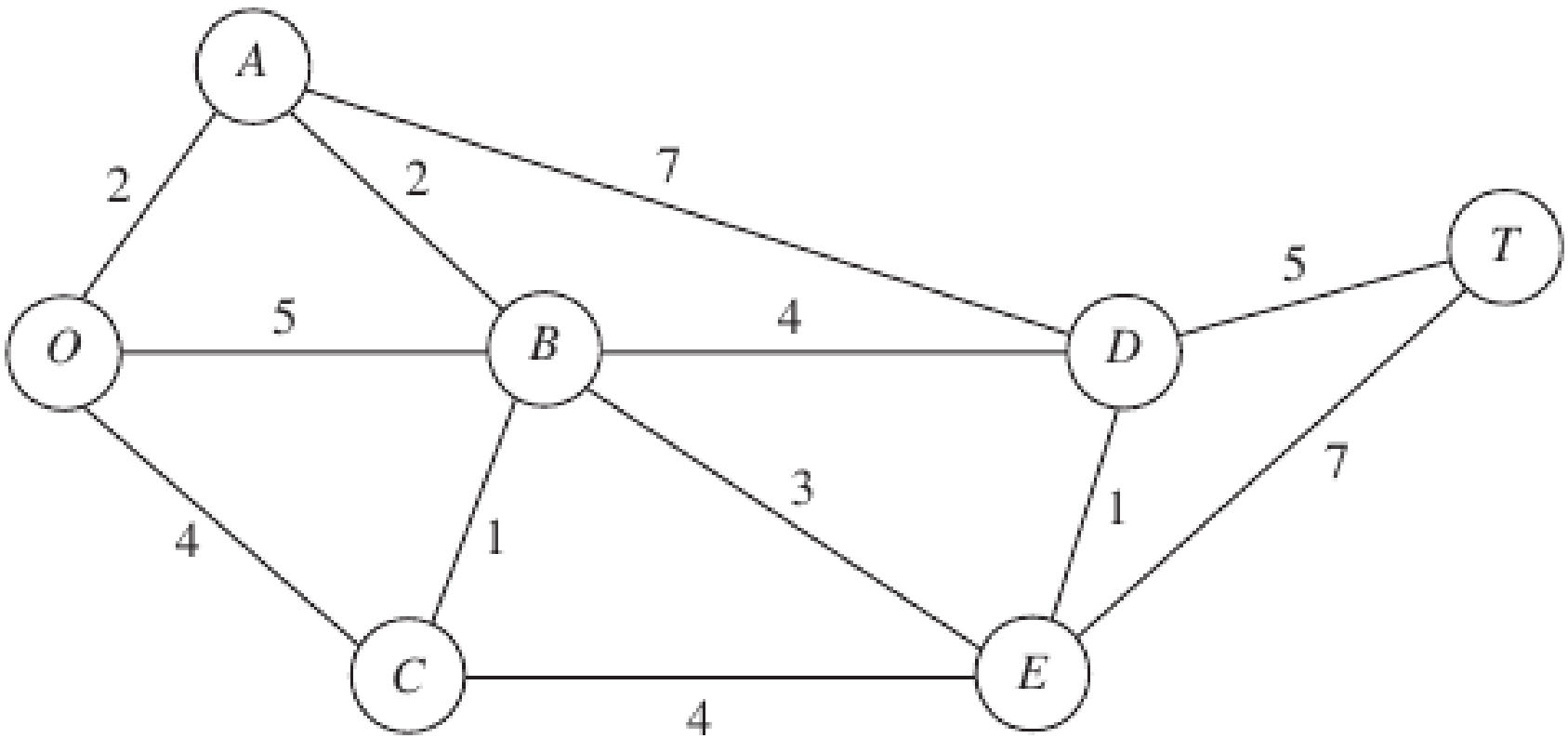
- El Algoritmo de Dijkstra es un método para encontrar el camino más corto entre dos nodos en un grafo ponderado con pesos positivos.
- La idea central es explorar el grafo de manera gradual, siempre eligiendo el nodo más cercano al origen que aún no ha sido visitado.

Concepto Fundamental

- El algoritmo mantiene un registro de la distancia mínima conocida desde el nodo origen hasta cada nodo del grafo.
- Inicialmente, solo conocemos que la distancia del origen a sí mismo es 0, y todas las demás distancias son infinitas.
- En cada paso, seleccionamos el nodo no visitado con la menor distancia conocida, lo visitamos, y actualizamos las distancias a sus vecinos si encontramos un camino más corto.

Ejemplo: Camino de O a T

Usando el grafo de la imagen, encontraremos el camino más corto desde O hasta T.



Paso 0: Inicialización

Comenzamos estableciendo las distancias iniciales:

- Distancia a O $= 0$ (es nuestro origen)
- Distancia a todos los demás nodos $= \infty$
- Ningún nodo ha sido visitado aún

Estado inicial:

- O: 0
- A: ∞
- B: ∞
- C: ∞
- D: ∞
- E: ∞
- T: ∞

Paso 1: Visitar O

Seleccionamos O (distancia 0, la menor). Examinamos sus vecinos:

- A: distancia $0 + 2 = 2$ (mejor que ∞)
- B: distancia $0 + 5 = 5$ (mejor que ∞)
- C: distancia $0 + 4 = 4$ (mejor que ∞)

Distancias actualizadas:

- O: 0 (visitado)
- A: 2
- B: 5
- C: 4
- D: ∞
- E: ∞
- T: ∞

Paso 2: Visitar A

Seleccionamos A (distancia 2, la menor entre no visitados).
Examinamos sus vecinos:

- B: distancia $2 + 2 = 4$ (mejor que 5, actualizamos)
- D: distancia $2 + 7 = 9$ (mejor que ∞)

Distancias actualizadas:

- O: 0 (visitado)
- A: 2 (visitado)
- B: 4
- C: 4
- D: 9
- E: ∞
- T: ∞

Paso 3: Visitar B o C

Seleccionamos B (distancia 4, empate con C pero elegimos B).
Examinamos sus vecinos:

- D: distancia $4 + 4 = 8$ (mejor que 9, actualizamos)
- E: distancia $4 + 3 = 7$ (mejor que ∞)
- C: distancia $4 + 1 = 5$ (peor que 4, no actualizamos)

Distancias actualizadas:

- O: 0 (visitado)
- A: 2 (visitado)
- B: 4 (visitado)
- C: 4
- D: 8
- E: 7
- T: ∞

Paso 4: Visitar C

Seleccionamos C (distancia 4). Examinamos sus vecinos:
E: distancia $4 + 4 = 8$ (peor que 7, no actualizamos)
Distancias actualizadas:

- O: 0 (visitado)
- A: 2 (visitado)
- B: 4 (visitado)
- C: 4 (visitado)
- D: 8
- E: 7
- T: ∞

Paso 5: Visitar E

Seleccionamos E (distancia 7). Examinamos sus vecinos:
D: distancia $7 + 1 = 8$ (igual que 8, no mejora)
Distancias actualizadas:

- O: 0 (visitado)
- A: 2 (visitado)
- B: 4 (visitado)
- C: 4 (visitado)
- D: 8 (visitado)
- E: 7 (visitado)
- T: ∞

Paso 6: Visitar D

Seleccionamos D (distancia 8). Examinamos sus vecinos:
T: distancia $8 + 5 = 13$ (mejor que ∞)
Distancias actualizadas:

- O: 0 (visitado)
- A: 2 (visitado)
- B: 4 (visitado)
- C: 4 (visitado)
- D: 8 (visitado)
- E: 7
- T: 13

Visitar T

Seleccionamos T (distancia 13).
Como T es nuestro destino, el algoritmo puede terminar aquí.
Resultado Final: La distancia más corta de O a T es 13.
El camino óptimo es: $O \longrightarrow A \longrightarrow B \longrightarrow D \longrightarrow T$

- O a A: 2
- A a B: 2
- B a D: 4
- D a T: 5

Total: $2 + 2 + 4 + 5 = 13$

Principio Clave

El algoritmo funciona porque siempre procesa los nodos en orden de distancia creciente desde el origen. Cuando visitamos un nodo, podemos estar seguros de que ya hemos encontrado el camino más corto hacia él, ya que cualquier otro camino pasaría por nodos no visitados que están más lejos.

Algoritmo de la Ruta Mas Corta (Dijkstra) - cont.

1. Inicializa la distancia del nodo de inicio a 0 y la de todos los demás nodos a infinito.
2. Marca todos los nodos como no visitados.
3. Tomando el nodo con la menor distancia conocida (inicialmente, el nodo de inicio), actualiza las distancias de sus vecinos.
4. Marca el nodo actual como visitado.
5. Repite el proceso para el siguiente nodo no visitado con la menor distancia y continúa hasta que todos los nodos hayan sido visitados.

Modelo de Prog. Matemática

Sea $G = (V, E)$ el grafo ponderado de la imagen, donde

- $V = \{O, A, B, C, D, E, T\}$ es el conjunto de nodos,
- $E \subset V \times V$ son las aristas con peso $w_{ij} > 0$ para cada $(i, j) \in E$.

Definimos la variable binaria

$$x_{ij} = \begin{cases} 1, & \text{si la arista } (i \to j) \text{ forma parte del camino,} \\ 0, & \text{en caso contrario.} \end{cases}$$

Función Objetivo, minimizar la longitud total del camino desde el nodo origen O hasta el destino T:

$$\text{mín} \sum_{(i,j) \in E} w_{ij}\, x_{ij}.$$

Restricciones

1. Nodo origen O: sale exactamente una unidad de flujo

$$\sum_{j:(O,j)\in E} x_{Oj} = 1$$

2. Nodo destino T: entra exactamente una unidad de flujo

$$\sum_{i:(i,T)\in E} x_{iT} = 1$$

3. Nodos intermedios $k \in V \setminus \{O, T\}$: flujo entrante igual a flujo saliente

$$\sum_{i:(i,k)\in E} x_{ik} - \sum_{j:(k,j)\in E} x_{kj} = 0$$

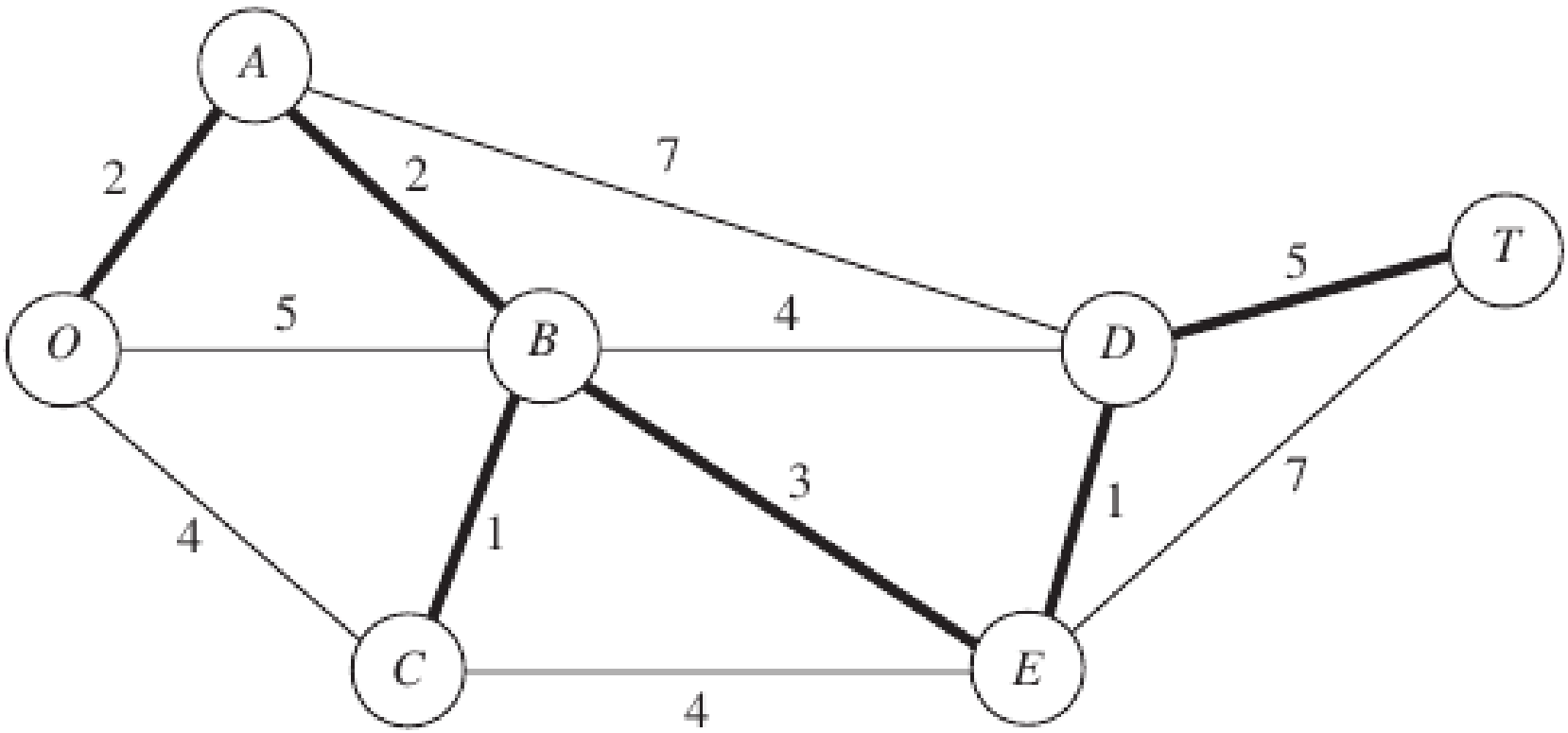
4. Dominio de las Variables

$$x_{ij} \in \{0, 1\}, \quad \forall (i,j) \in E.$$

Árbol o esqueleto de recubrimiento mínimo

¿Qué es un árbol de recubrimiento mínimo?
Es el sub-grafo obtenido uniendo los nodos usando la menor cantidad posible de aristas, pero sin crear circuitos cerrados y usando la menor suma de distancias/pesos.

Arboles de recubrimiento o expansión mínima (esqueleto)

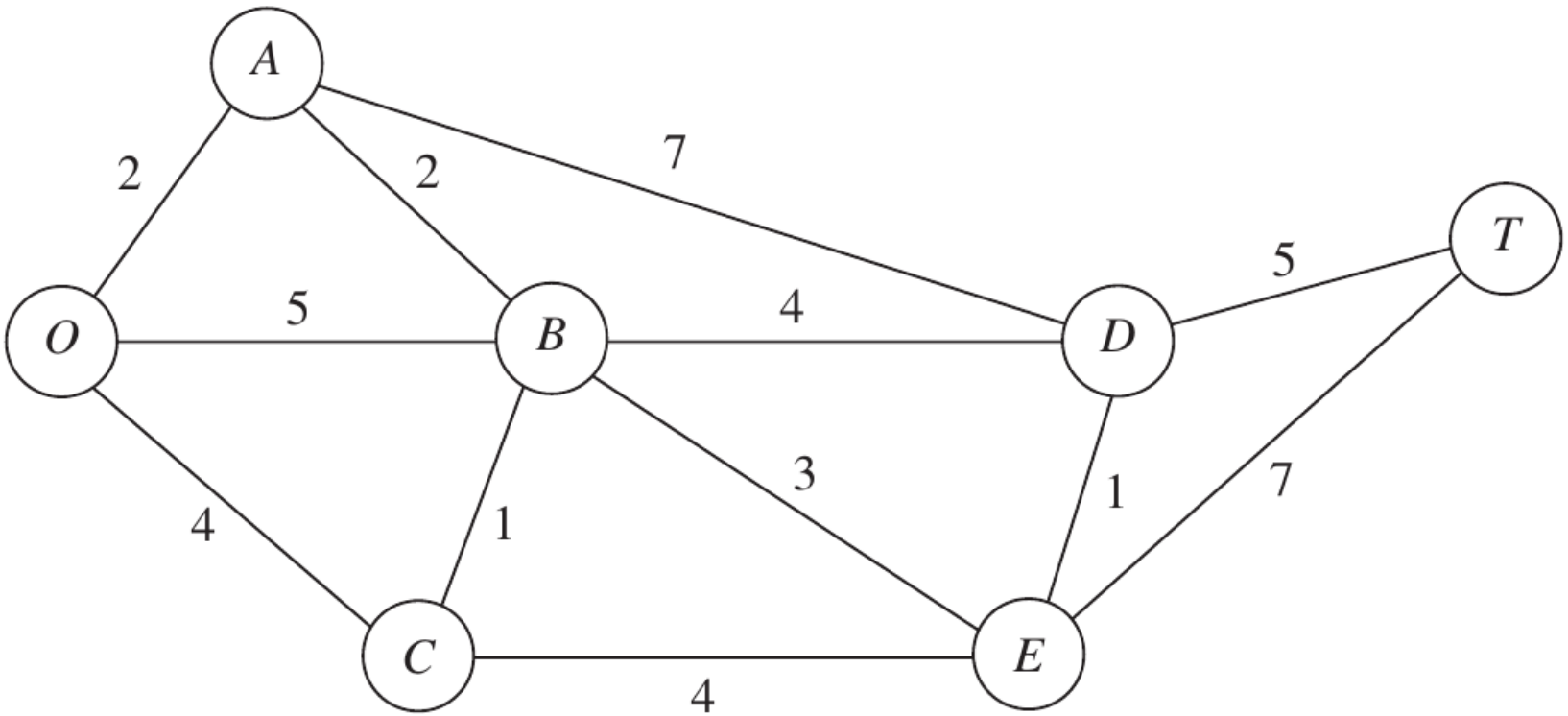


ALGORITMO DE PRIM

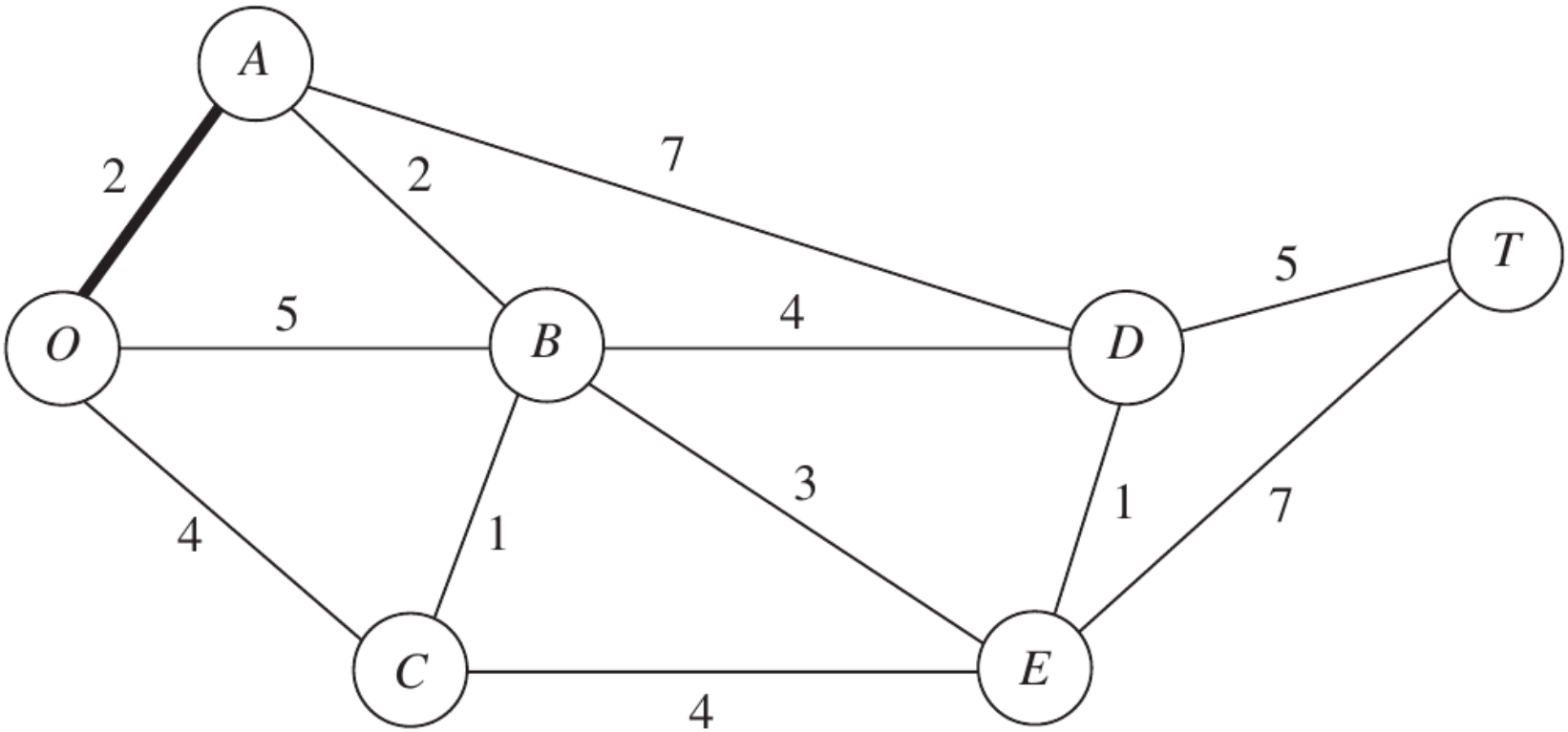
Repita hasta que el árbol T tenga (n-1) aristas :

1. Al comienzo tome cualquier arista que tenga el menor valor asignado.
2. Agregue a T la arista de valor mínimo, conformada por un vértice en T y otro vértice que no pertenezca a T.

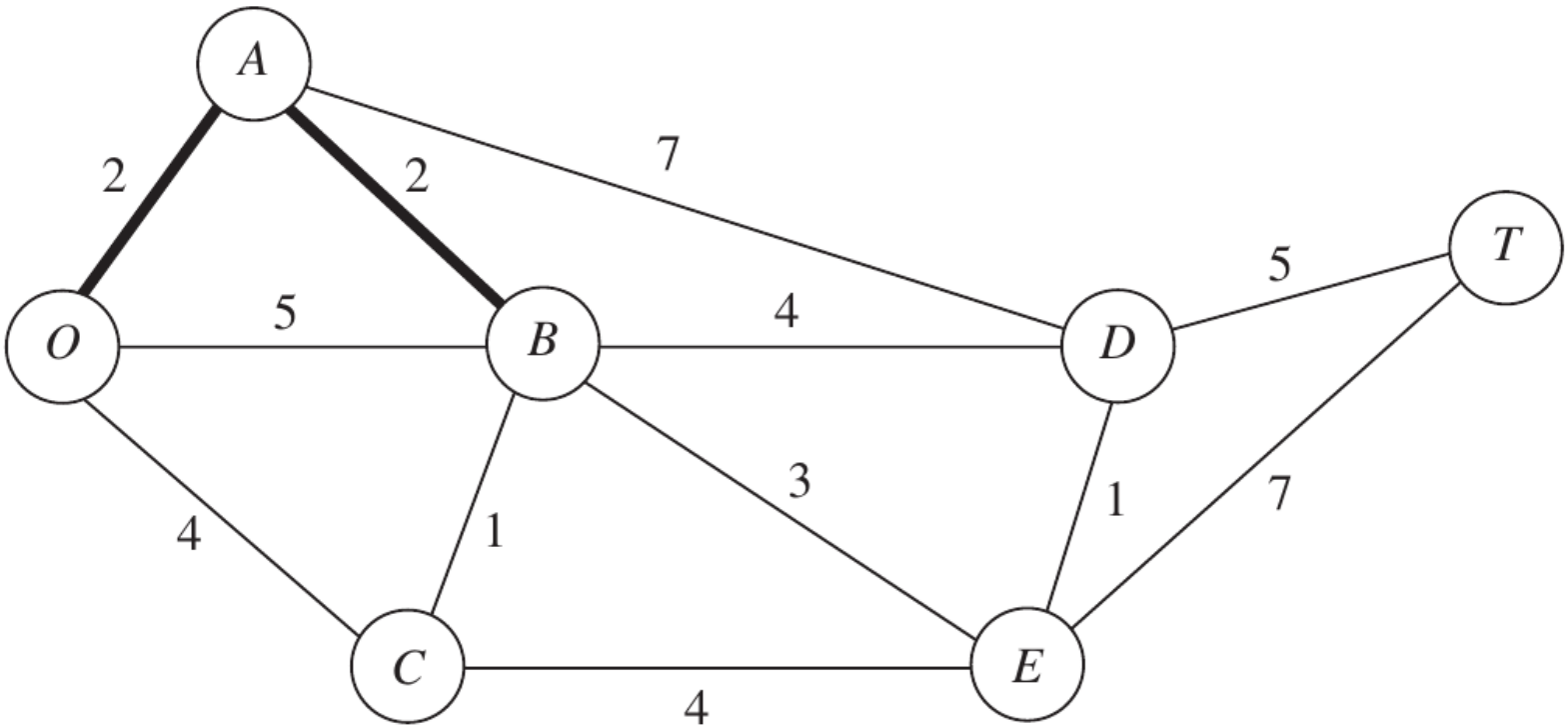
Algoritmo de Prim - 0



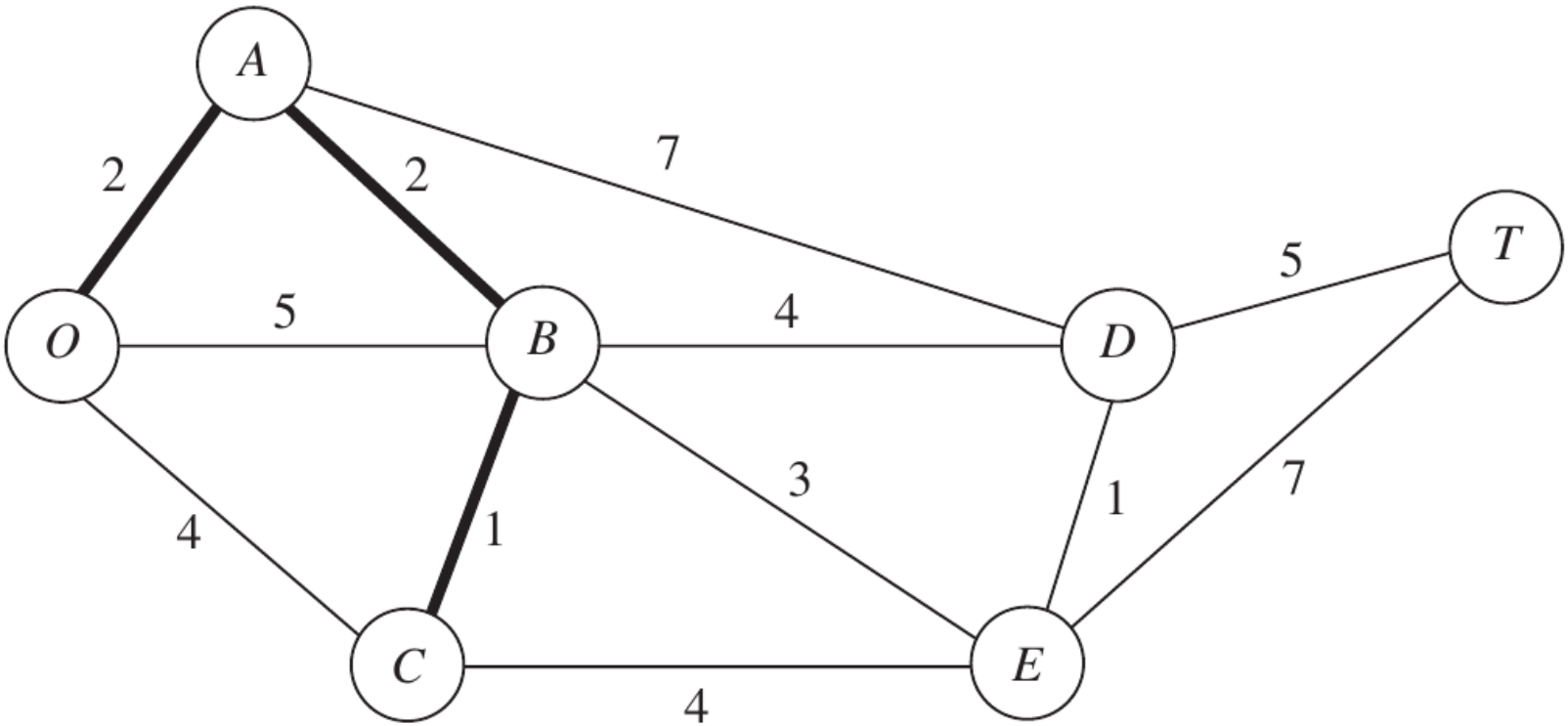
Algoritmo de Prim - 1



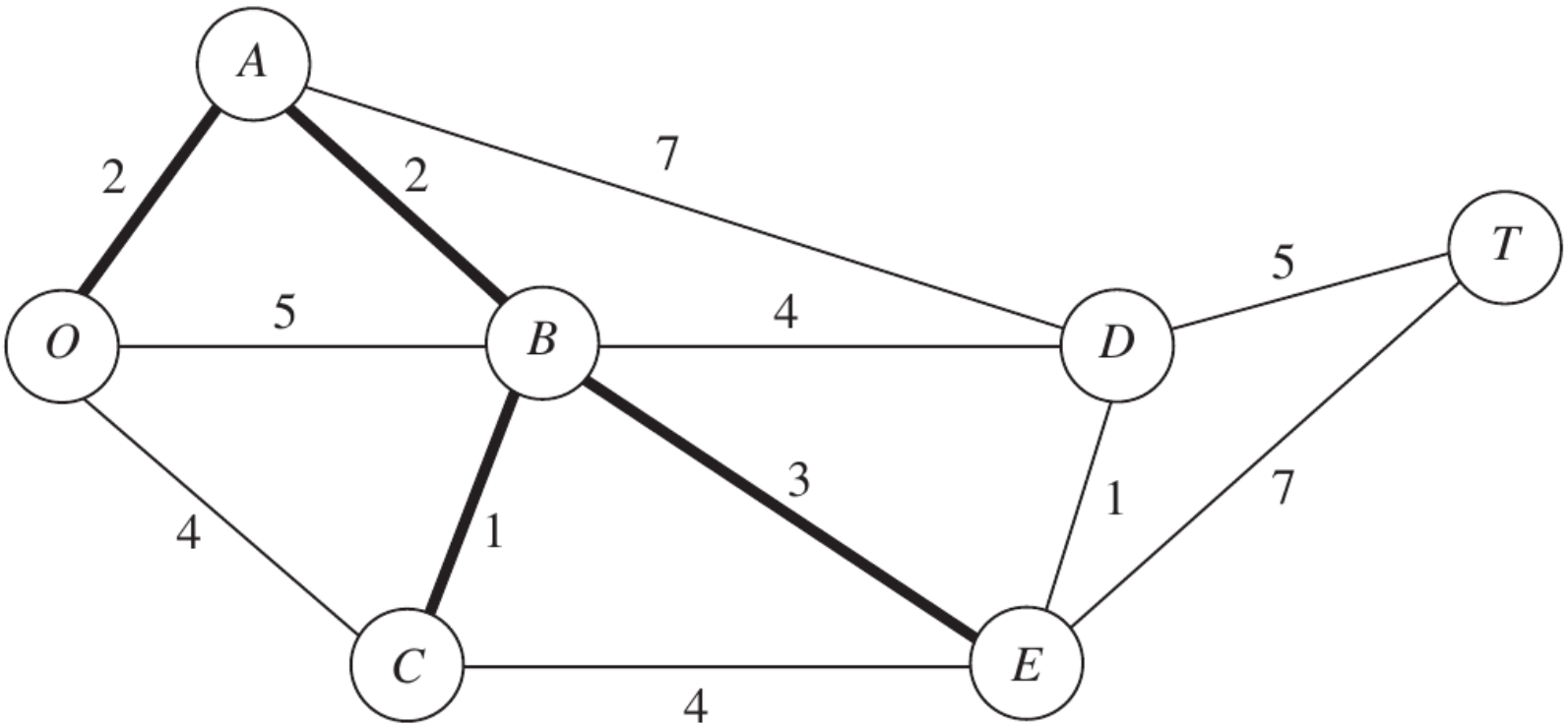
Algoritmo de Prim - 2



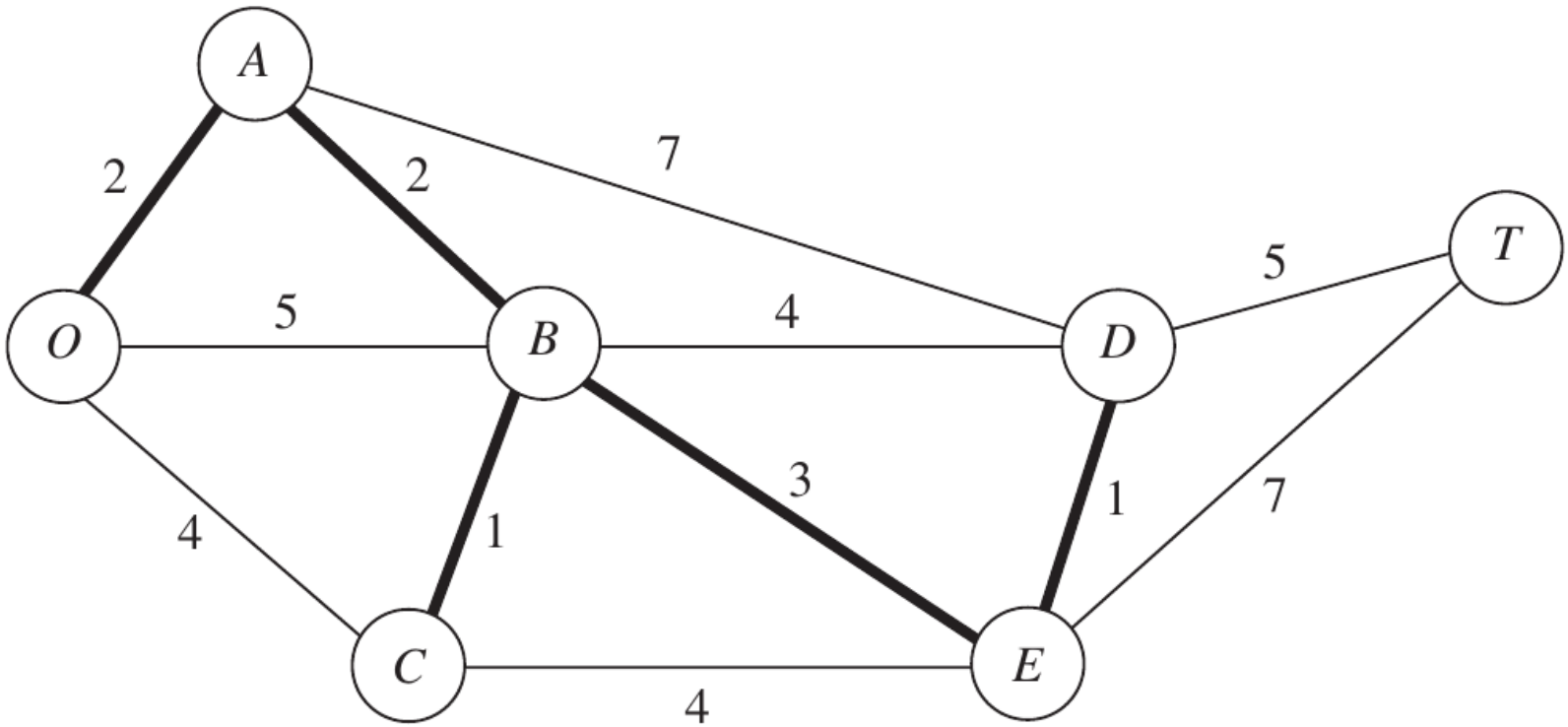
Algoritmo de Prim - 3



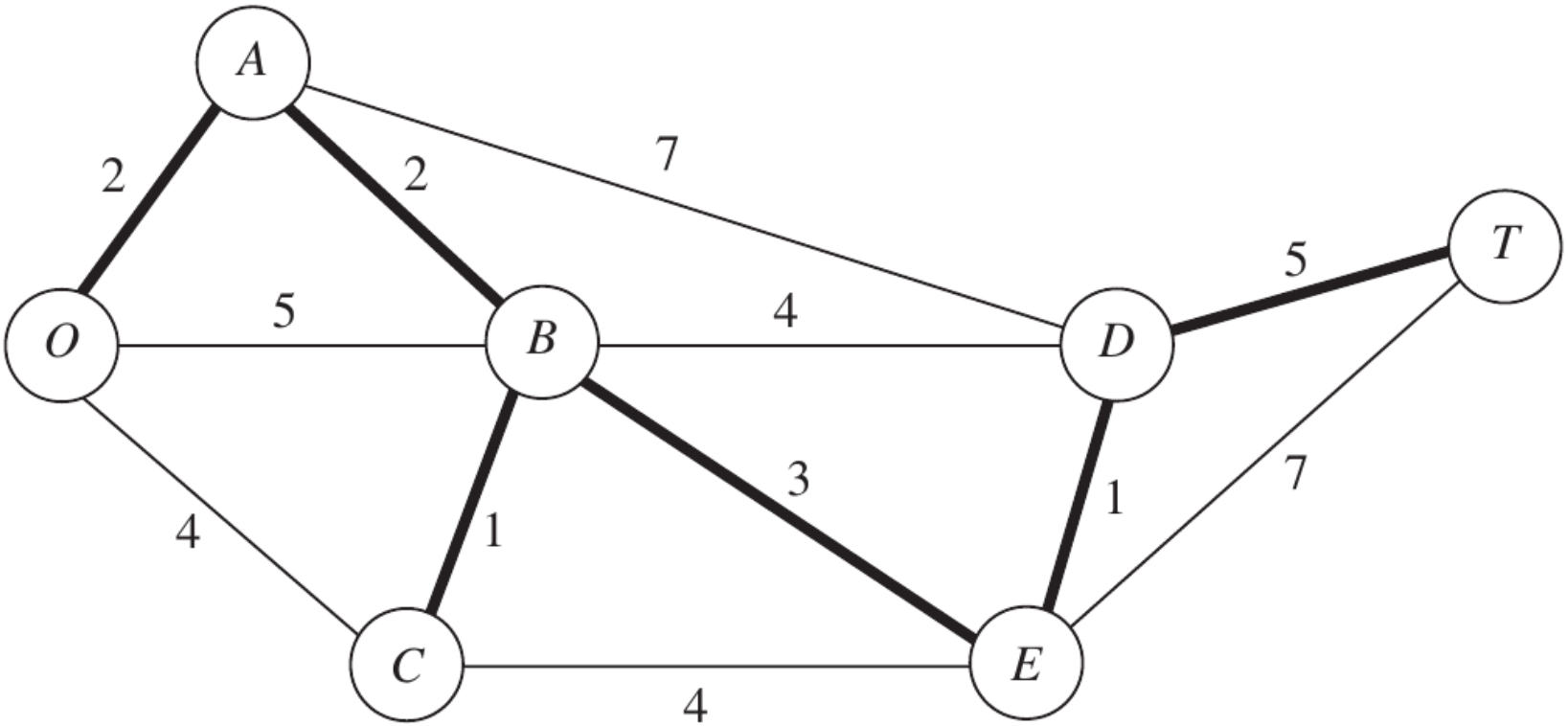
Algoritmo de Prim - 4



Algoritmo de Prim - 5



Algoritmo de Prim - 6



ALGORITMO DE KRUSKAL

Repita los siguientes pasos hasta que el conjunto T tenga (n-1) aristas ($|T| =$ n-1):

1. Al comenzar $T = \emptyset$
2. Agregue a T las aristas de menor valor que no formen un ciclo con las aristas que ya están en T.