COMPLEJIDAD ALGORÍTMICA

INTRODUCCIÓN A GRAFOS

ALGORITMOS DE DFS Y BFS

Andrés Leiva

¿QUÉ ES UN GRAFO?

¿QUÉ ES UN GRAFO?

Grafo = Nodos + Aristas

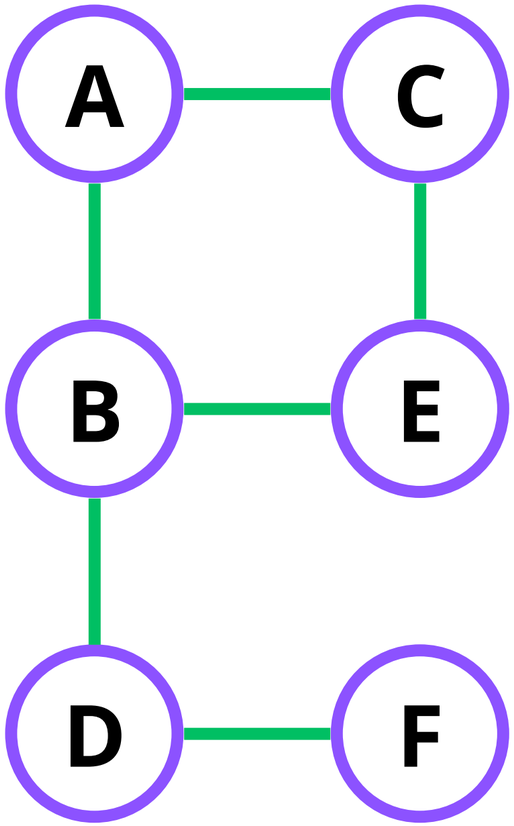
¿QUÉ ES UN GRAFO?

Grafo = Nodos + Aristas

¿QUÉ ES UN GRAFO?

Grafo = Nodos + Aristas

Grafo = Nodos + Aristas



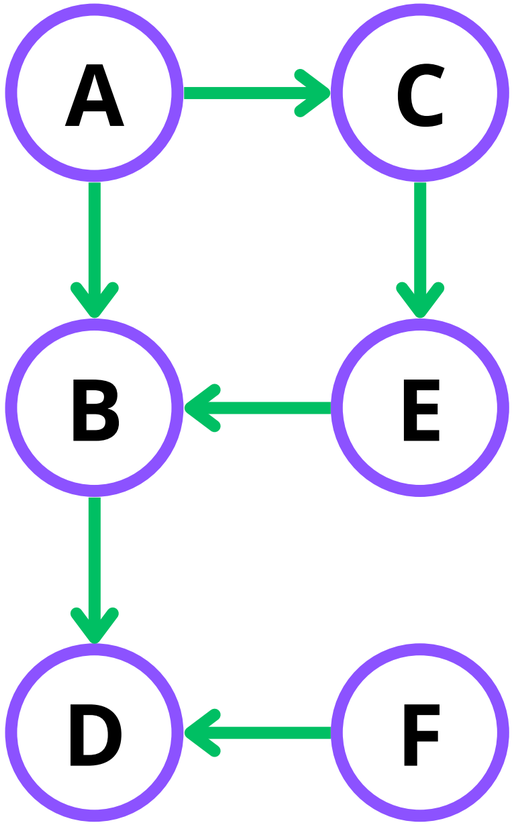
Grafo NO dirigido

Las aristas NO tienen un sentido

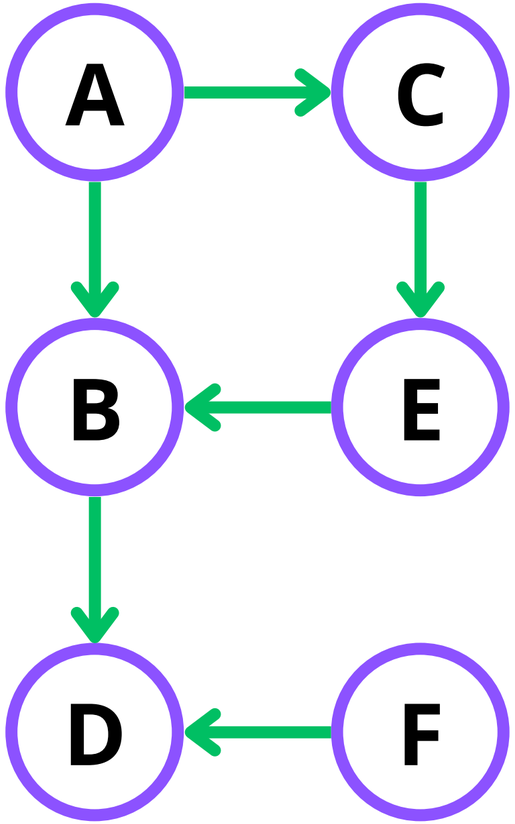
Grafo = Nodos + Aristas

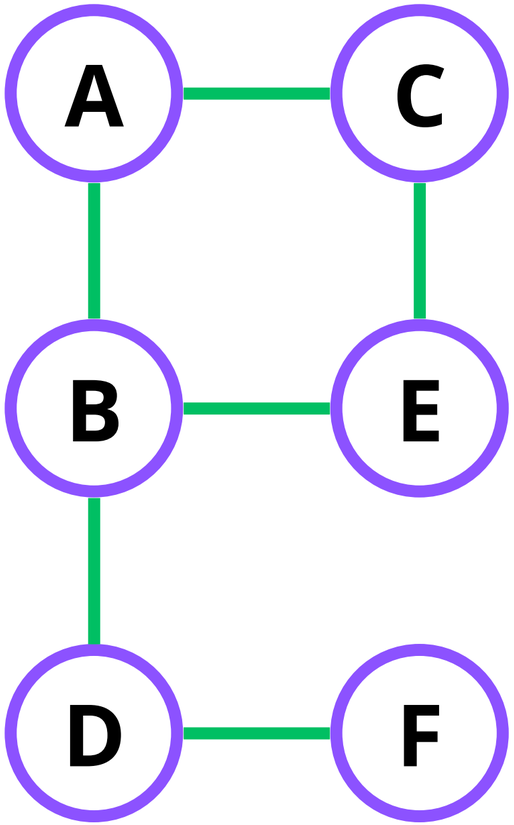
Grafo dirigido

Las aristas SI tienen un sentido

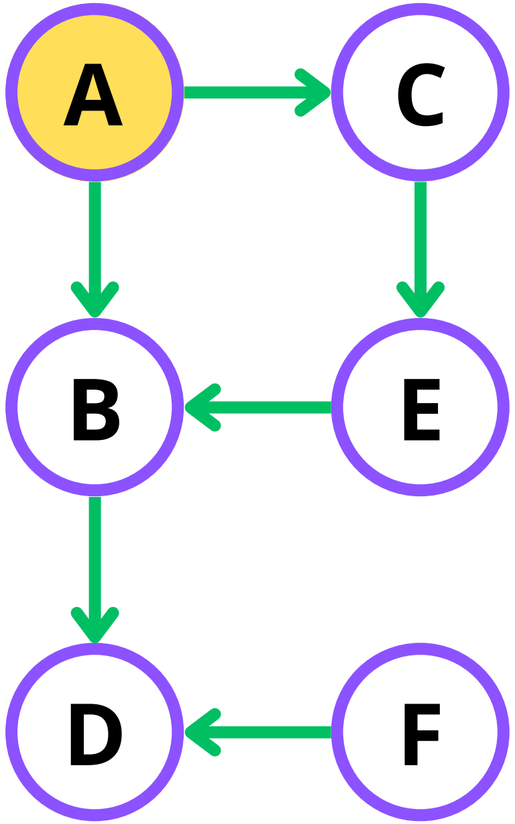


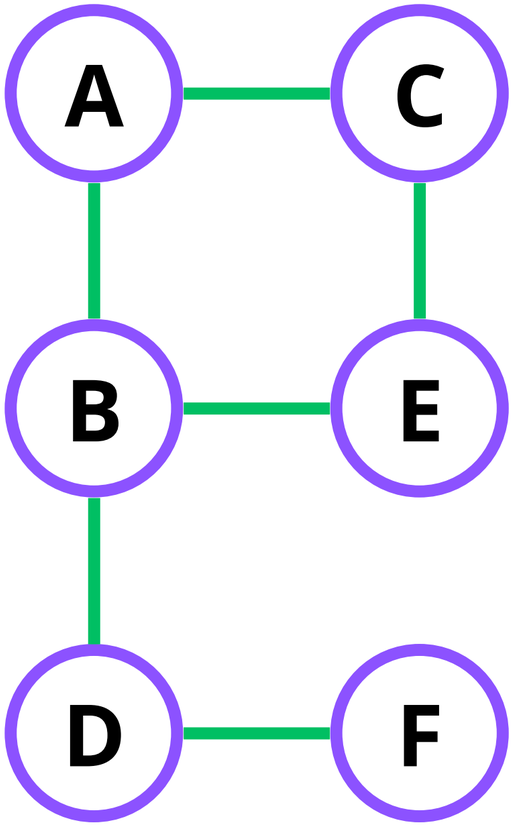
Grafo = Nodos + Aristas



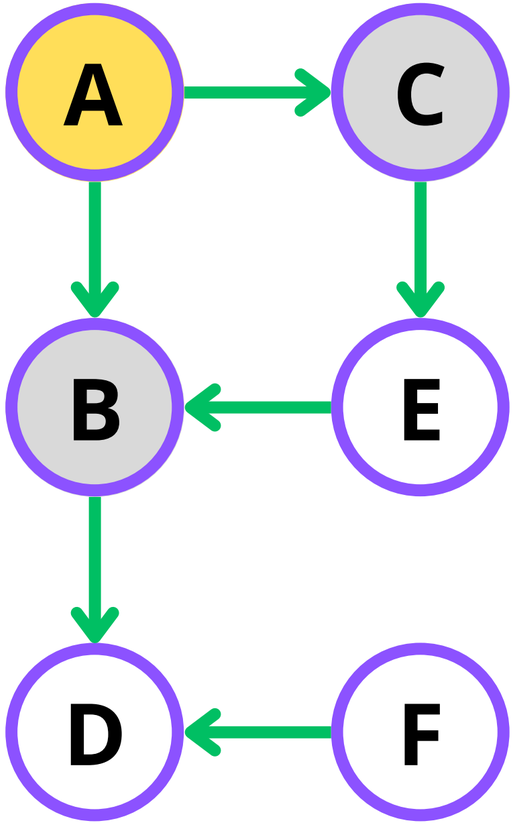


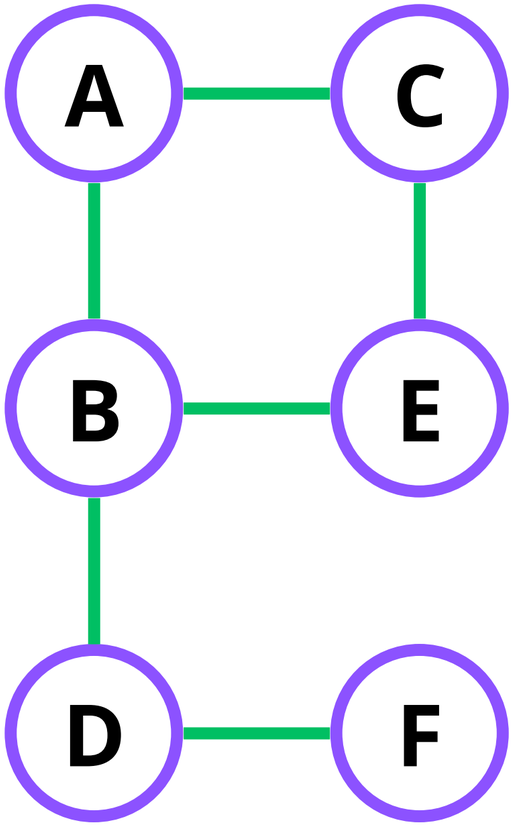
Grafo = Nodos + Aristas



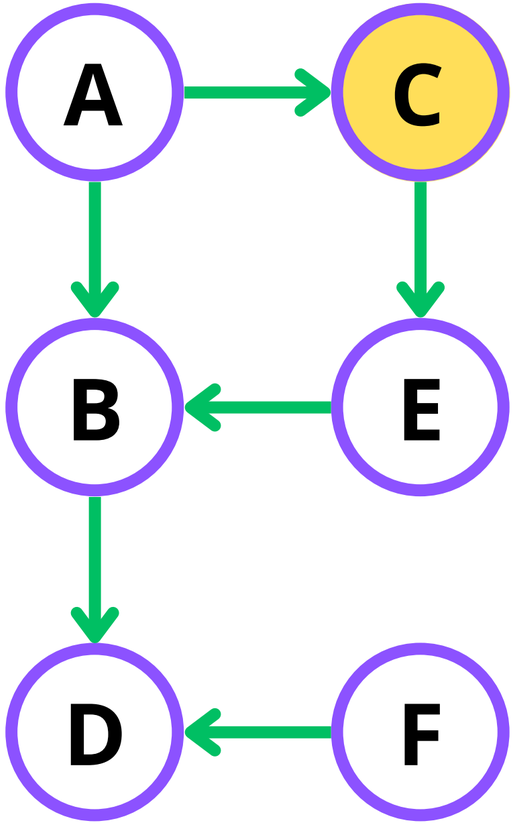


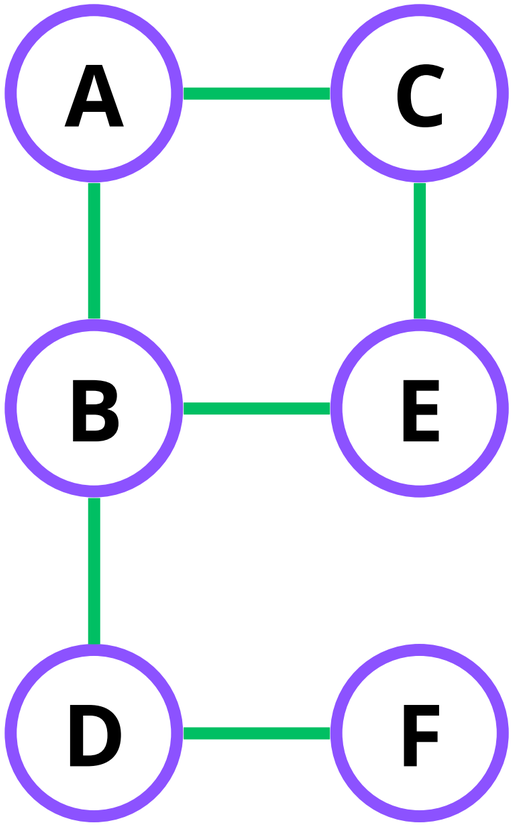
Grafo = Nodos + Aristas



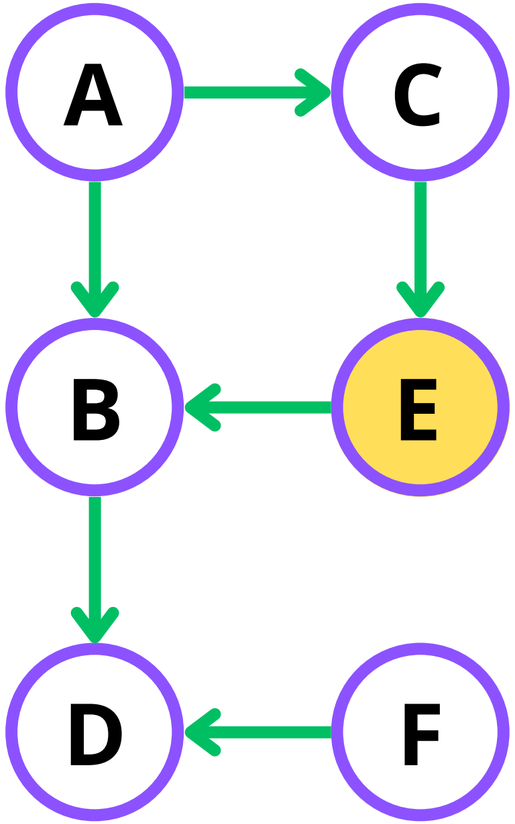


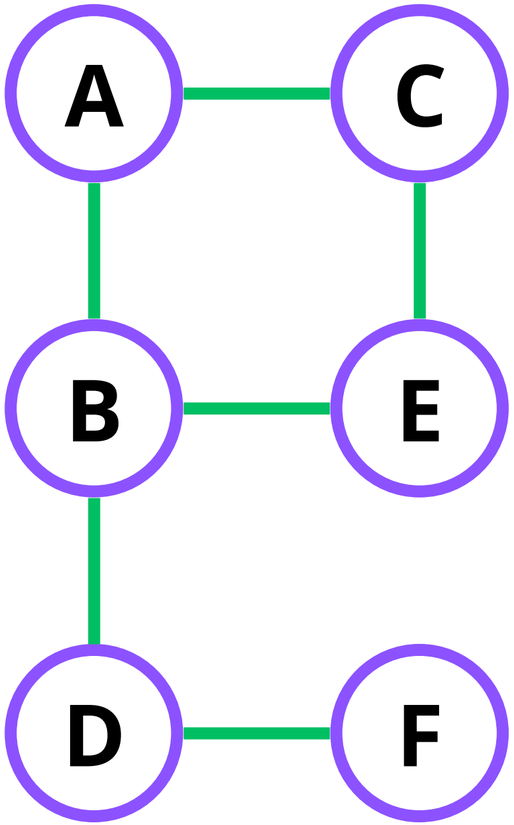
Grafo = Nodos + Aristas



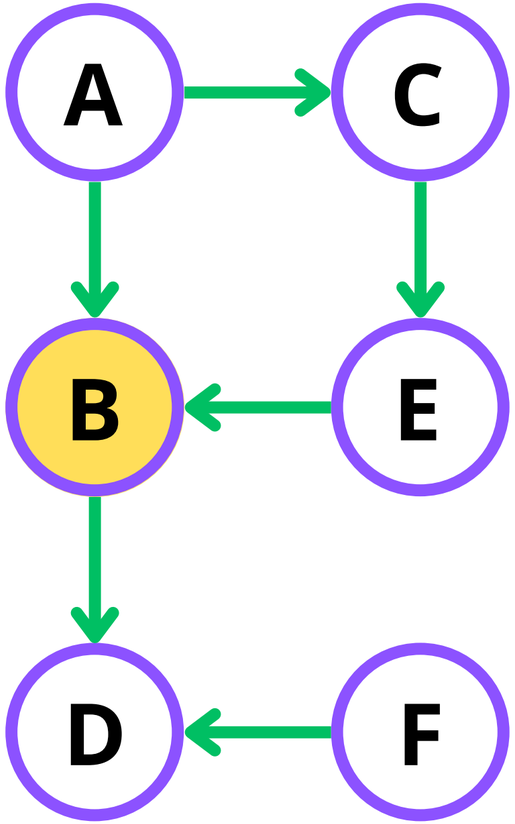


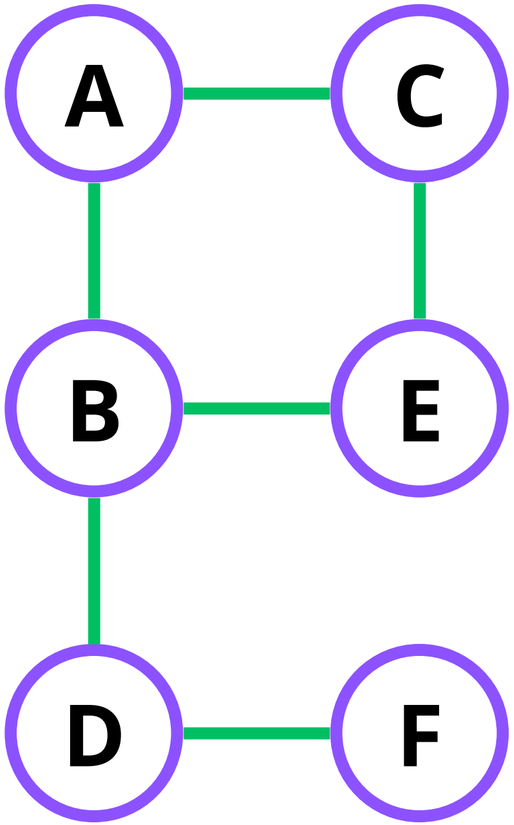
Grafo = Nodos + Aristas



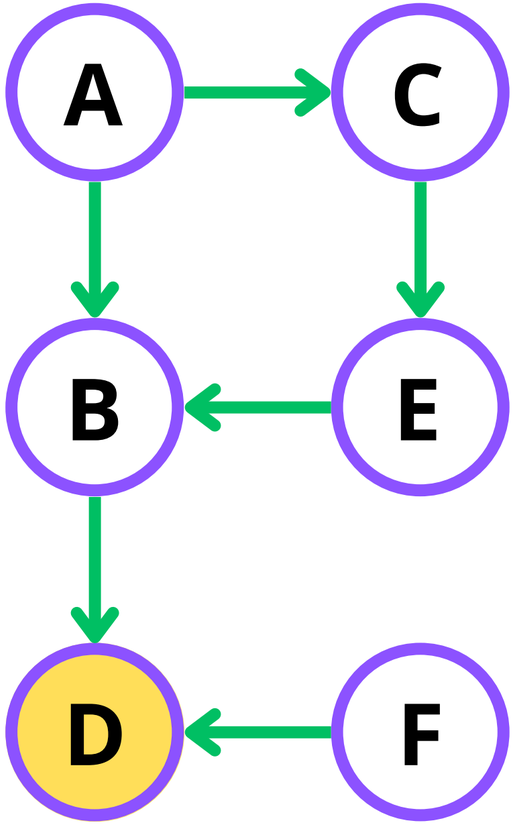


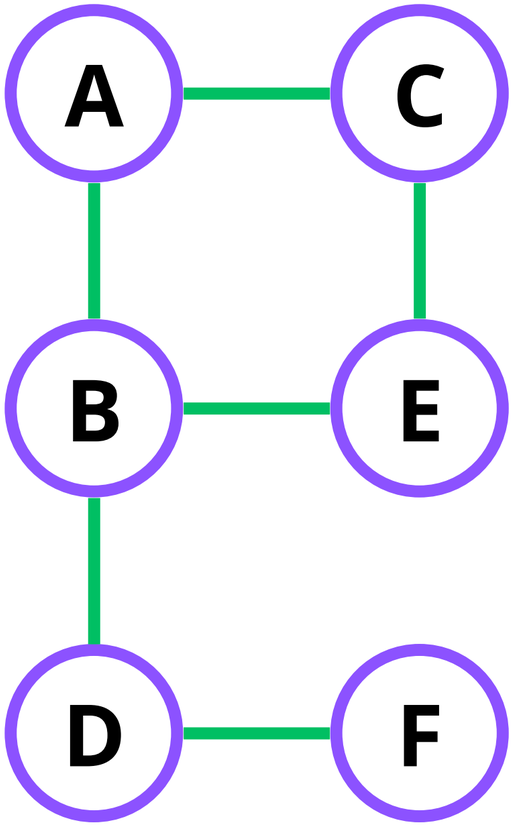
Grafo = Nodos + Aristas



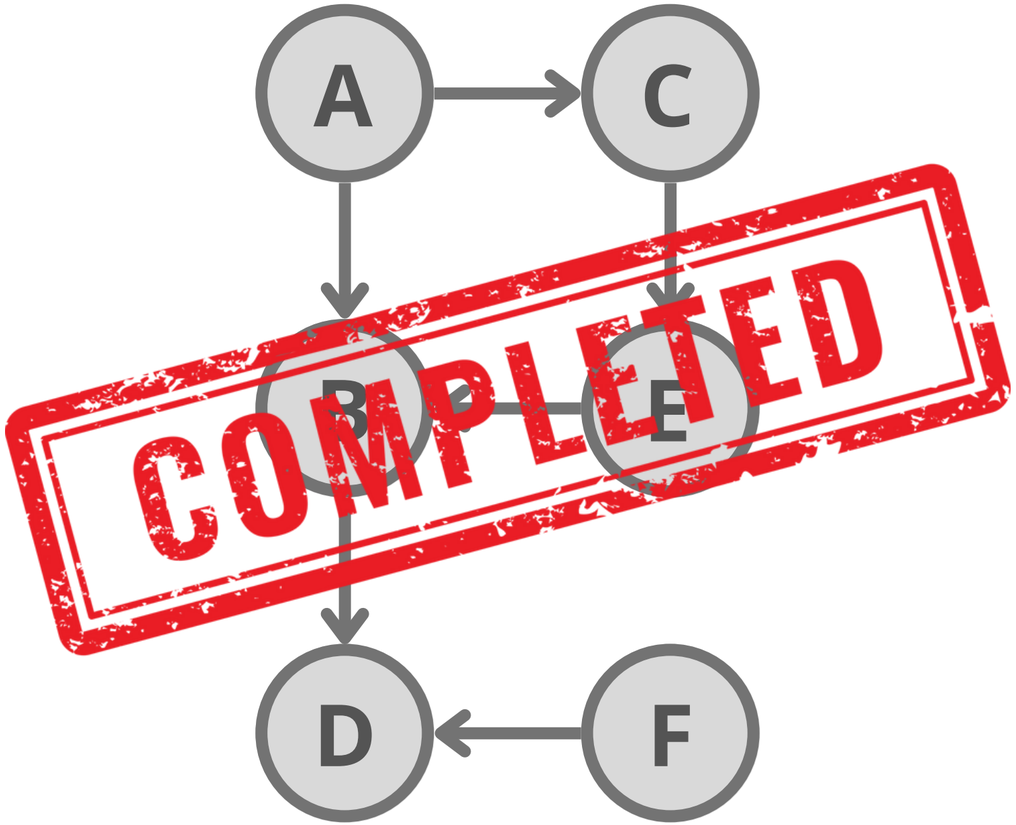


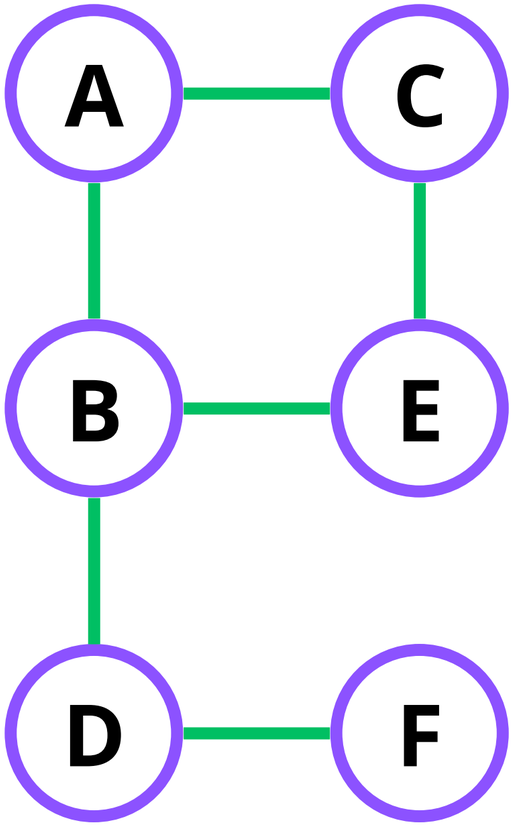
Grafo = Nodos + Aristas



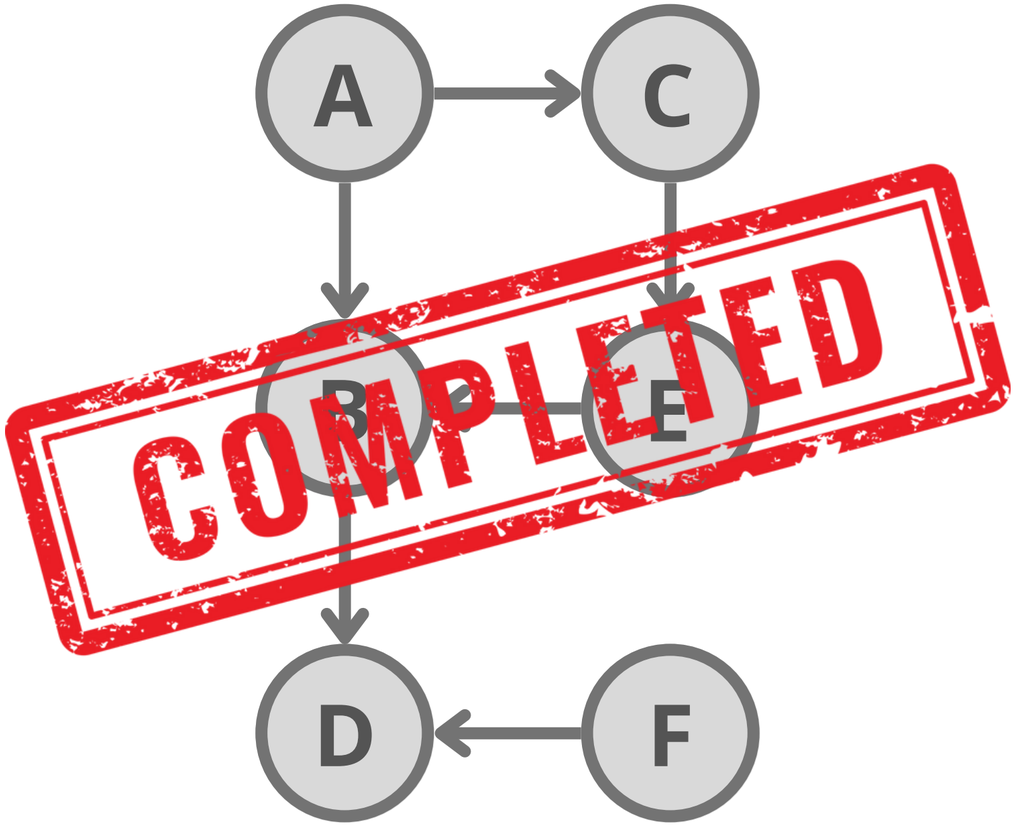


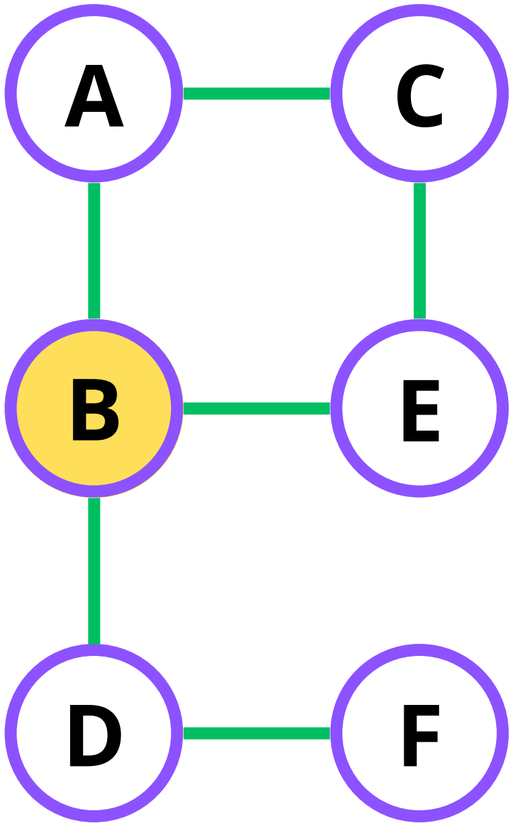
Grafo = Nodos + Aristas



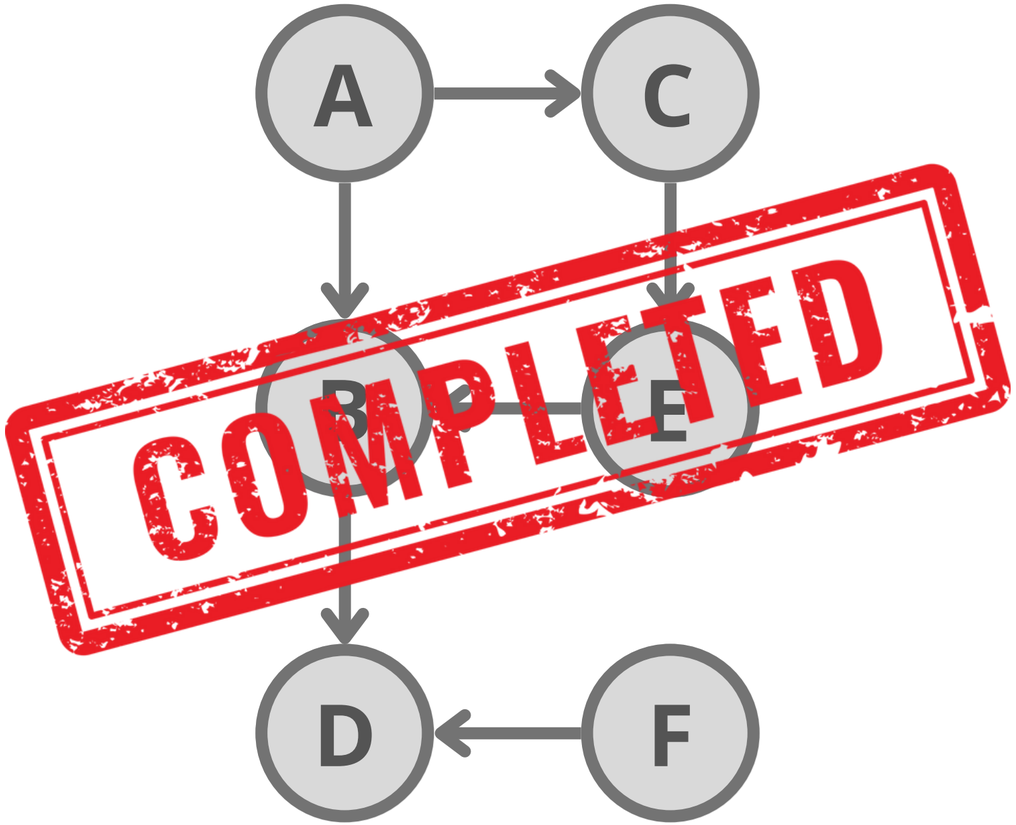


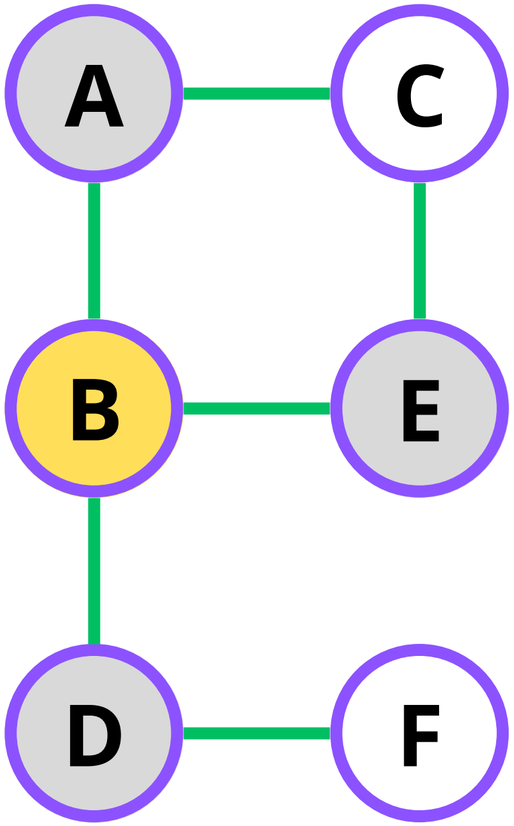
Grafo = Nodos + Aristas



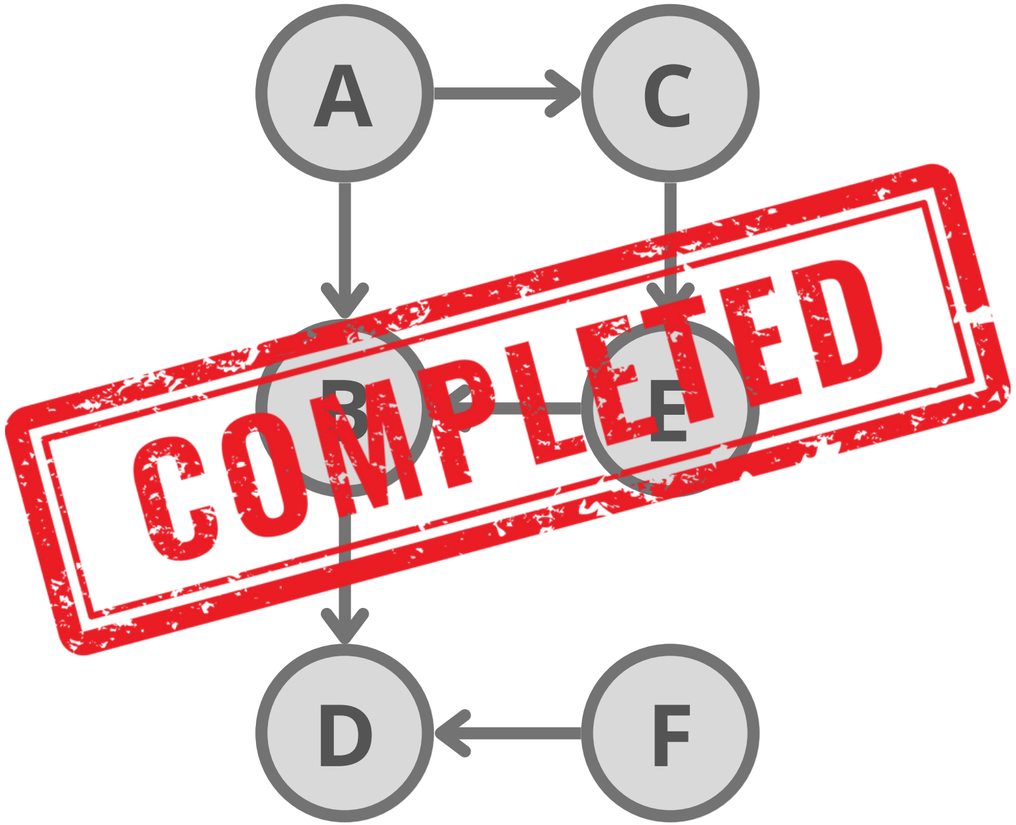


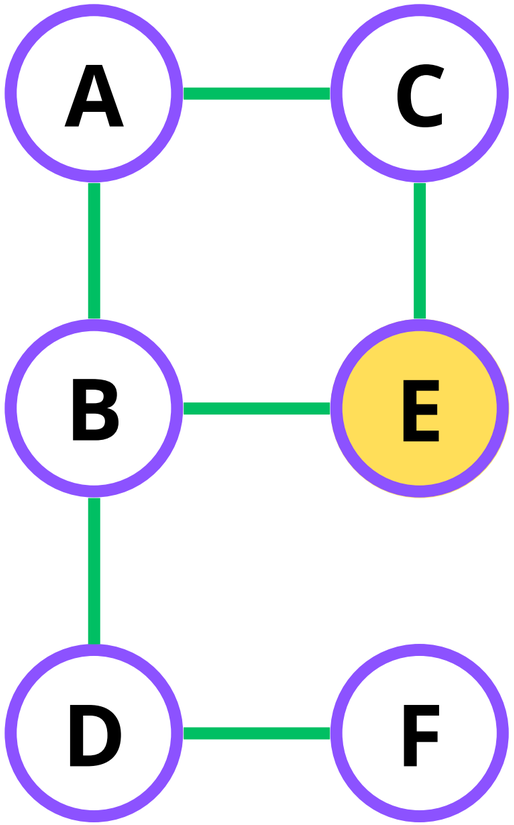
Grafo = Nodos + Aristas



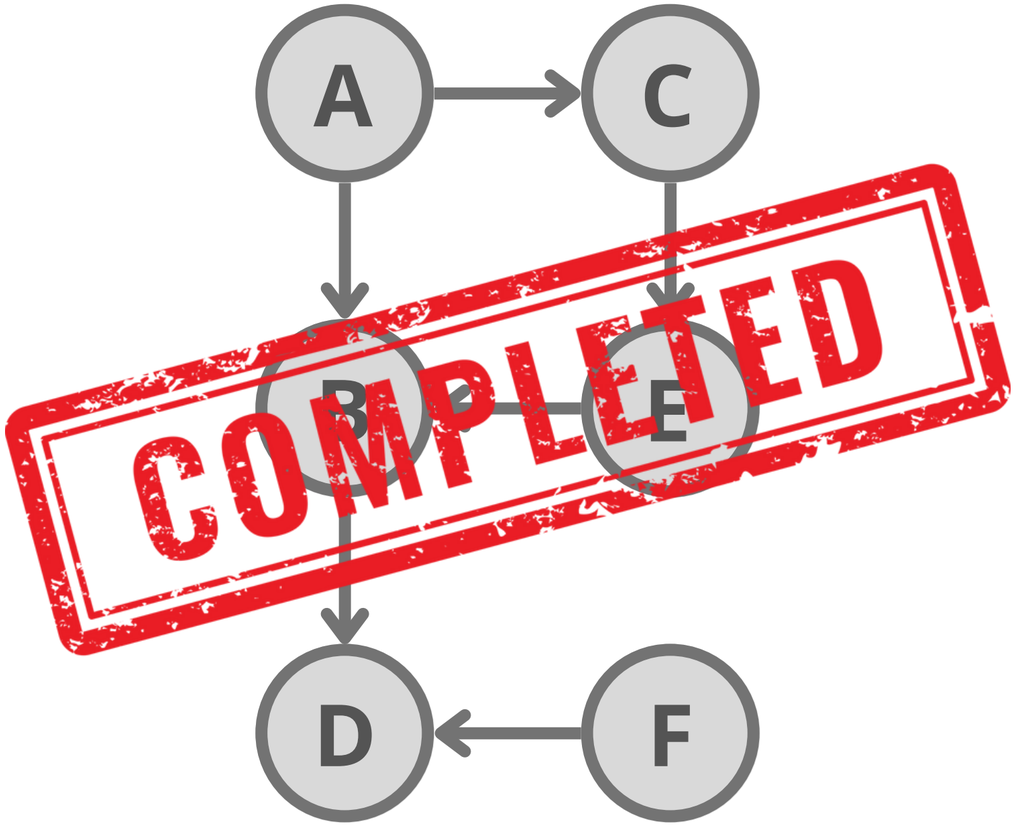


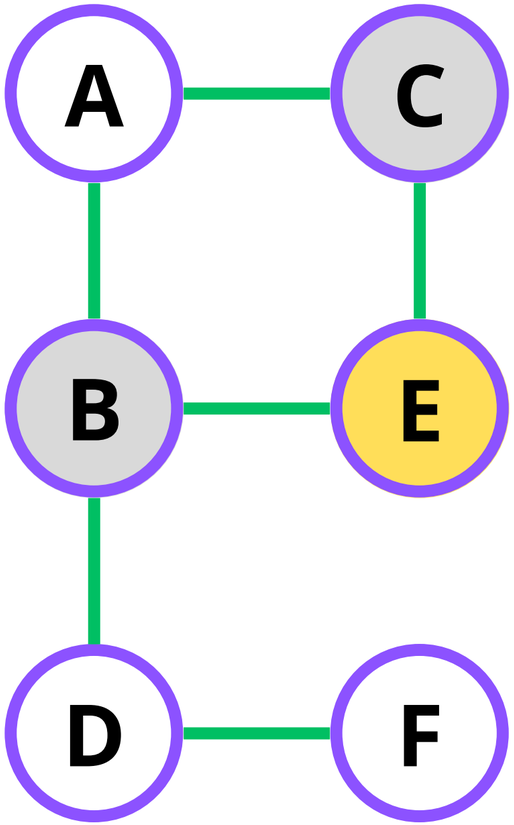
Grafo = Nodos + Aristas



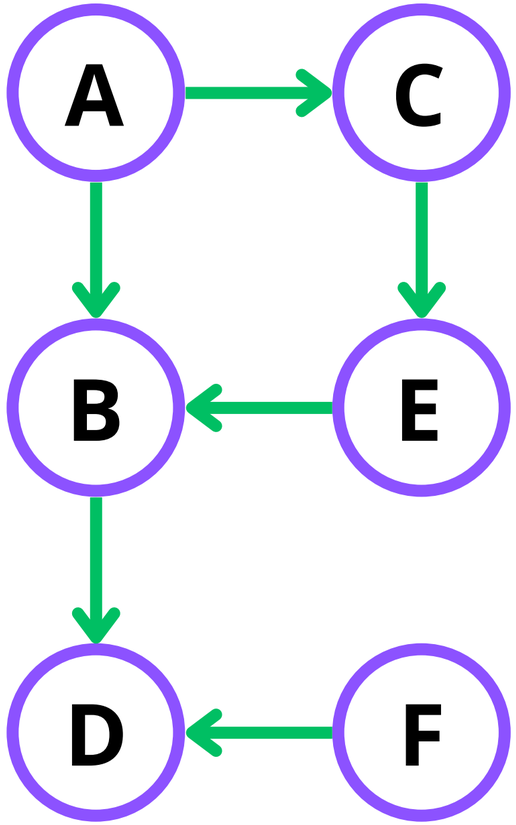


Grafo = Nodos + Aristas

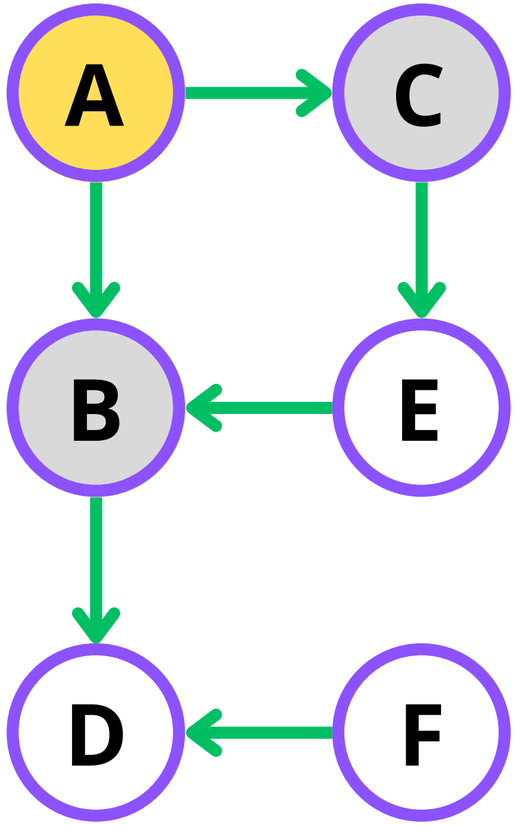




Terminologías en Grafos



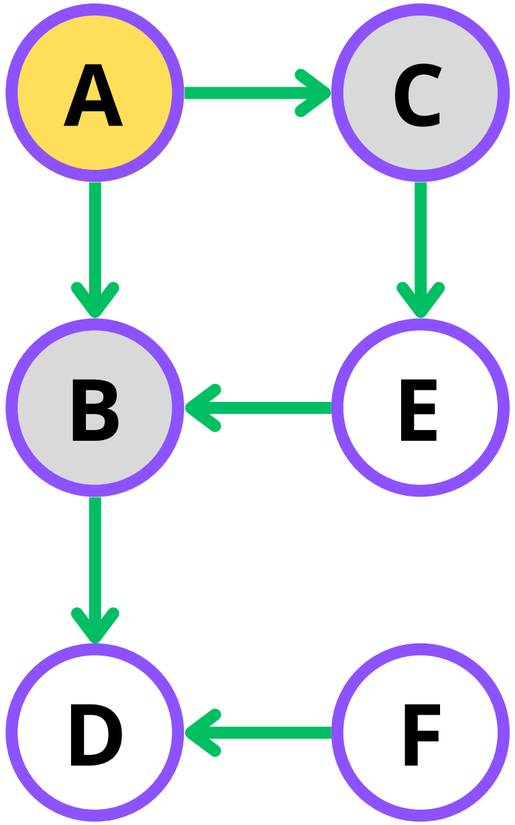
Terminologías en Grafos



Terminologías en Grafos

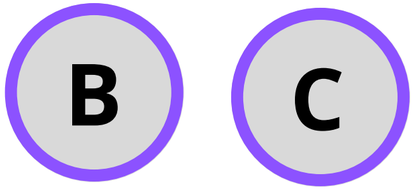
Vecinos del nodo A

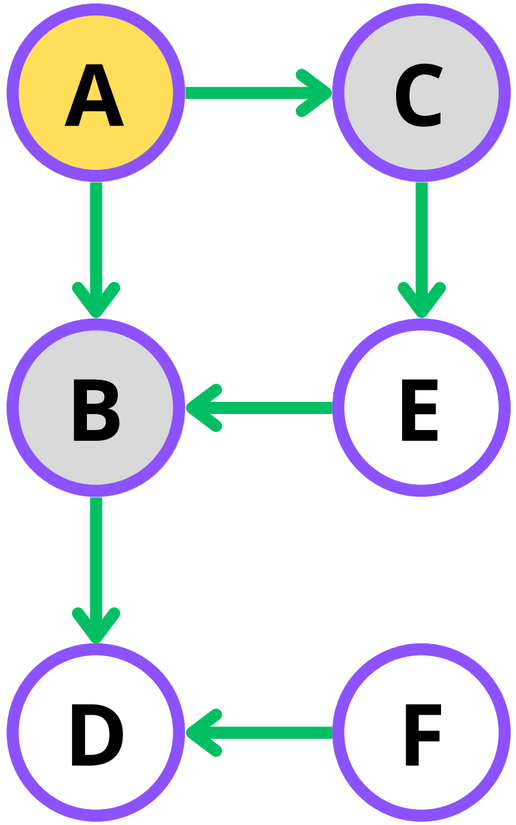
B C



Terminologías en Grafos

Vecinos del nodo A



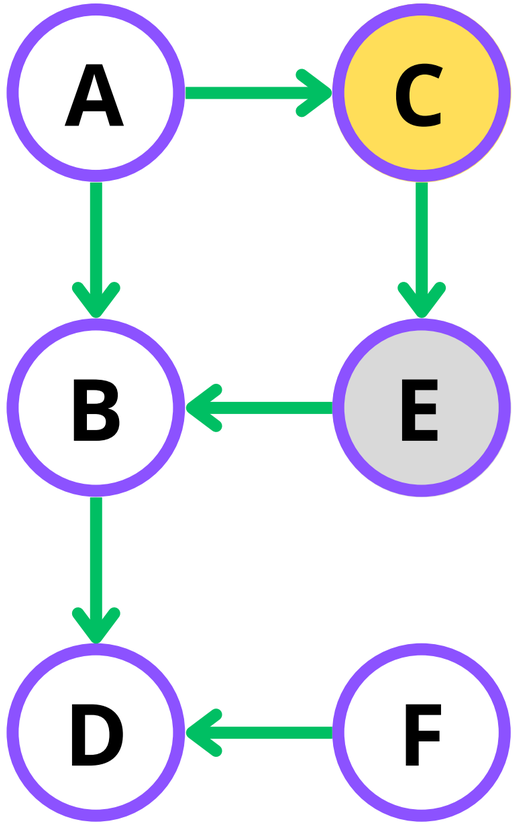


Un nodo vecino es cualquier nodo que es accesible mediante una arista desde otro nodo

Terminologías en Grafos

Vecinos del nodo C

E



Un nodo vecino es cualquier nodo que es accesible mediante una arista desde otro nodo

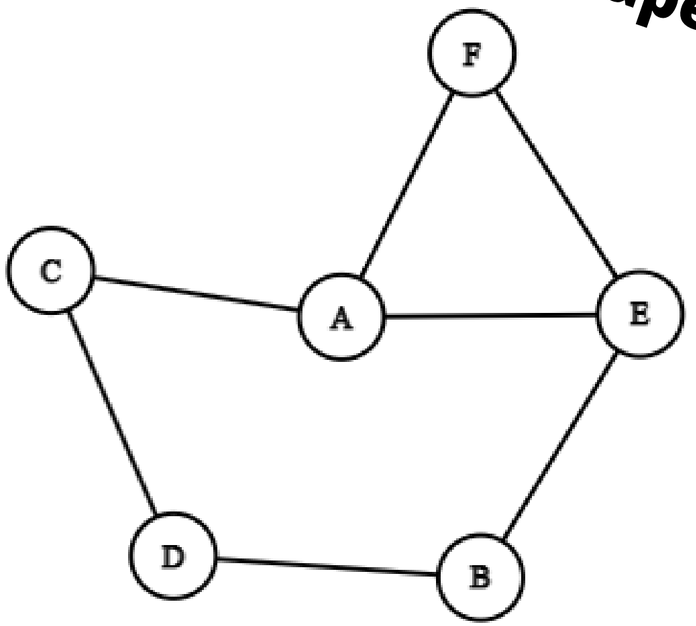
¿CÓMO REPRESENTAR UN GRAFO?

¿CÓMO REPRESENTAR UN GRAFO?

Cuando quieras ver cómo funciona un algoritmo de grafos, lo mejor que puedes hacer es graficarlo.

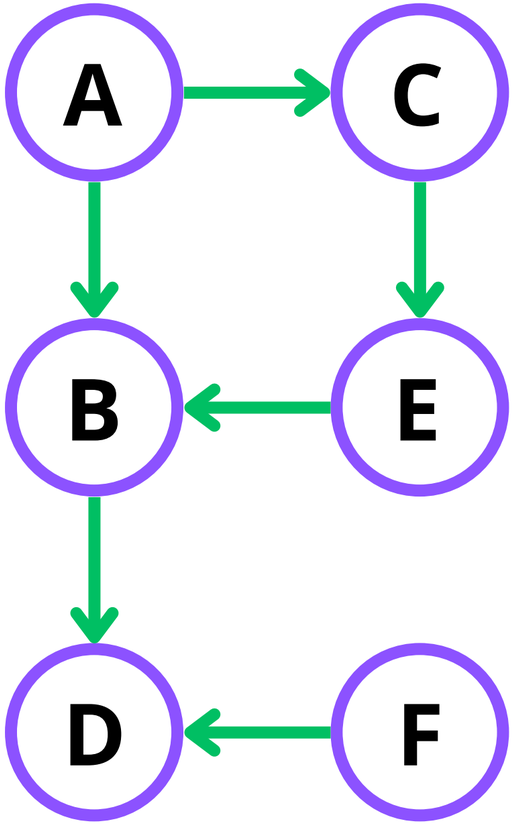
Paint

Papel y Lápiz



csacademy.com/app/graph_editor

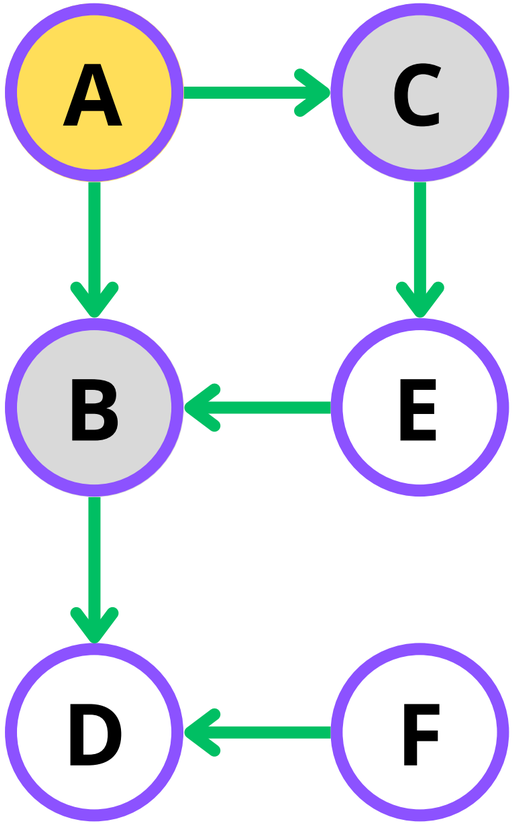
¿CÓMO REPRESENTAR UN GRAFO?




```
graph = {
    "A": ["B", "C"],
    "B": ["D"],
    "C": ["E"],
    "D": [],
    "E": ["B"],
    "F": ["D"],
}
```

Lista de adyacencia

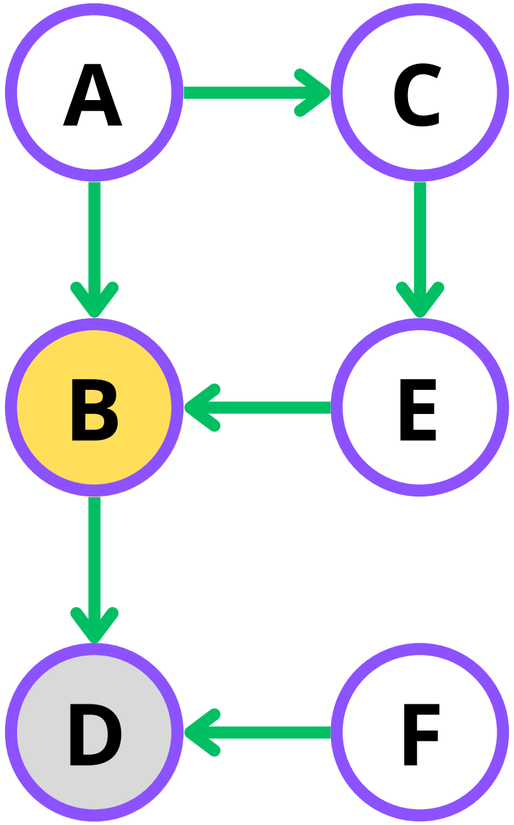
¿CÓMO REPRESENTAR UN GRAFO?




```
graph = {
  "A": ["B", "C"],
  "B": ["D"],
  "C": ["E"],
  "D": [],
  "E": ["B"],
  "F": ["D"],
}
```

Lista de adyacencia

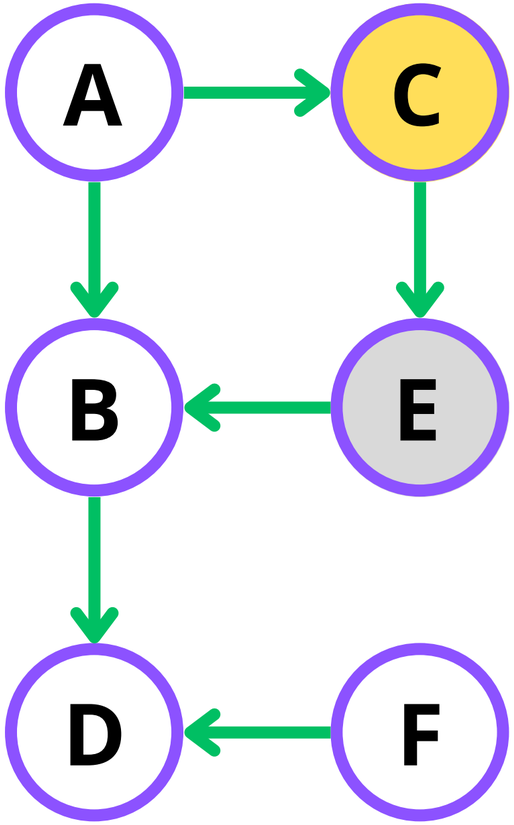
¿CÓMO REPRESENTAR UN GRAFO?




```
graph = {
    "A": ["B", "C"],
    "B": ["D"],
    "C": ["E"],
    "D": [],
    "E": ["B"],
    "F": ["D"],
}
```

Lista de adyacencia

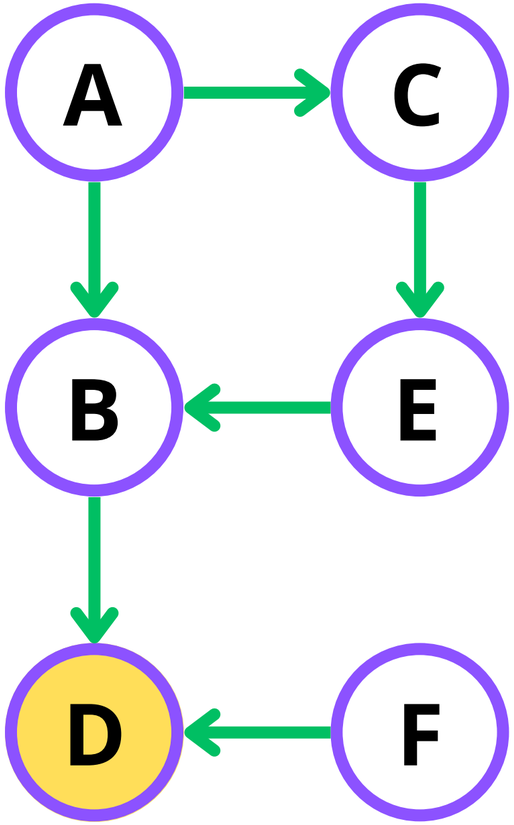
¿CÓMO REPRESENTAR UN GRAFO?




```
graph = {
    "A": ["B", "C"],
    "B": ["E"],
    "C": ["E"],
    "D": [],
    "E": ["B"],
    "F": ["D"],
}
```

Lista de adyacencia

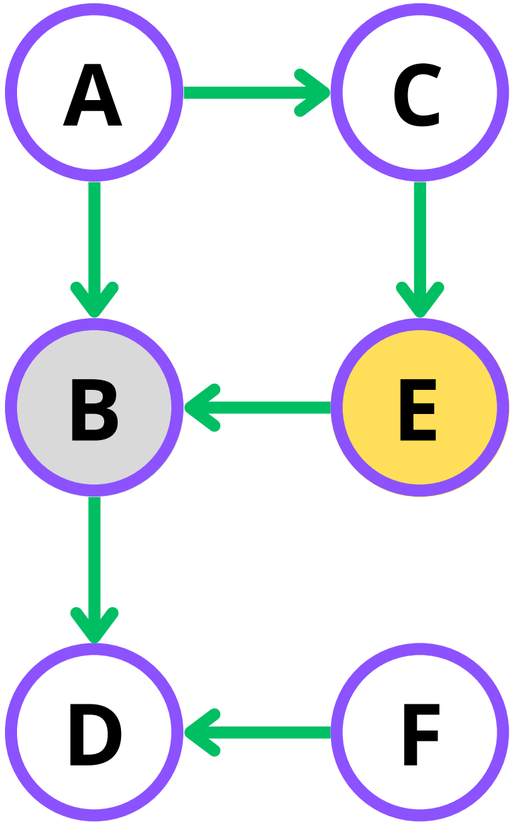
¿CÓMO REPRESENTAR UN GRAFO?




```
graph = {
    "A": ["B", "C"],
    "B": ["D"],
    "C": ["E"],
    "D": [],
    "E": ["B"],
    "F": ["D"],
}
```

Lista de adyacencia

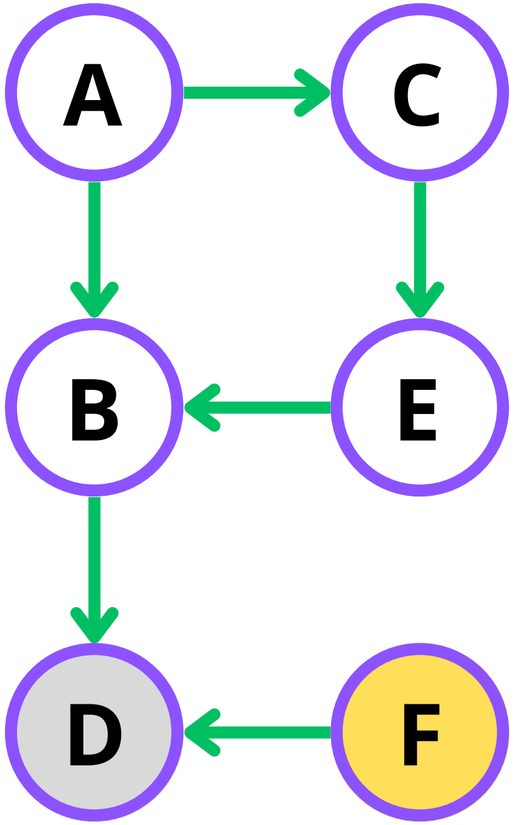
¿CÓMO REPRESENTAR UN GRAFO?




```
graph = {
    "A": ["B", "C"],
    "B": ["D"],
    "C": ["E"],
    "D": [],
    "E": ["B"],
    "F": ["D"],
}
```

Lista de adyacencia

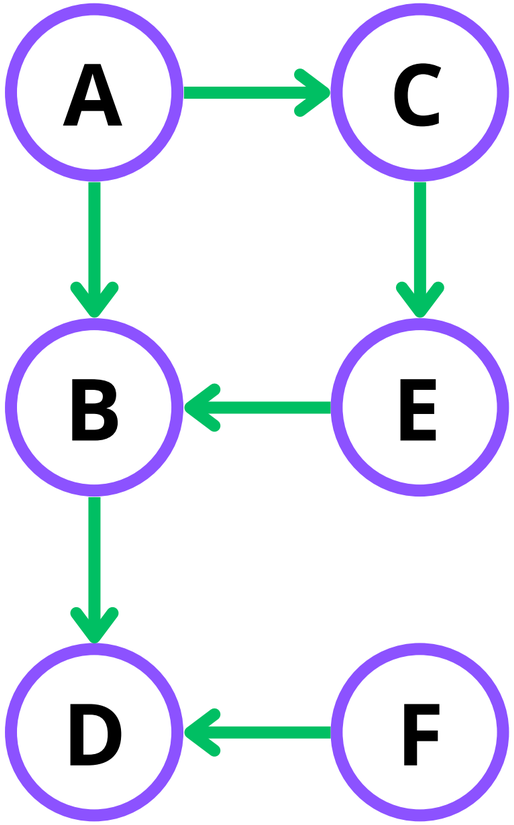
¿CÓMO REPRESENTAR UN GRAFO?




```
graph = {
  "A": ["B", "C"],
  "B": ["D"],
  "C": ["E"],
  "D": [],
  "E": ["F"],
  "F": ["D"],
}
```

Lista de adyacencia

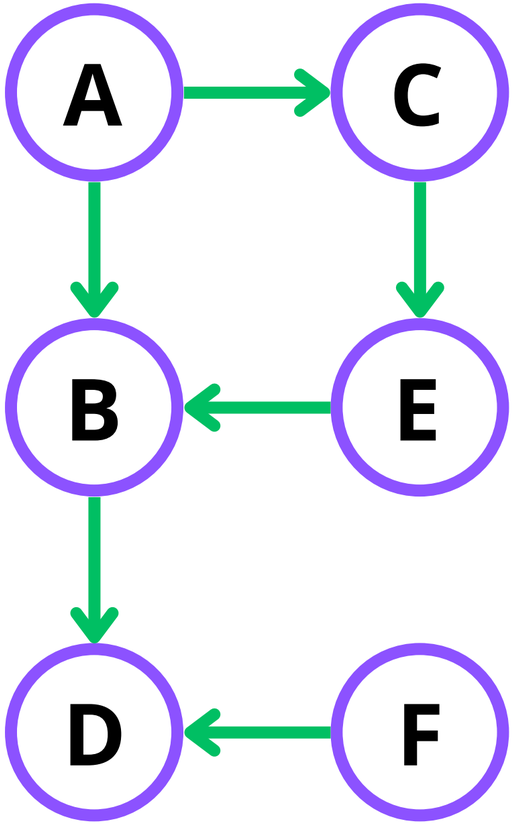
¿CÓMO REPRESENTAR UN GRAFO?




```
graph = [
  [0, 1, 1, 0, 0, 0],
  [0, 0, 0, 1, 0, 0],
  [0, 0, 0, 0, 1, 0],
  [0, 0, 0, 0, 0, 0],
  [0, 1, 0, 0, 0, 0],
  [0, 0, 0, 1, 0, 0]
]
```

Matriz de adyacencia

¿CÓMO REPRESENTAR UN GRAFO?

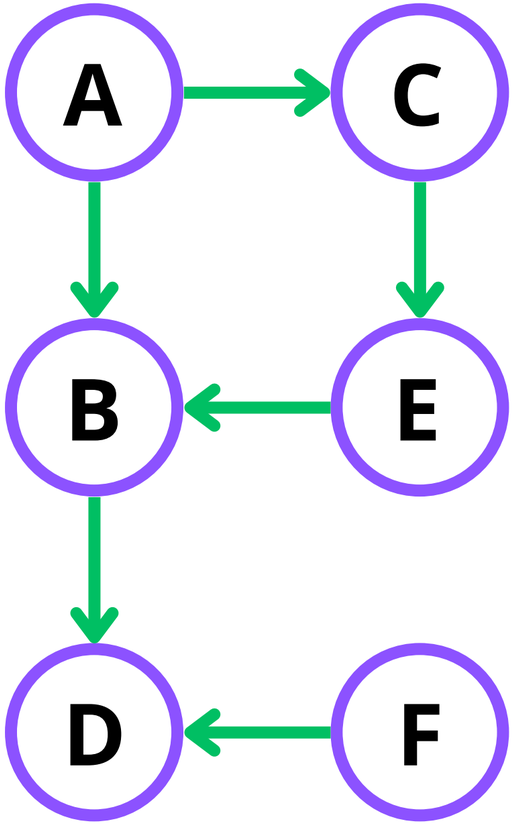


Cada fila representa los vecinos del nodo

```
graph = [
A[0, 1, 1, 0, 0, 0],
B[0, 0, 0, 1, 0, 0],
C[0, 0, 0, 0, 1, 0],
D[0, 0, 0, 0, 0, 0],
E[0, 1, 0, 0, 0, 0],
F[0, 0, 0, 1, 0, 0]
]
```

Matriz de adyacencia

¿CÓMO REPRESENTAR UN GRAFO?



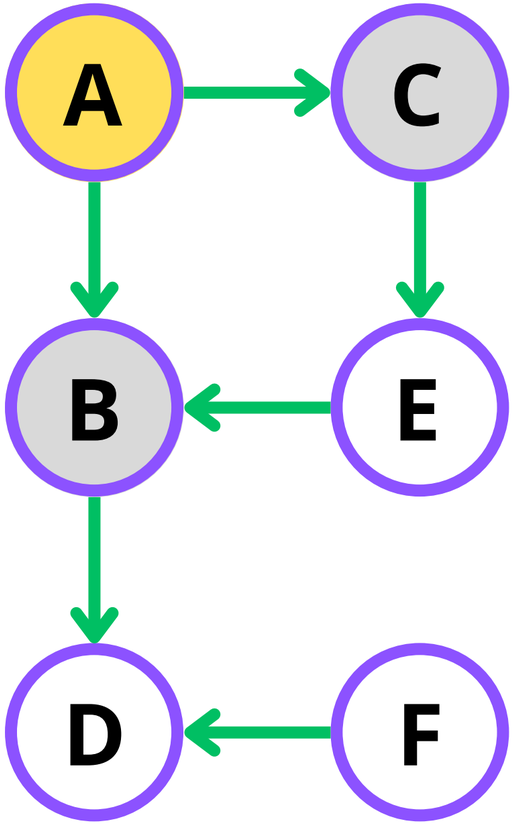
Cada fila representa los vecinos del nodo

Cada columna es un vértice

```
graph = [
  A[0, 1, 1, 0, 0, 0],
  B[0, 0, 0, 1, 0, 0],
  C[0, 0, 0, 0, 1, 0],
  D[0, 0, 0, 0, 0, 0],
  E[0, 1, 0, 0, 0, 0],
  F[0, 0, 0, 1, 0, 0]
] A  B  C  D  E  F
```

Matriz de adyacencia

¿CÓMO REPRESENTAR UN GRAFO?



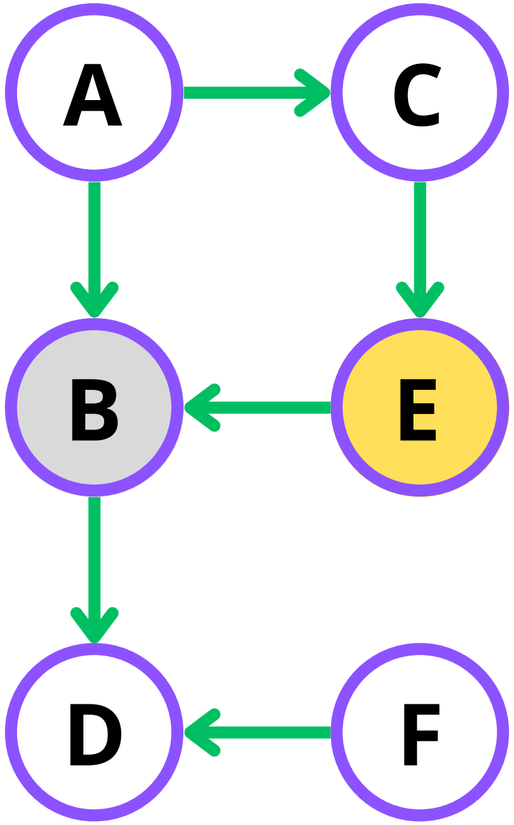
Cada fila representa los vecinos del nodo

Cada columna es un vértice

```
1  graph = [
2    A[0, 1, 1, 0, 0, 0],
3    B[0, 0, 0, 1, 0, 0],
4    C[0, 0, 0, 0, 1, 0],
5    D[0, 0, 0, 0, 0, 0],
6    E[0, 1, 0, 0, 0, 0],
7    F[0, 0, 0, 1, 0, 0]
8  ]  A  B  C  D  E  F
```

Matriz de adyacencia

¿CÓMO REPRESENTAR UN GRAFO?



Cada fila representa los vecinos del nodo

Cada columna es un vértice

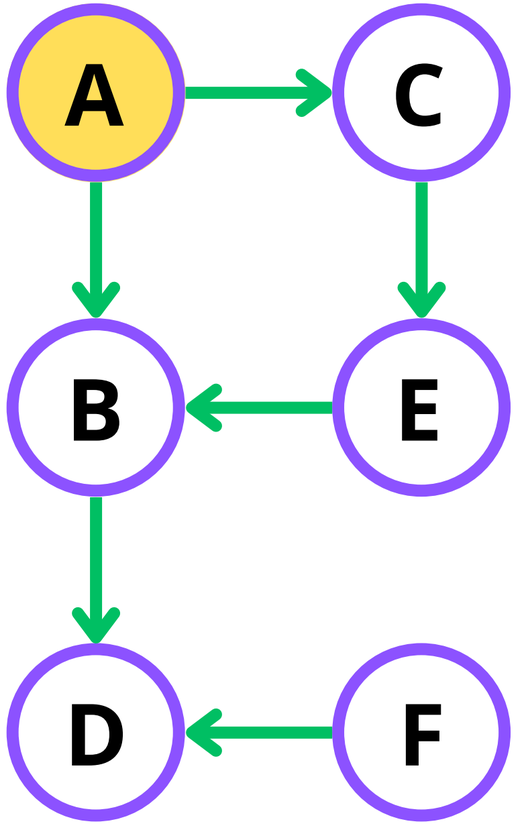
```
1  graph = [
2    A[0, 1, 1, 0, 0, 0],
3    B[0, 0, 0, 1, 0, 0],
4    C[0, 0, 0, 0, 1, 0],
5    D[0, 0, 0, 0, 0, 0],
6    E[0, 1, 0, 0, 0, 0],
7    F[0, 0, 0, 1, 0, 0]
8  ]  A  B  C  D  E  F
```

Matriz de adyacencia

ALGORITMOS DE BÚSQUEDA:

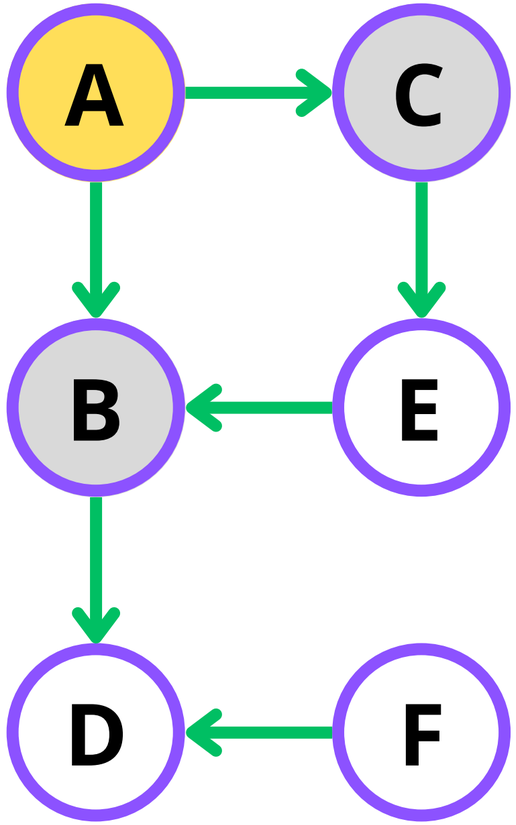
DFS

ALGORITMOS DE BÚSQUEDA: DFS



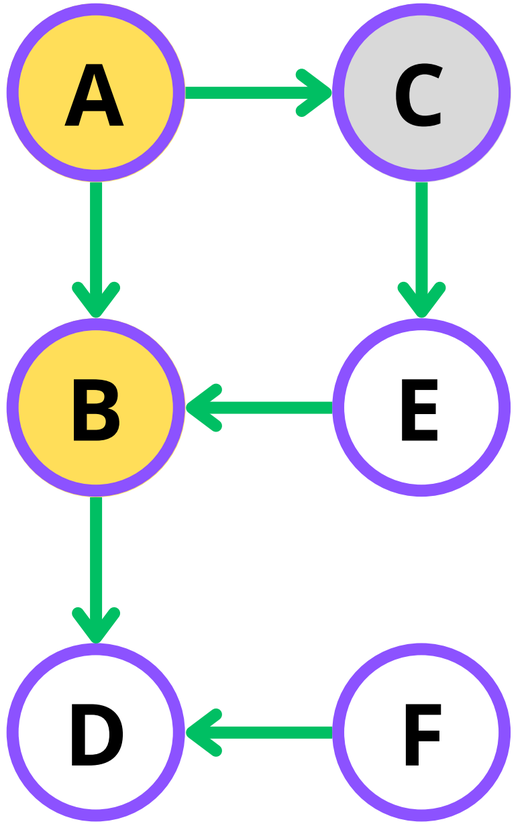
1. Empezamos de un nodo inicial

ALGORITMOS DE BÚSQUEDA: DFS



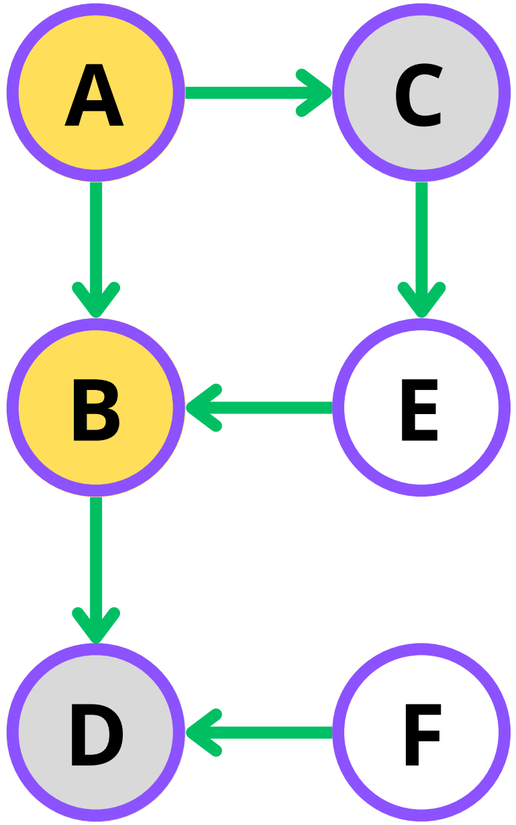
1. Empezamos de un nodo inicial
2. Procedemos a visitar sus vecinos que no hayan sido visitados (y volvemos a realizar el paso 2 con los vecinos del vecino recursivamente hasta que no podamos más)

ALGORITMOS DE BÚSQUEDA: DFS



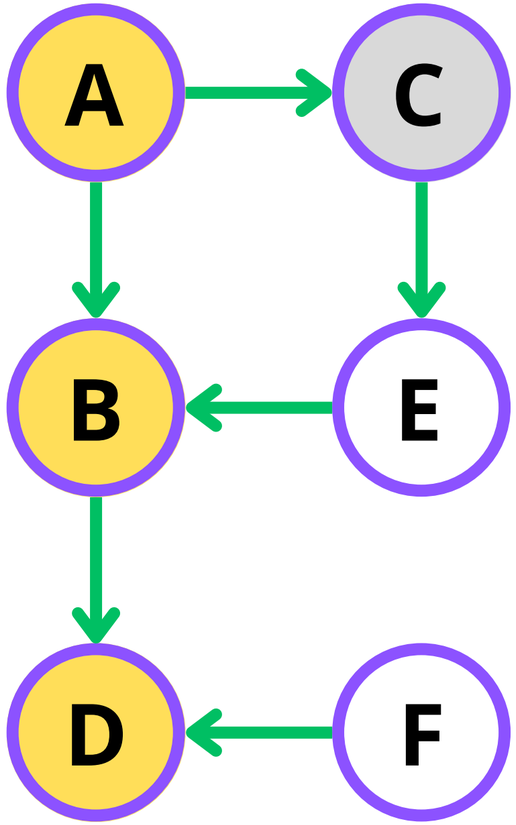
1. Empezamos de un nodo inicial
2. Procedemos a visitar sus vecinos que no hayan sido visitados (y volvemos a realizar el paso 2 con los vecinos del vecino recursivamente hasta que no podamos más)

ALGORITMOS DE BÚSQUEDA: DFS



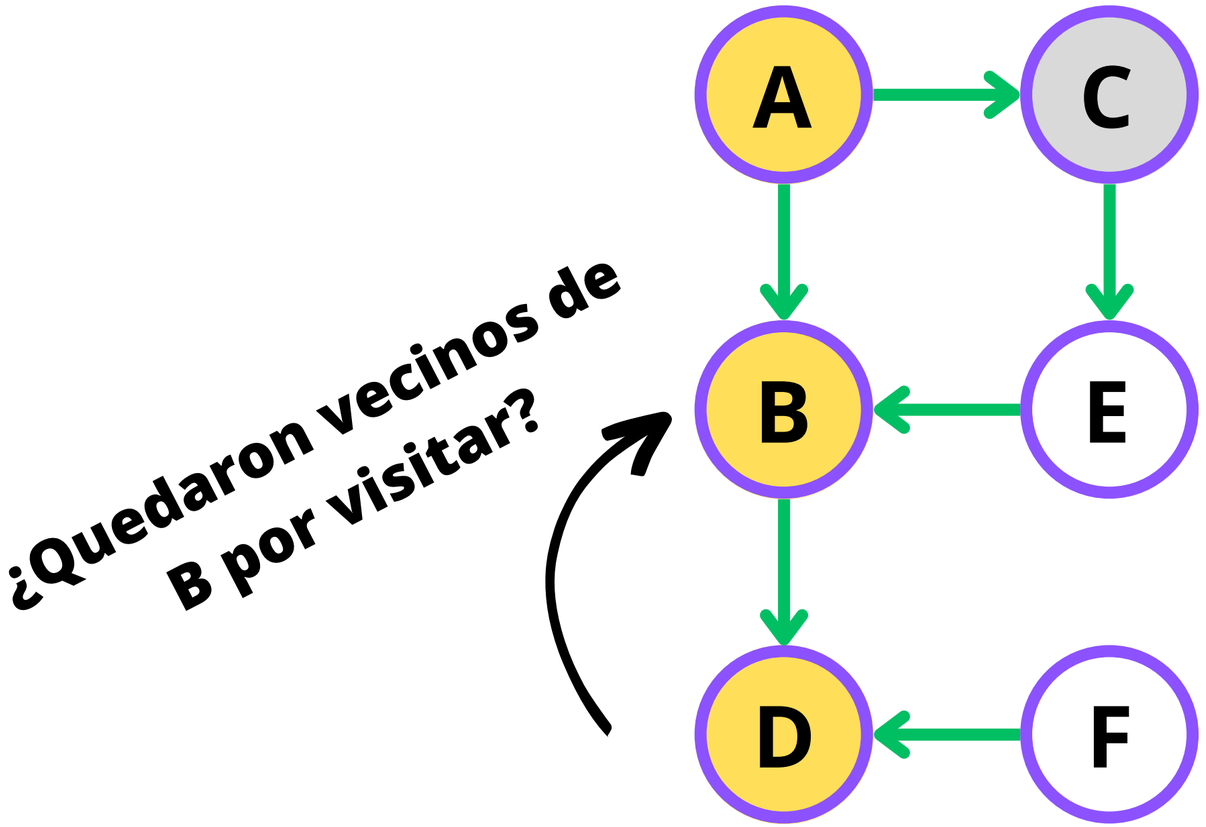
1. Empezamos de un nodo inicial
2. Procedemos a visitar sus vecinos que no hayan sido visitados (y volvemos a realizar el paso 2 con los vecinos del vecino recursivamente hasta que no podamos más)

ALGORITMOS DE BÚSQUEDA: DFS



1. Empezamos de un nodo inicial
2. Procedemos a visitar sus vecinos que no hayan sido visitados (y volvemos a realizar el paso 2 con los vecinos del vecino recursivamente hasta que no podamos más)

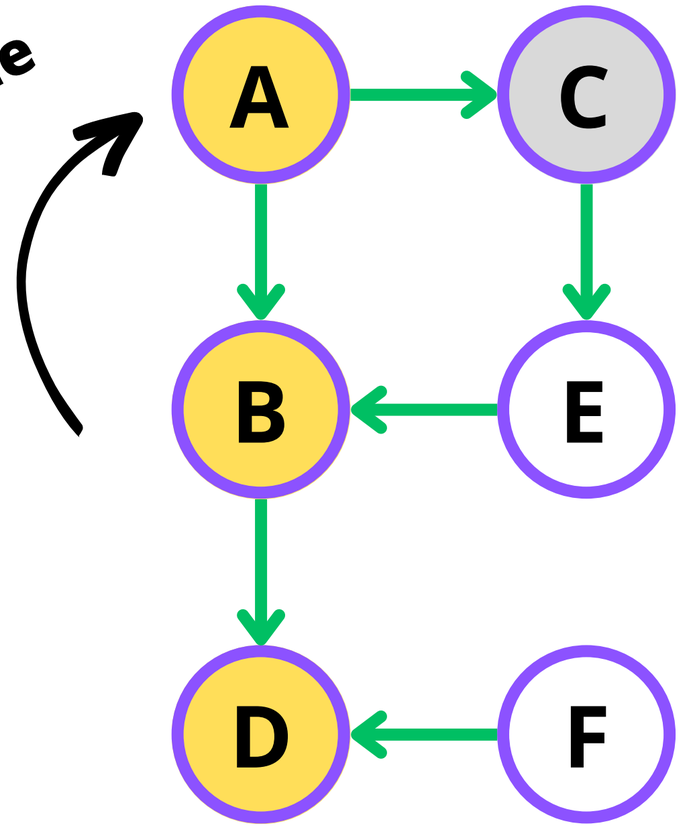
ALGORITMOS DE BÚSQUEDA: DFS



1. Empezamos de un nodo inicial
2. Procedemos a visitar sus vecinos que no hayan sido visitados (y volvemos a realizar el paso 2 con los vecinos del vecino recursivamente hasta que no podamos más)

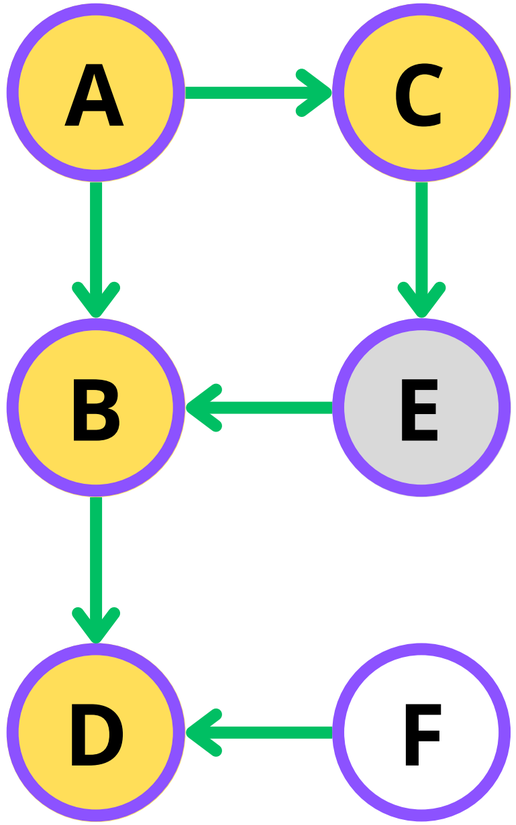
ALGORITMOS DE BÚSQUEDA: DFS

¿Quedaron vecinos de A por visitar?



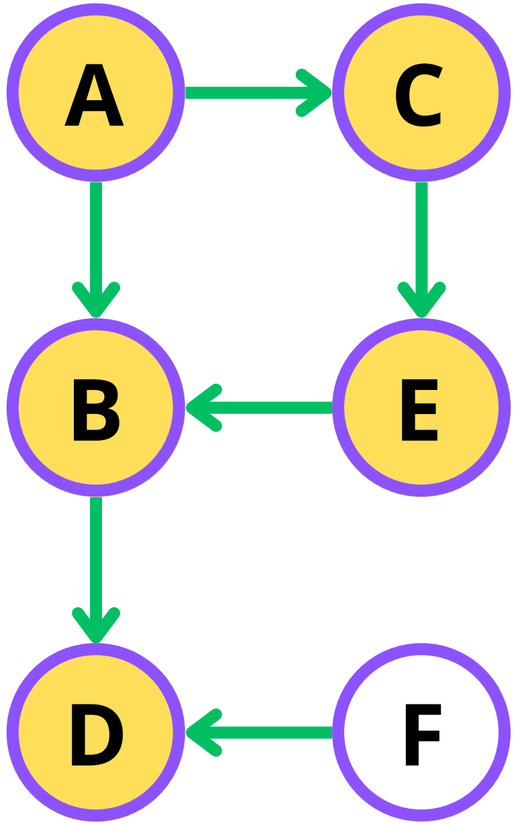
1. Empezamos de un nodo inicial
2. Procedemos a visitar sus vecinos que no hayan sido visitados (y volvemos a realizar el paso 2 con los vecinos del vecino recursivamente hasta que no podamos más)

ALGORITMOS DE BÚSQUEDA: DFS



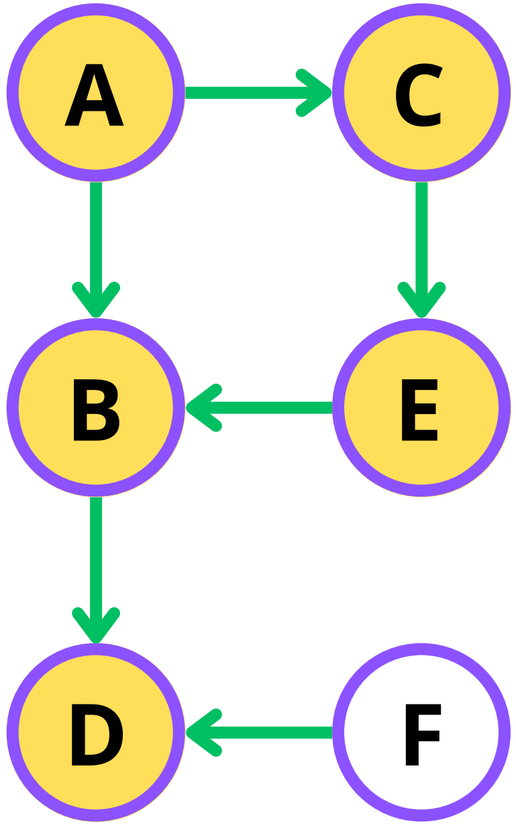
1. Empezamos de un nodo inicial
2. Procedemos a visitar sus vecinos que no hayan sido visitados (y volvemos a realizar el paso 2 con los vecinos del vecino recursivamente hasta que no podamos más)

ALGORITMOS DE BÚSQUEDA: DFS



1. Empezamos de un nodo inicial
2. Procedemos a visitar sus vecinos que no hayan sido visitados (y volvemos a realizar el paso 2 con los vecinos del vecino recursivamente hasta que no podamos más)

ALGORITMOS DE BÚSQUEDA: DFS



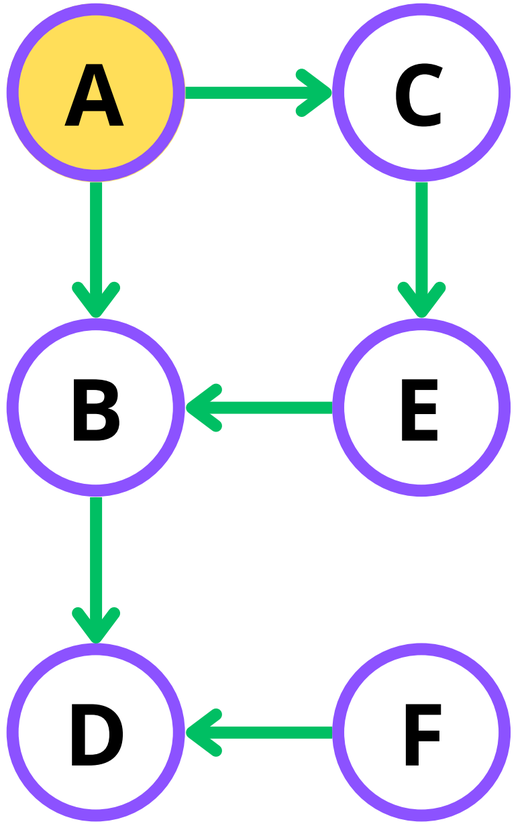
1. Empezamos de un nodo inicial
2. Procedemos a visitar sus vecinos que no hayan sido visitados (y volvemos a realizar el paso 2 con el vecino con sus vecinos recursivamente hasta que no podamos más)

A → B → D → C → E

ALGORITMOS DE BÚSQUEDA:

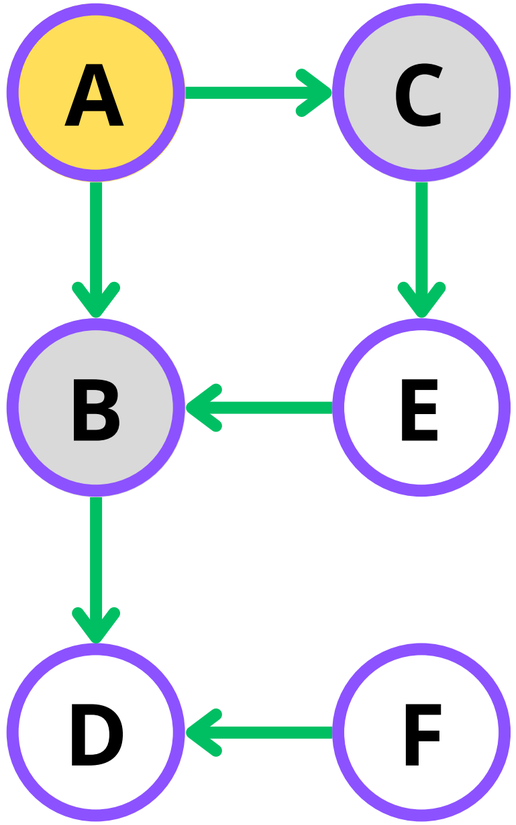
BFS

ALGORITMOS DE BÚSQUEDA: BFS



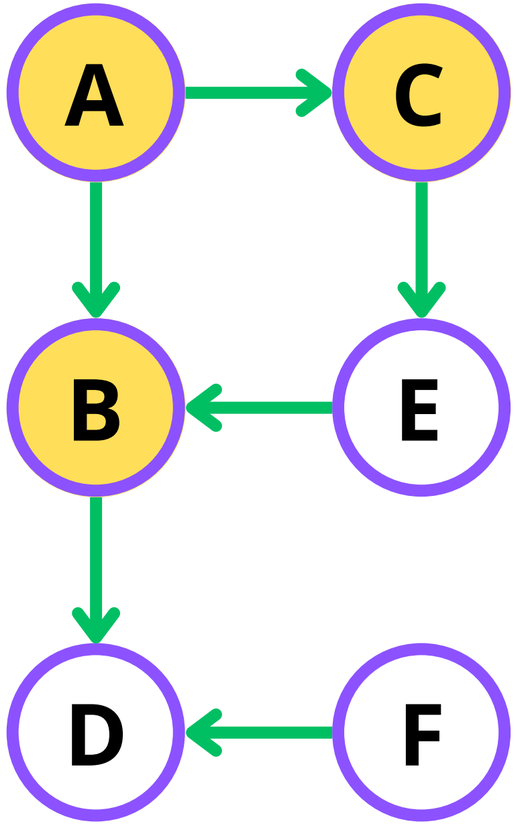
1. Empezamos de un nodo inicial

ALGORITMOS DE BÚSQUEDA: BFS



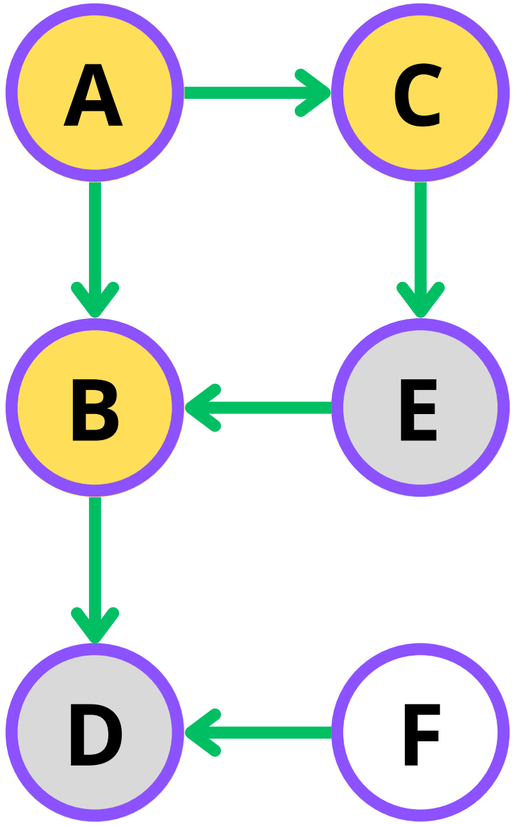
1. Empezamos de un nodo inicial
2. Procedemos a visitar todo sus vecinos que no hayan sido visitados y realizamos esto mismo para los vecinos que se visten hasta que termine

ALGORITMOS DE BÚSQUEDA: BFS



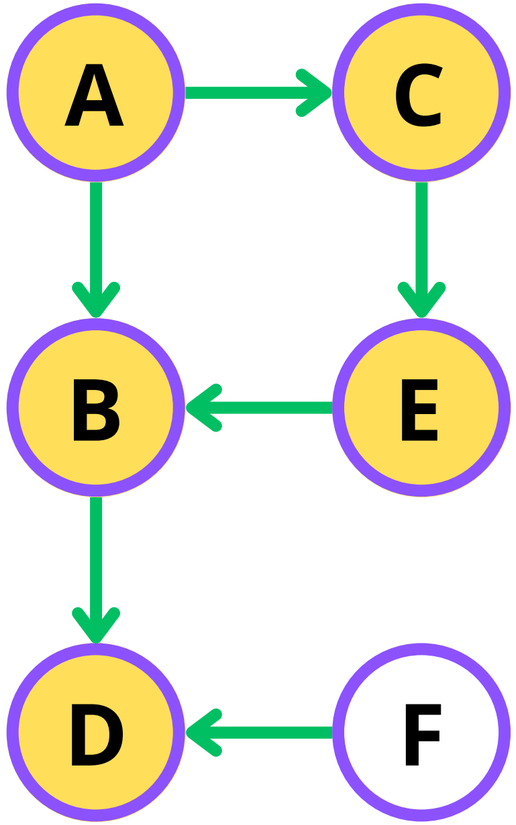
1. Empezamos de un nodo inicial
2. Procedemos a visitar todo sus vecinos que no hayan sido visitados y realizamos esto mismo para los vecinos que se visten hasta que termine

ALGORITMOS DE BÚSQUEDA: BFS



1. Empezamos de un nodo inicial
2. Procedemos a visitar todo sus vecinos que no hayan sido visitados y realizamos esto mismo para los vecinos que se visten hasta que termine

ALGORITMOS DE BÚSQUEDA: BFS



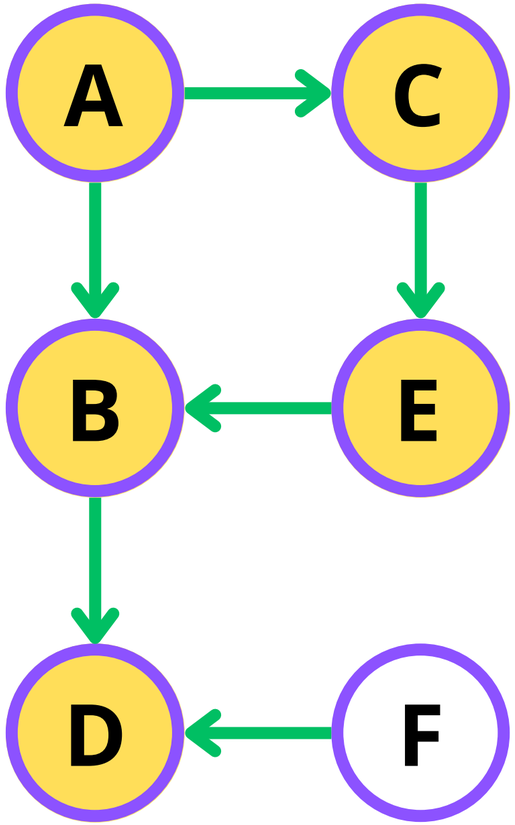
1. Empezamos de un nodo inicial
2. Procedemos a visitar todo sus vecinos que no hayan sido visitados y realizamos esto mismo para los vecinos que se visten hasta que termine

A → B → C → D → E

ALGORITMOS DE BÚSQUEDA: BFS y DFS

DFS

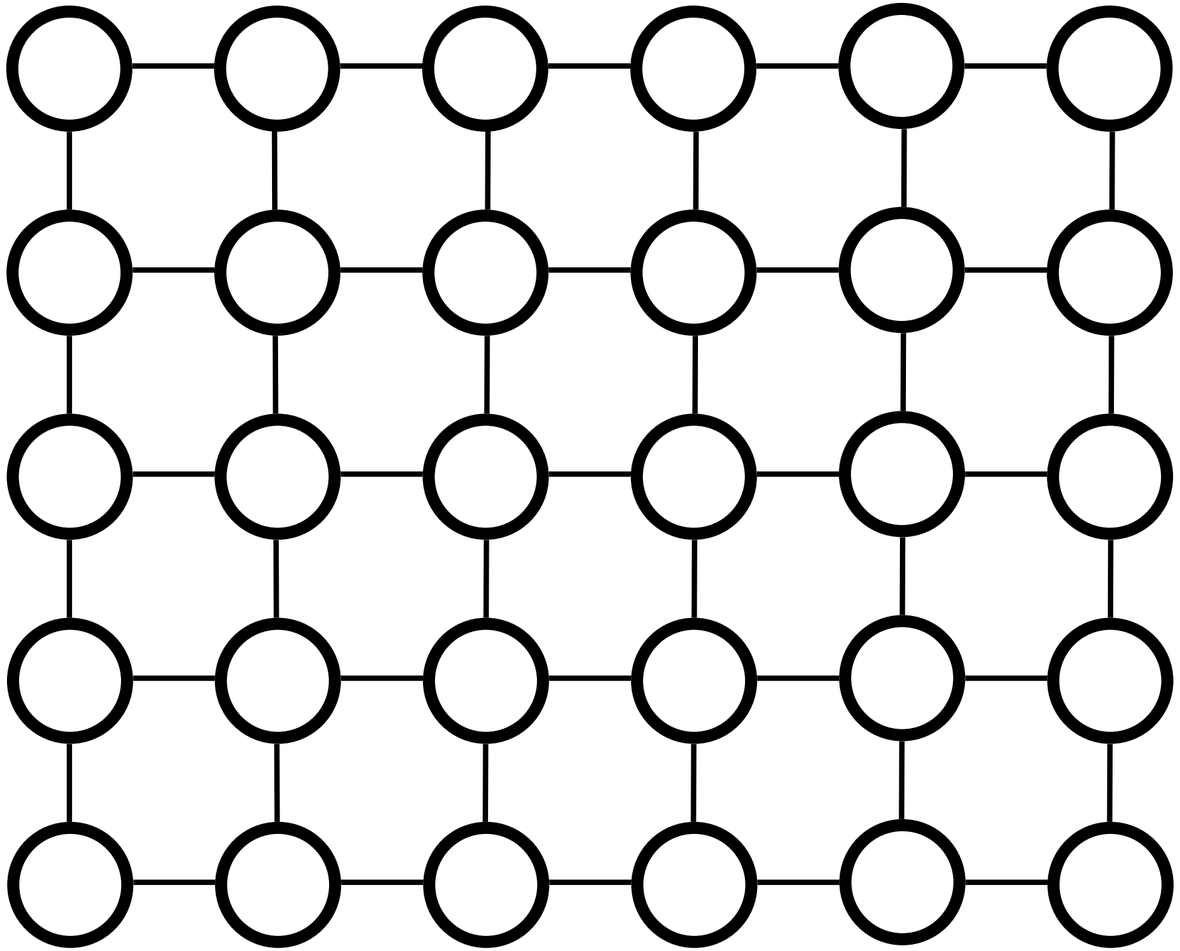
A → B → D → C → E



BFS

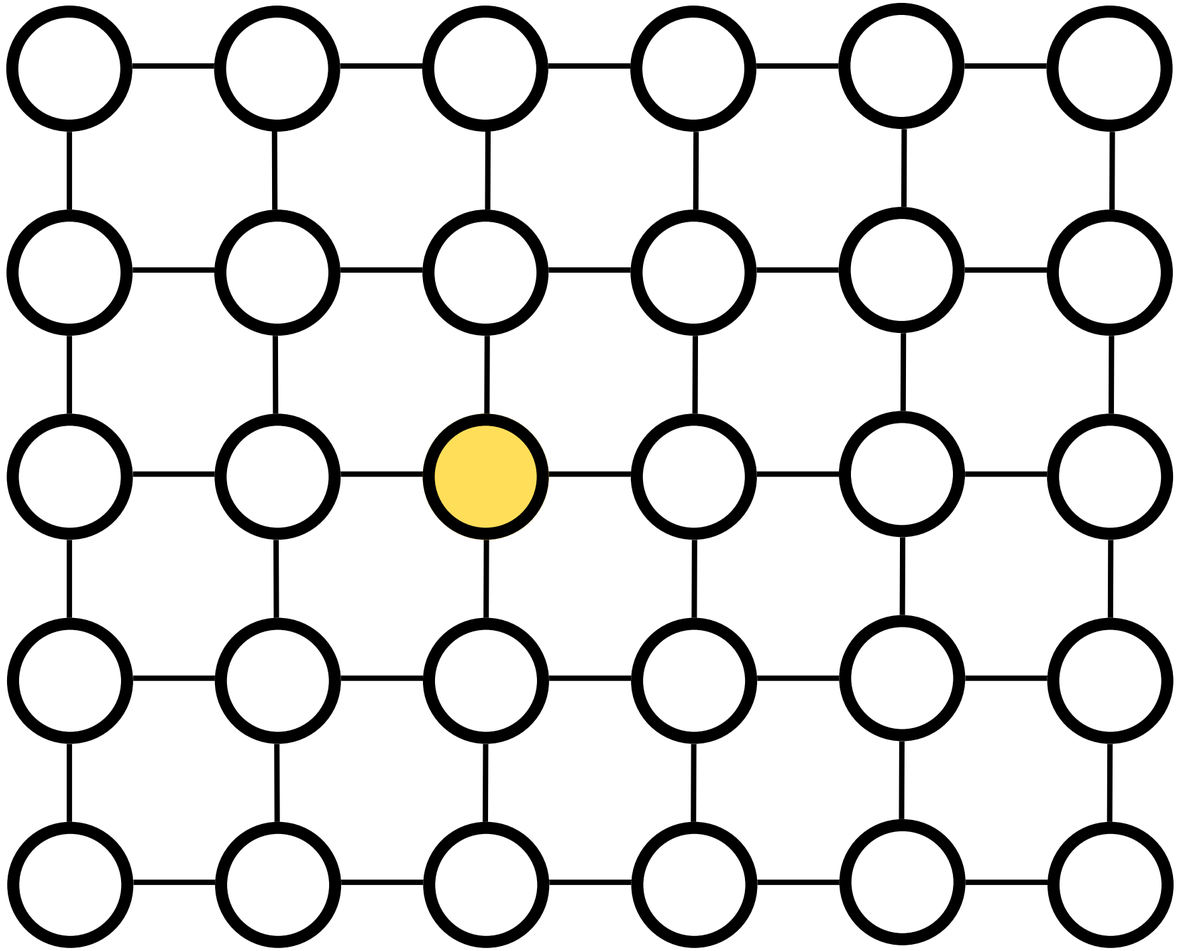
A → B → C → D → E

ALGORITMOS DE BÚSQUEDA: BFS y DFS



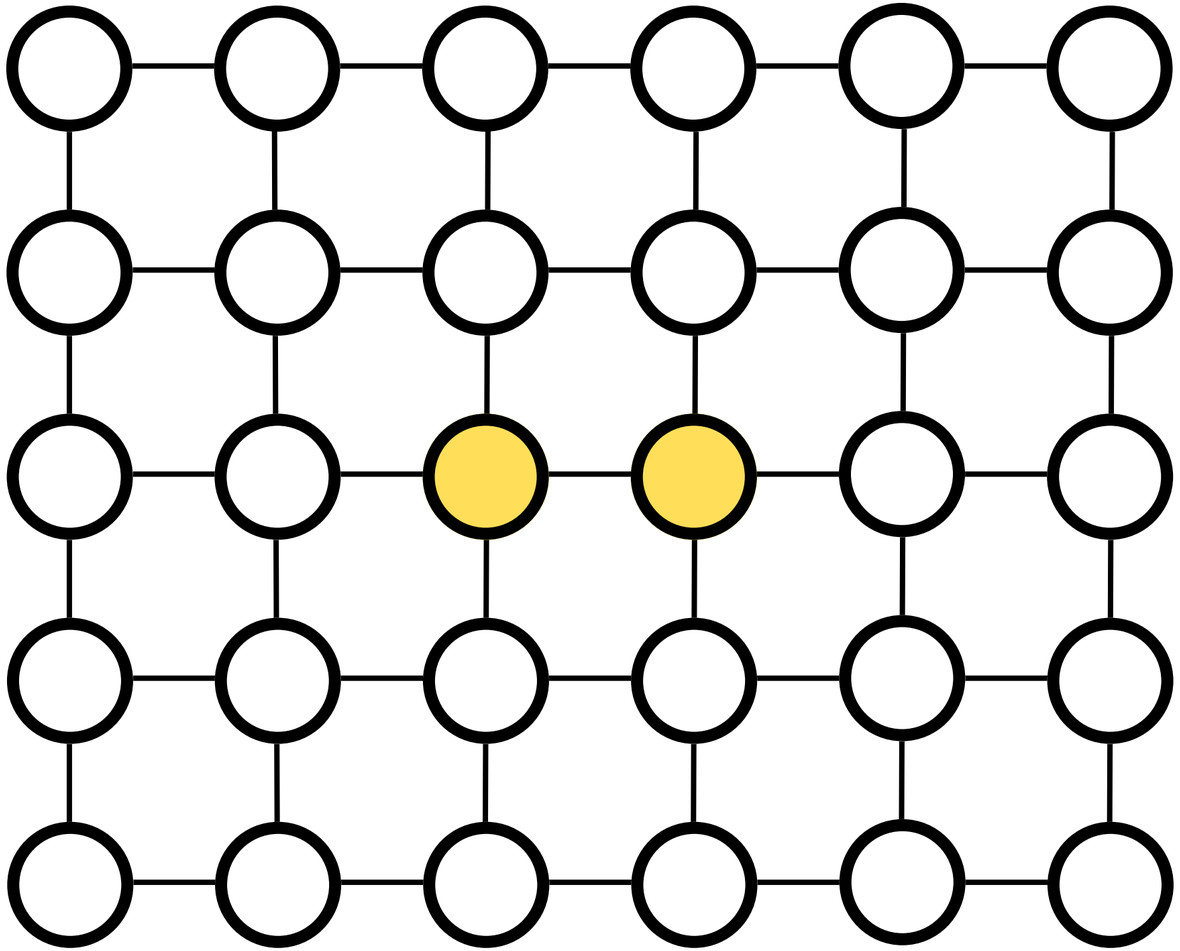
ALGORITMOS DE BÚSQUEDA: BFS y DFS

DFS



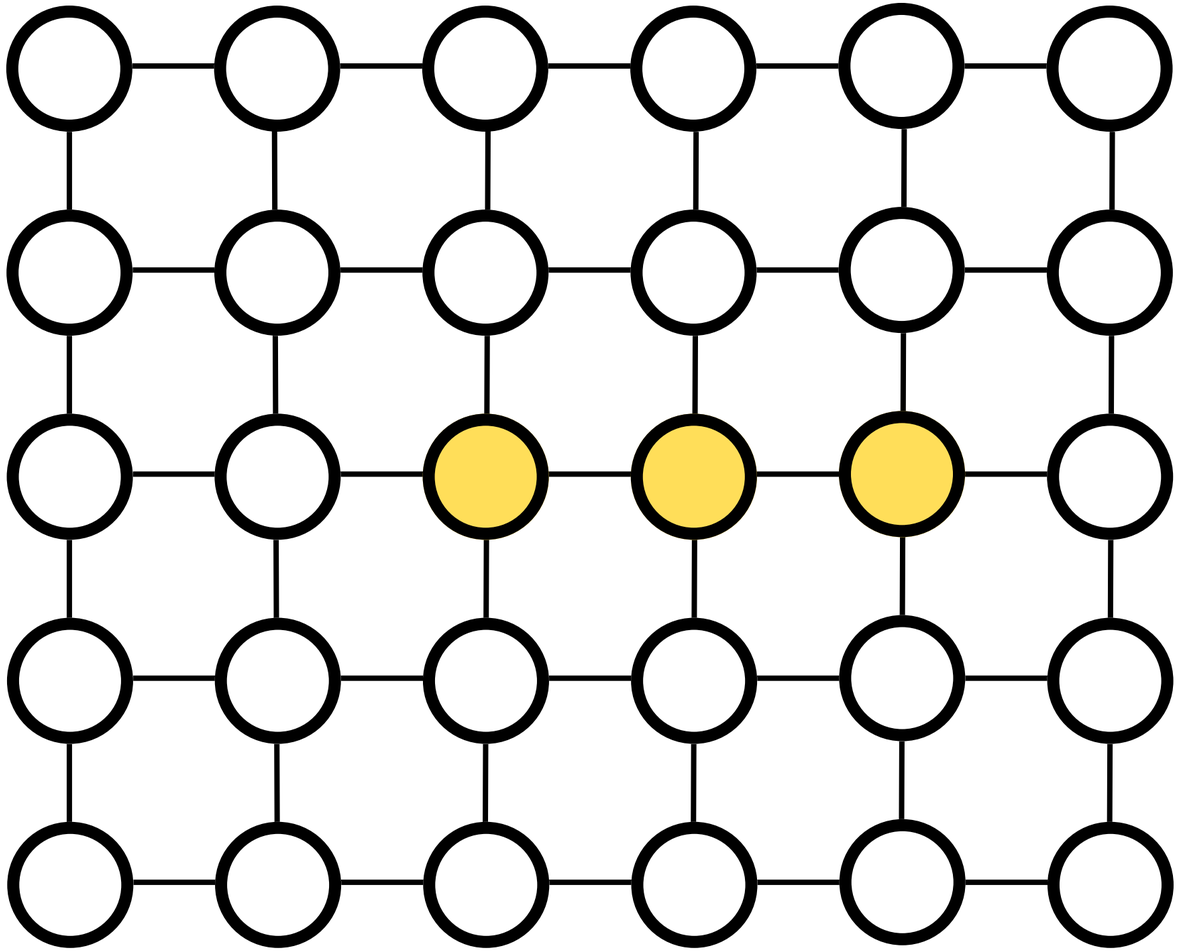
ALGORITMOS DE BÚSQUEDA: BFS y DFS

DFS



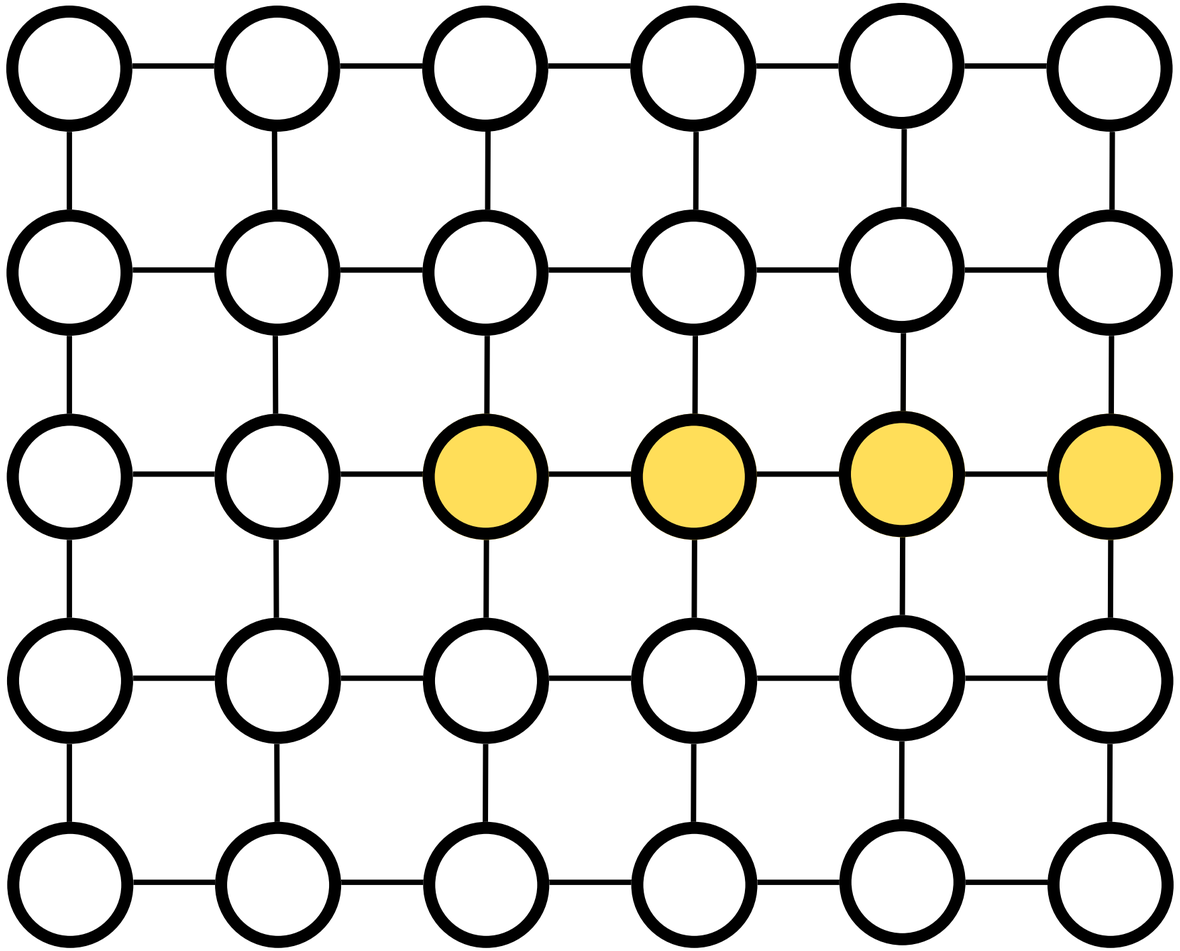
ALGORITMOS DE BÚSQUEDA: BFS y DFS

DFS



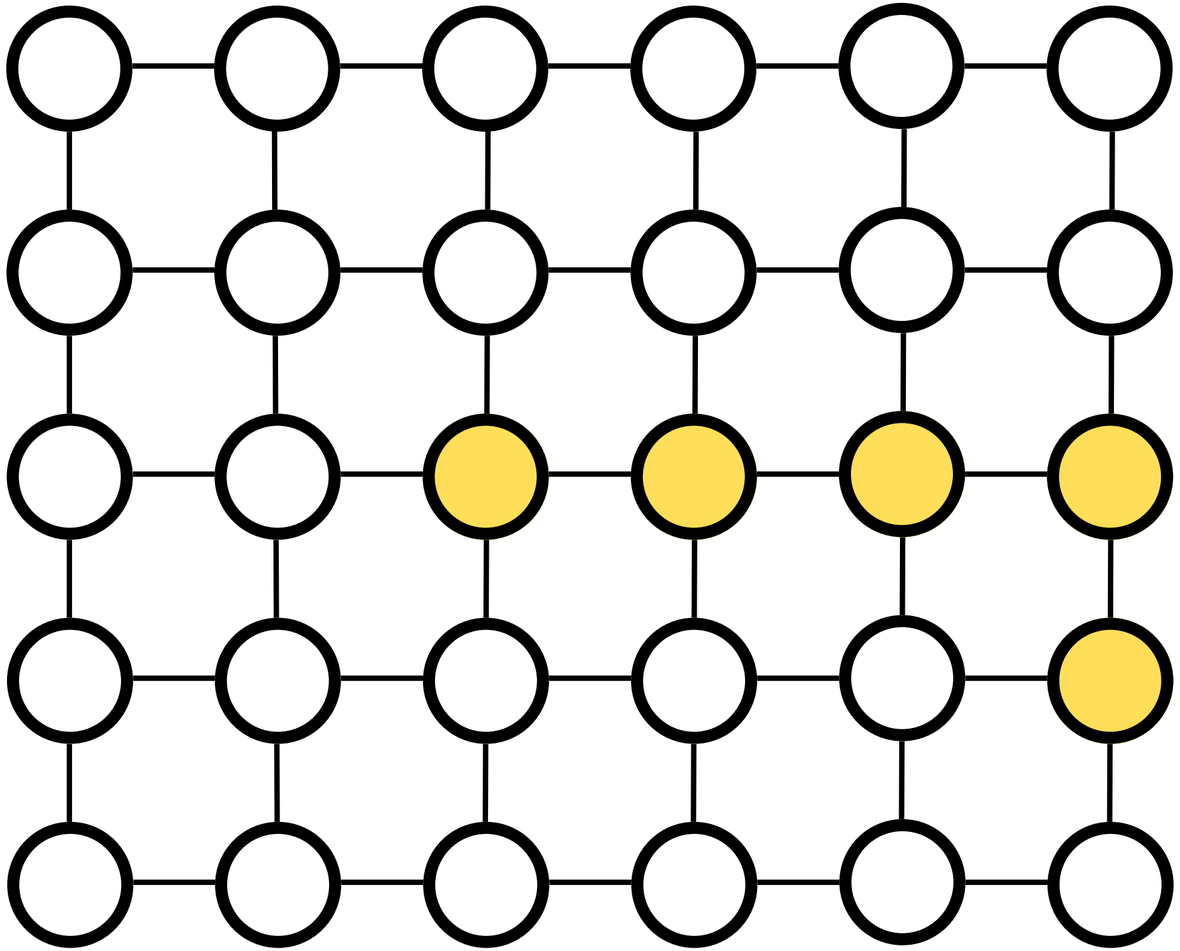
ALGORITMOS DE BÚSQUEDA: BFS y DFS

DFS



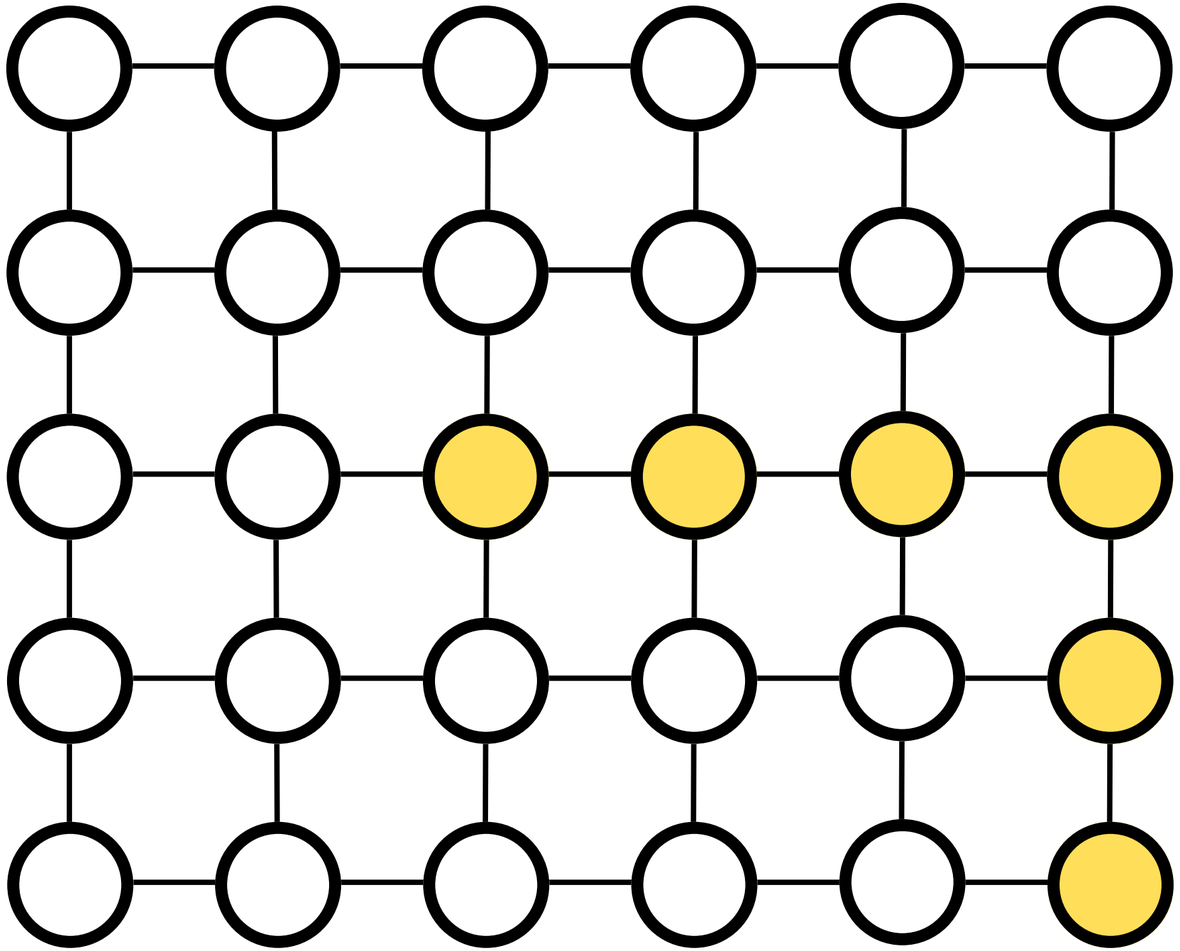
ALGORITMOS DE BÚSQUEDA: BFS y DFS

DFS



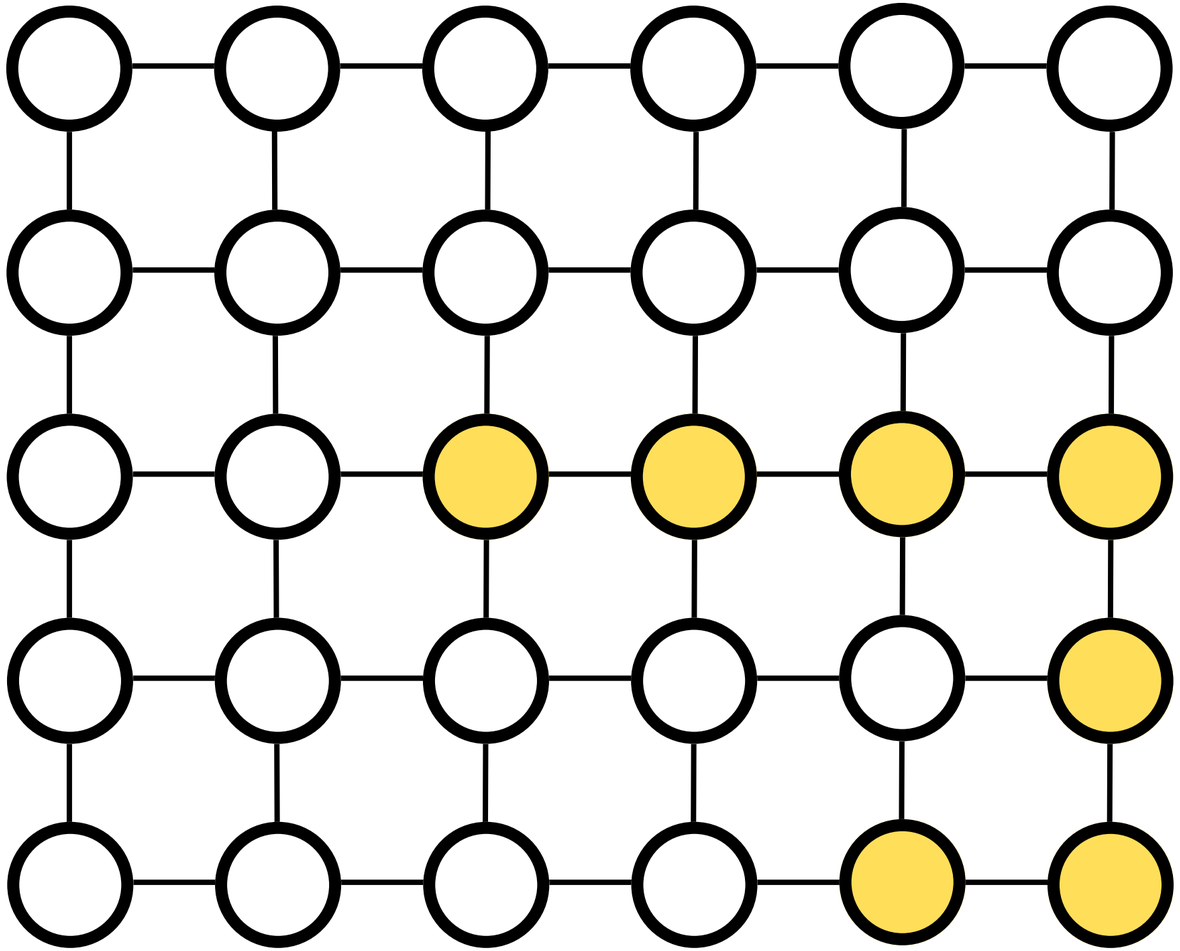
ALGORITMOS DE BÚSQUEDA: BFS y DFS

DFS



ALGORITMOS DE BÚSQUEDA: BFS y DFS

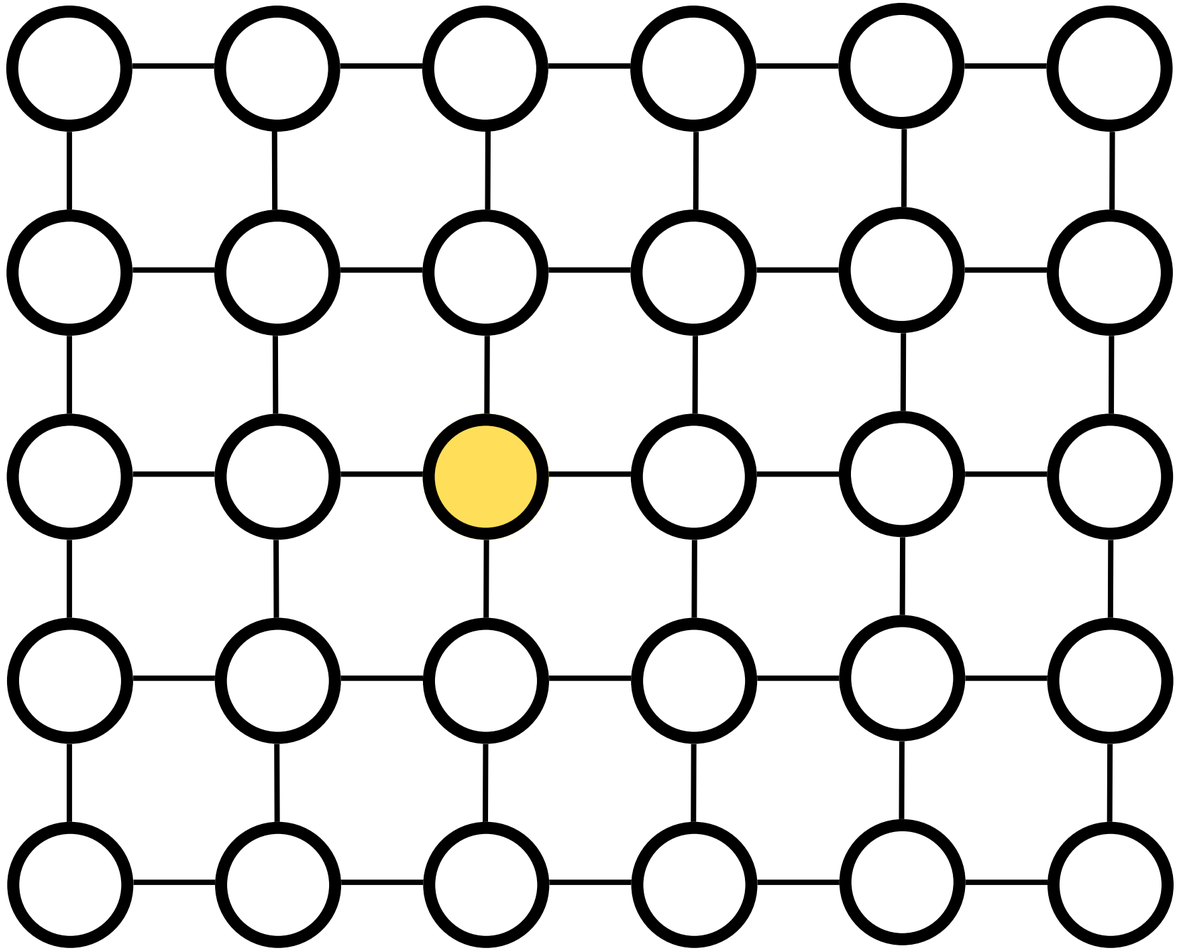
DFS



Así continuará hasta que termine de visitar todo

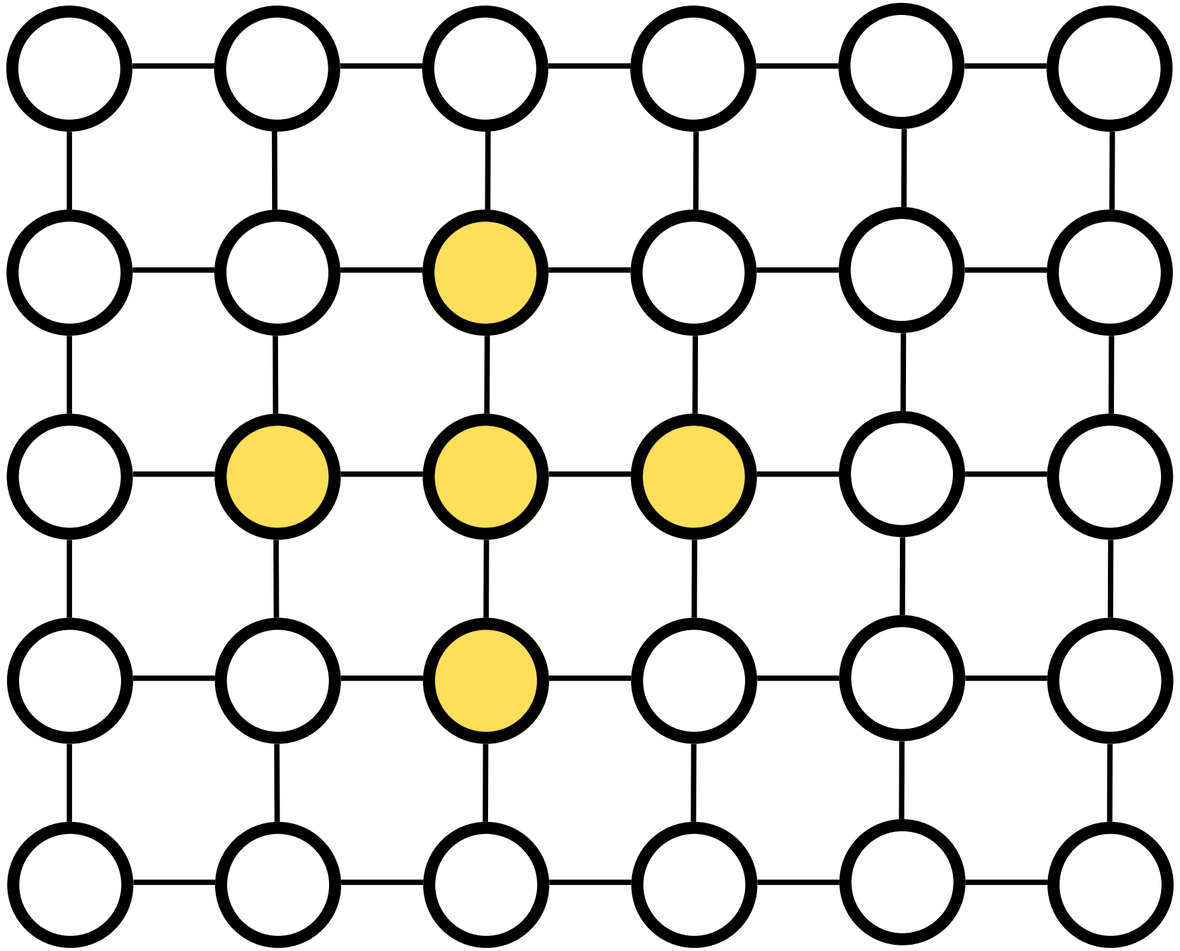
ALGORITMOS DE BÚSQUEDA: BFS y DFS

BFS



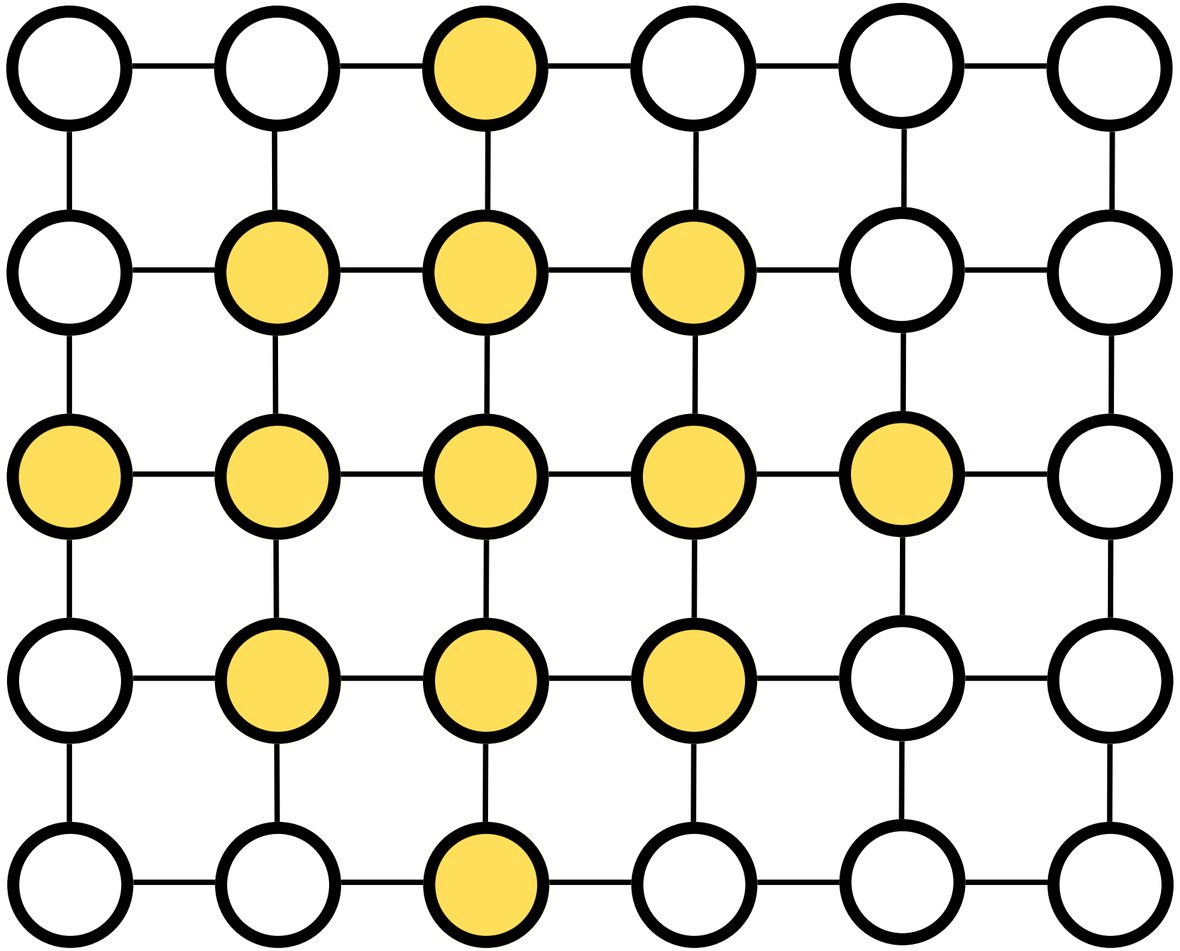
ALGORITMOS DE BÚSQUEDA: BFS y DFS

BFS



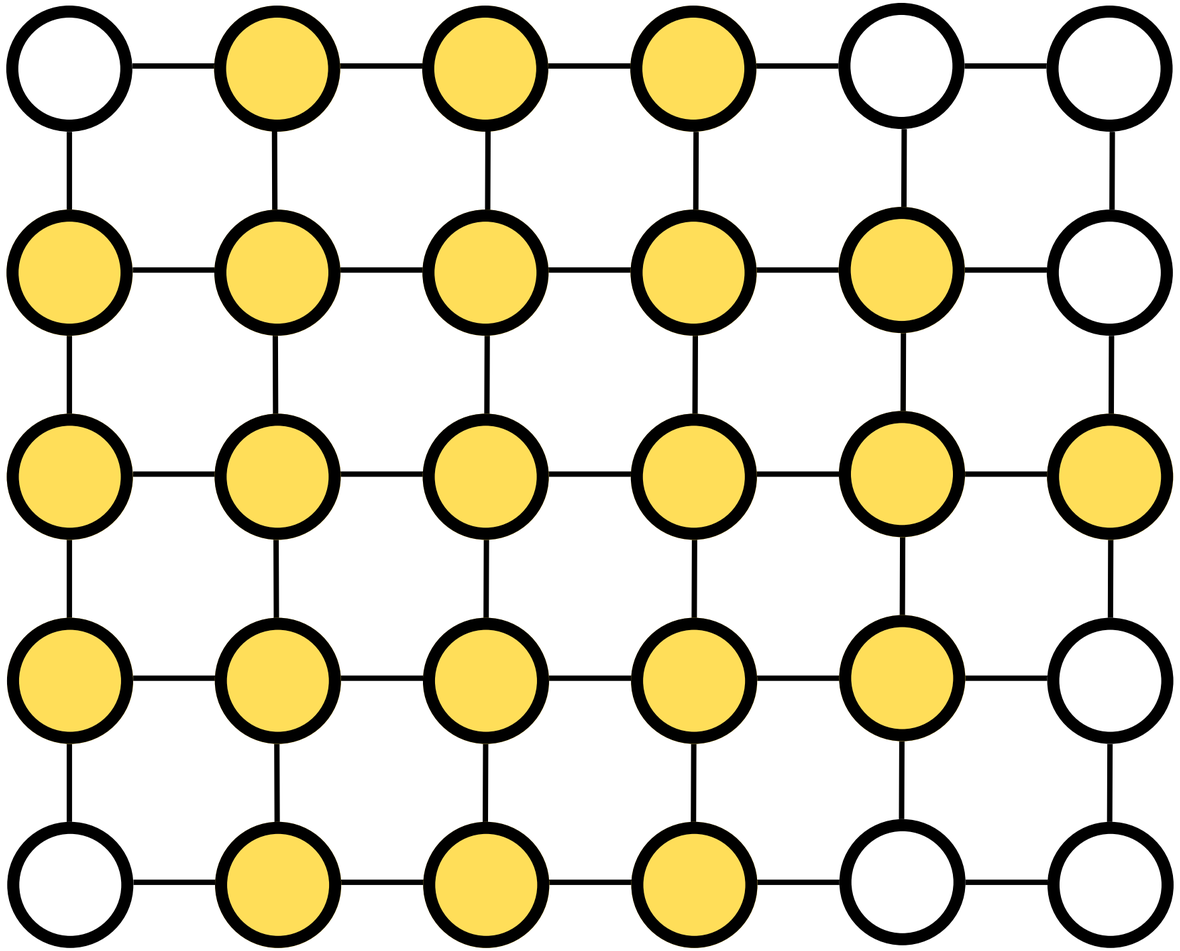
ALGORITMOS DE BÚSQUEDA: BFS y DFS

BFS



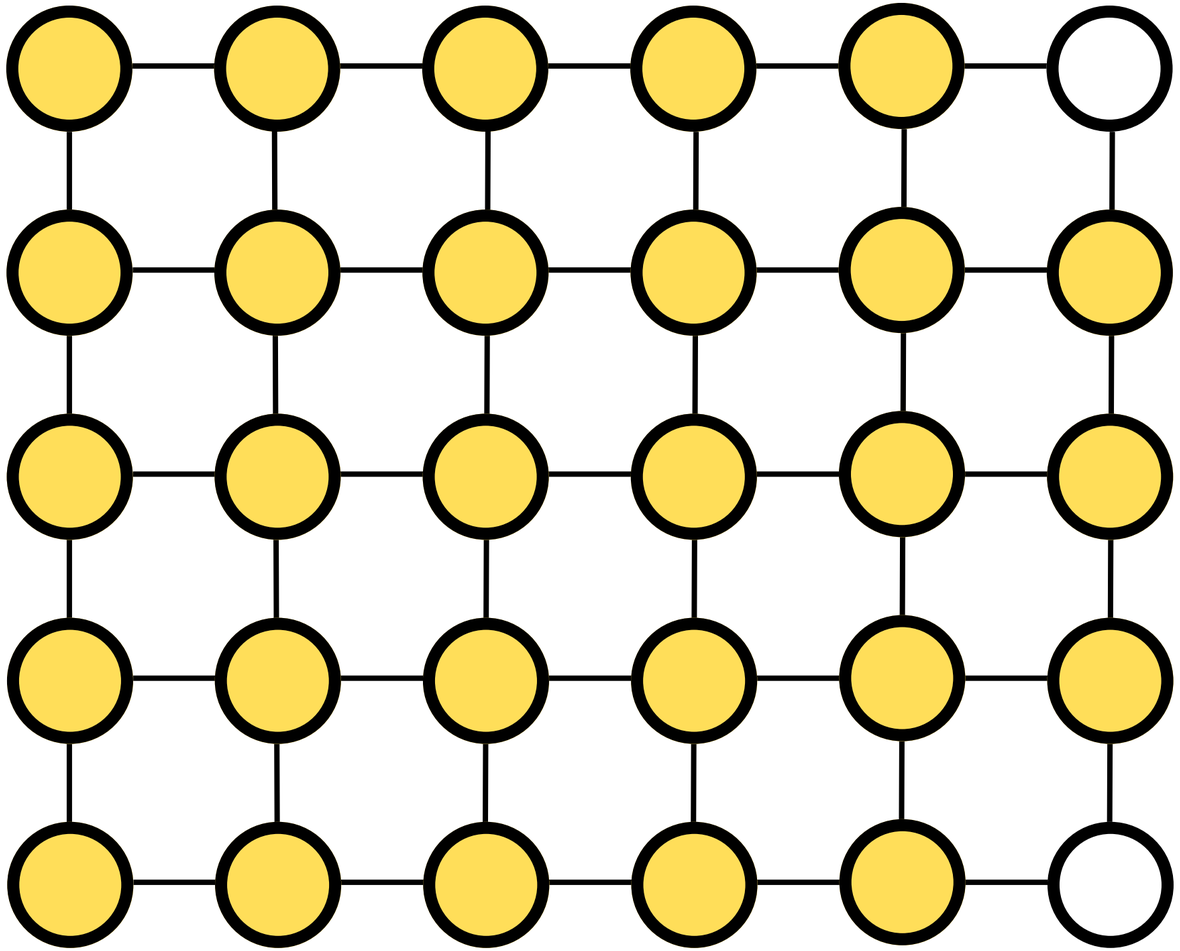
ALGORITMOS DE BÚSQUEDA: BFS y DFS

BFS



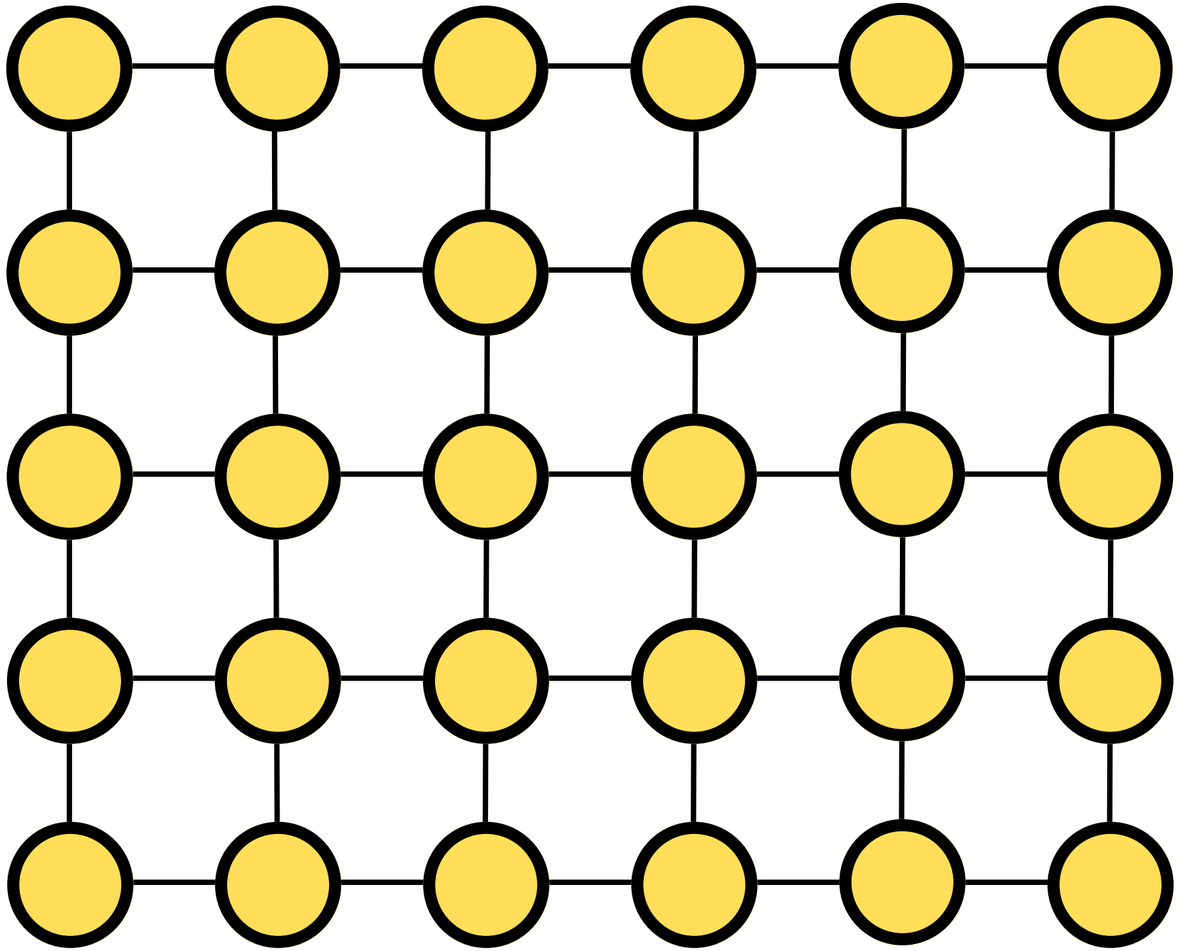
ALGORITMOS DE BÚSQUEDA: BFS y DFS

BFS



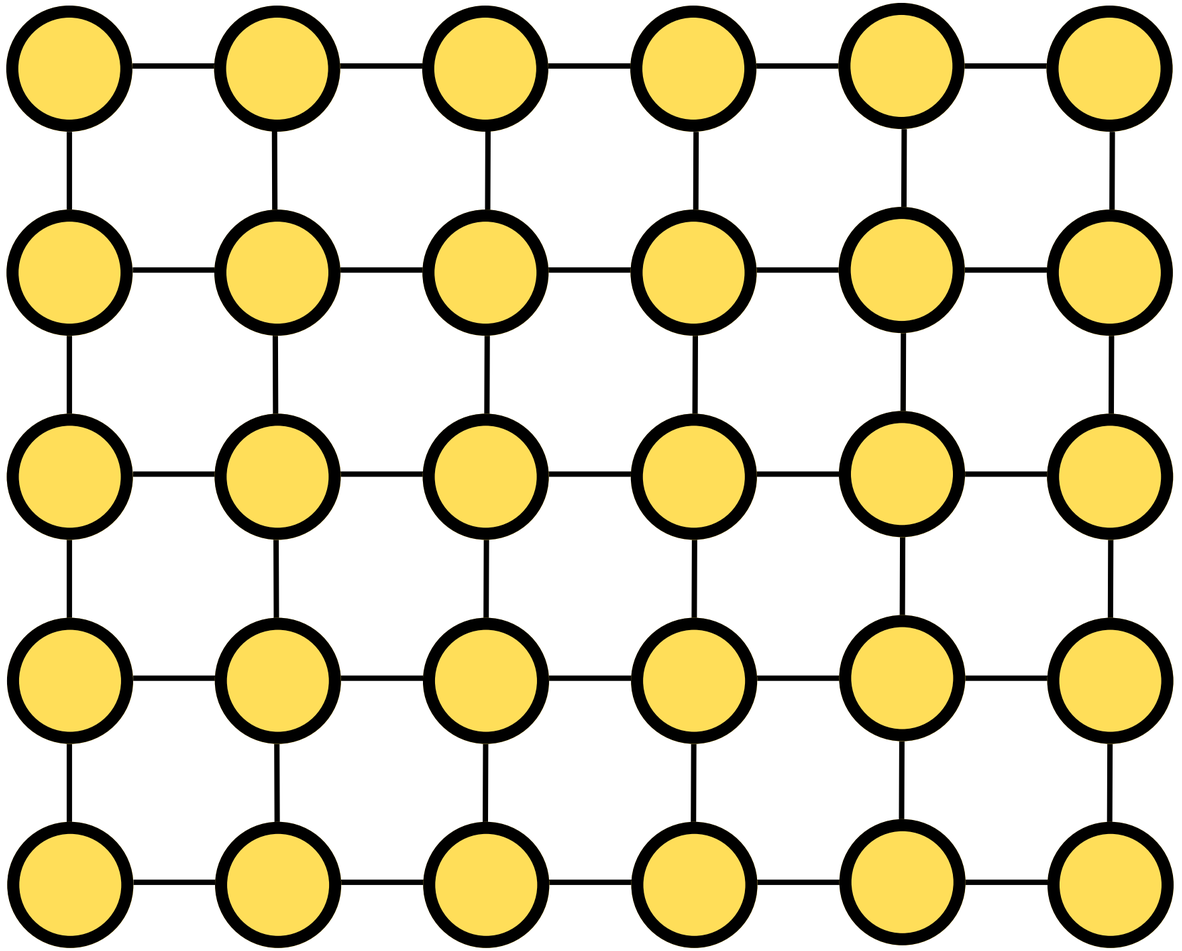
ALGORITMOS DE BÚSQUEDA: BFS y DFS

BFS



ALGORITMOS DE BÚSQUEDA: BFS y DFS

BFS



IMPLEMENTACIÓN

DFS

BFS

DFS

Stack

BFS

DFS

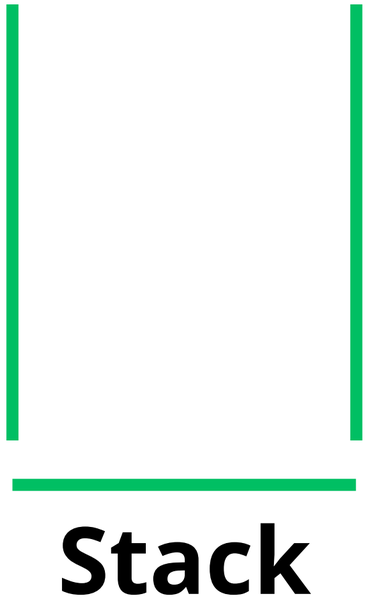
Stack

BFS

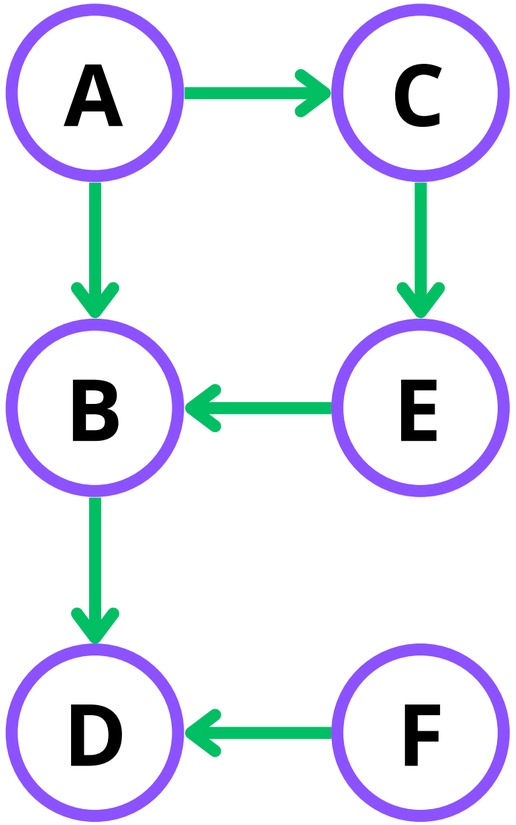
Queue

DFS

Recorrido esperado: A → B → D → C → E



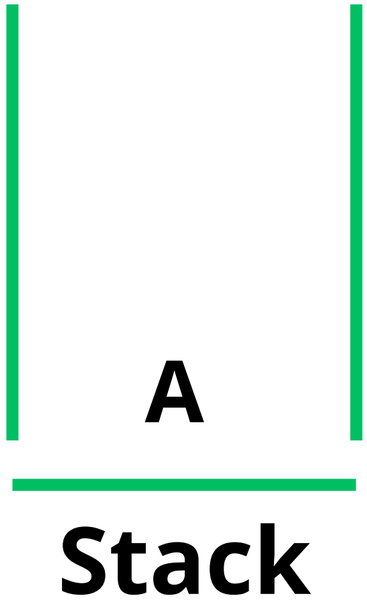
Actual

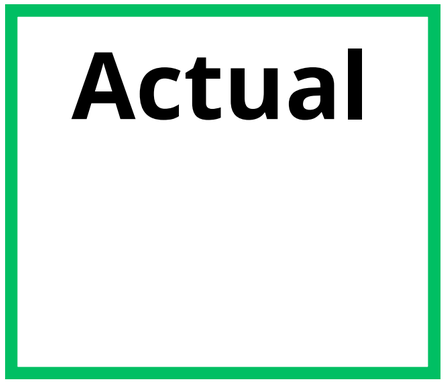


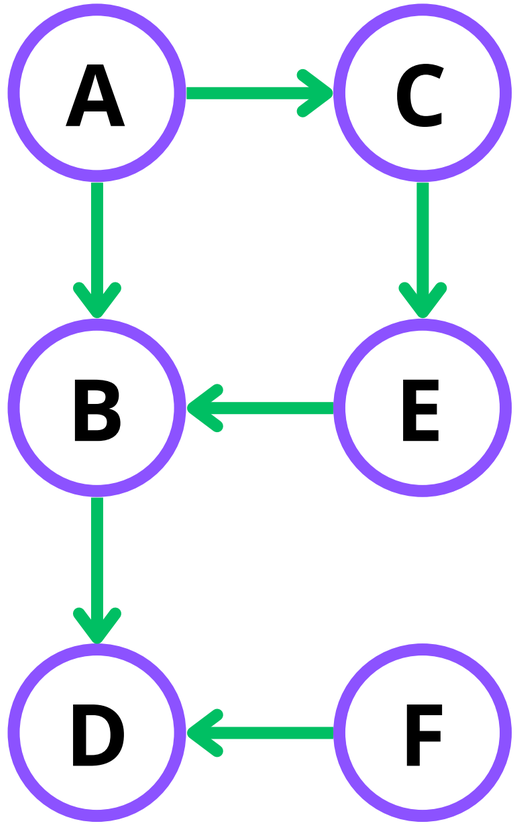
Recorrido actual:

DFS

Recorrido esperado: A → B → D → C → E



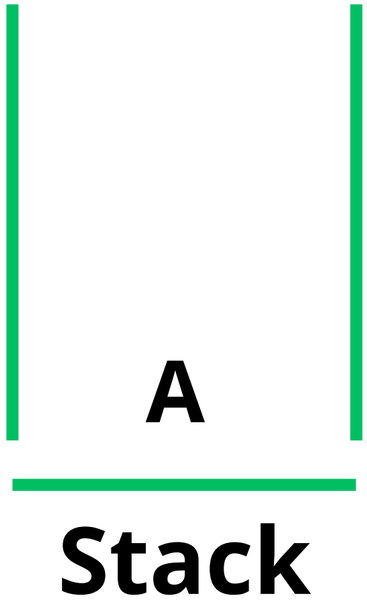


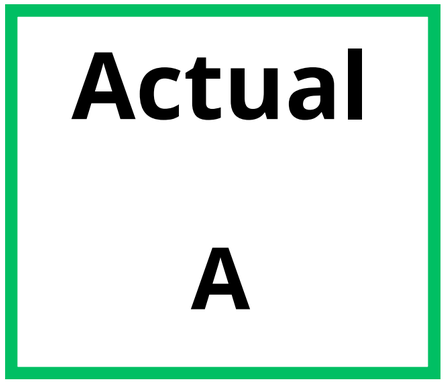


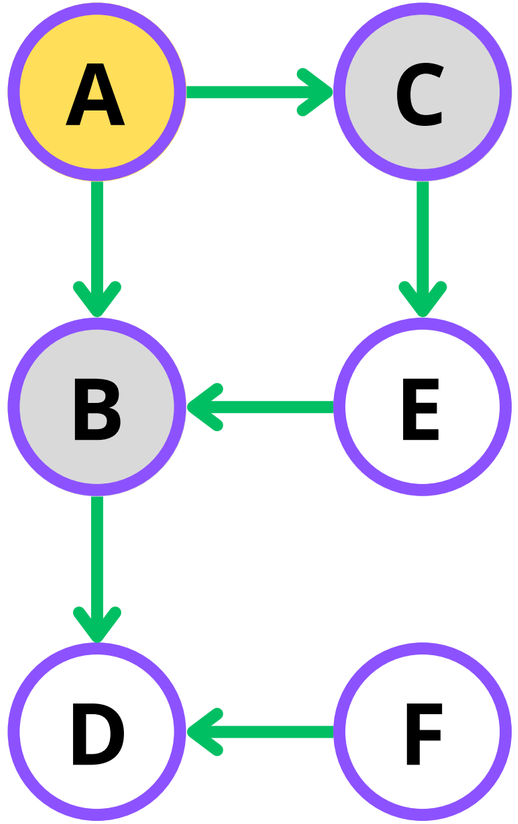
Recorrido actual:

DFS

Recorrido esperado: A → B → D → C → E



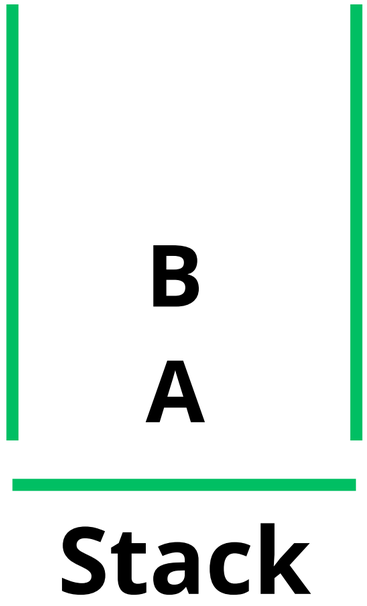


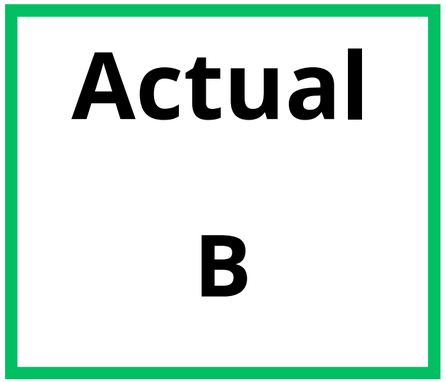


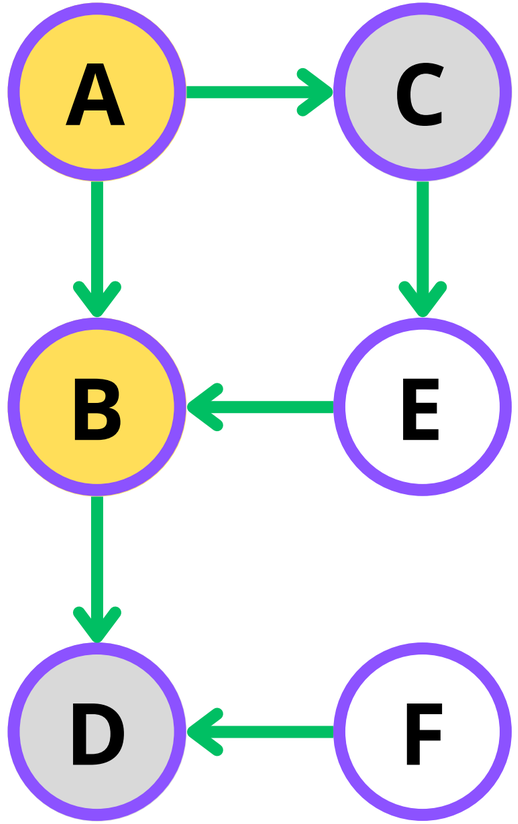
Recorrido actual: A

DFS

Recorrido esperado: A → B → D → C → E



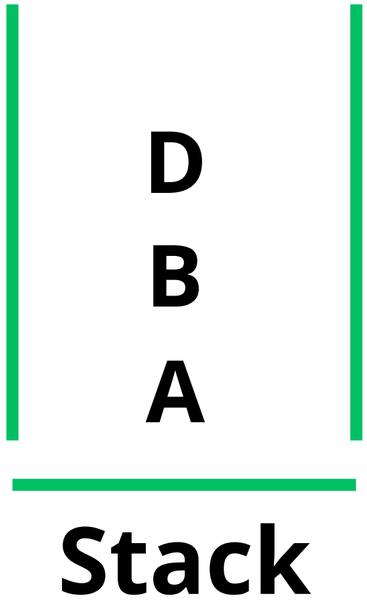


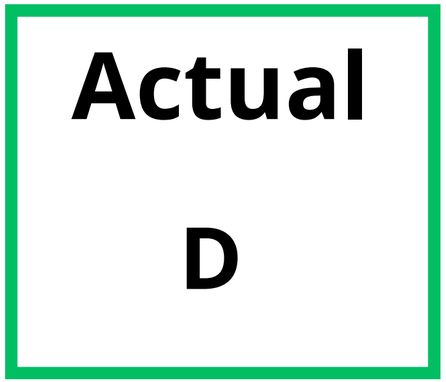


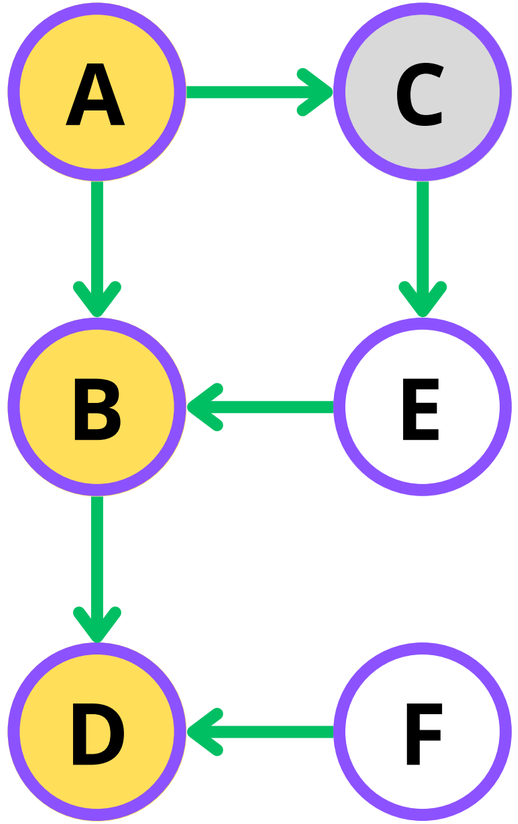
Recorrido actual: A → B

DFS

Recorrido esperado: A → B → D → C → E



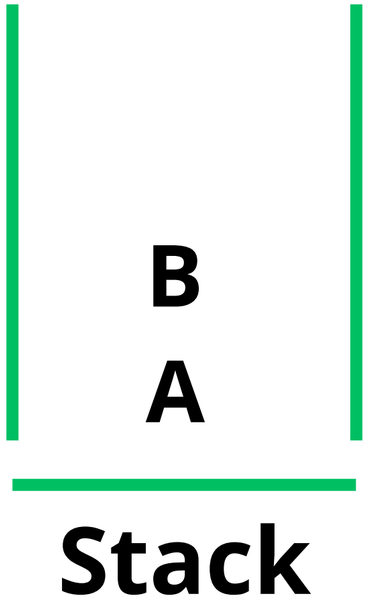


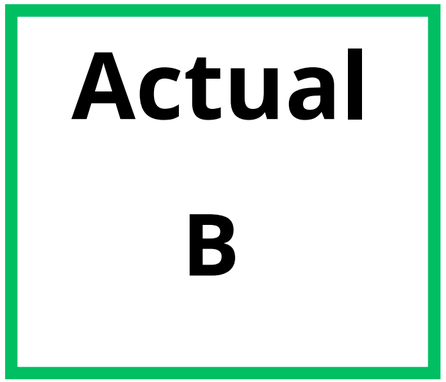


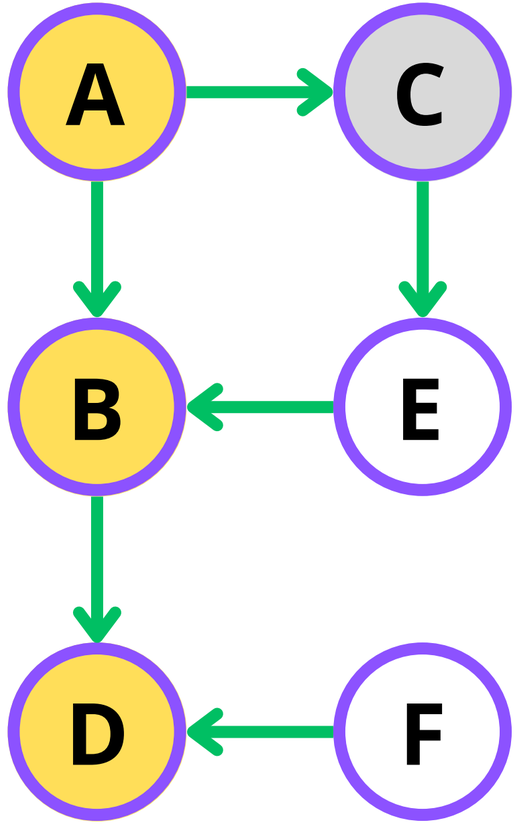
Recorrido actual: A → B → D

DFS

Recorrido esperado: A → B → D → C → E



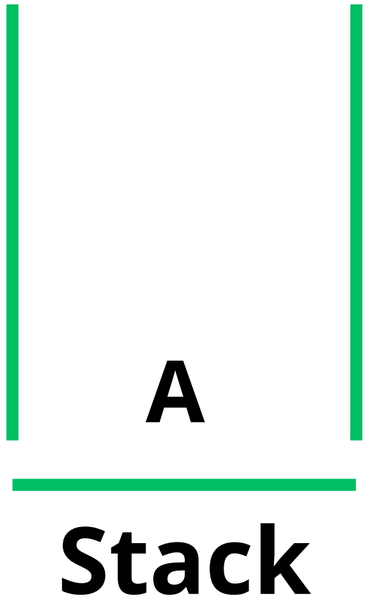


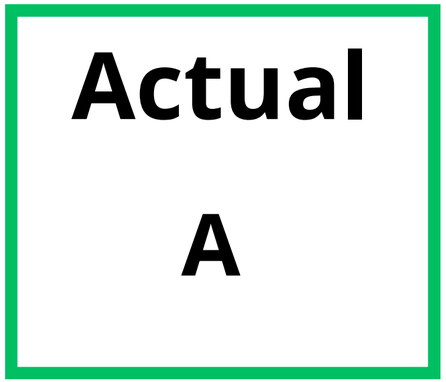


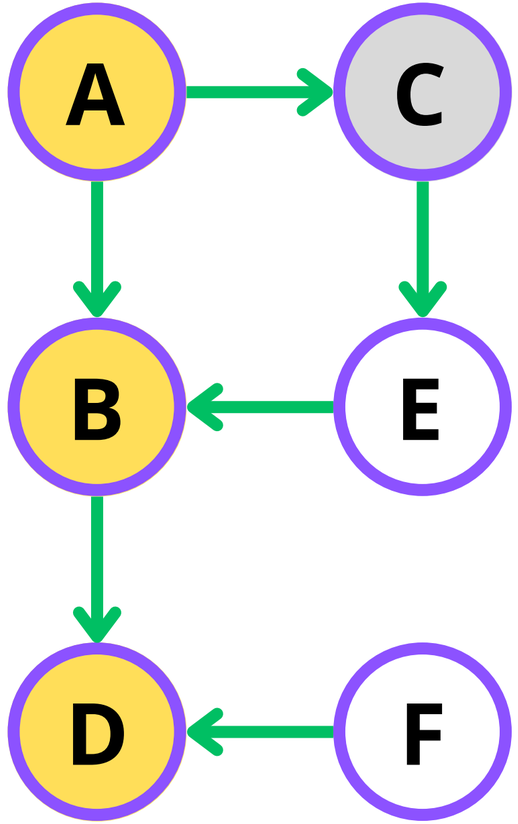
Recorrido actual: A → B → D

DFS

Recorrido esperado: A → B → D → C → E



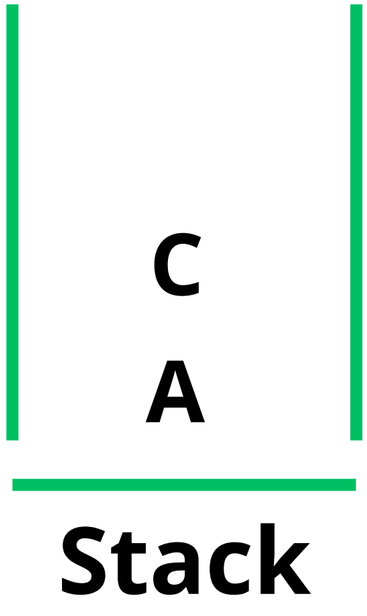


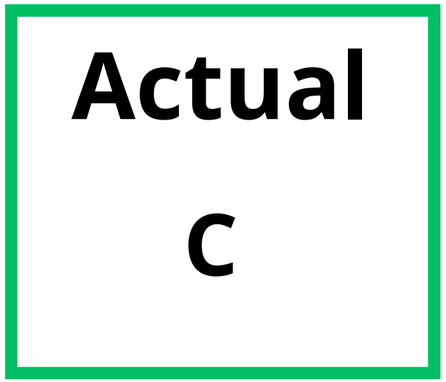


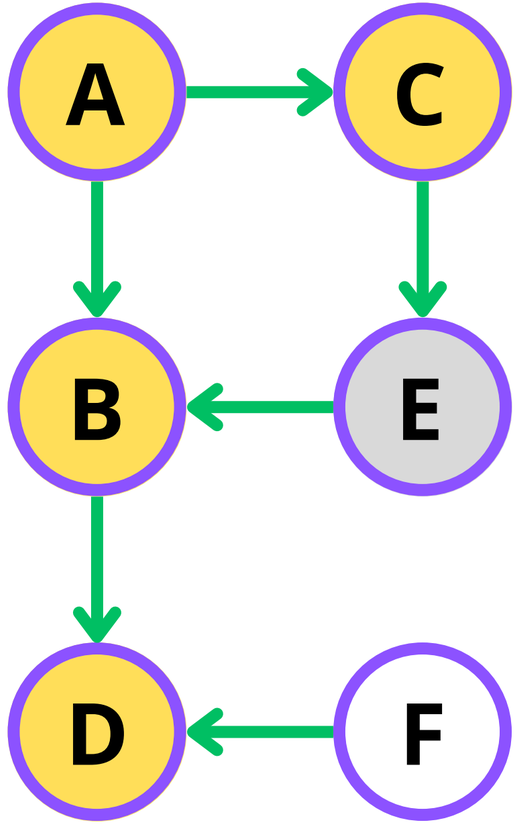
Recorrido actual: A → B → D

DFS

Recorrido esperado: A → B → D → C → E



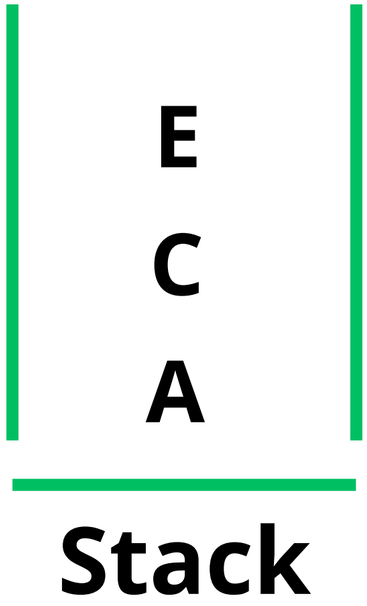


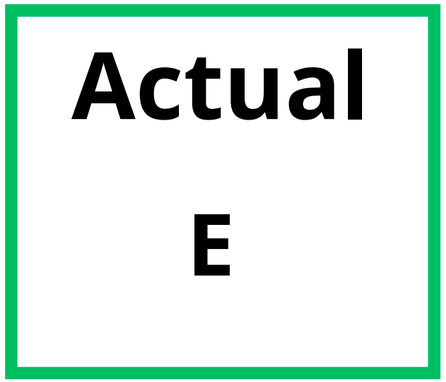


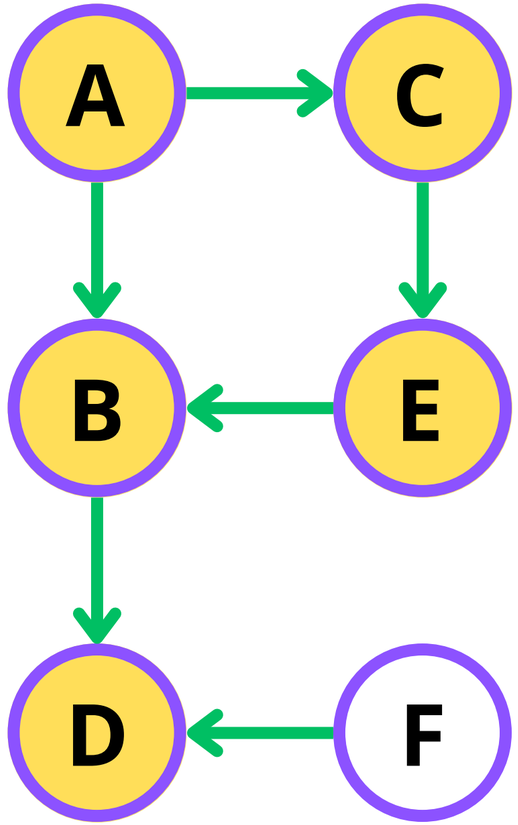
Recorrido actual: A → B → D → C

DFS

Recorrido esperado: A → B → D → C → E



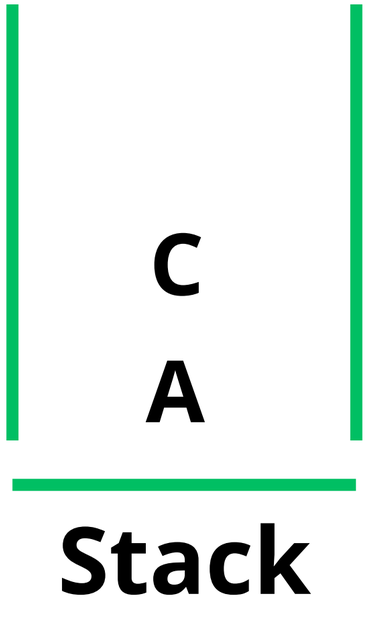


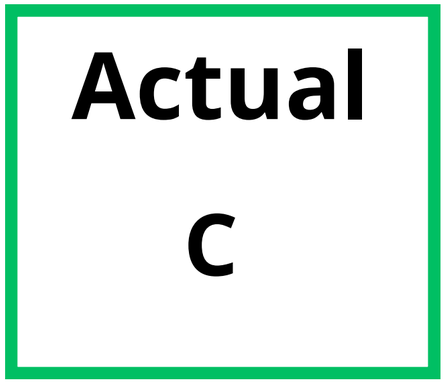


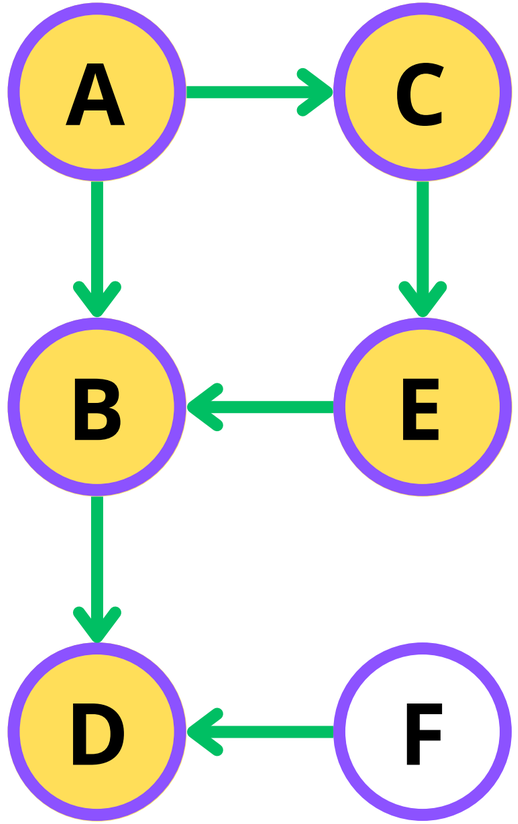
Recorrido actual: A → B → D → C → E

DFS

Recorrido esperado: A → B → D → C → E



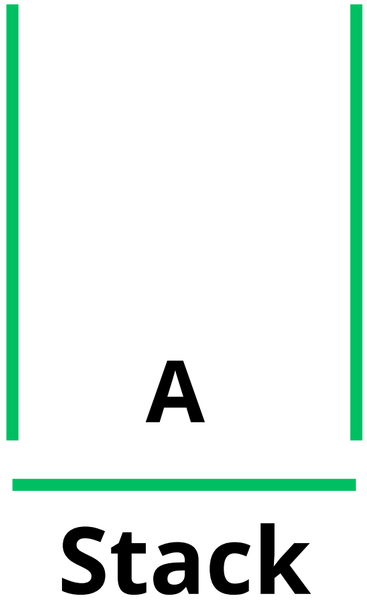


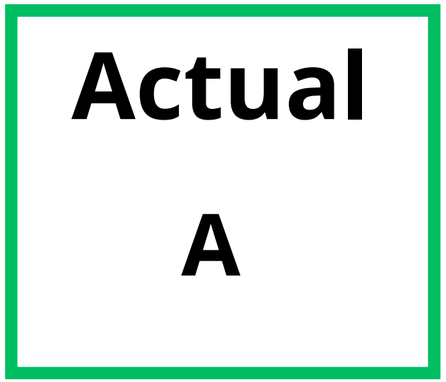


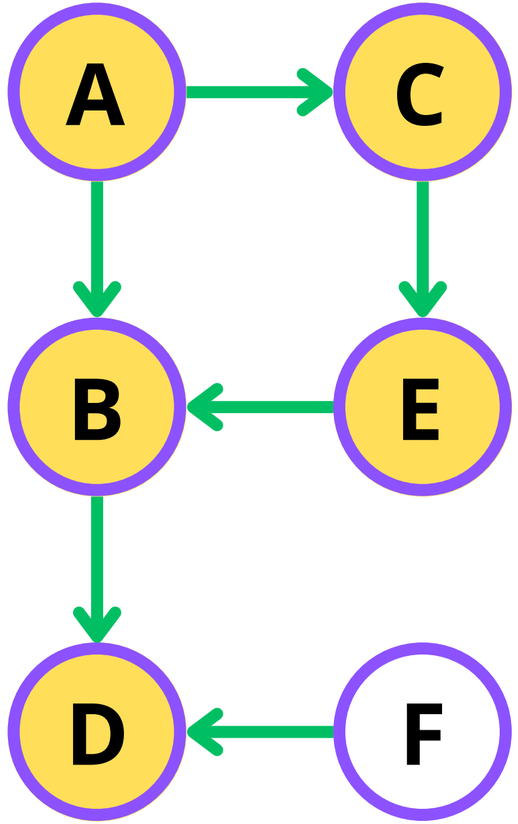
Recorrido actual: A → B → D → C → E

DFS

Recorrido esperado: A → B → D → C → E







Recorrido actual: A → B → D → C → E

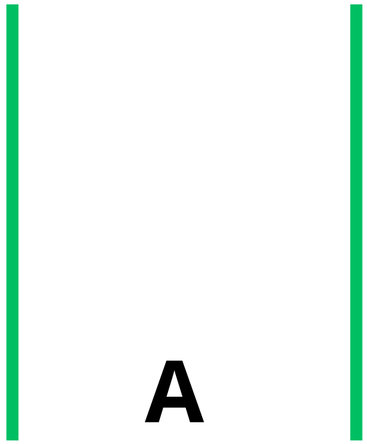
DFS

Recorrido esperado: A → B → D → C → E

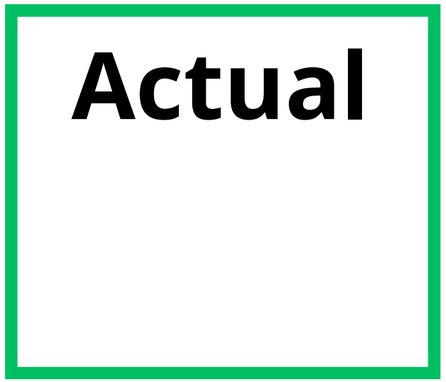
Recorrido actual: A → B → D → C → E

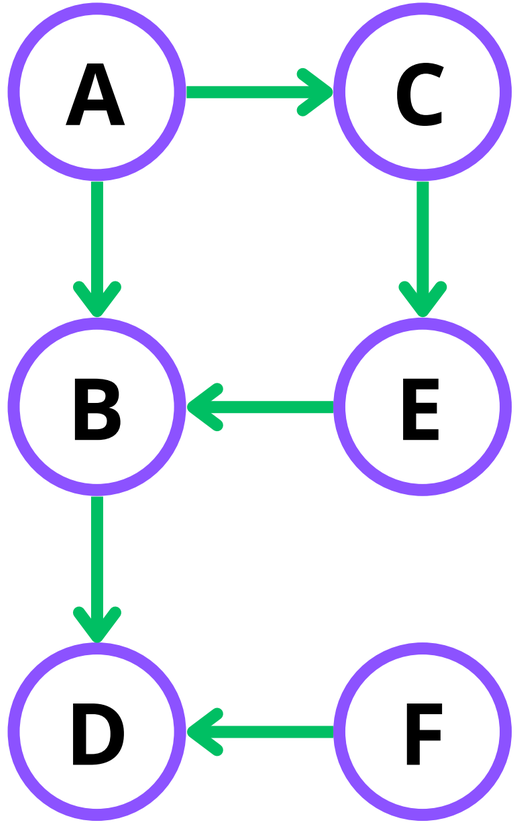
BFS

Recorrido esperado: A → B → C → D → E



QUEUE

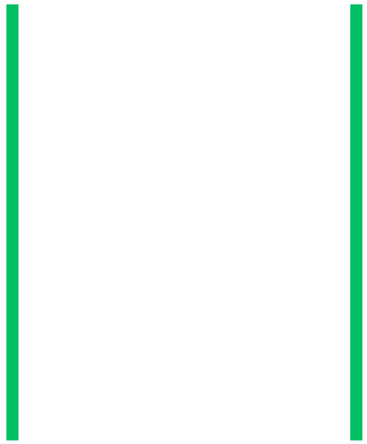




Recorrido actual:

BFS

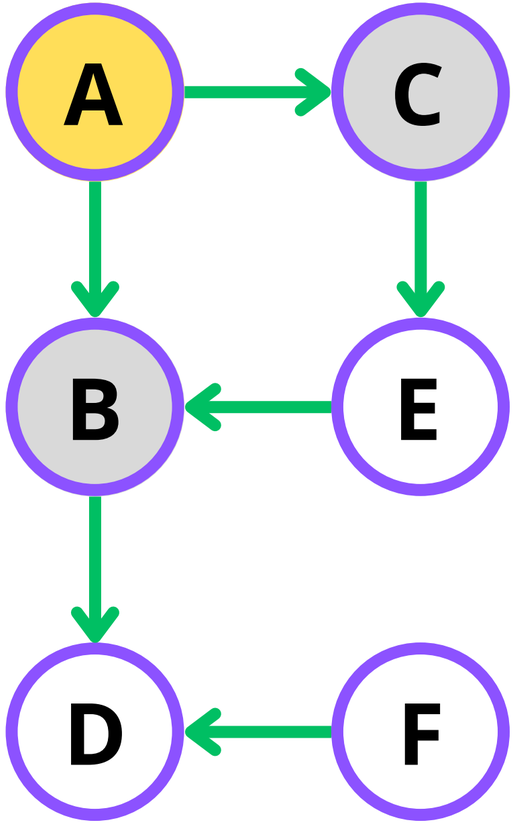
Recorrido esperado: A → B → C → D → E



QUEUE

Actual

A



Recorrido actual: A

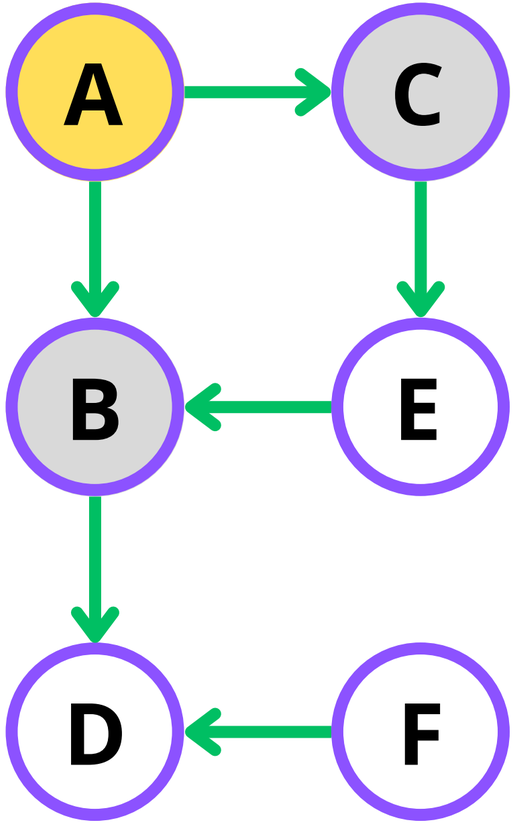
BFS

Recorrido esperado: A → B → C → D → E

C
B

QUEUE

Actual
A



Recorrido actual: A

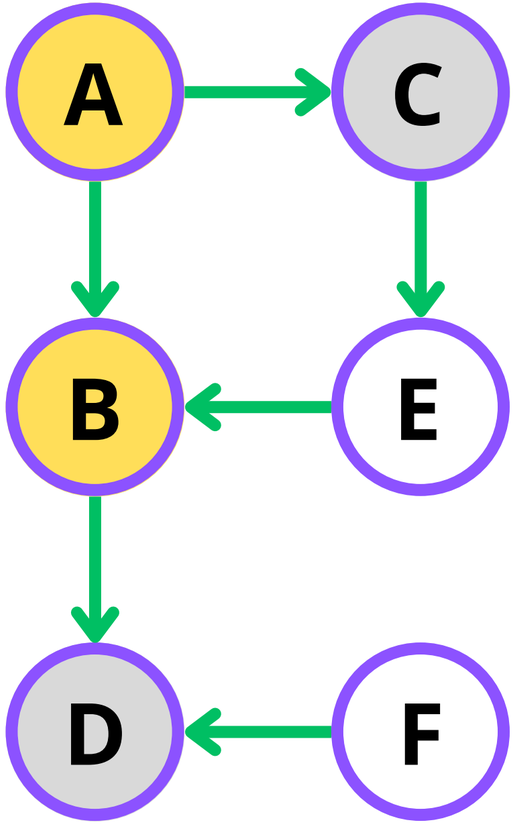
BFS

Recorrido esperado: A → B → C → D → E

D
C

QUEUE

Actual
B



Recorrido actual: A → B

BFS

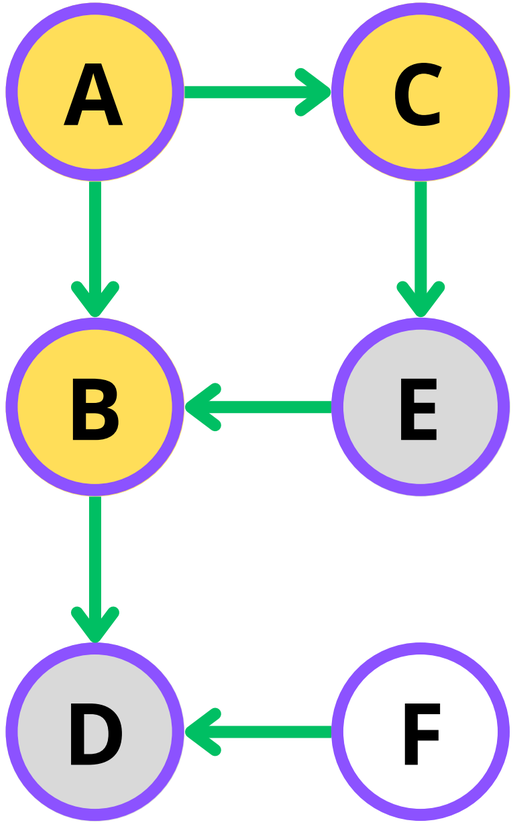
Recorrido esperado: A → B → C → D → E

E
D

QUEUE

Actual

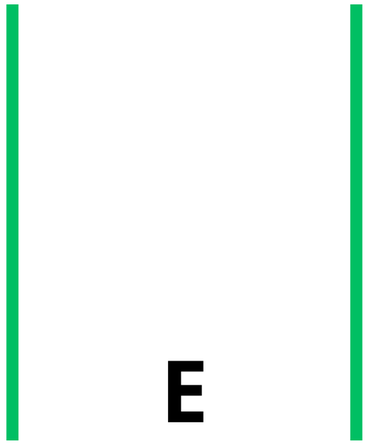
C



Recorrido actual: A → B → C

BFS

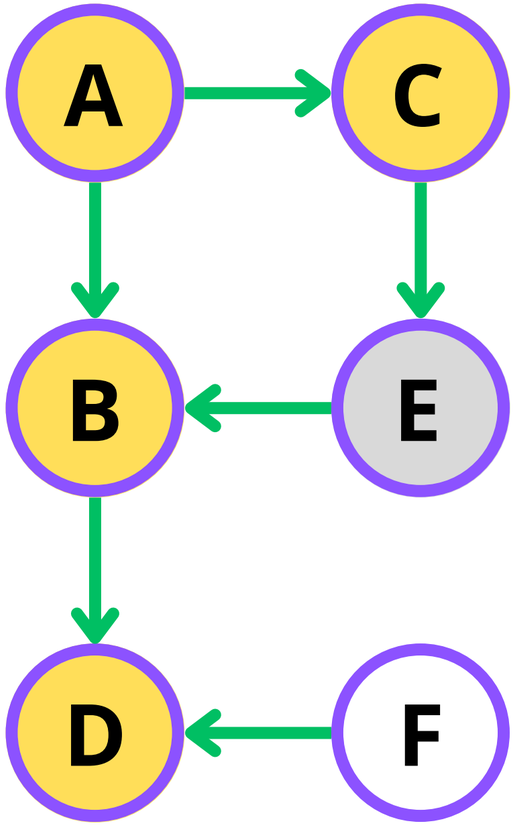
Recorrido esperado: A → B → C → D → E



QUEUE

Actual

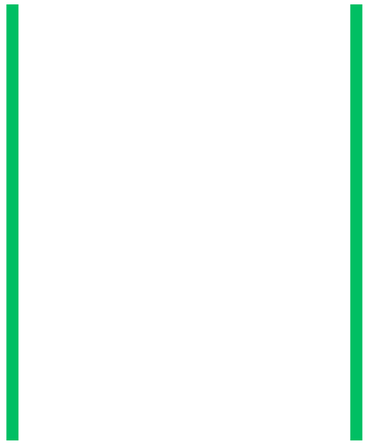
D



Recorrido actual: A → B → C → D

BFS

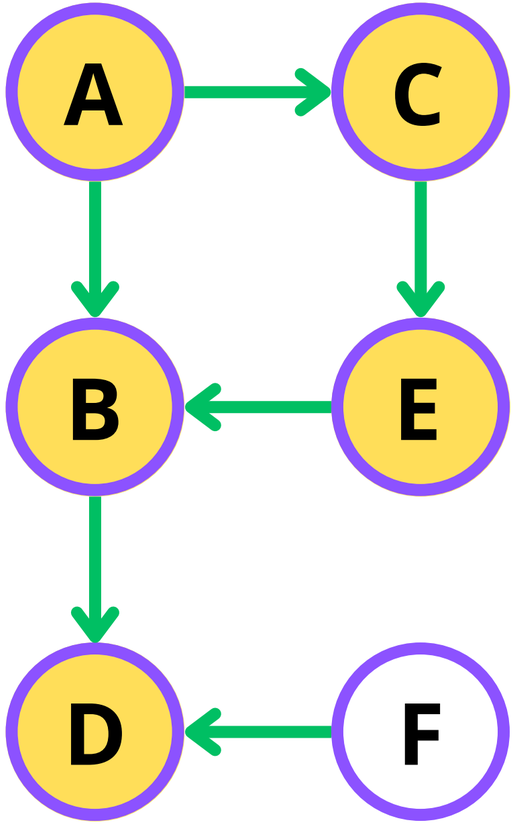
Recorrido esperado: A → B → C → D → E



QUEUE

Actual

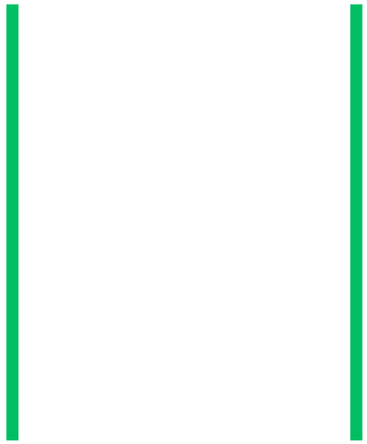
E



Recorrido actual: A → B → C → D → E

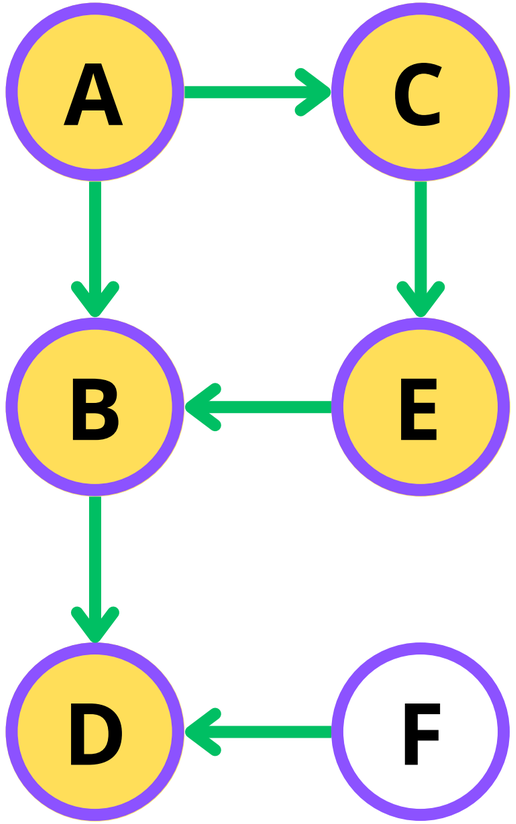
BFS

Recorrido esperado: A → B → C → D → E



QUEUE

Actual



Recorrido actual: A → B → C → D → E

Listo, vamos a implementarlo

Habrá un PDF adicional sobre más ejemplos paso a paso de DFS y BFS en el repositorio de Github