ESCUELA POLITÉCNICA NACIONAL
ESCUELA DE FORMACIÓN DE TECNÓLOGOS

ALGORITMOS Y ESTRUCTURAS DE DATOS (TDSD222)

ASIGNATURA:	Algoritmos y Estructuras de Datos
PROFESOR:	Ing. Lorena Chulde
FECHA:	28 – 01 - 2026
PERÍODO ACADÉMICO:	2025-8

TALLER EN CLASE
(individual)

TÍTULO:
Grafos solo

ESTUDIANTE

Chasi Alexis

OBJETIVO

Determinar el recorrido de las estructuras de datos no lineales.

PARTE 1: TALLER - GRAFOS

Grafo no dirigido y no ponderado (lista de adyacencia)

```
grafo = {
    'A': ['B', 'C'],
    'B': ['A', 'D'],
    'C': ['A', 'D'],
    'D': ['B', 'C']
}
```

BFS (Breadth-First Search)

```
### # RECORRIDO BSF POR AnCHURA ###
A B C D
```

(DFS - Depth First Search)

```
### Recorrido DFS por profundidad ###
A B D C
```

BFS en grafo desconectado

Si el grafo **tiene componentes desconectados**, y recorres desde cada nodo, puedes **descubrir cada componente por separado**.

```
grafo = {
    'A': ['B'],
    'B': ['A'],
    'C': ['D'],
    'D': ['C']
}
```

Verificar si hay camino entre dos nodos:

```
def hay_camino(grafo, origen, destino):
    visitados = set()
    cola = deque([origen])

    while cola:
        nodo = cola.popleft()
        if nodo == destino:
            return True
        visitados.add(nodo)
        for vecino in grafo[nodo]:
            if vecino not in visitados and vecino not in cola:
                cola.append(vecino)

    return False


print("Hay camino entre A y D?", hay_camino(grafo, 'A', 'D'))
print("Hay camino entre A y B?", hay_camino(grafo, 'A', 'B'))
print("Hay camino entre C y D?", hay_camino(grafo, 'C', 'D'))
print("Hay camino entre B y C?", hay_camino(grafo, 'B', 'C'))
```

```
Hay camino entre A y D? False
Hay camino entre A y B? True
Hay camino entre C y D? True
Hay camino entre B y C? False
```

DFS para contar componentes conexas

- Recorre el grafo con **DFS** desde cada nodo **no visitado**.

```
Desde B
B A C D
Desde C
C A B D
Desde D
D B A C
```

- Cada vez que comienza un nuevo DFS, significa que encontró una nueva **componente conexa**.

Ejercicios de Dijkstra, Kruskal, TopoSort

Dijkstra – Camino mínimo

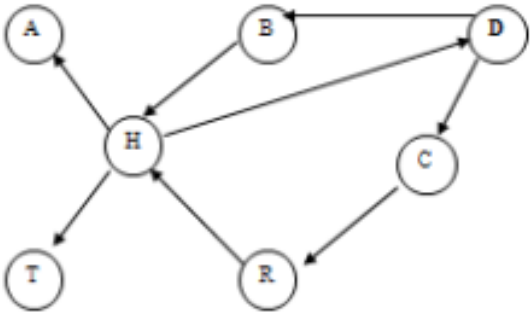
```
{'A': 0, 'B': 1, 'C': 1, 'D': 2}
```

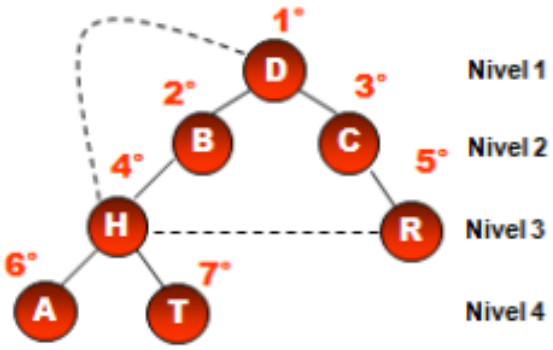
Topological Sort (Orden topológico)

```
['A', 'B', 'D', 'C']
```

PARTE 2: TAREA - GRAFOS

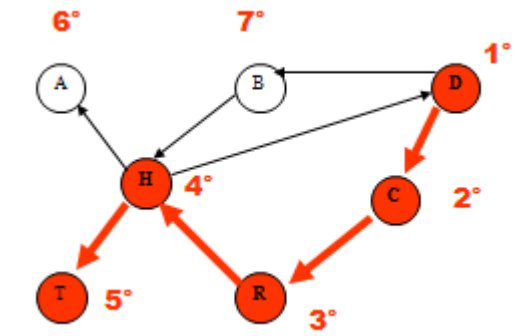
BFS (Breadth-First Search)






```
Recorrido BFS
D B C H R A T
```

(DFS - Depth First Search)




```
Recorrido DFS
D B C R H A T
```

PRESENTACIÓN:

Al finalizar la clase, por favor entrega el taller en una hoja al profesor