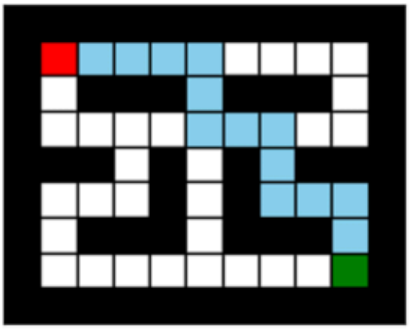
UNIVERSIDAD SAN FRANCISCO

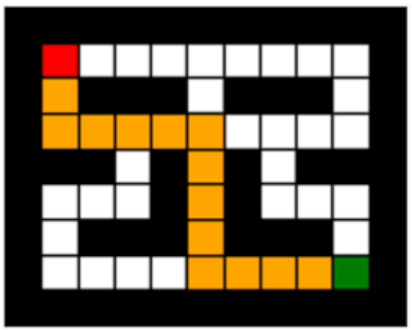
Taller #2

Grupo 4:
Cesar Carrera 00344613
Edgar Guzmán 00344822
Juan Diego Sánchez 00344455
Byron Vinueza 00345908

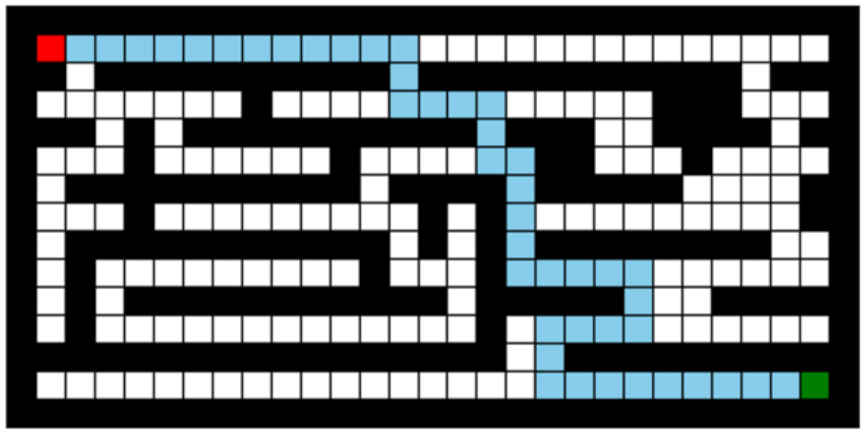
Algoritmos de búsqueda en laberintos

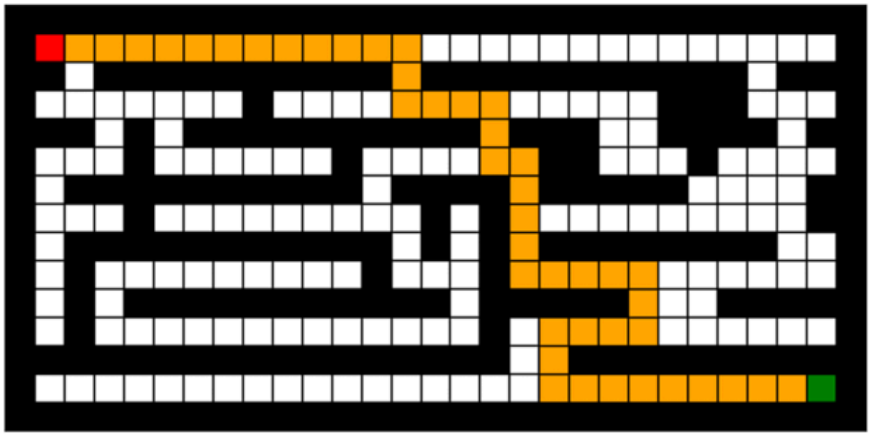
```
Comparación de Algoritmos:
BFS   → Tiempo: 0.000049s, Nodos: 43, Longitud: 15
A*    → Tiempo: 0.000058s, Nodos: 33, Longitud: 15
```



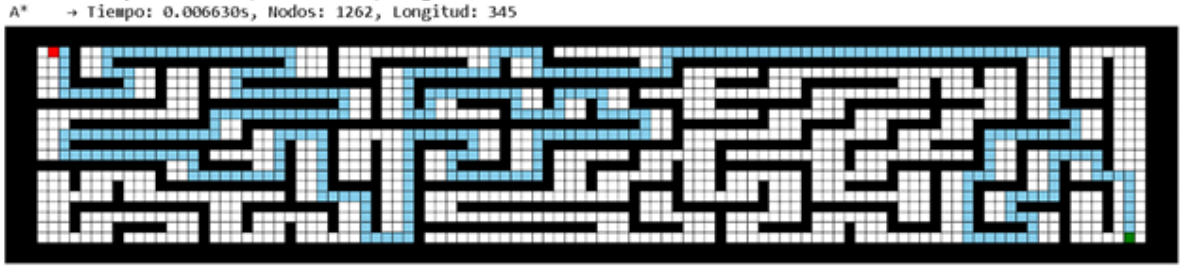


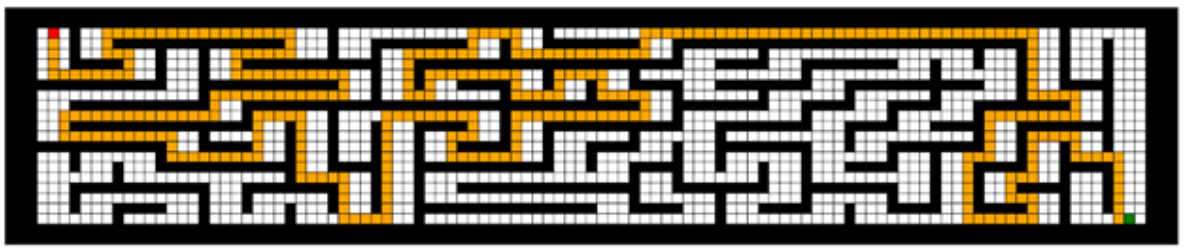
```
Comparación de Algoritmos:
BFS   → Tiempo: 0.000517s, Nodos: 158, Longitud: 45
A*    → Tiempo: 0.000341s, Nodos: 72, Longitud: 45
```





```
Comparación de Algoritmos:
BFS   → Tiempo: 0.003794s, Nodos: 1268, Longitud: 345
A*    → Tiempo: 0.006630s, Nodos: 1262, Longitud: 345
```





Para la resolución de los laberintos se decidió usar únicamente los algoritmos BFS y A* por las siguientes razones:

1. BFS:

BFS explora nivel por nivel, garantizando encontrar el camino más corto en términos de número de pasos en un grafo no ponderado lo cual nos da simplicidad y garantía de optimalidad para este tipo de laberintos. Sin embargo, se expande a muchos nodos lo que puede ser ineficiente en grandes laberintos como se puede evidenciar en el último laberinto.

2. A*

   Es un algoritmo de búsqueda informada que usa una heurística (en este caso, la distancia Manhattan) para guiar la exploración. A diferencia del BFS, este algoritmo es muy eficiente en laberintos grandes o con caminos más largos, ya que expande menos nodos al priorizar direcciones más prometedoras. Se podría decir que A* combina lo mejor de BFS (optimalidad) y Dijkstra (eficiencia) cuando se usa una buena heurística.

Si bien es cierto que existen otros algoritmos de búsqueda, hay ciertas limitantes con las otras opciones. Usar un algoritmo DFS no garantiza el camino más corto ya que se puede quedar atrapado en caminos profundos sin salidas y es muy sensible a cómo es la estructura del laberinto. Es por eso que fue descartado ya que no es eficiente para este tipo de problemas.
Por otra parte, usar el algoritmo Dijkstra tampoco era una solución viable ya que estamos usando A* y la diferencia radica en que A* si tiene una heurística. Este algoritmo es más lento que A* por lo que no representa una alternativa buena.

A continuación se detallan las observaciones encontradas después de analizar los resultados de los diferentes laberintos:

Metrica	**BFS**	**A***
Tiempo de ejecución	Normal/lento	Rapido
Nodos expandidos	Muchos	Pocos (por heuristica)
Optimalidad	Si	Si
Escalabilidad	Disminuye en laberintos grandes	Alta

¿Se puede establecer alguna métrica para evaluar los algoritmos en este problema?

Se pueden definir las siguientes métricas de evaluación para analizar el desempeño de cada uno de los algoritmos que se puedan usar:

Tiempo de ejecución

- **Definición**: Tiempo total que tarda el algoritmo en encontrar la solución desde que inicia hasta que termina.

- **Explicación**: Menor tiempo indica mayor eficiencia computacional.

Nodos expandidos

- **Definición**: Cantidad de nodos (celdas del laberinto) que fueron explorados por el algoritmo.

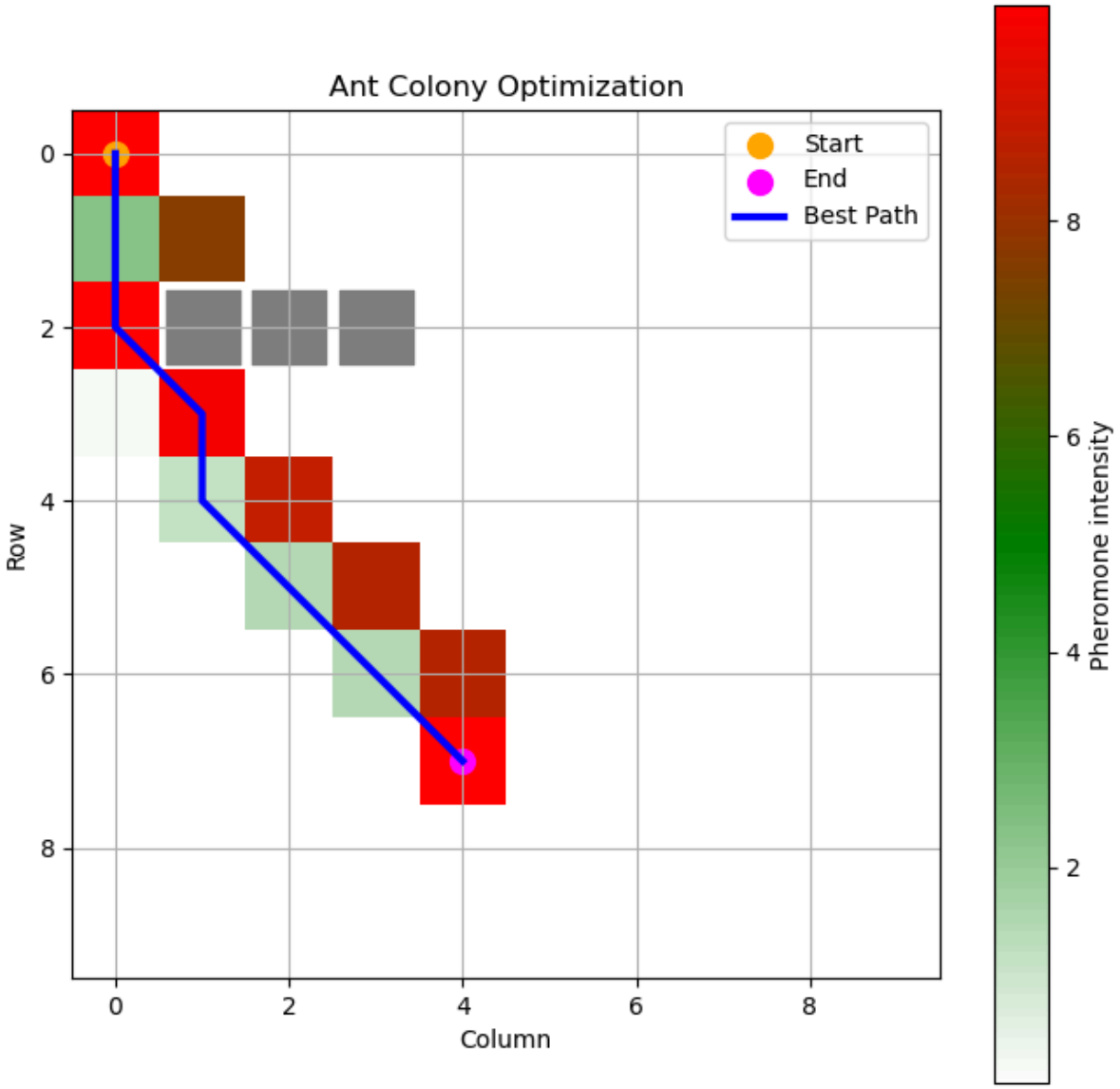
- **Explicación**: Un menor número de nodos expandidos indica un algoritmo más selectivo y eficiente.

Longitud del camino encontrado

- **Definición**: Número de pasos en el camino desde el nodo inicial (S) hasta el objetivo (E).

- **Explicación**: En laberintos sin pesos, el camino más corto tiene menor longitud lo cual indica que es más óptimo.

Optimización de colonia de hormigas



Parámetros definidos:

- Inicio: (0, 0)
- Fin: (4, 7)
- Obstáculos: tres posiciones bloqueadas en una fila vertical:(1,2), (2,2), (3,2)
- Iteraciones: 100
- Tamaño de la grilla: por defecto (10,10)
- Número de hormigas: 10
- Tasa de evaporación de feromonas: 0.1
- α (influencia de la feromona): 0.1 → poca importancia
- β (influencia de la heurística): 15 → muy importante

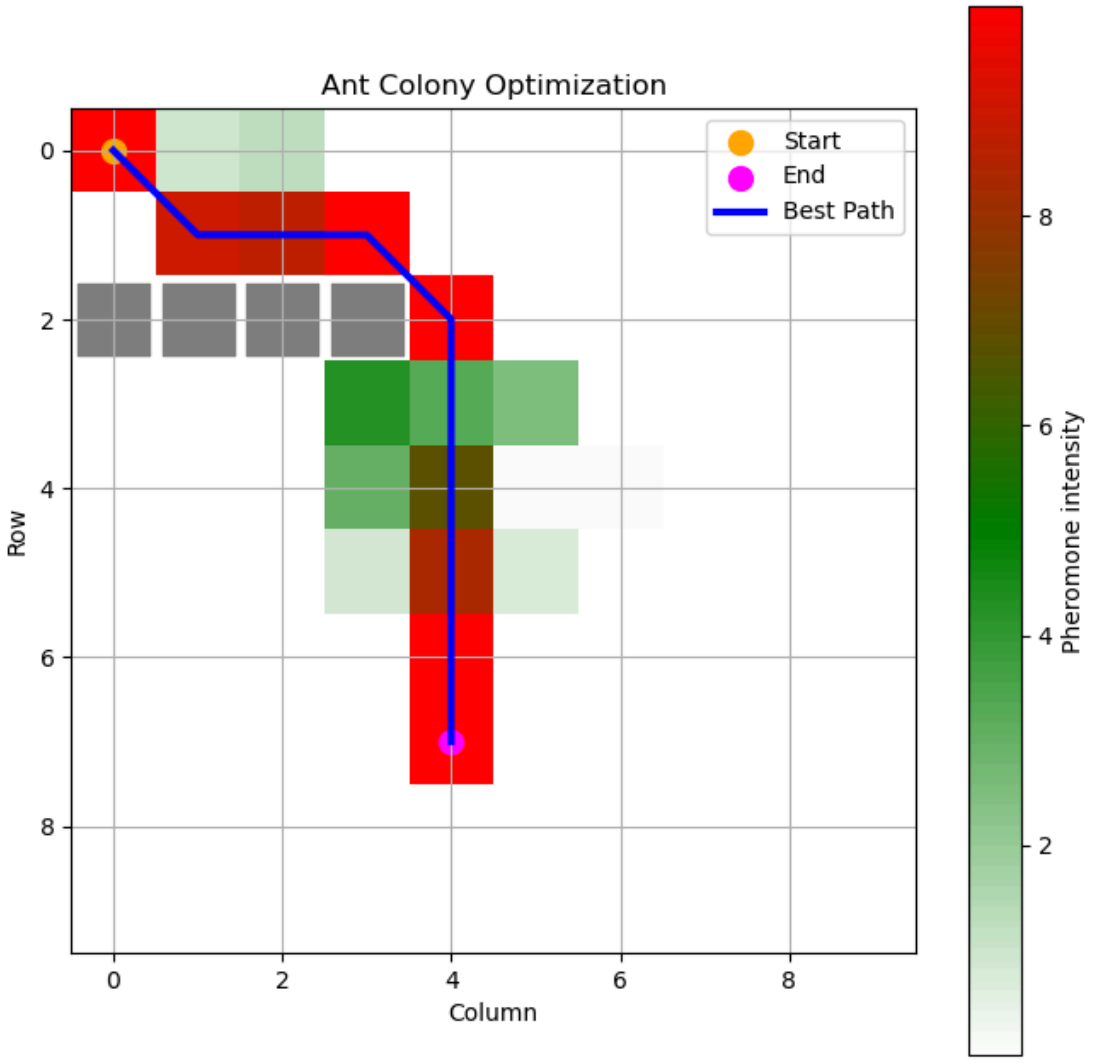
Inicialización:

- Se prepara una grilla de 10x10 con todas las celdas disponibles, excepto los obstáculos.
- Se asigna un nivel de feromona inicial de 1 en todas las celdas.

Búsqueda iterativa (100 ciclos):

En cada iteración:

- 10 hormigas son lanzadas desde el inicio.
- Cada hormiga construye un camino hasta el nodo final o se detiene si no puede avanzar.
- Los caminos encontrados se almacenan.
- Se selecciona el mejor camino (el más corto) entre todos.
- Se aplica evaporación de feromonas en toda la grilla.
- Se deposita feromona adicional en el mejor camino.
- Se actualiza self.best_path si este nuevo camino es mejor o igual al anterior**.**



1.Problema:

- El algoritmo ACO no encontraba un camino hasta el final (self.end) en el caso de estudio 2.
- El gráfico mostraba que las hormigas daban vueltas sin llegar a destino.

2. Código original

- En el método find_best_path, se elegía el camino más corto de todos, sin importar si llegaba o no al destino.

3. Solución

- Se añadió una validación para quedarse solo con caminos que llegaban a self.end.
- Si no hay ningún camino válido, la iteración se ignora (no se reforzaba nada).
- Se selecciona el camino más corto entre los válidos.
- Se refuerza con feromonas solo esos caminos válidos, y se evita aprender caminos incompletos.

4. Resultado

- Ahora el algoritmo encuentra correctamente los caminos que llegan al destino.
- Ya no se refuerzan trayectorias inútiles.
- El mejor camino se actualiza solo si es igual o mejor que el anterior.

Implementación de ACO para resolver el Travelling Salesman Problem (TSP)

- Definir las ciudades con coordenadas 2D
- Calcular la matriz de distancias entre ciudades
- Inicializar parámetros:
    - Número de hormigas
    - Número de iteraciones
    - Peso de feromonas (alpha)
    - Peso de la heurística (beta)
    - Tasa de evaporación de feromonas
    - Cantidad de feromonas depositadas por una buena solución
- Inicio en una ciudad aleatoria
- Calcular la probabilidad de ir a una ciudad no visitada
- Seleccionar la próxima ciudad aleatoriamente (ponderado por P)
- Calcular la longitud del recorrido de cada hormiga
- Encontrar la mejor ruta
- Aplicar la evaporación de feromonas
- Reforzar las feromonas en los caminos recorridos
- Iterar este proceso hasta encontrar la mejor ruta global
- Visualizar

Conclusiones

Algoritmos de Búsqueda: BFS es ideal para garantizar la solución más corta sin necesidad de heurística, pero no es escalable por otra parte, A* supera a BFS en eficiencia sin sacrificar optimalidad, lo que lo convierte en la mejor elección práctica.
La elección de Manhattan como heurística en A* es correcta, dado que el movimiento es en 4 direcciones y no se permiten diagonales.

ACO: Se concluye que el problema respondía a condiciones de lógica no de parámetros. La solución correcta fue aplicar una validación del destino antes de elegir el mejor camino, no se requirió realizar modificaciones en los parámetros iniciales

Repartición de tareas

César Carrera
- Generación de algoritmos BFS y A* para cada laberinto

Byron Vinueza
- Creación de laberintos a partir de los grafos obtenidos en la carpeta

Juan Diego Sánchez
- Correr la implementación del algoritmo de colonia de hormigas.
- Pregunta de investigación

Edgar Guzmán
- Correr el segundo caso de estudio y descripción de los parámetros.
- Conclusiones