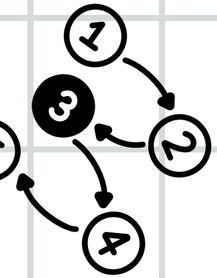
UNAM - FACULTAD DE CIENCIAS
LICENCIATURA EN MATEMÁTICAS APLICADAS

DFS (DEPTH-FIRST SEARCH)
BÚSQUEDA EN PROFUNDIDAD



MATERIA: INTRODUCCIÓN A LAS MATEMÁTICAS DISCRETAS

2026-I

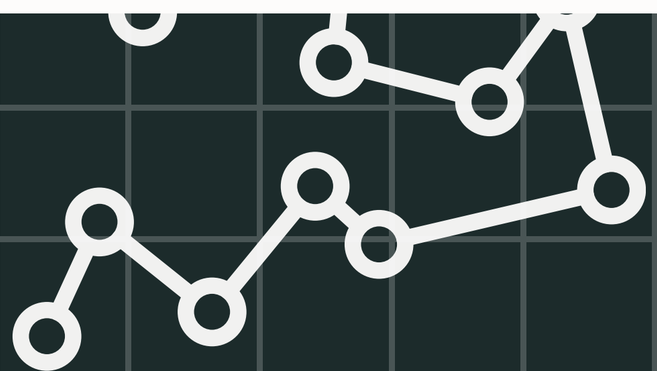
Alumno: Hector Silva Hernández
Profesor: Dr. Germán Benítez Bobadilla
Ayudantes: Montserrat Arias Basilio - Laura Daniela Zamudio Alcantar

Objetivos de aprendizaje

- Comprender la búsqueda en profundidad (DFS) como algoritmo de recorrido en gráficas, su relación con árboles y su papel dentro de la teoría de gráficas.
- Entender las características del algoritmo de DFS.
- Conocer la relevancia y algunas aplicaciones de DFS.

Agenda

1. Historia y motivación
2. Definiciones
3. DFS y para qué sirve
4. Algoritmo
5. Ejemplo paso a paso
6. Complejidad computacional
7. Aplicaciones
8. Conclusiones y Key take-aways
9. Referencias



1. Historia y motivación

- Muchas situaciones reales se pueden modelar con diagramas: redes de computadoras, mapas de transporte, redes sociales, etc...
- Estos dibujos se formalizan como un área de las matemáticas: Teoría de Gráficas/Grafos: nace con Euler en 1736 (problema de los puentes de Königsberg).
- Siglo XIX: surgen retos para formalizar formas efectivas para recorrer gráficas (1870 - Charles Pierre Trémaux - DFS - solución de laberintos)
- Siglo XX: el desarrollo de las matemáticas y las ciencias de la computación traen múltiples aplicaciones relacionadas.

2. Definiciones

Antes de analizar DFS debemos abordar algunas definiciones:

Gráfica (Bondy & Murty, 2008)

- Una **gráfica** G es un par ordenado (V(G), E(G)) que consiste de un conjunto de vértices (V(G)) y un conjunto de aristas (E(G)), y que junto a una función de incidencia ψ que asigna a cada arista $e \in E(G)$ un par no ordenado de vértices (posiblemente iguales):

$$\psi_G : E(G) \to \{\{u, v\} : u, v \in V(G)\}$$

- Donde si ψ(e)={*u*,*v*} decimos que la arista *e* une a *u* y *v*; *u* y *v* son los extremos de *e*.

Camino y Gráfica Conexa

- **Camino**: secuencia de vértices v_0 ,v_1, ..., v_k donde cada par consecutivo está unido por una arista.
- Una gráfica es **conexa** si: para cualquier par de vértices $u, v \in V$ existe un camino de *u* a *v*.
- En términos simples: la gráfica es conexa si "una sola pieza", no hay componentes aislados.

2. Definiciones (cont.)

Antes de analizar DFS debemos abordar algunas definiciones:

Ciclo, Gráfica simple y ponderación

- Un **ciclo** es un camino que cumple que $v_0=v_k$ (empieza y termina en el mismo vértice) y los vértices intermedios son todos distintos.
- Una **gráfica simple** es aquella no tiene lazos (ninguna arista del tipo $\{v,v\}$), y no tiene aristas múltiples entre el mismo par de vértices.
- Cada arista puede tener un valor de **ponderación/peso** diferente, en cuyo caso son **gráficas ponderadas**.

Árbol

- Un **árbol** es una gráfica simple que cumple, de forma equivalente:
  - Es conexa y no tiene ciclos.
  - Tiene *n* vértices y exactamente *n-1* aristas.
  - Entre cualquier par de vértices hay un unico camino simple.
- Es una estructura mínima que mantiene todos los vértices conectados.
- Si quitamos una arista -> la gráfica se desconecta.

2. Definiciones (cont.)

Antes de analizar DFS debemos abordar algunas definiciones:

Árbol generador

- Sea G=(V,E) una gráfica conexa, un árbol generador (spanning tree) T de G:
- V(T)=V(G): contiene todos los vértices de G.
- T es un árbol: conexo y acíclico, con |V| - 1 aristas.
- Es un “esqueleto mínimo” de la gráfica original.
- Conecta todos los nodos usando el mínimo número de aristas.
- DFS produce naturalmente un árbol generador DFS cuando se aplica a una gráfica conexa.

Backtracking

- El **backtracking** (retroceso) es una técnica algorítmica de resolución de problemas que consiste en encontrar una solución de forma incremental probando diferentes opciones y deshaciéndolas si conducen a un callejón sin salida.
- En DFS, cuando un vértice ya no tiene vecinos no visitados el algoritmo “retrocede” al vértice desde el cual llegó.
- En términos simples: se deshace el último movimiento y se intenta otra rama pendiente.
- Laberintos: sigues un pasillo hasta que no hay salida, regreses al último cruce para probar otro camino.

2. Definiciones (cont.)

Antes de analizar DFS debemos abordar algunas definiciones:

Pila

- Una pila es una estructura de datos LIFO (Last In, First Out): El último elemento que entra es el primero en salir.
- Operaciones básicas:
  - push(x): insertar elemento en la cima.
  - pop(): retirar el elemento de la cima.
- En DFS: la pila puede ser explícita (estructura de datos) o implícita (recursión).
- El vértice en la cima de la pila es el vértice actual que se está explorando.

Profundidad de un vértice

- Al aplicar DFS a un grafo, obtenemos un árbol DFS (si el grafo es conexo) o un bosque.
- Elegimos una raíz del árbol DFS y la profundidad de un vértice v en el árbol DFS es el número de aristas en el camino que va desde la raíz hasta v.
- Profundidad(raíz) = 0.
- Si v es hijo de u en el árbol DFS: $prof(v)=prof(u)+1$.
- La profundidad en el árbol DFS no necesariamente coincide con la distancia mínima en el grafo original, DFS no está diseñado para encontrar caminos más cortos, sino para recorrer la gráfica en profundidad.

3. DFS y para qué sirve

- **DFS** (Depth-First Search) = Es un **algoritmo** de búsqueda en profundidad.
- **Estrategia**:
  - desde un vértice inicial, avanzar siempre que se pueda a un vecino no visitado,
  - solo cuando se agotan los vecinos, se hace backtracking.
- Se construye un árbol DFS:
  - vértices = los nodos alcanzables,
  - aristas = las que se usan para entrar por primera vez a cada vértice.
- **Propósito principal**: recorrer sistemáticamente una gráfica y conocer su estructura interna.
- **Usos clásicos**: Verificar si una gráfica es conexa, encontrar componentes conexas, detectar ciclos, puentes y puntos de articulación.
- **En gráficas dirigidas**: componentes fuertemente conexas, orden topológico.

4. Algoritmo

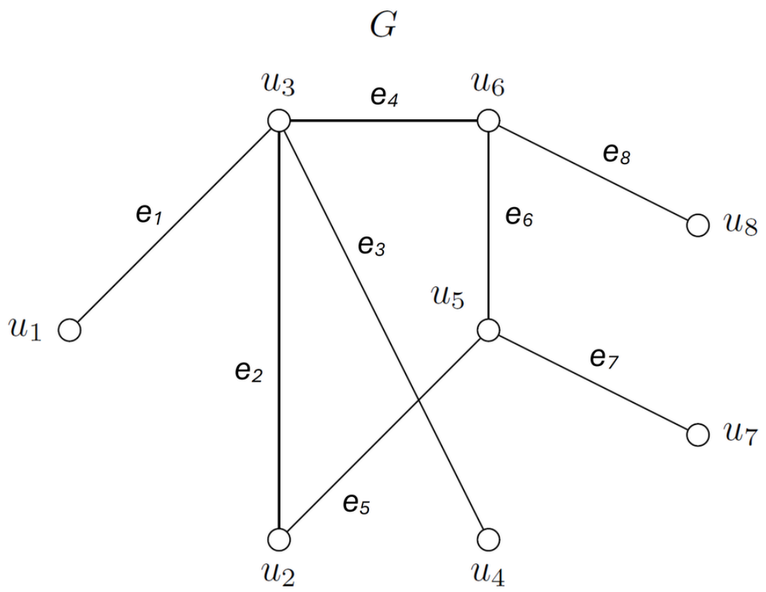
Definamos ahora el algoritmo DFS

Algoritmo

1. Comenzamos el recorrido en el vértice raíz r.
2. Marcamos r gris, le asignamos prof(r) = 0 y lo metemos en la pila: estamos parados en r.
3. Mientras la pila no esté vacía: miramos la cima de la pila: es el vértice donde estamos actualmente, u.
4. Buscamos vecinos no visitados (blancos) de u.
5. Si existe un vecino no visitado v: caminamos por la arista $\{u,v\}$.
6. Marcamos v gris, ponemos prof(v) = prof(u) + 1.
7. Añadimos $\{u,v\}$ al árbol DFS.
8. Hacemos push de v en la pila: ahora estamos en v.
9. Si el vértice en el que estamos ya no tiene vecinos blancos: no podemos seguir avanzando desde, lo marcamos en negro y hacemos pop: retrocedemos al vértice anterior (backtracking).
10. Repetimos: avanzamos a vecinos blancos cuando los haya, retrocedemos cuando ya no haya.
11. Cuando la pila se vacía: hemos recorrido todo lo alcanzable desde r, las aristas guardadas forman el árbol DFS.

5. Ejemplo paso a paso

Nuestra gráfica



- G: gráfica **simple no dirigida y no ponderada y conexa**.
- Vértices: $V(G) = \{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8\}.$
- Aristas: $E(G) = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}.$
- Función de incidencia:

$$\psi_G : E(G) \to \{\{u, v\} : u, v \in V(G)\}:$$

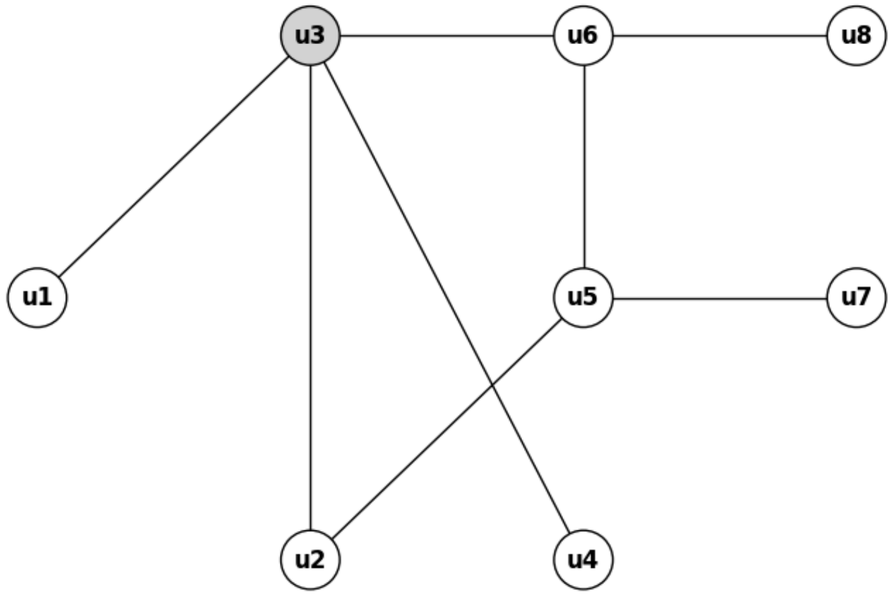
$$\begin{aligned} \psi_G(e_1) &= \{u_1, u_3\}, & \psi_G(e_2) &= \{u_2, u_3\}, \\ \psi_G(e_3) &= \{u_3, u_4\}, & \psi_G(e_4) &= \{u_3, u_6\}, \\ \psi_G(e_5) &= \{u_2, u_5\}, & \psi_G(e_6) &= \{u_5, u_6\}, \\ \psi_G(e_7) &= \{u_5, u_7\}, & \psi_G(e_8) &= \{u_6, u_8\}. \end{aligned}$$

- Empezaremos el DFS en u_3

5. Ejemplo paso a paso (cont.)

Paso 1

Grafo original
Paso 1



Descripción del paso

Descripción: Inicializamos DFS en u3. Definimos prof(u3) = 0 (raíz).

En este paso no se añadió ninguna arista al árbol DFS.

Cálculo de profundidad en este paso:
prof(u3) = 0

Pila (cima al final):
u3

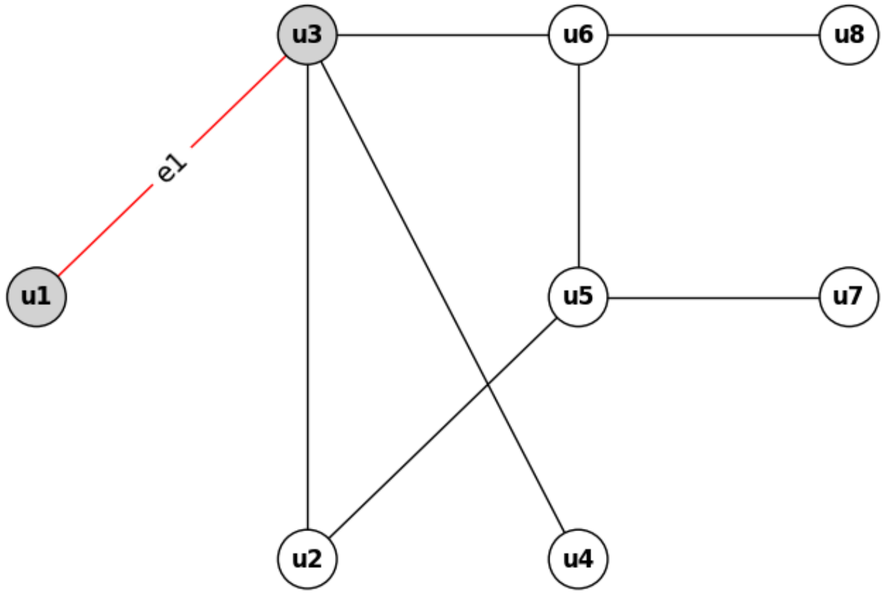
Profundidades actuales:
u1: –
u2: –
* u3: 0
u4: –
u5: –
u6: –
u7: –
u8: –

Árbol DFS
(construido hasta ahora)

5. Ejemplo paso a paso (cont.)

Paso 2

Grafo original
Paso 2



Descripción del paso

Descripción: Visitamos u1 desde u3. prof(u1) = prof(u3) + 1 = 0 + 1 = 1. Añadimos la arista e1 = ('u1', 'u3') al árbol DFS.

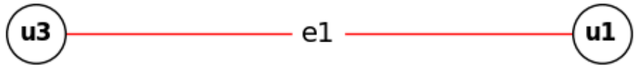
Arista añadida al árbol DFS en este paso: e1 = ('u1', 'u3')

Cálculo de profundidad en este paso:
prof(u1) = prof(u3) + 1 = 0 + 1 = 1

Pila (cima al final):
u3 → u1

Profundidades actuales:
* u1: 1
u2: –
u3: 0
u4: –
u5: –
u6: –
u7: –
u8: –

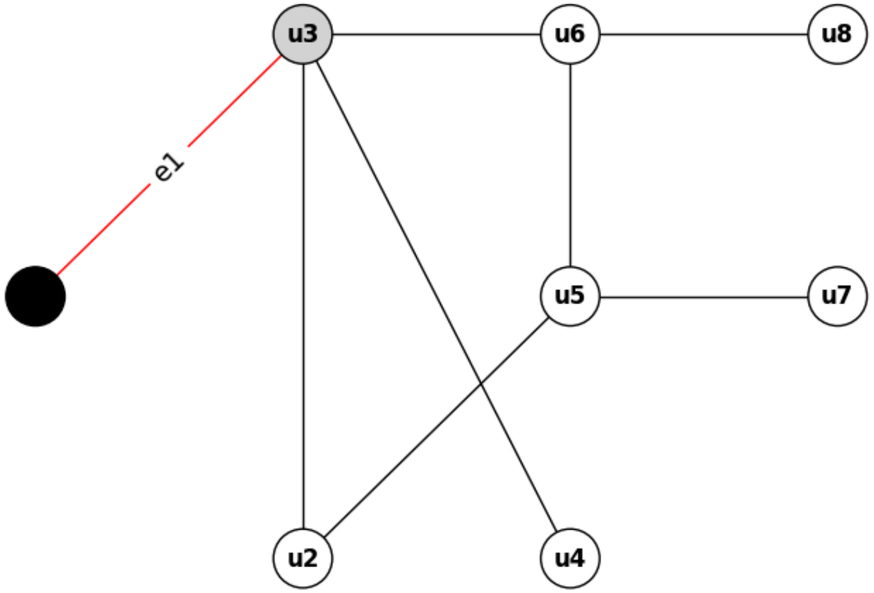
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Paso 3

Grafo original
Paso 3



Descripción del paso

Descripción: No quedan vecinos blancos de u1. Marcamos u1 como negro y hacemos backtracking.

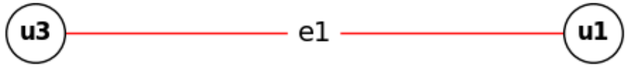
En este paso no se añadió ninguna arista al árbol DFS.

No se actualizó ninguna profundidad en este paso.

Pila (cima al final):
u3

Profundidades actuales:
u1: 1
u2: –
u3: 0
u4: –
u5: –
u6: –
u7: –
u8: –

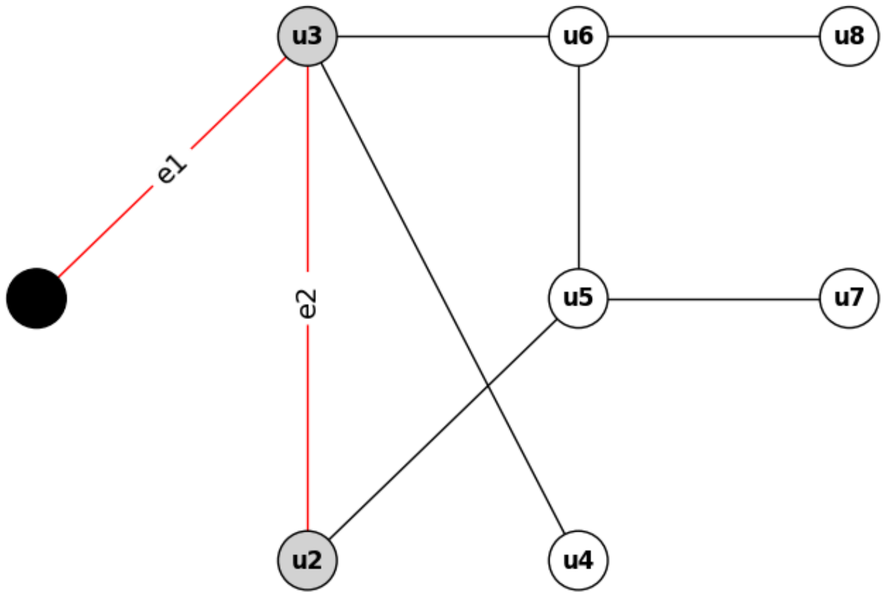
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Paso 4

Grafo original
Paso 4



Descripción del paso

Descripción: Visitamos u2 desde u3. prof(u2) = prof(u3) + 1 = 0 + 1 = 1. Añadimos la arista e2 = ('u2', 'u3') al árbol DFS.

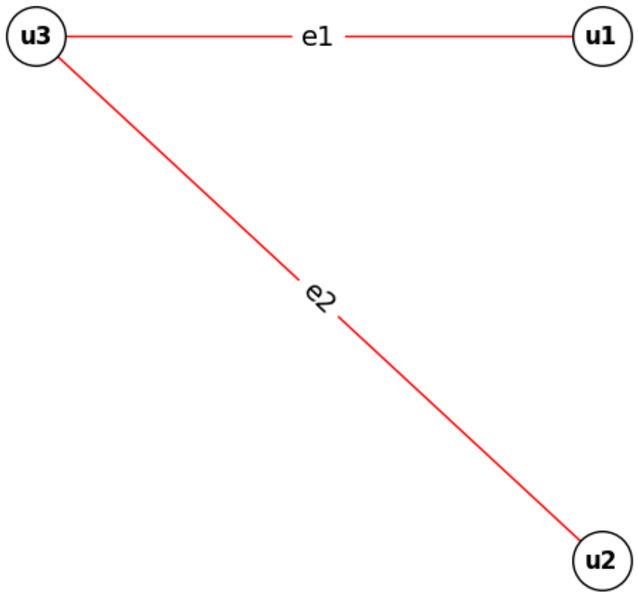
Arista añadida al árbol DFS en este paso: e2 = ('u2', 'u3')

Cálculo de profundidad en este paso:
prof(u2) = prof(u3) + 1 = 0 + 1 = 1

Pila (cima al final):
u3 → u2

Profundidades actuales:
u1: 1
* u2: 1
u3: 0
u4: –
u5: –
u6: –
u7: –
u8: –

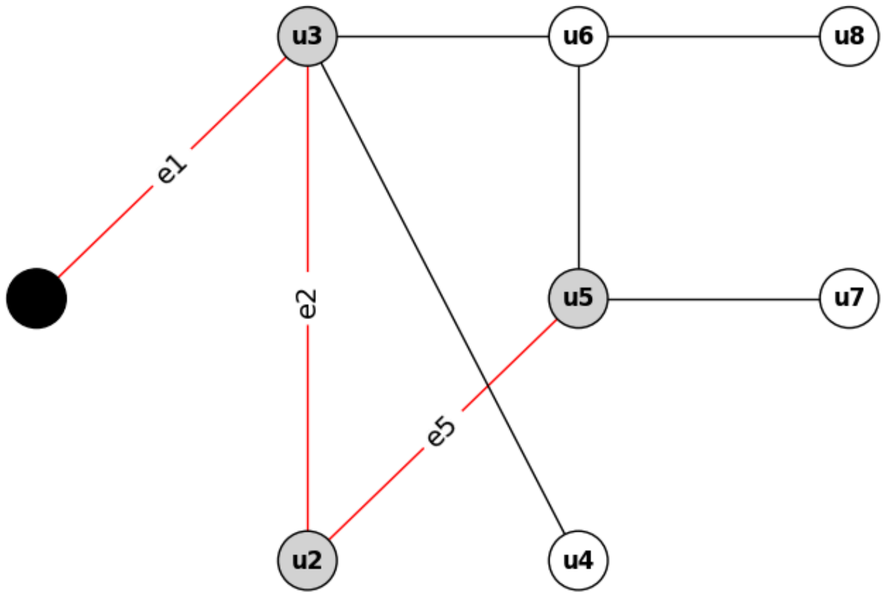
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Paso 5

Grafo original
Paso 5



Descripción del paso

Descripción: Visitamos u5 desde u2. prof(u5) = prof(u2) + 1 = 1 + 1 = 2. Añadimos la arista e5 = ('u2', 'u5') al árbol DFS.

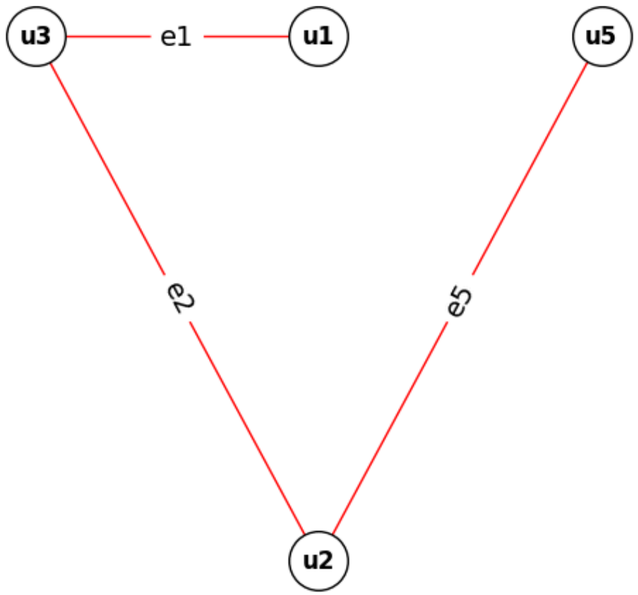
Arista añadida al árbol DFS en este paso: e5 = ('u2', 'u5')

Cálculo de profundidad en este paso:
prof(u5) = prof(u2) + 1 = 1 + 1 = 2

Pila (cima al final):
u3 → u2 → u5

Profundidades actuales:
u1: 1
u2: 1
u3: 0
u4: –
* u5: 2
u6: –
u7: –
u8: –

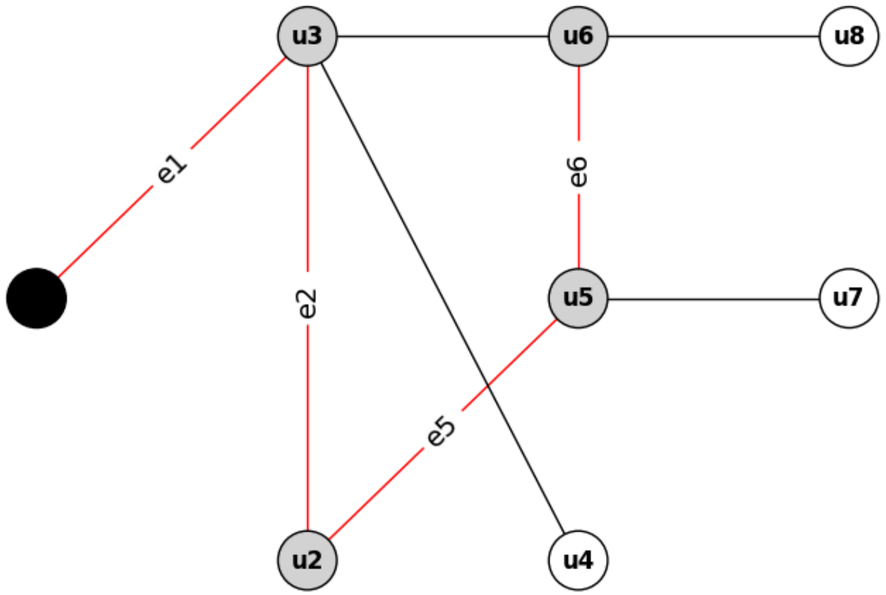
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Paso 6

Grafo original
Paso 6



Descripción del paso

Descripción: Visitamos u6 desde u5. prof(u6) = prof(u5) + 1 = 2 + 1 = 3. Añadimos la arista e6 = ('u5', 'u6') al árbol DFS.

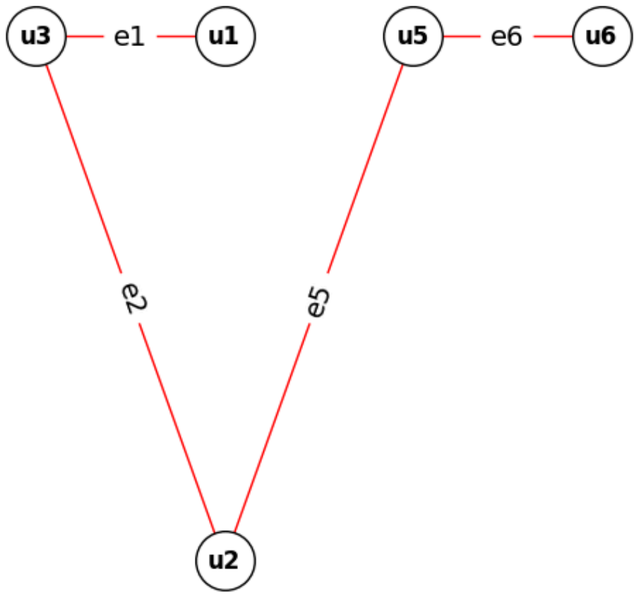
Arista añadida al árbol DFS en este paso: e6 = ('u5', 'u6')

Cálculo de profundidad en este paso:
prof(u6) = prof(u5) + 1 = 2 + 1 = 3

Pila (cima al final):
u3 → u2 → u5 → u6

Profundidades actuales:
u1: 1
u2: 1
u3: 0
u4: –
u5: 2
* u6: 3
u7: –
u8: –

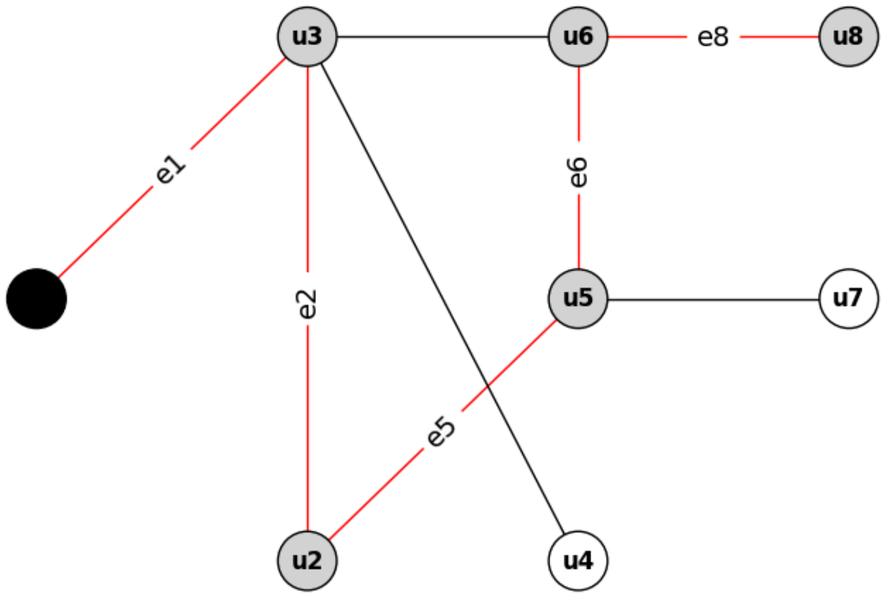
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Paso 7

Grafo original
Paso 7



Descripción del paso

Descripción: Visitamos u8 desde u6. prof(u8) = prof(u6) + 1 = 3 + 1 = 4. Añadimos la arista e8 = ('u6', 'u8') al árbol DFS.

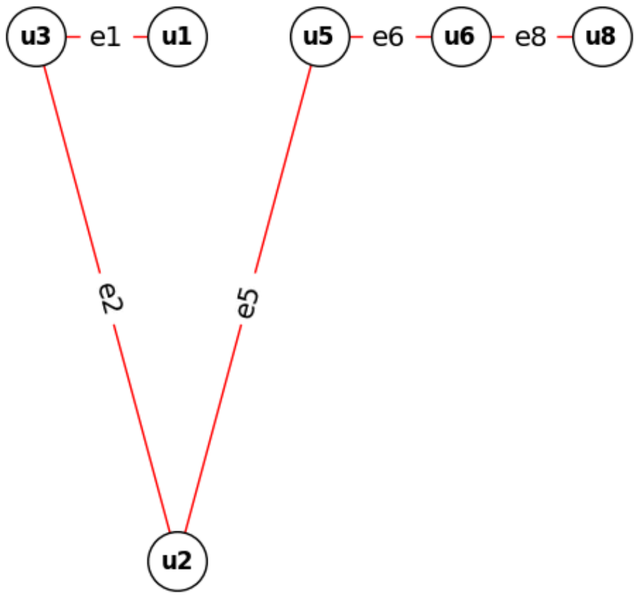
Arista añadida al árbol DFS en este paso: e8 = ('u6', 'u8')

Cálculo de profundidad en este paso:
prof(u8) = prof(u6) + 1 = 3 + 1 = 4

Pila (cima al final):
u3 → u2 → u5 → u6 → u8

Profundidades actuales:
u1: 1
u2: 1
u3: 0
u4: –
u5: 2
u6: 3
u7: –
* u8: 4

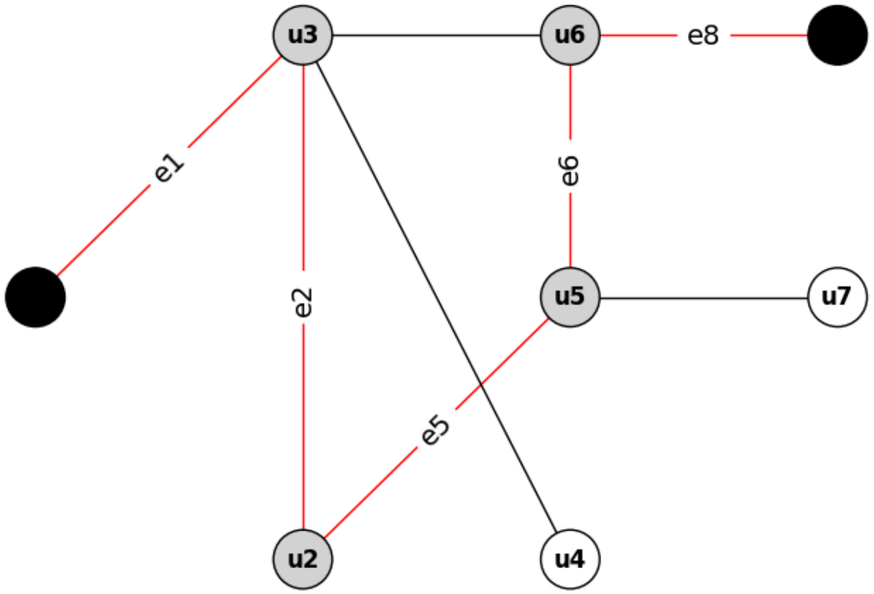
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Paso 8

Grafo original
Paso 8



Descripción del paso

Descripción: No quedan vecinos blancos de u8. Marcamos u8 como negro y hacemos backtracking.

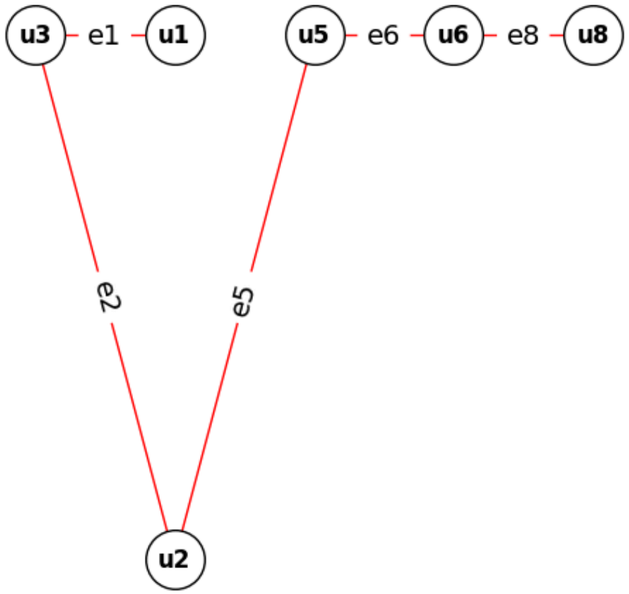
En este paso no se añadió ninguna arista al árbol DFS.

No se actualizó ninguna profundidad en este paso.

Pila (cima al final):
u3 → u2 → u5 → u6

Profundidades actuales:
u1: 1
u2: 1
u3: 0
u4: –
u5: 2
u6: 3
u7: –
u8: 4

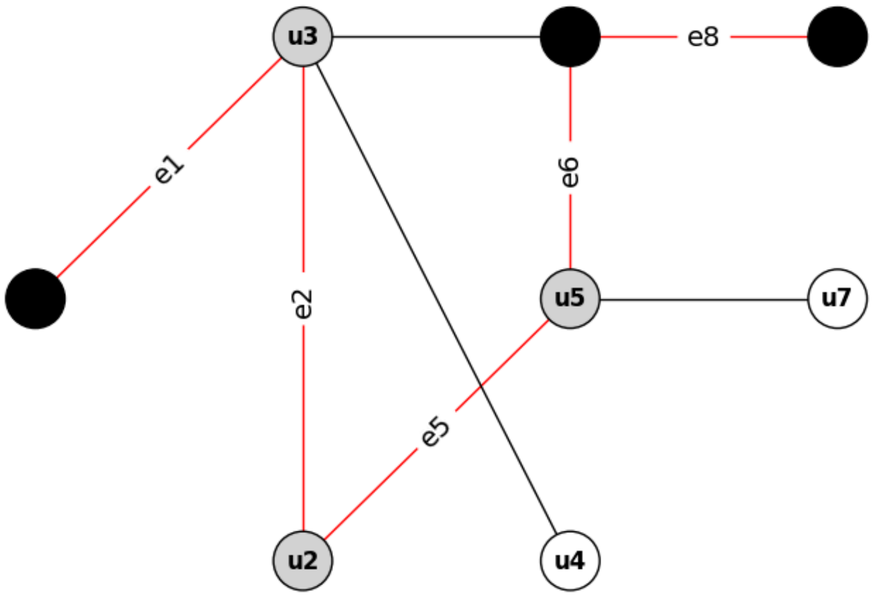
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Paso 9

Grafo original
Paso 9



Descripción del paso

Descripción: No quedan vecinos blancos de u6. Marcamos u6 como negro y hacemos backtracking.

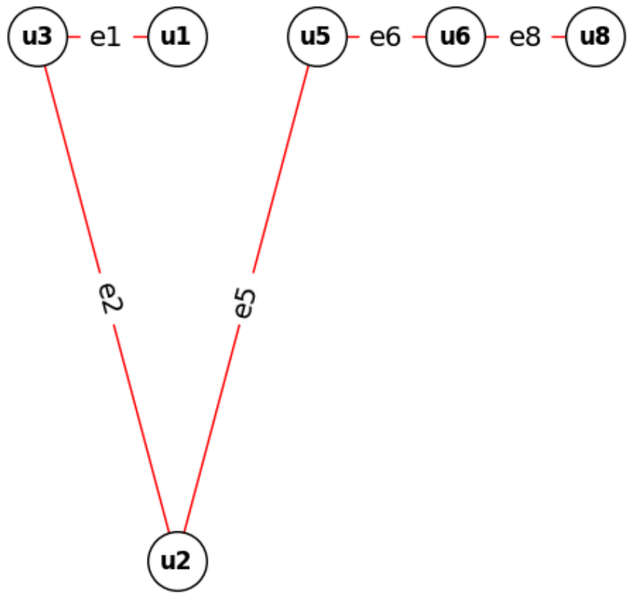
En este paso no se añadió ninguna arista al árbol DFS.

No se actualizó ninguna profundidad en este paso.

Pila (cima al final):
u3 → u2 → u5

Profundidades actuales:
u1: 1
u2: 1
u3: 0
u4: –
u5: 2
u6: 3
u7: –
u8: 4

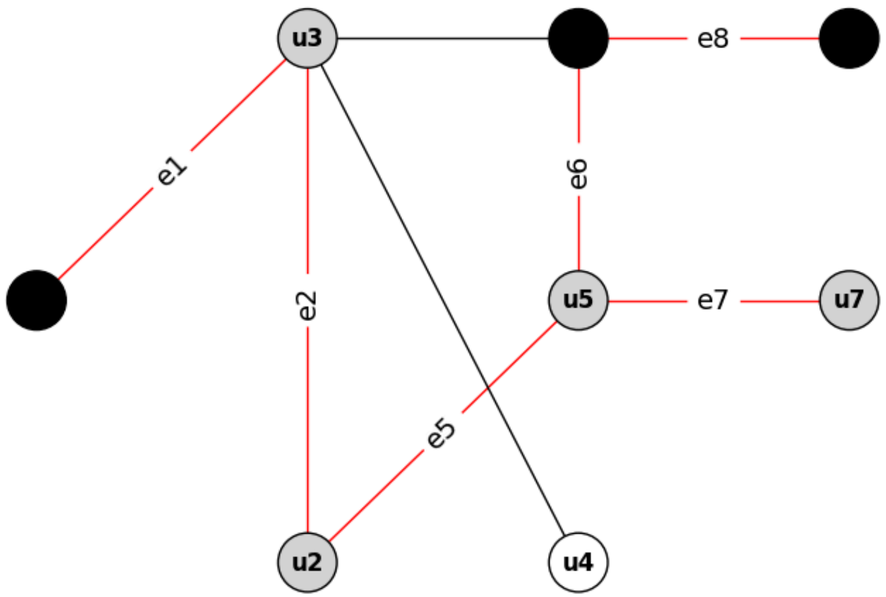
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Paso 10

Grafo original
Paso 10



Descripción del paso

Descripción: Visitamos u7 desde u5. prof(u7) = prof(u5) + 1 = 2 + 1 = 3. Añadimos la arista e7 = ('u5', 'u7') al árbol DFS.

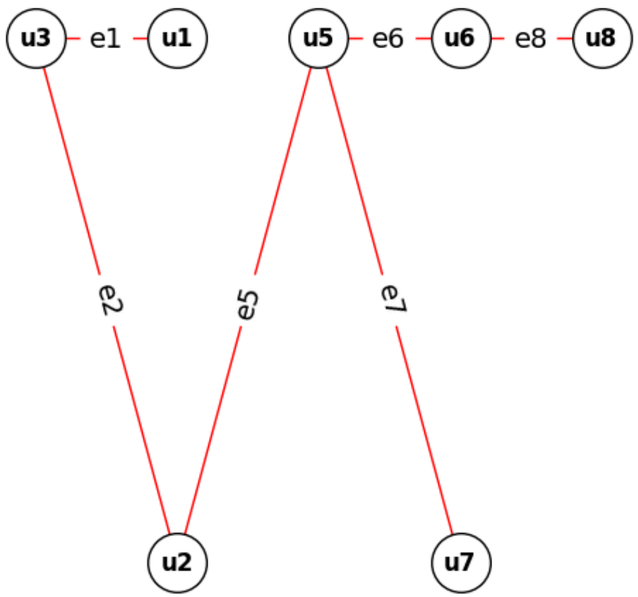
Arista añadida al árbol DFS en este paso: e7 = ('u5', 'u7')

Cálculo de profundidad en este paso:
prof(u7) = prof(u5) + 1 = 2 + 1 = 3

Pila (cima al final):
u3 → u2 → u5 → u7

Profundidades actuales:
u1: 1
u2: 1
u3: 0
u4: –
u5: 2
u6: 3
* u7: 3
u8: 4

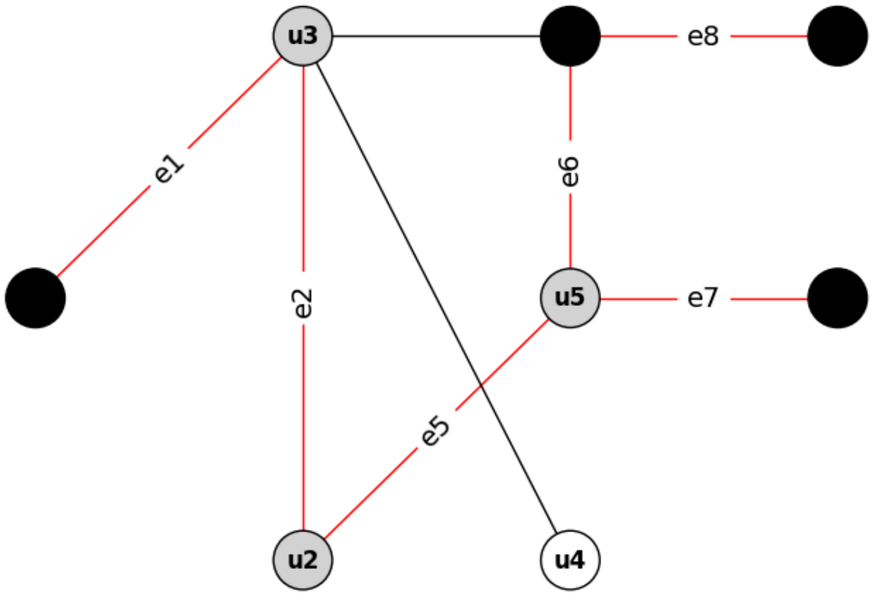
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Paso 11

Grafo original
Paso 11



Descripción del paso

Descripción: No quedan vecinos blancos de u7. Marcamos u7 como negro y hacemos backtracking.

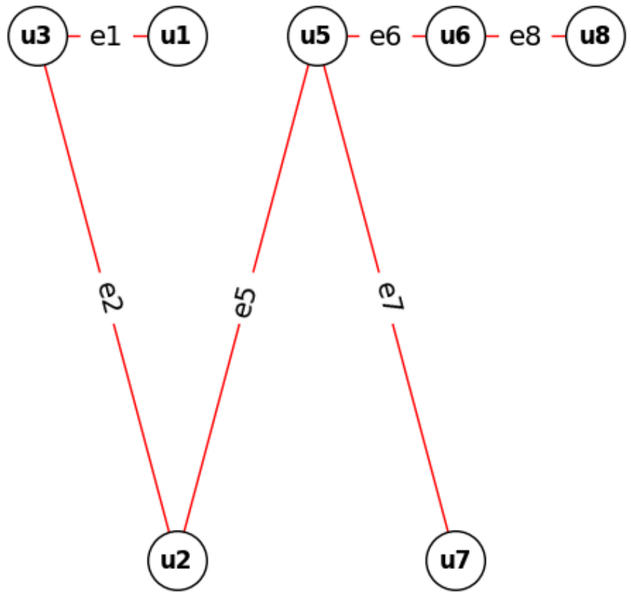
En este paso no se añadió ninguna arista al árbol DFS.

No se actualizó ninguna profundidad en este paso.

Pila (cima al final):
u3 → u2 → u5

Profundidades actuales:
u1: 1
u2: 1
u3: 0
u4: –
u5: 2
u6: 3
u7: 3
u8: 4

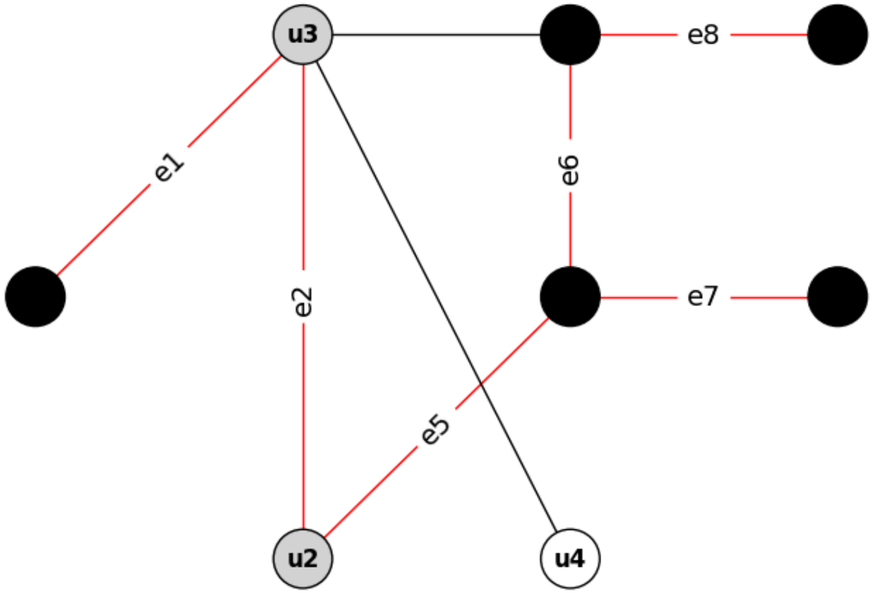
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Paso 12

Grafo original
Paso 12



Descripción del paso

Descripción: No quedan vecinos blancos de u5. Marcamos u5 como negro y hacemos backtracking.

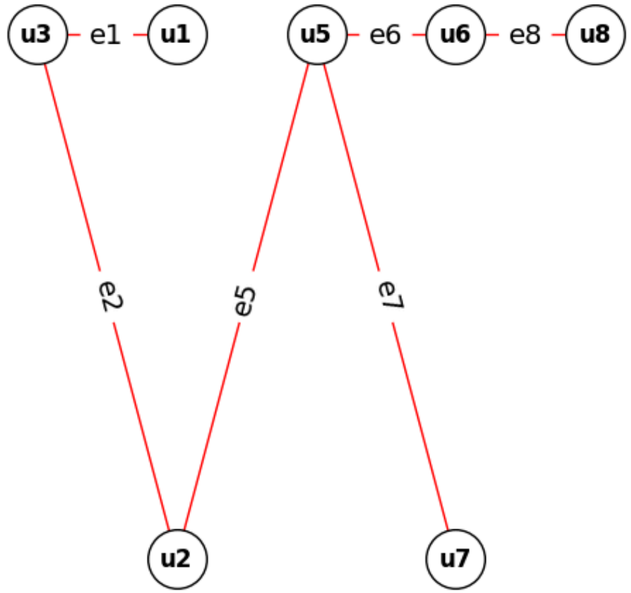
En este paso no se añadió ninguna arista al árbol DFS.

No se actualizó ninguna profundidad en este paso.

Pila (cima al final):
u3 → u2

Profundidades actuales:
u1: 1
u2: 1
u3: 0
u4: –
u5: 2
u6: 3
u7: 3
u8: 4

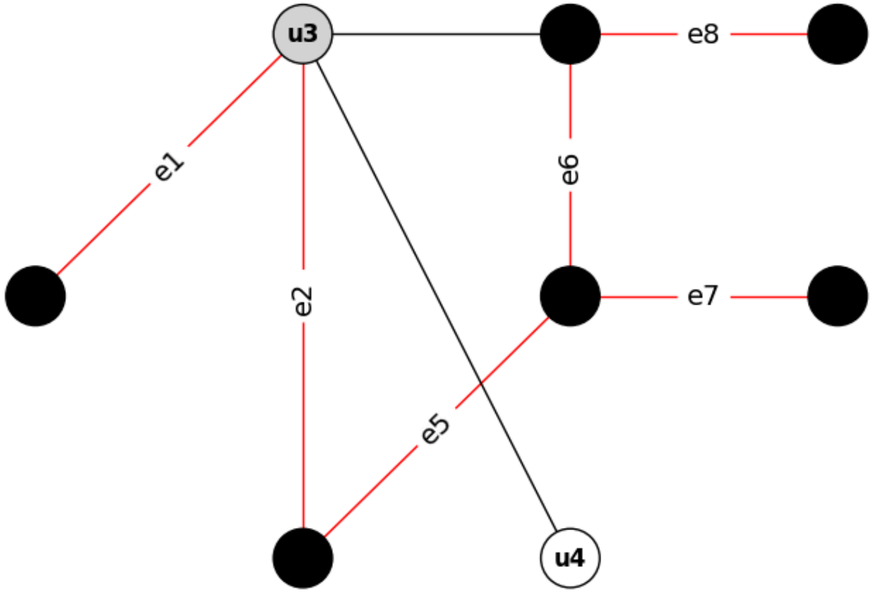
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Paso 13

Grafo original
Paso 13



Descripción del paso

Descripción: No quedan vecinos blancos de u2. Marcamos u2 como negro y hacemos backtracking.

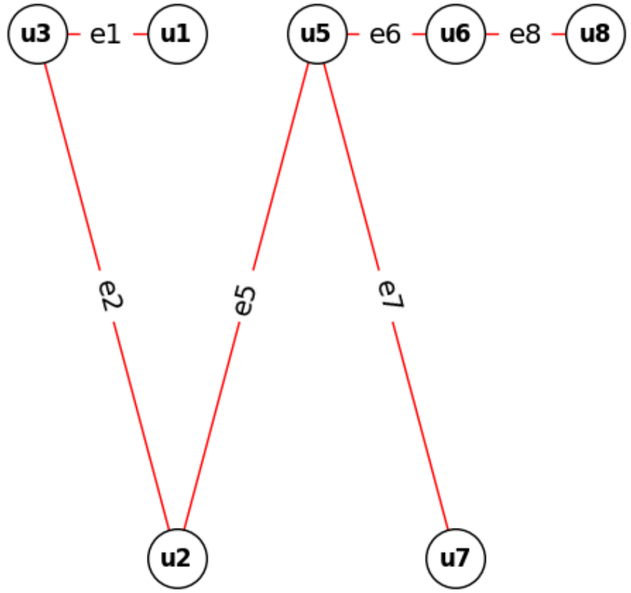
En este paso no se añadió ninguna arista al árbol DFS.

No se actualizó ninguna profundidad en este paso.

Pila (cima al final):
u3

Profundidades actuales:
u1: 1
u2: 1
u3: 0
u4: –
u5: 2
u6: 3
u7: 3
u8: 4

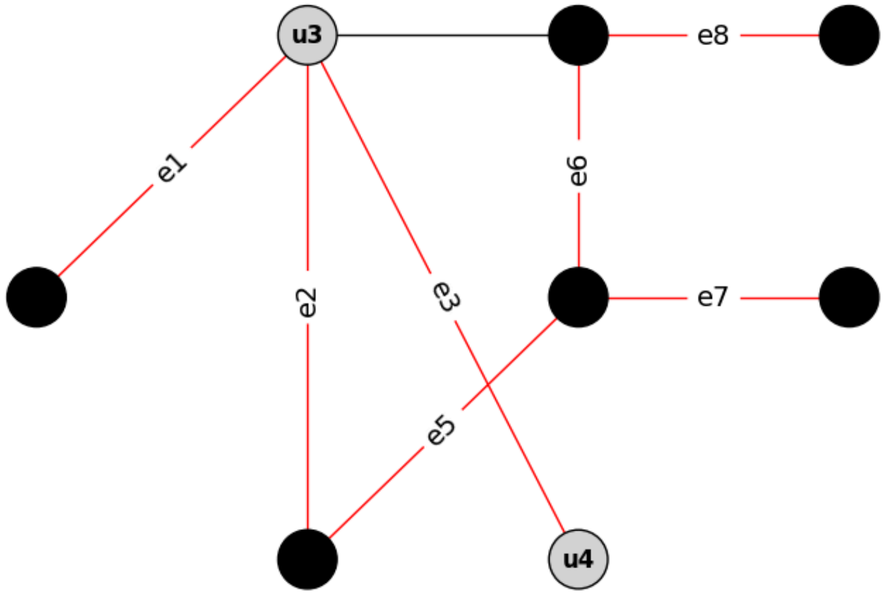
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Paso 14

Grafo original
Paso 14



Descripción del paso

Descripción: Visitamos u4 desde u3. prof(u4) = prof(u3) + 1 = 0 + 1 = 1. Añadimos la arista e3 = ('u3', 'u4') al árbol DFS.

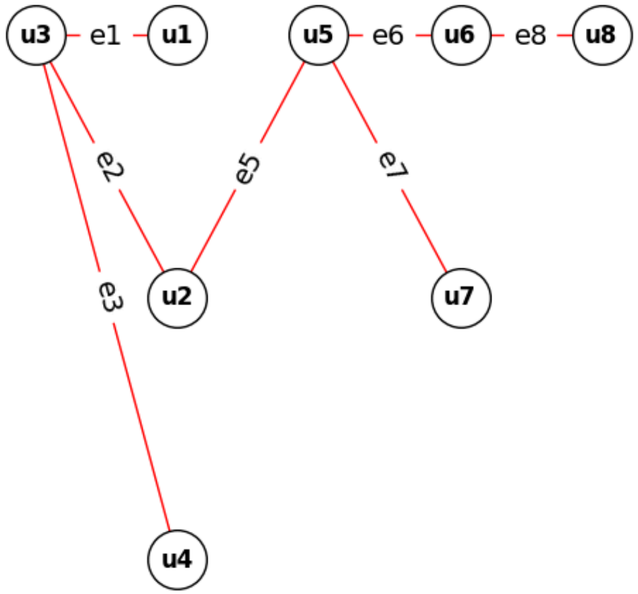
Arista añadida al árbol DFS en este paso: e3 = ('u3', 'u4')

Cálculo de profundidad en este paso:
prof(u4) = prof(u3) + 1 = 0 + 1 = 1

Pila (cima al final):
u3 → u4

Profundidades actuales:
u1: 1
u2: 1
u3: 0
* u4: 1
u5: 2
u6: 3
u7: 3
u8: 4

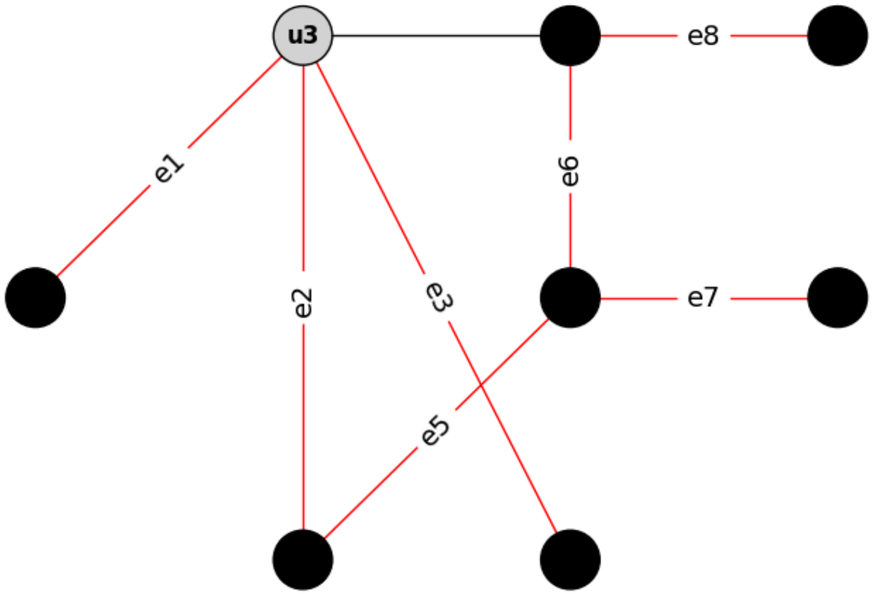
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Paso 15

Grafo original
Paso 15



Descripción del paso

Descripción: No quedan vecinos blancos de u4. Marcamos u4 como negro y hacemos backtracking.

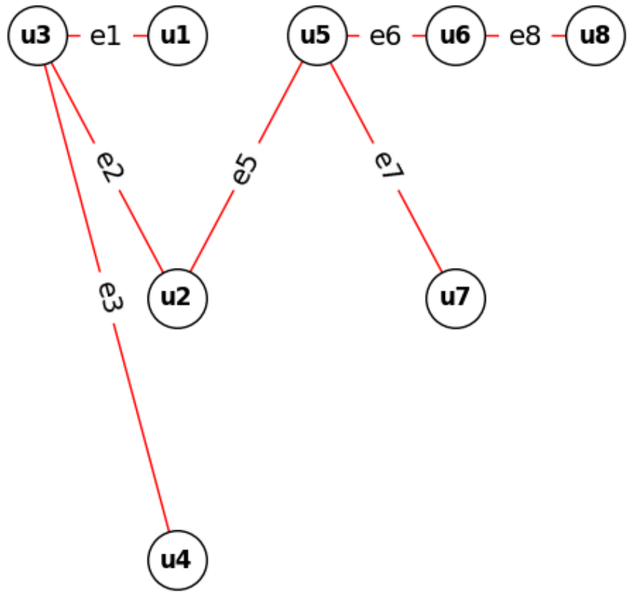
En este paso no se añadió ninguna arista al árbol DFS.

No se actualizó ninguna profundidad en este paso.

Pila (cima al final):
u3

Profundidades actuales:
u1: 1
u2: 1
u3: 0
u4: 1
u5: 2
u6: 3
u7: 3
u8: 4

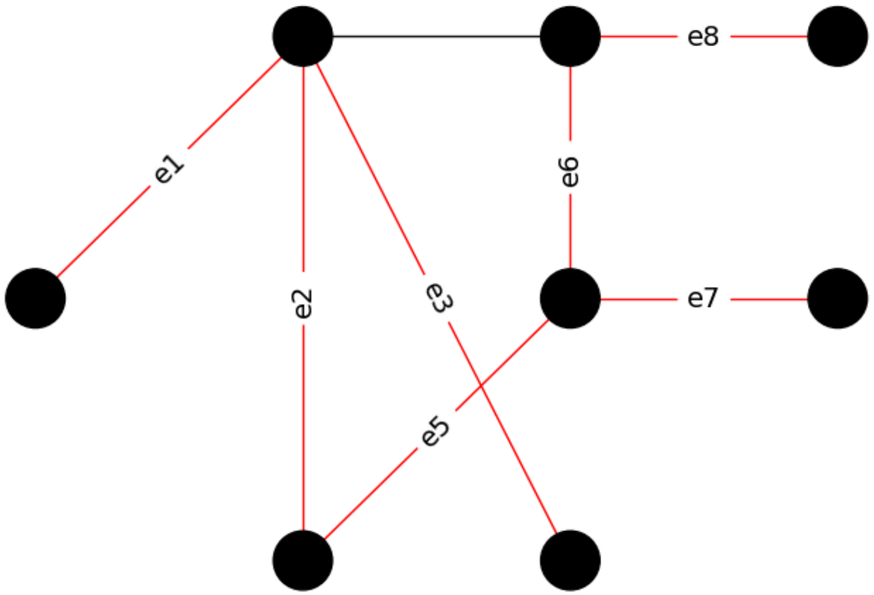
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Paso 16 - Fin

Grafo original
Paso 16



Descripción del paso

Descripción: No quedan vecinos blancos de u3. Marcamos u3 como negro y hacemos backtracking.

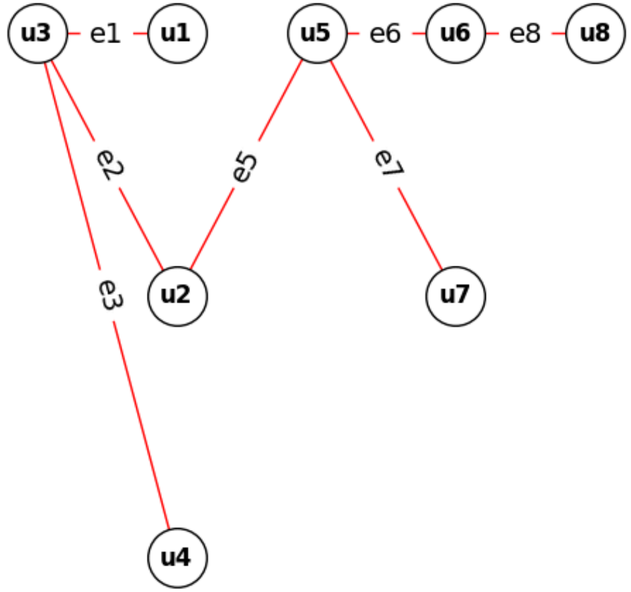
En este paso no se añadió ninguna arista al árbol DFS.

No se actualizó ninguna profundidad en este paso.

Pila (cima al final):
(vacía)

Profundidades actuales:
u1: 1
u2: 1
u3: 0
u4: 1
u5: 2
u6: 3
u7: 3
u8: 4

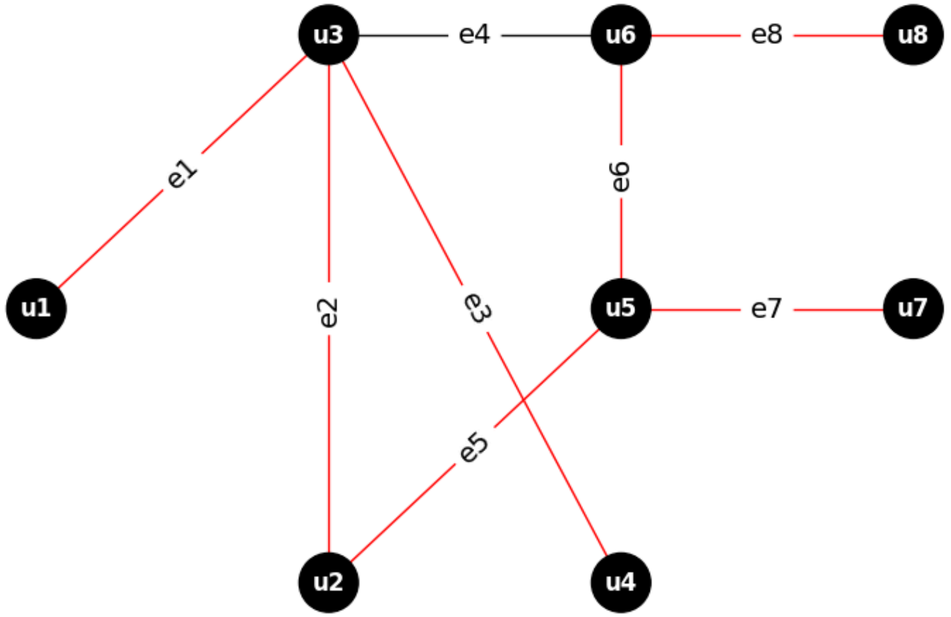
Árbol DFS
(construido hasta ahora)



5. Ejemplo paso a paso (cont.)

Resultado

Grafo original y árbol DFS (en rojo)



Datos finales del DFS

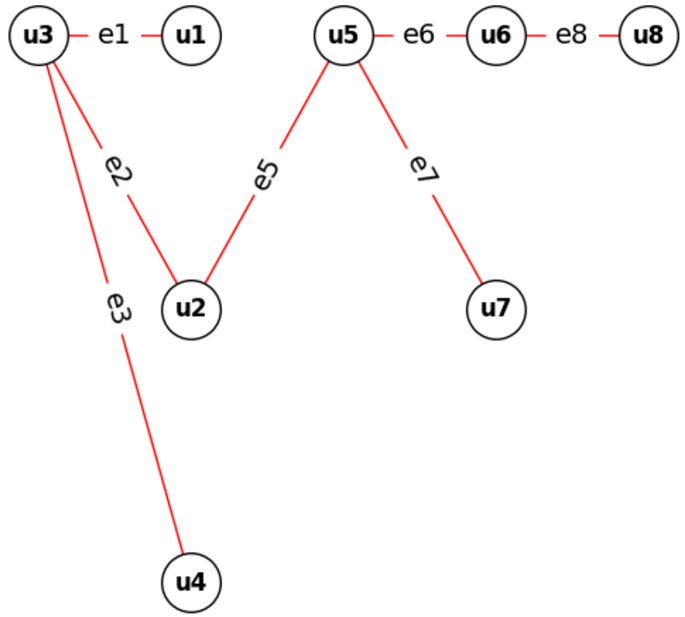
Resumen final del DFS

Profundidades finales (prof(u)):
- u1: 1
- u2: 1
- u3: 0
- u4: 1
- u5: 2
- u6: 3
- u7: 3
- u8: 4

Aristas en el árbol DFS:
- e1 = ('u1', 'u3')
- e2 = ('u2', 'u3')
- e3 = ('u3', 'u4')
- e5 = ('u2', 'u5')
- e6 = ('u5', 'u6')
- e7 = ('u5', 'u7')
- e8 = ('u6', 'u8')

Aristas que NO aparecen en el árbol DFS:
- e4 = ('u3', 'u6')

Árbol DFS final



6. Complejidad computacional

Temporal

- Sea G=(V,E) con |V| = *n* vértices y |E| = *m* aristas.
- DFS examina cada vértice al menos una vez, y revisa cada arista a lo más dos veces (ida y vuelta).
- Tiempo total: O(*n*+*m*), lineal en el tamaño de la gráfica.

Espacial

- Memoria utilizada: arreglos y pila de tamaño O(n) en el peor caso.
- Resultado: un bosque DFS (uno o varios árboles) que cubren todos los vértices.
- En una gráfica conexa, se obtiene un único árbol DFS.

7. Aplicaciones

- **Sistemas de archivos y compiladores:** análisis de dependencias, orden de compilación, optimización.
- **Redes de comunicación:** detección de enlaces críticos, análisis de conectividad.
- **Juegos:** resolver laberintos, sudokus o problemas con una sola solución mediante backtracking.

$a^{\circ} = 1$

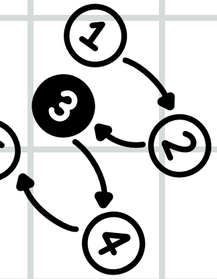
8. Conclusiones y Key take-aways

1. **DFS** es un algoritmo de recorrido en gráficas que usa una pila y explora en profundidad, con backtracking.
2. Produce un bosque DFS que revela la estructura de la gráfica.
3. Su complejidad es lineal en el número de vértices y aristas.
4. Es una herramienta básica que aparece como subrutina en muchos algoritmos avanzados.
5. Se complementa con BFS (no compiten)

DFS es un algoritmo que recorre una gráfica en profundidad usando una pila, construye un árbol generador y se aplica para analizar la estructura del grafo.

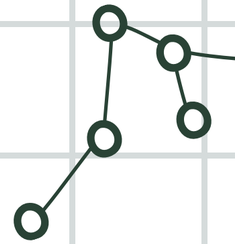
9. Referencias

- Béla Bollobás. (2010). *Modern graph theory*. New York, Ny Springer.
- Cormen, T. H., Charles Eric Leiserson, Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms*. The Mit Press.
- Diestel, R. (2025). *Graph Theory*. Springer Nature.
- J.A. Bondy, & Murty, U. S. R. (2008). *Graph Theory*, (Cap. 6). Springer.
- Ringer, L., Turner, R., & Carroll, G. (n.d.). *Graph Searching*. Cse442-17f.github.io. https://cse442-17f.github.io/A-Star-Search/
- Zhao, Y. (2023). *Graph Theory and Additive Combinatorics*. Cambridge University Press.



FIN

¡GRACIAS!



Herramientas utilizadas: PowerPoint, Canva, Sora, Python (Google Colab)

Repositorio github: https://github.com/hsilva-unam/THC_Semestre_06_01/blob/main/IMD_26_01_VideoDFS_HectorSilva.ipynb