Introducción a Grafos

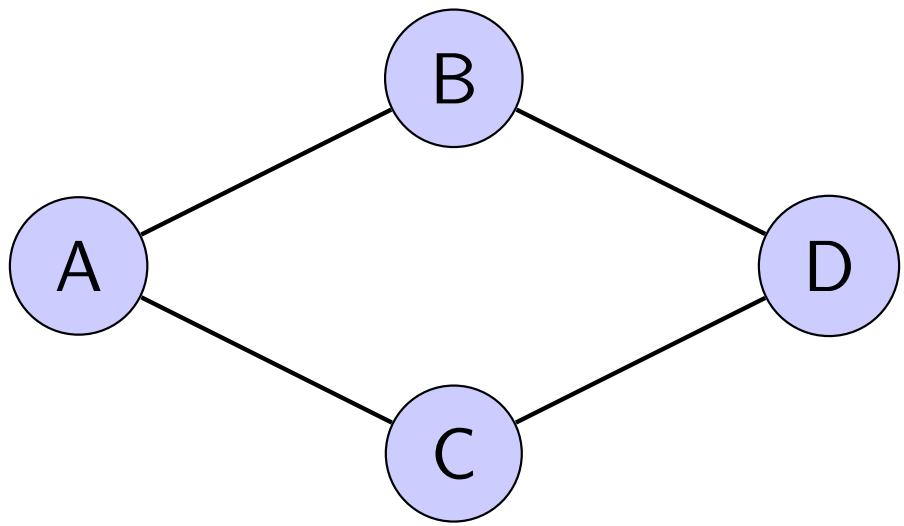
Cosijopii García

Introducción a Grafos

- Los grafos son estructuras matemáticas utilizadas para modelar relaciones entre objetos.
- Representan problemas de conectividad en diversas áreas:
  - Ciencias de la computación.
  - Redes sociales.
  - Logística y transporte.
  - Biología y química.
- Formalmente, un grafo se define como $G = (V, E)$, donde:
  - V: Conjunto de vértices o nodos.
  - E: Conjunto de aristas que conectan los vértices.

¿Qué es un Grafo?

- Un grafo es una estructura compuesta de:
  - **Vértices (nodos):** Representan objetos o entidades.
  - **Aristas (enlaces):** Representan relaciones entre los nodos.
- Ejemplo gráfico:



Características de los Grafos

- Los grafos pueden clasificarse en diferentes tipos según sus propiedades:
  - **Dirigidos:** Las aristas tienen un sentido definido.
  - **No dirigidos:** Las aristas no tienen un sentido específico.
  - **Ponderados:** Las aristas tienen un peso asociado (costo, distancia, etc.).
  - **No ponderados:** Todas las aristas tienen el mismo valor.
- Propiedades fundamentales:
  - **Grado de un vértice:** Número de aristas conectadas a un nodo.
  - **Conexión:** Dos nodos están conectados si existe al menos un camino entre ellos.
  - **Ciclo:** Camino cerrado que comienza y termina en el mismo nodo.

Propiedades de los Grafos

Los grafos poseen diversas propiedades fundamentales que ayudan a describir sus características y comportamiento. Estas propiedades incluyen:

- **Grado de un vértice ($deg(v)$)**:
  - Número de aristas que inciden en un vértice.
  - Para grafos dirigidos:
    - **Grado de entrada ($deg_{in}(v)$)**: Número de aristas que llegan al vértice v.
    - **Grado de salida ($deg_{out}(v)$)**: Número de aristas que salen del vértice v.
- **Densidad del grafo (D)**:
  - Medida de cuán completo es el grafo.
  - Fórmula:

$$D = \frac{2 \times |E|}{|V| \times (|V| - 1)}$$

    donde $|E|$ es el número de aristas y $|V|$ el número de vértices.
- **Caminos y ciclos**:
  - **Camino**: Secuencia de vértices conectados por aristas.
  - **Ciclo**: Camino cerrado donde el primer y último vértice son el mismo.

Propiedades Avanzadas de los Grafos

- **Conectividad**:
  - Un grafo es **conexo** si existe un camino entre cualquier par de vértices.
  - En grafos dirigidos:
    - **Fuertemente conexo**: Hay un camino dirigido entre cada par de vértices.
    - **Débilmente conexo**: El grafo se vuelve conexo si se ignoran las direcciones de las aristas.
- **Diámetro (d)**:
  - Máxima distancia entre cualquier par de vértices.
  - Para grafos ponderados, se considera la suma de los pesos.
- **Centro del grafo**:
  - Vértice con la mínima distancia máxima a cualquier otro vértice.
- **Clustering coefficient**:
  - Medida de qué tan "conectado" está un vértice con sus vecinos.
  - Fórmula:

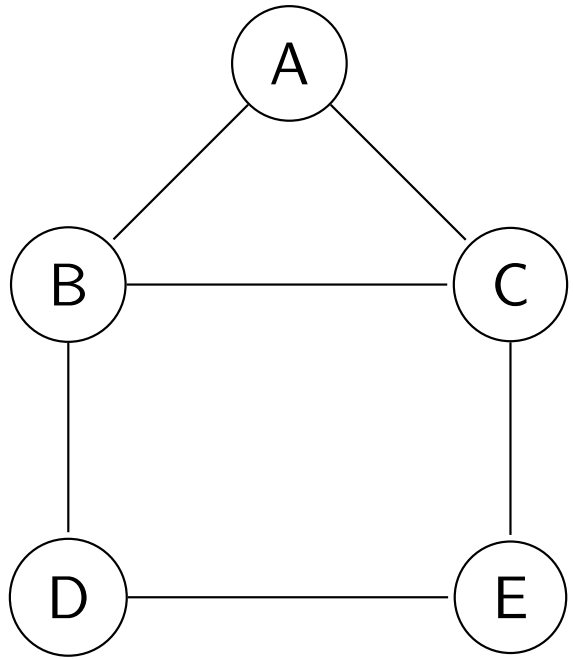
$$C(v) = \frac{2 \times \text{número de triángulos conectados a } v}{deg(v) \times (deg(v) - 1)}$$

Propiedades Globales de los Grafos

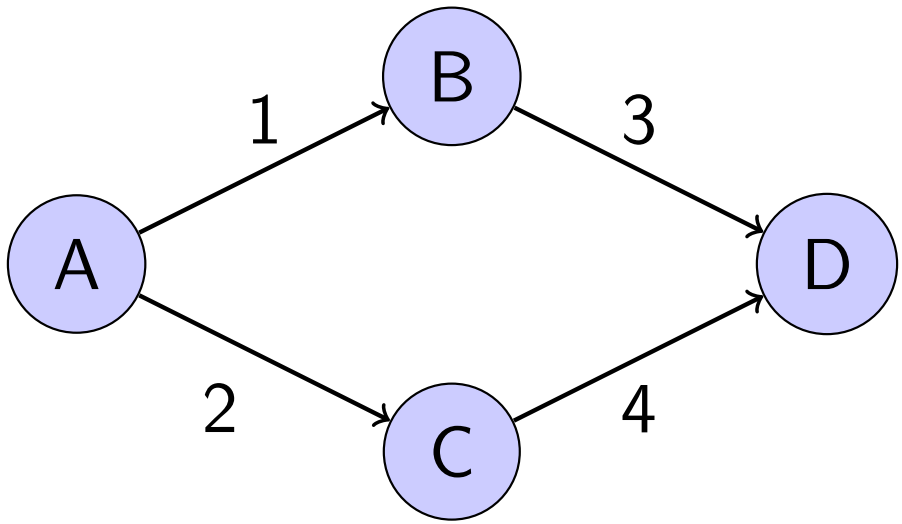
- **Número cromático ($\chi(G)$)**:
  - Número mínimo de colores necesarios para colorear los vértices de un grafo de manera que no haya dos vértices adyacentes con el mismo color.
- **Planaridad**:
  - Un grafo es **plano** si se puede dibujar en un plano sin que las aristas se crucen.
- **Árbol de expansión mínima**:
  - Subgrafo conexo que incluye todos los vértices con el peso total mínimo (para grafos ponderados).
- **Propiedades espectrales**:
  - Analizan el espectro del grafo (valores propios de la matriz de adyacencia o Laplaciana).
  - Aplicaciones en análisis de redes y teoría de grafos avanzada.

Ejercicios de Propiedades de Grafos I

1. Dado el siguiente grafo, calcule:
   - El grado de cada vértice.
   - La densidad del grafo.
   - El diámetro del grafo.



Ejemplo de un Grafo Dirigido



Resumen

- Los grafos son una herramienta poderosa para modelar relaciones entre objetos.
- Compuestos por vértices y aristas que pueden ser dirigidas o no, ponderadas o no.
- Las características principales incluyen adyacencia, conexión y ciclos.

Representación de Grafos

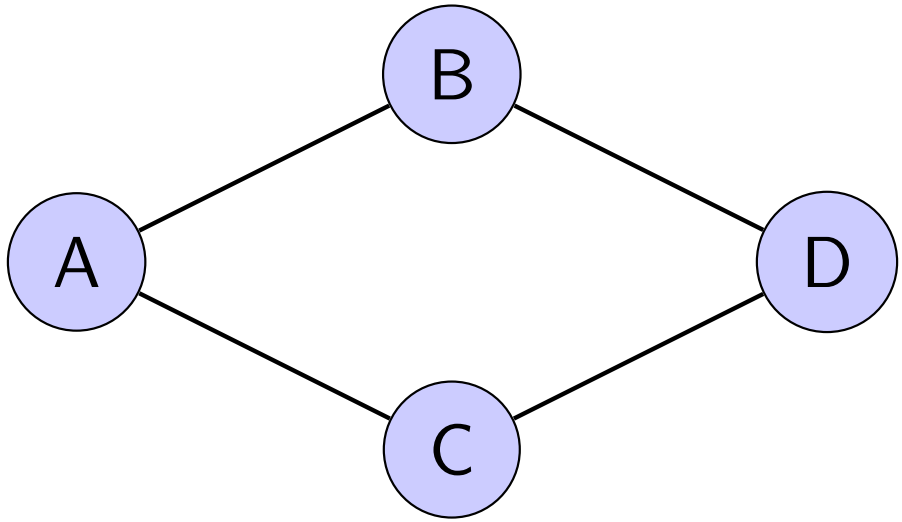
- Los grafos pueden representarse de varias formas en estructuras de datos.
- Las representaciones más comunes son:
  - **Matriz de adyacencia:** Utiliza una tabla bidimensional para registrar conexiones entre nodos.
  - **Lista de adyacencia:** Usa listas para registrar nodos conectados a cada vértice.
- Ambas tienen ventajas y desventajas dependiendo del tipo y tamaño del grafo.

Matriz de Adyacencia

- Es una tabla bidimensional de tamaño $n \times n$, donde n es el número de nodos.
- Cada celda $[i][j]$ indica si existe una arista entre los nodos i y j:
  - 1 si existe una conexión.
  - 0 si no existe conexión.
- Ideal para grafos densos (muchas conexiones).

Ejemplo de Matriz de Adyacencia

- Grafo ejemplo:



- Matriz de adyacencia:

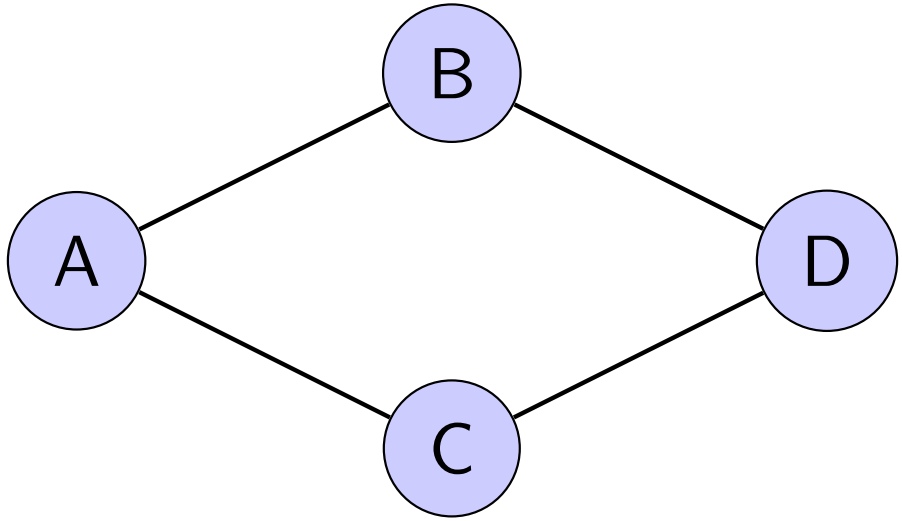
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Lista de Adyacencia

- Cada nodo tiene una lista que contiene los nodos a los que está conectado.
- Adecuado para grafos dispersos (pocas conexiones).
- Requiere menos memoria que la matriz de adyacencia para grafos grandes.

Ejemplo de Lista de Adyacencia

- Grafo ejemplo (mismo que antes):



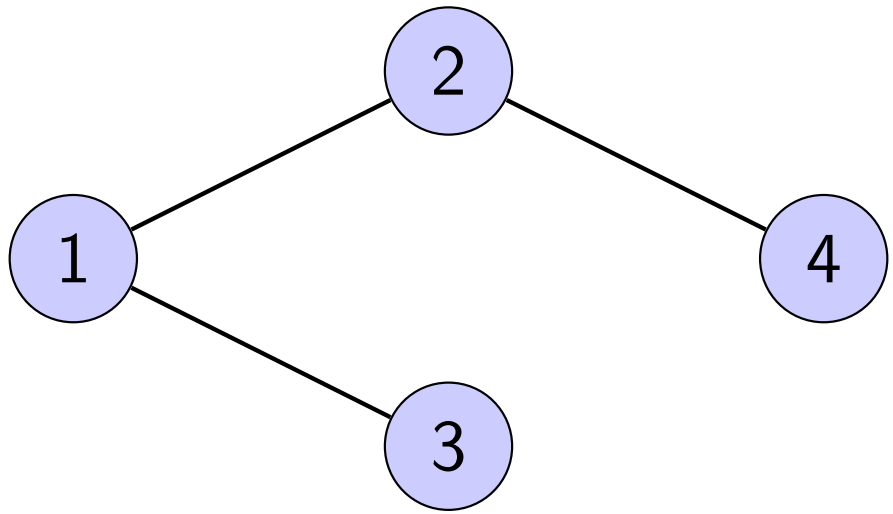
- Lista de adyacencia:
- `A -> B, C`
- `B -> A, D`
- `C -> A, D`
- `D -> B, C`

Comparación de Representaciones

- **Matriz de Adyacencia:**
  - Ventaja: Acceso rápido para verificar conexiones ($O(1)$).
  - Desventaja: Alto uso de memoria (n^2).
- **Lista de Adyacencia:**
  - Ventaja: Uso eficiente de memoria para grafos dispersos.
  - Desventaja: Acceso más lento para verificar conexiones ($O(d)$, donde d es el grado del nodo).
- Elección depende del problema y la naturaleza del grafo.

Ejercicios

- Representa el siguiente grafo como matriz y lista de adyacencia:



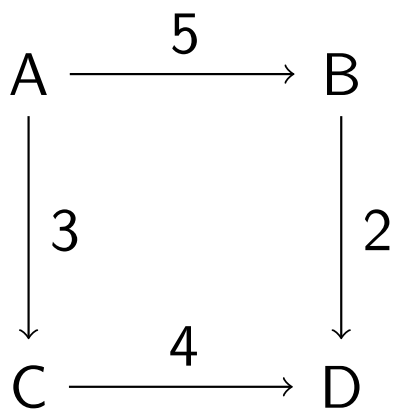
- Determina cuál representación es más eficiente para este grafo y justifica tu respuesta.

Grafos Ponderados I

Un grafo ponderado es un grafo en el que cada arista tiene un valor asociado, conocido como peso o costo. Este peso puede representar cualquier tipo de información, como distancias, costos de transporte, tiempos, etc.

- En un grafo ponderado, las aristas tienen un valor numérico asociado, generalmente representado como un número.
- Los pesos pueden ser positivos, negativos o incluso cero, dependiendo de la naturaleza del problema que se está modelando.

Ejemplo de un grafo ponderado:



Grafos Ponderados II

Aquí, las aristas tienen los siguientes pesos:

- De A a B el peso es 5,
- De A a C el peso es 3,
- De B a D el peso es 2,
- De C a D el peso es 4.

Representación de Grafos Ponderados

La representación de un grafo ponderado puede hacerse de las siguientes maneras:

- **Matriz de Adyacencia Ponderada**: Cada celda de la matriz contiene el peso de la arista entre los vértices correspondientes. Si no existe una arista entre los vértices, se coloca un valor especial (por ejemplo, infinito o cero).
- **Lista de Adyacencia Ponderada**: Cada vértice tiene una lista de adyacencia donde cada entrada contiene un par de valores: el vértice adyacente y el peso de la arista que lo conecta.

Ejemplo usando lista de adyacencia ponderada:

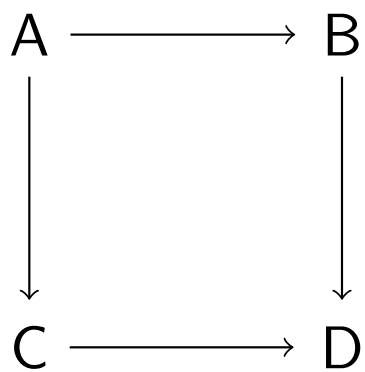
- $A : \{(B, 5), (C, 3)\}$
- $B : \{(D, 2)\}$
- $C : \{(D, 4)\}$
- $D : \{\}$

Grafos Dirigidos I

Un grafo dirigido (también llamado digrafo) es un tipo de grafo en el cual las aristas tienen una dirección, es decir, cada arista es un par ordenado de vértices (u, v), donde u es el vértice de origen y v es el vértice de destino.

- Los grafos dirigidos son utilizados para modelar relaciones unidireccionales, como en la representación de redes de transporte, flujos de información, y procesos secuenciales.
- Las aristas se representan con flechas para indicar la dirección.

Ejemplo de un grafo dirigido:



Grafos Dirigidos II

En este ejemplo, las flechas indican la dirección de las aristas entre los vértices. Por ejemplo, la arista de A a B indica que hay una relación dirigida de A a B, pero no necesariamente viceversa.

Representación de Grafos Dirigidos

Al igual que con los grafos no dirigidos, los grafos dirigidos se pueden representar usando una **matriz de adyacencia** o **lista de adyacencia**:

- **Matriz de Adyacencia Dirigida**: La matriz contiene un valor diferente de cero en la posición (i, j) si existe una arista dirigida desde el vértice i hacia el vértice j.
- **Lista de Adyacencia Dirigida**: En la lista, cada vértice mantiene una lista de otros vértices hacia los cuales tiene aristas dirigidas.

Ejemplo usando lista de adyacencia dirigida:

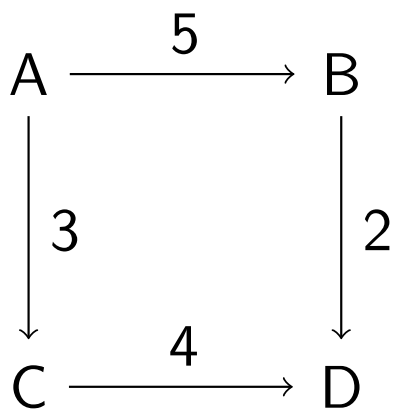
- $A : \{B, C\}$ (desde A hacia B y C)
- $B : \{D\}$ (desde B hacia D)
- $C : \{D\}$ (desde C hacia D)
- $D : \{\}$ (no hay aristas desde D)

Grafos Ponderados I

Un grafo ponderado es un grafo en el que cada arista tiene un valor asociado, conocido como peso o costo. Este peso puede representar cualquier tipo de información, como distancias, costos de transporte, tiempos, etc.

- En un grafo ponderado, las aristas tienen un valor numérico asociado, generalmente representado como un número.
- Los pesos pueden ser positivos, negativos o incluso cero, dependiendo de la naturaleza del problema que se está modelando.

Ejemplo de un grafo ponderado:



Grafos Ponderados II

Aquí, las aristas tienen los siguientes pesos:

- De A a B el peso es 5,
- De A a C el peso es 3,
- De B a D el peso es 2,
- De C a D el peso es 4.

Representación de Grafos Ponderados I

La representación de un grafo ponderado puede hacerse de las siguientes maneras:

- **Matriz de Adyacencia Ponderada**: Cada celda de la matriz contiene el peso de la arista entre los vértices correspondientes. Si no existe una arista entre los vértices, se coloca un valor especial (por ejemplo, infinito o cero).
- **Lista de Adyacencia Ponderada**: Cada vértice tiene una lista de adyacencia donde cada entrada contiene un par de valores: el vértice adyacente y el peso de la arista que lo conecta.

Ejemplo usando lista de adyacencia ponderada:

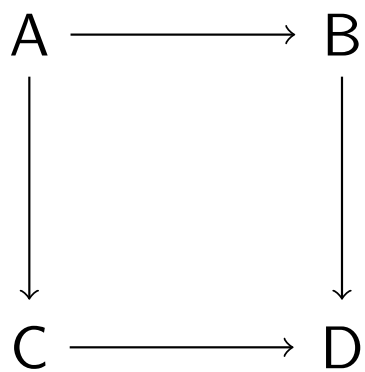
- $A : \{(B, 5), (C, 3)\}$
- $B : \{(D, 2)\}$
- $C : \{(D, 4)\}$
- $D : \{\}$

Grafos Dirigidos I

Un grafo dirigido (también llamado digrafo) es un tipo de grafo en el cual las aristas tienen una dirección, es decir, cada arista es un par ordenado de vértices (u, v), donde u es el vértice de origen y v es el vértice de destino.

- Los grafos dirigidos son utilizados para modelar relaciones unidireccionales, como en la representación de redes de transporte, flujos de información, y procesos secuenciales.
- Las aristas se representan con flechas para indicar la dirección.

Ejemplo de un grafo dirigido:



Grafos Dirigidos II

En este ejemplo, las flechas indican la dirección de las aristas entre los vértices. Por ejemplo, la arista de A a B indica que hay una relación dirigida de A a B, pero no necesariamente viceversa.

Representación de Grafos Dirigidos

Al igual que con los grafos no dirigidos, los grafos dirigidos se pueden representar usando una **matriz de adyacencia** o **lista de adyacencia**:

- **Matriz de Adyacencia Dirigida**: La matriz contiene un valor diferente de cero en la posición (i, j) si existe una arista dirigida desde el vértice i hacia el vértice j.
- **Lista de Adyacencia Dirigida**: En la lista, cada vértice mantiene una lista de otros vértices hacia los cuales tiene aristas dirigidas.

Ejemplo usando lista de adyacencia dirigida:

- $A : \{B, C\}$ (desde A hacia B y C)
- $B : \{D\}$ (desde B hacia D)
- $C : \{D\}$ (desde C hacia D)
- $D : \{\}$ (no hay aristas desde D)

Grafos Dirigidos Ponderados

Un **grafo dirigido ponderado** es un grafo en el que cada arista tiene una dirección y un peso asociado. - Las aristas conectan un vértice de origen a un vértice de destino. - El peso de una arista representa el costo o la distancia entre los vértices.

- Las aristas son dirigidas, es decir, tienen un punto de inicio y un punto de fin.
- El peso o costo de cada arista es representado por un valor numérico.
- Usualmente, este tipo de grafos se utilizan en algoritmos de optimización, como el algoritmo de Dijkstra, para encontrar caminos más cortos en redes.

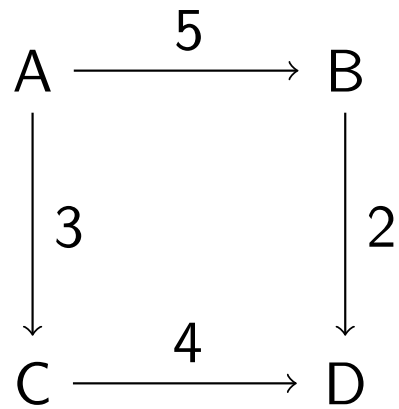
Representación de Grafos Dirigidos Ponderados

Un grafo dirigido ponderado se puede representar de las siguientes maneras:

1. **Lista de adyacencia ponderada**:
   - Cada vértice tiene una lista de sus vértices vecinos junto con los pesos de las aristas que lo conectan.
2. **Matriz de adyacencia ponderada**:
   - Es una matriz cuadrada donde el valor en la fila i y columna j representa el peso de la arista de v_i a v_j. Si no hay arista, el valor es 0 o infinito.

Ejemplo de Grafo Dirigido Ponderado

Consideremos el siguiente grafo dirigido ponderado:



En este grafo: - La arista $A \to B$ tiene un peso de 5. - La arista $A \to C$ tiene un peso de 3. - La arista $B \to D$ tiene un peso de 2. - La arista $C \to D$ tiene un peso de 4.

Lista de Adyacencia Ponderada

La lista de adyacencia ponderada para el grafo anterior sería:

- $A : \{(B, 5), (C, 3)\}$
- $B : \{(D, 2)\}$
- $C : \{(D, 4)\}$
- $D : \{\}$

Cada vértice está asociado con una lista de sus vértices vecinos y los pesos de las aristas que los conectan.

Matriz de Adyacencia Ponderada

La matriz de adyacencia ponderada para el mismo grafo sería:

$$\begin{bmatrix} 0 & 5 & 3 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

En esta matriz: - El valor 5 en la fila 1, columna 2 representa el peso de la arista $A \to B$. - El valor 3 en la fila 1, columna 3 representa el peso de la arista $A \to C$. - El valor 2 en la fila 2, columna 4 representa el peso de la arista $B \to D$. - El valor 4 en la fila 3, columna 4 representa el peso de la arista $C \to D$.

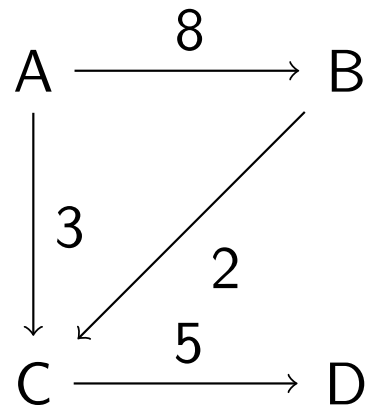
Operaciones en Grafos Dirigidos Ponderados

En un grafo dirigido ponderado, las operaciones más comunes son:

- **Búsqueda de caminos más cortos**: Utilizando algoritmos como Dijkstra o el algoritmo de Bellman-Ford.
- **Actualización de pesos**: Cambiar el peso de una arista en el grafo.
- **Detección de ciclos**: Verificar si el grafo contiene ciclos, lo que podría ser relevante en la optimización de rutas o en la programación de redes.

Ejercicio de Representación

Considera el siguiente grafo dirigido ponderado:



1. Escribe la lista de adyacencia ponderada de este grafo. 2. Escribe la matriz de adyacencia ponderada. 3. Si deseas cambiar el peso de la arista $A \rightarrow C$ de 3 a 7, ¿cómo afectaría a la representación?

Introducción

Objetivo: Explorar dos estrategias fundamentales para recorrer grafos:

- **Búsqueda en Anchura (Breadth-First Search, BFS)**.
- **Búsqueda en Profundidad (Depth-First Search, DFS)**.

Estas técnicas se utilizan ampliamente en problemas como:

- Encontrar rutas más cortas.
- Detectar ciclos en grafos.
- Resolver laberintos o problemas de conexión.

Búsqueda en Anchura (BFS)

Definición: Es un algoritmo que explora los nodos de un grafo por niveles:

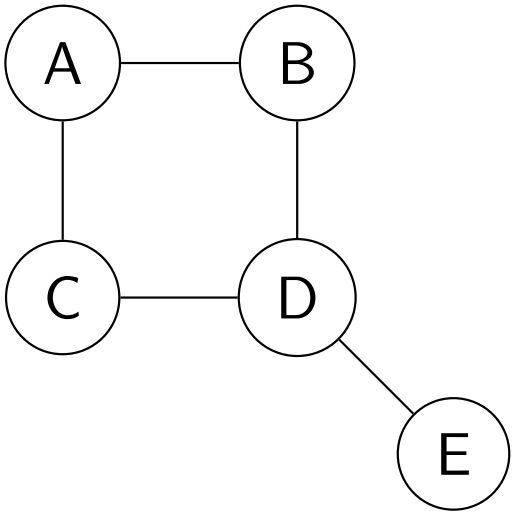
- Parte de un nodo inicial y visita todos sus vecinos primero.
- Luego avanza al siguiente nivel, visitando los vecinos de los nodos ya explorados.

Características:

- Utiliza una **cola** para mantener el orden de visita.
- Garantiza encontrar la ruta más corta en grafos no ponderados.
- Complejidad: $O(V + E)$, donde V son los nodos y E las aristas.

Ejemplo: Búsqueda en Anchura (BFS)

Grafo Ejemplo: Supongamos el siguiente grafo no dirigido:



Orden de visita: Iniciando desde A:

- Nivel 0: A.
- Nivel 1: B, C.
- Nivel 2: D.
- Nivel 3: E.

Algoritmo BFS (Pseudocódigo) I

Algoritmo: Búsqueda en Anchura (BFS)

```
Entrada: Grafo G representado como matriz de adyace
         Nodo inicial inicio
Salida: Orden de visita de los nodos en BFS

1. Crear una cola vacia Q
2. Crear un arreglo visitados[]
e inicializarlo en 0
3. Encolar el nodo inicial en Q
4. Marcar visitados[inicio] = 1

5. Mientras Q no este vacia:
    6. Desencolar un nodo actual de Q
    7. Imprimir nodo actual
    8. Para cada vecino i de actual:
        9. Si G[actual][i] == 1 y visitados[i] == 0:
```

Algoritmo BFS (Pseudocódigo) II

```
10. Encolar i en Q
11. Marcar visitados[i] = 1
```

Notas: - La matriz de adyacencia $G[i][j]$ es 1 si hay una arista entre i y j, 0 en caso contrario. - El orden de visita depende de los vecinos explorados.

Búsqueda en Profundidad (DFS)

Definición: Es un algoritmo que explora los nodos de un grafo tan profundo como sea posible antes de retroceder:

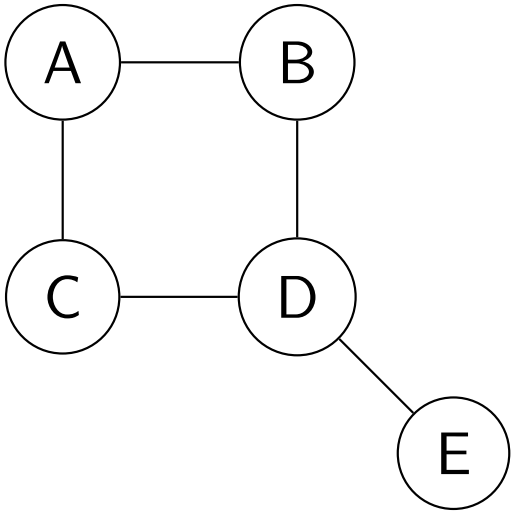
- Parte de un nodo inicial y sigue un camino hasta llegar al final.
- Luego retrocede y explora otros caminos.

Características:

- Utiliza una **pila** (o recursión) para manejar el orden de visita.
- No garantiza encontrar la ruta más corta.
- Complejidad: $O(V + E)$.

Ejemplo: Búsqueda en Profundidad (DFS)

Grafo Ejemplo: Supongamos el mismo grafo no dirigido:



Orden de visita: Iniciando desde A:

- Ruta inicial: $A \rightarrow B \rightarrow D \rightarrow E$.
- Retroceso y exploración restante: C.

Algoritmo DFS con Pila (Pseudocódigo) I

Algoritmo: Búsqueda en Profundidad con Pila (DFS)

```
Entrada: Grafo G como matriz de adyacencia,
         Nodo inicial inicio
Salida: Orden de visita de los nodos en DFS
1. Crear una pila vacia S
2. Crear un arreglo visitados[] = 0
3. Apilar el nodo inicial en S
4. Mientras S no estr vacia:
    5. Desapilar un nodo actual de S
    6. Si visitados[actual] == 0:
        7. Imprimir nodo actual
        8. Marcar visitados[actual] = 1
        9. Para cada vecino i de actual "for --":
            10. Si G[actual][i] == 1 y visitados[i] =
                11. Apilar i en S
```

Algoritmo DFS con Pila (Pseudocódigo) II

Notas: - La matriz de adyacencia $G[i][j]$ es 1 si hay una arista entre i y j, 0 en caso contrario. - Se utiliza orden inverso en los vecinos para mantener el orden lógico de recorrido.

Comparación: BFS vs DFS

Diferencias principales:

- BFS explora niveles por completo antes de pasar al siguiente.
- DFS se enfoca en un camino hasta llegar al final antes de retroceder.

Usos comunes:

- BFS: Rutas más cortas, problemas de conectividad.
- DFS: Detección de ciclos, recorrido completo.

Ejemplo de Grafo

Representación del grafo como matriz de adyacencia:

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Descripción del grafo:

- Nodos: $0, 1, 2, 3, 4$
- Aristas:
  - $0 \leftrightarrow 1$, $0 \leftrightarrow 2$
  - $1 \leftrightarrow 3$, $2 \leftrightarrow 3$
  - $3 \leftrightarrow 4$

Ejercicios:

1. Aplica BFS partiendo del nodo 0. ¿Cuál es el orden de visita?
2. Aplica DFS partiendo del nodo 0. ¿Cuál es el orden de visita?

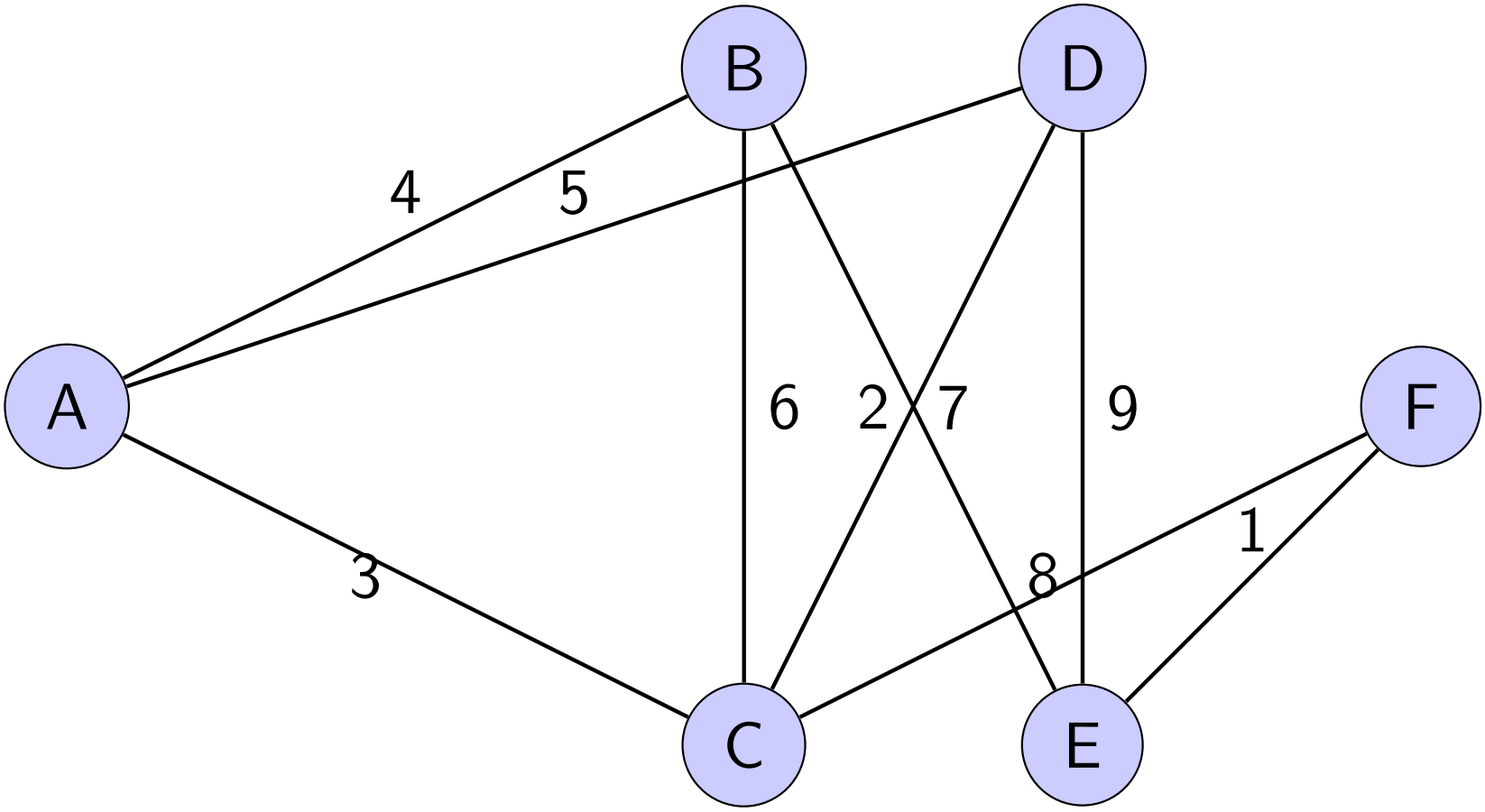
Árbol de Expansión Mínima

- Un grafo conexo y no dirigido ponderado tiene al menos un árbol de expansión.
- Un **Árbol de Expansión Mínima (MST)** es un árbol de expansión con el menor peso total de aristas.
- **Aplicaciones:** Redes de comunicación, diseño de circuitos, análisis de clusters, optimización de rutas, etc.

Definiciones Clave

- **Grafo Conexo:** Existe un camino entre cualquier par de vértices.
- **Árbol de Expansión:** Subgrafo que contiene todos los vértices del grafo original y es un árbol (sin ciclos).
- **Peso de un Árbol:** Suma de los pesos de sus aristas.

Ejemplo



Prim's Algorithm - Detailed Pseudocode I

```
# Algoritmo de Prim para Arbol de Expansion Minima (MST)

# Entrada: Un grafo G representado por una matriz de adyacencia mat
#          donde mat[i][j] es el peso de la arista entre los vertices i y j.
#          Si no existe una arista entre i y j, mat[i][j] =
# Salida: Un Arbol de Expansion Minima (MST), representado por el conjunto de aristas

Para cada vertice v en V:
    key[v] = infinito          # Inicializamos las claves
    parent[v] = NULL           # Inicializamos los padres de los vertices
    visitado[v] = Falso        # Marcamos todos los vertices como no visitados

key[s] = 0                    # Establecemos la clave del vertice inicial s a 0

Mientras haya vertices no visitados:
    u = Buscar el vertice con la clave minima que no haya sido visitado
    Marcar a u como visitado

    Para cada vertice v de V:
        Si v no esta visitado y mat[u][v] < key[v]:  # Si el vertice v tiene un costo ma
            key[v] = mat[u][v]      # Actualizamos la clave de v
            parent[v] = u           # El vertice u es el nuevo padre de v

# El MST se forma con las aristas entre cada vertice y su parent
```

Definición del Problema

Dado un grafo con 4 vértices (A, B, C, D) y la siguiente matriz de adyacencia que representa los pesos de las aristas:

$$\text{mat} = \begin{pmatrix} 0 & 2 & 3 & \infty \\ 2 & 0 & 4 & 5 \\ 3 & 4 & 0 & 6 \\ \infty & 5 & 6 & 0 \end{pmatrix}$$

Esta matriz muestra lo siguiente:

- A está conectado con B con peso 2, con C con peso 3, y no tiene conexión directa con D.
- B está conectado con A con peso 2, con C con peso 4, y con D con peso 5.
- C está conectado con A con peso 3, con B con peso 4, y con D con peso 6.
- D está conectado con B con peso 5, con C con peso 6, y no tiene conexión directa con A.

Paso 1: Inicialización

Comenzamos con el vértice **A**. Los arreglos de claves y padres iniciales son los siguientes:

$$\text{key} = \begin{pmatrix} 0 & \infty & \infty & \infty \end{pmatrix}, \quad \text{parent} = \begin{pmatrix} NULL & NULL & NULL & NULL \end{pmatrix}$$

Insertamos todos los vértices en la cola de prioridad con sus valores correspondientes de clave. Inicialmente, el vértice **A** tiene un valor de clave de 0, mientras que los demás vértices tienen clave infinita.

Paso 2: Selección del vértice A

Extraemos el vértice **A** de la cola de prioridad (tiene el valor de clave más pequeño, 0) y lo añadimos al MST. Luego, actualizamos las claves de los vértices adyacentes de **A**:

$$\text{key}[B] = \min(\infty, 2) = 2, \quad \text{parent}[B] = A$$

$$\text{key}[C] = \min(\infty, 3) = 3, \quad \text{parent}[C] = A$$

$$\text{key}[D] = \infty \quad \text{(no hay conexión directa con A)}$$

Después de extraer **A**, ya no estará en la cola de prioridad. La cola de prioridad ahora es:

$$\text{Q} = \{B, C, D\}$$

Paso 3: Selección del vértice B

Extraemos el vértice **B** de la cola de prioridad (tiene la clave más pequeña, 2). Lo añadimos al MST y actualizamos las claves de sus vértices adyacentes:

$$\text{key}[C] = \min(3, 4) = 3, \quad \text{parent}[C] = A$$

$$\text{key}[D] = \min(\infty, 5) = 5, \quad \text{parent}[D] = B$$

Después de extraer **B**, la cola de prioridad ahora es:

$$Q = \{C, D\}$$

Paso 4: Selección del vértice C

Extraemos el vértice **C** de la cola de prioridad (tiene la clave más pequeña, 3). Lo añadimos al MST y actualizamos las claves de sus vértices adyacentes:

$$\text{key}[D] = \min(5, 6) = 5, \quad \text{parent}[D] = B$$

Después de extraer **C**, la cola de prioridad ahora es:

$$Q = \{D\}$$

Paso 5: Selección del vértice D

Finalmente, extraemos el vértice **D** de la cola de prioridad (que tiene la clave más pequeña, 5). Lo añadimos al MST. Ya no hay vértices para actualizar.
La cola de prioridad ahora está vacía, lo que indica que hemos terminado el algoritmo.

Resultado Final

El Árbol de Expansión Mínima (MST) está formado por las siguientes aristas:

$$\{(A, B), (A, C), (B, D)\}$$

El costo total del MST es:

$$2 + 3 + 5 = 10$$

Hemos conectado todos los vértices con el costo mínimo total.