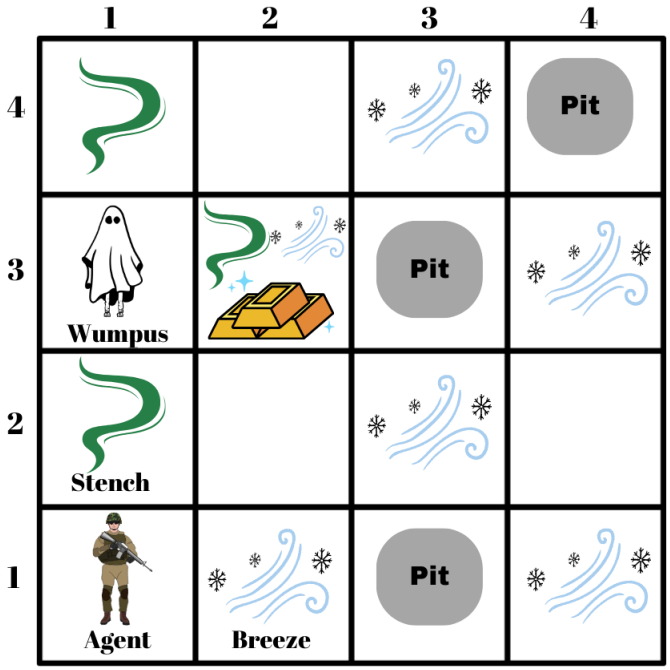
Guía de Taller 1: Mundo del Wumpus (Simplificado)

*Comparación de estrategias de búsqueda: BFS, DFS y A**

Contexto

Trabajaremos con una versión *simplificada* del Mundo del Wumpus sobre una grilla pequeña. El agente conoce el mapa (no hay percepciones ni incertidumbre) y su objetivo es llegar al **oro (G)** desde su **posición inicial (A)**, evitando **pozos (P)**. Usaremos este escenario para comparar las estrategias **BFS**, **DFS** y **A* (Manhattan)**.



Objetivos de aprendizaje

- Modelar un problema de **búsqueda** (estados, acciones, transición, objetivo, costo).
- Implementar y comparar **BFS**, **DFS** y **A*** (sin incertidumbre).
- Medir **nodos expandidos**, **longitud/costo** y discutir **optimalidad/eficiencia**.

Escenario

Grilla 4×4 (ejemplo ilustrativo; el docente presentará el mapa a usar en clase):

```
.   P   .   G
.   .   P   .
.   .   .   .
A   .   .   .
```

Convenciones: `A`=Agente (inicio), `G`=Oro (meta), `P`=Pozo (obstáculo), `.`=libre.
Movimientos: 4 direcciones (N, S, E, O). Costo uniforme: 1 por paso.

Esoecificaciones

1. **Modelamiento** (usar el *starter* en Python de abajo, sin modificar la firma de funciones).
2. **Implementar** por separado:
   - 2.1. **BFS** (cola FIFO). Debe retornar *camino*, *orden de expansión*.
   - 2.2. **DFS** (pila LIFO). Debe retornar *camino*, *orden de expansión*.
   - 2.3. **A*** con **Manhattan**. Debe retornar *camino*, *orden de expansión*.
3. **Medir** y **comparar**: costo del camino (nº pasos), nodos expandidos, y optimalidad.
4. **Reflexionar**: ¿Por qué A* expande menos nodos que BFS en este dominio? ¿Cuándo DFS falla en optimalidad?

Starter: Modelamiento en Python (sin búsquedas)

No implementen aquí BFS/DFS/A*. Sólo modelamiento (estados, acciones, transición, reconstrucción de caminos) que será usado por sus algoritmos en archivos o celdas aparte.

Listing 1: Modelamiento del Mundo del Wumpus (simplificado)

```python
# --- Mundo del Wumpus simplificado: modelamiento base ---
from collections import deque
import heapq

# Grilla 4x4 de ejemplo (el docente puede cambiarla en clase)
grid = [
    [".","P",".","G"],
    [".",".","P","."],
    [".",".",".","."],
    ["A",".",".","."]
]

ROWS, COLS = len(grid), len(grid[0])

# Estado inicial (A) y objetivo (G)
```

```python
def find(symbol):
    for r in range(ROWS):
        for c in range(COLS):
            if grid[r][c] == symbol:
                return (r, c)
    return None

INI  = find("A")
GOAL = find("G")

# Movimientos vlidos: 4 direcciones (N,S,E,O)
DIRS4 = [(1,0),(-1,0),(0,1),(0,-1)]

def in_bounds(r, c):
    return 0 <= r < ROWS and 0 <= c < COLS

def passable(r, c):
    # No se puede pasar por pozos
    return grid[r][c] != "P"

def neighbors(r, c):
    """Acciones aplicables: movimiento a celdas adyacentes libres."""
    for dr, dc in DIRS4:
        nr, nc = r + dr, c + dc
        if in_bounds(nr, nc) and passable(nr, nc):
            yield (nr, nc)

def manhattan(a, b):
    """Heurstica para A*: distancia Manhattan (admisible en 4-dir, costo=1)."""
    (r1, c1), (r2, c2) = a, b
    return abs(r1 - r2) + abs(c1 - c2)

def reconstruct(parent, goal):
    """Reconstruye la ruta desde el inicio hasta 'goal' usando el diccionario '
    parent'."""
    path = []
    cur = goal
    while cur is not None:
        path.append(cur)
        cur = parent.get(cur)
    return path[::-1]

# --- Puntos de integracin (implementar en otro archivo o debajo):
# def bfs(start=INI, goal=GOAL): ...
# def dfs(start=INI, goal=GOAL): ...
# def astar(start=INI, goal=GOAL): ...
```

Guía de experimentación

- Usar el mismo mapa para las tres estrategias.
- **BFS**: reportar camino (lista de coordenadas), costo (len(camino)-1), nodos expandidos.
- **DFS**: idem. Notar si el camino es más largo (no óptimo) y cuántos nodos expande.
- **A***: idem. Verificar misma solución óptima que BFS, pero con menos nodos expandidos.

Entregables

1. Código de **BFS**, **DFS**, **A*** (cada uno en función separada), reutilizando el *modelamiento*.
2. Resultados: camino y costo para cada algoritmo; # de nodos expandidos.
3. Breve discusión técnica: optimalidad (BFS/A*), eficiencia (A* vs BFS), limitaciones de DFS.
4. Presentación grupal sobre el resultado final y los puntos antes mencionados (grupos max 3 integrantes)

Rúbrica de evaluación (100 %)

Criterio	**Puntaje**	**Indicadores**
Modelamiento correcto (estados, acciones, transición, heurística)	25 %	Define INI/GOAL; vecinos válidos; Manhattan correcta; código limpio.
BFS (correctitud y reporte)	20 %	Encuentra solución si existe; camino y costo; orden/nodos expandidos.
DFS (correctitud y reporte)	15 %	Implementación funcional; reconoce falta de optimalidad; reporte de métricas.
A* (correctitud, heurística, métricas)	25 %	Usa Manhattan; devuelve camino óptimo; expande menos nodos que BFS.
Análisis y comparación	10 %	Discusión clara sobre optimalidad/eficiencia; conclusiones justificadas.
Presentación (claridad, comentarios, organización 10 min maximo)	5 %	Código comentado; resultados ordenados; buena redacción.

Notas

- Mantener **costo uniforme** (=1 por paso) para que BFS sea óptimo y la comparación con A* sea justa.
- La heurística Manhattan es **admisible y consistente** en 4 direcciones con costo uniforme.
- No hay percepciones ni incertidumbre: el mapa es **totalmente observable y estático**.