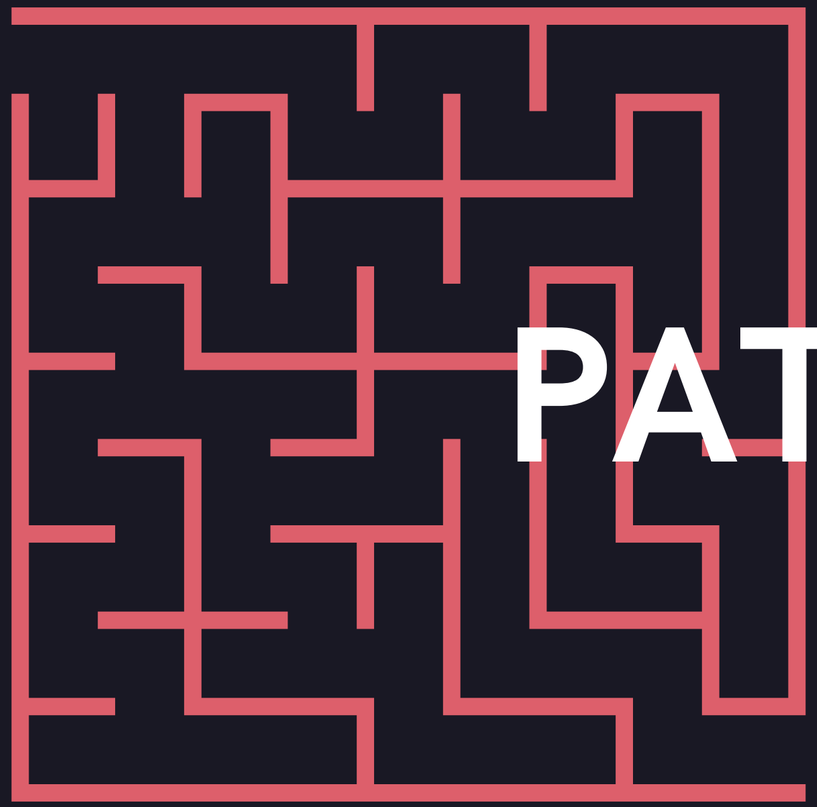
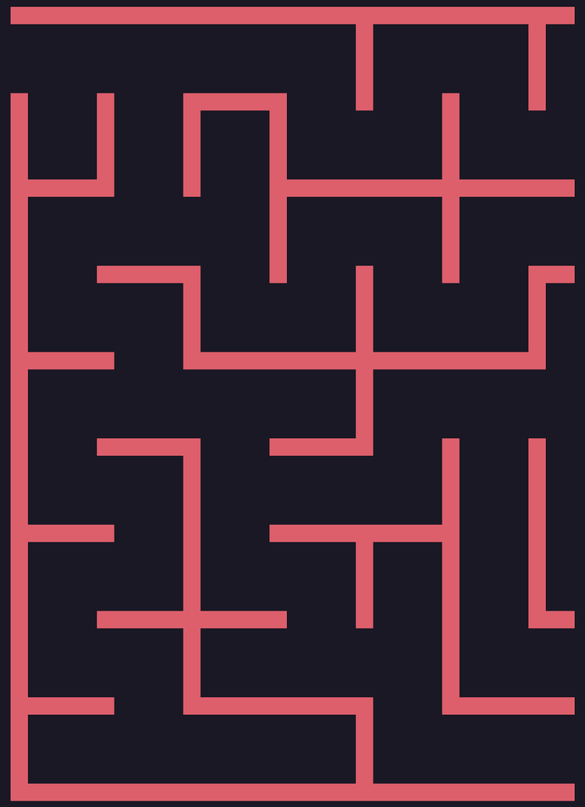


PATHFINDER

Algoritmos Avanzados de Busqueda y Optimización

Esteban Casalas
Rafael Filardi
Sebastian Fripp
Carol Glass

Agenda



1. Motivacion: Problemas de Pathfinder
2. Modelado del problema usando grafos
    a. Reglas
3. Algoritmos y estrategias
4. Resultados
5. Demo

MOTIVACION: PROBLEMAS DE PATHFINDING

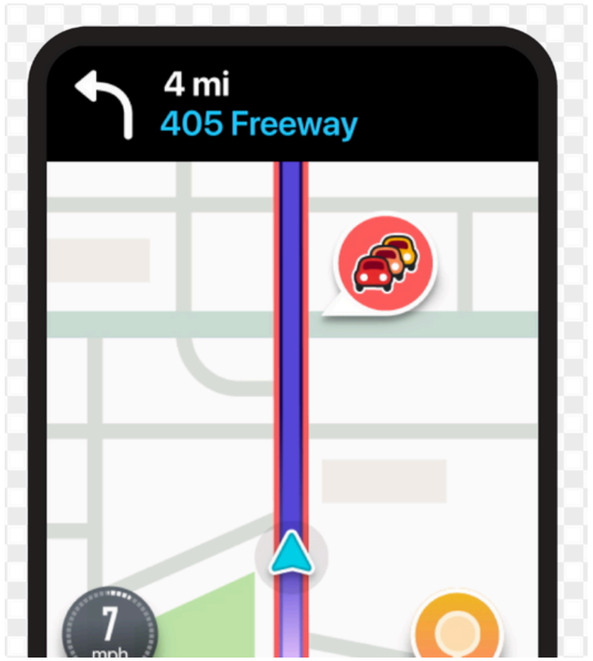
El proyecto aborda la resolución de un problema de búsqueda en grafos, representando un laberinto como un grafo, donde cada celda es un nodo y cada movimiento posible es una arista con un costo asociado.

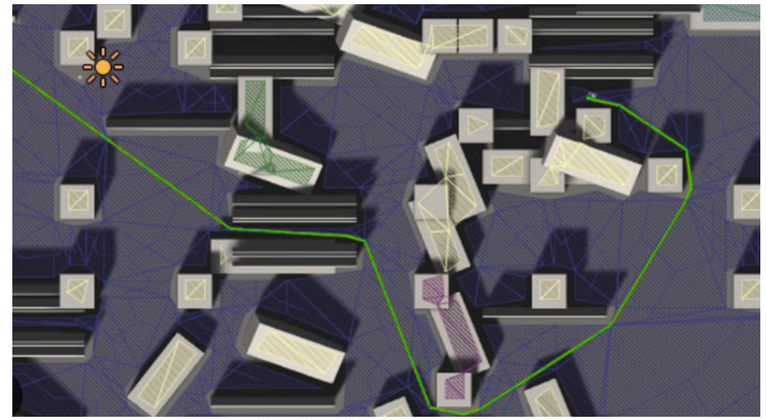
Problema:
¿Existe un camino desde la entrada hasta la salida del laberinto? (Problema de decision)
En otras palabras, **“Dado un laberinto M, un comienzo s y un destino d, existe un camino que conecta s y d”**

Approach:
Explorar diferentes algoritmos que permitan resolver laberintos.
Se evaluará su desempeño en términos de eficiencia y complejidad.

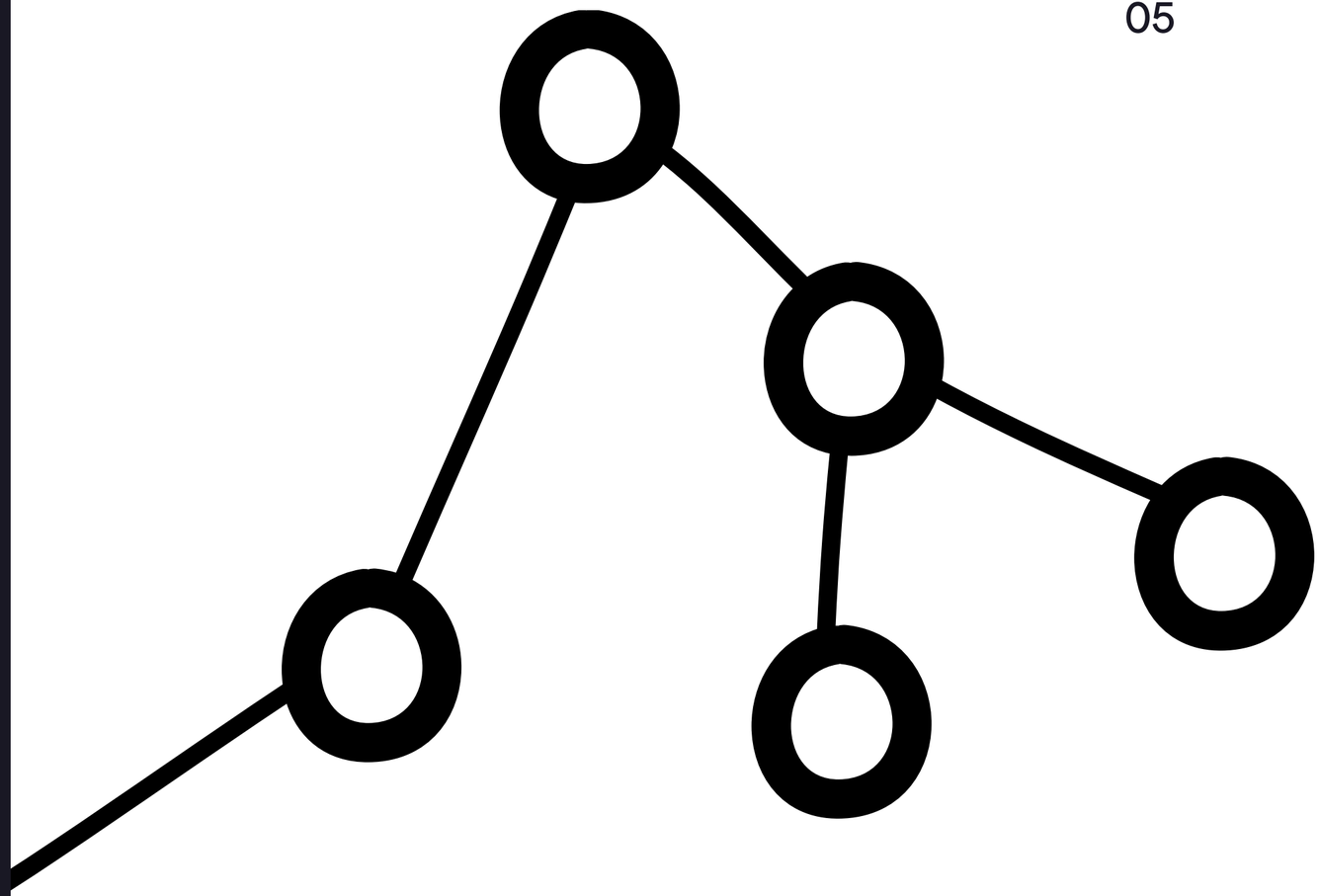
04

MAS MOTIVACION: PATHFINDING EN LA VIDA REAL

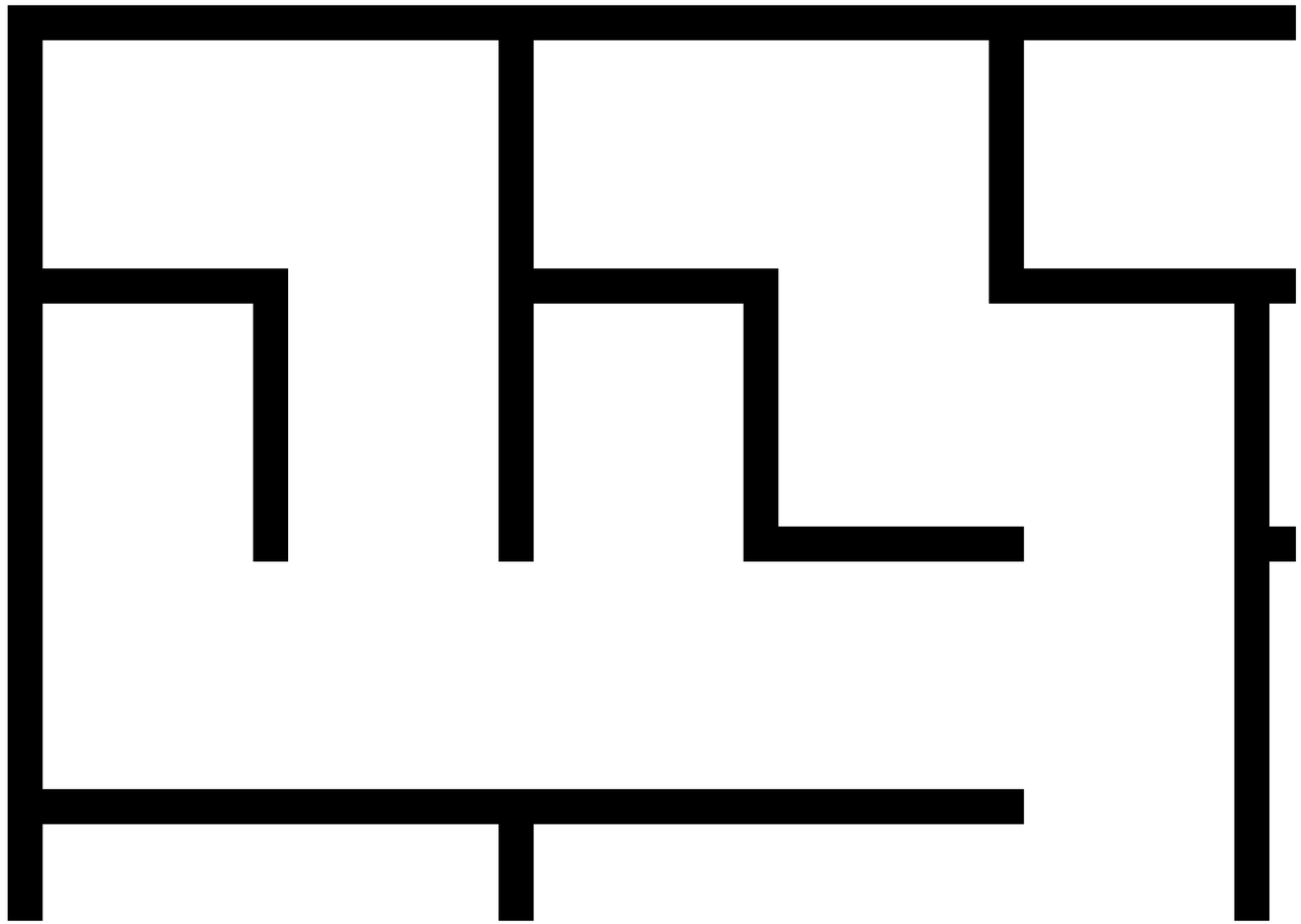




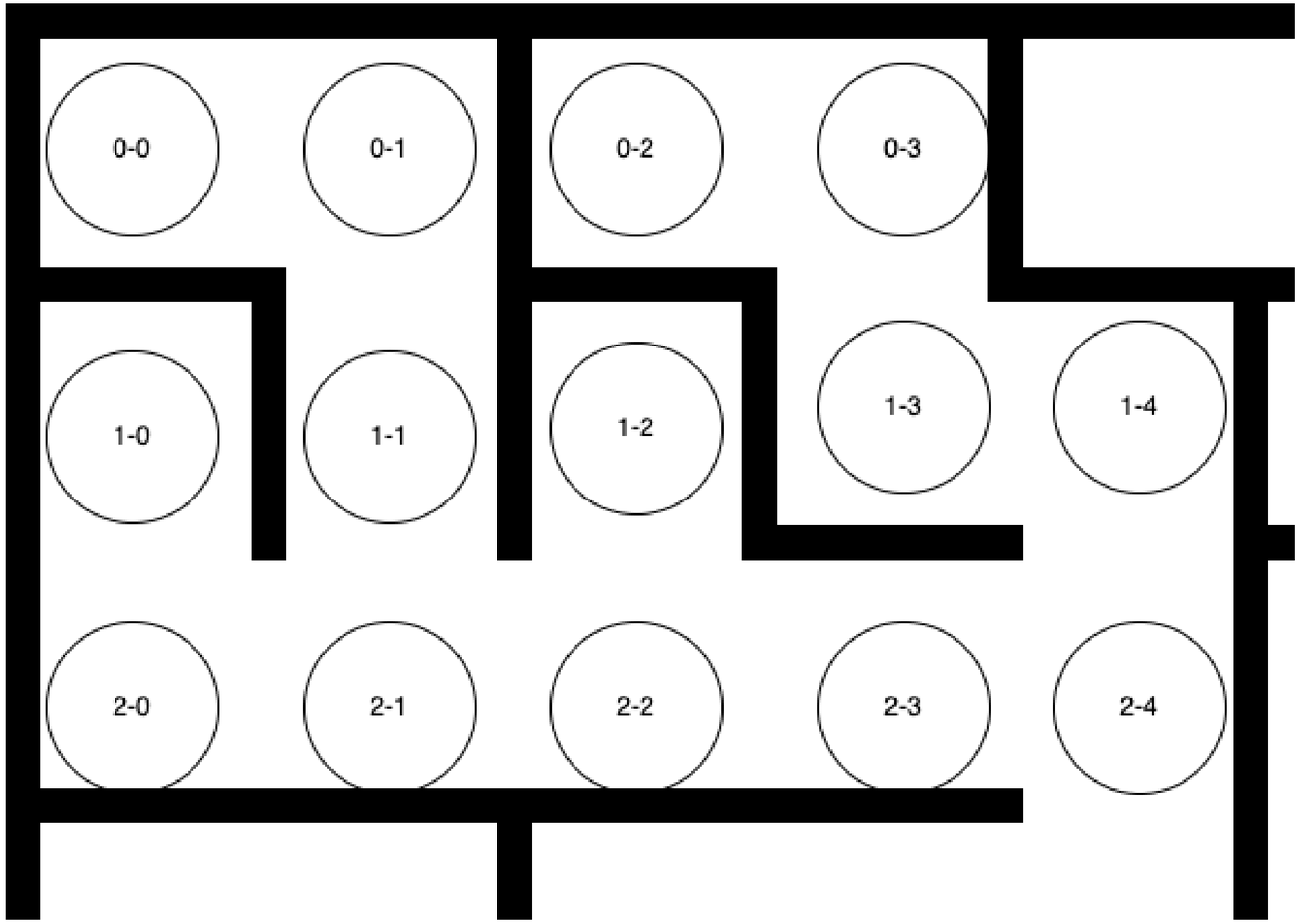
MODELADO DEL PROBLEMA USANDO GRAFOS



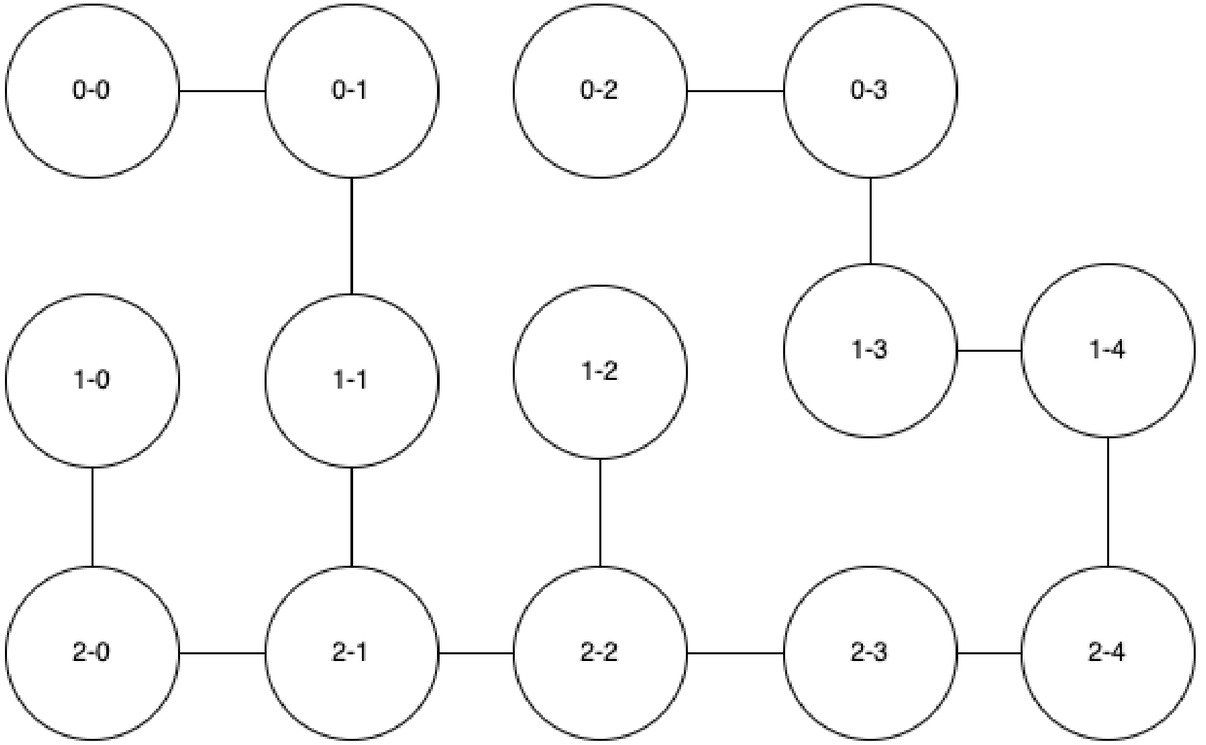
LABERINTO COMO GRAFO



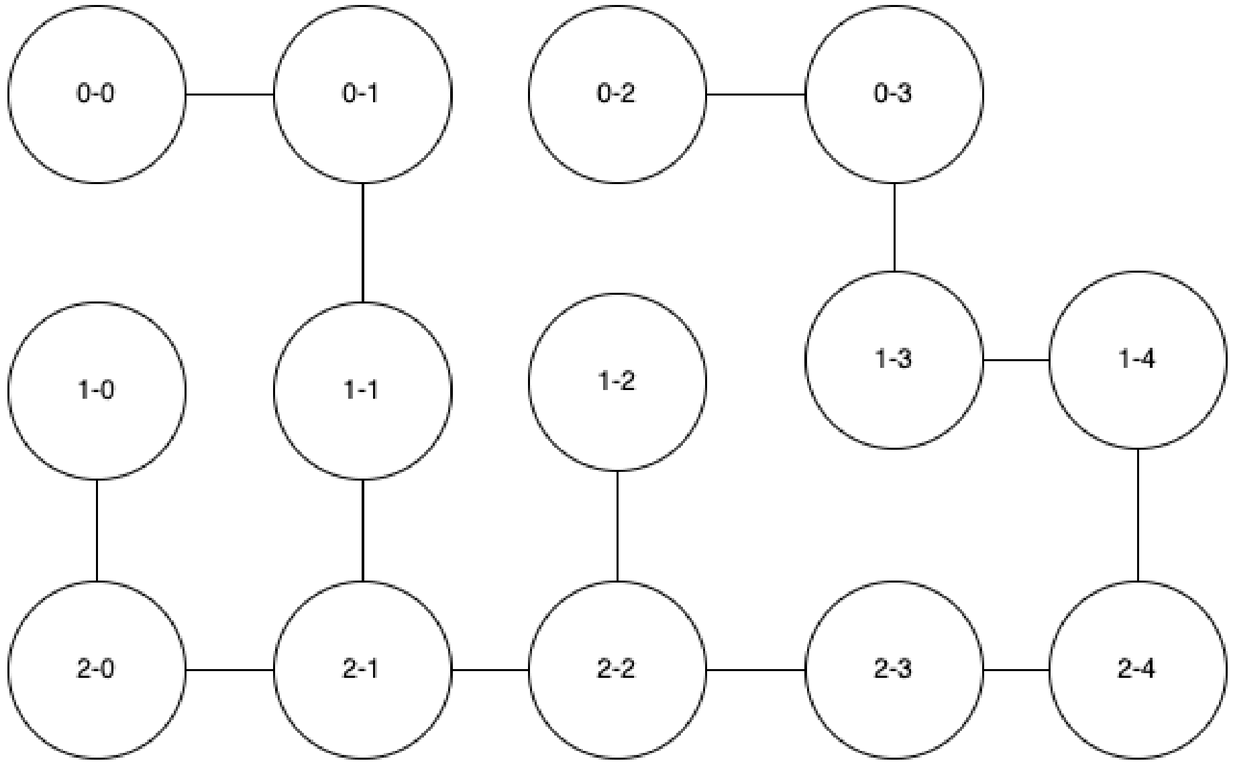
LABERINTO COMO GRAFO



LABERINTO COMO GRAFO



LABERINTO COMO GRAFO



Laberinto = G(V,E)

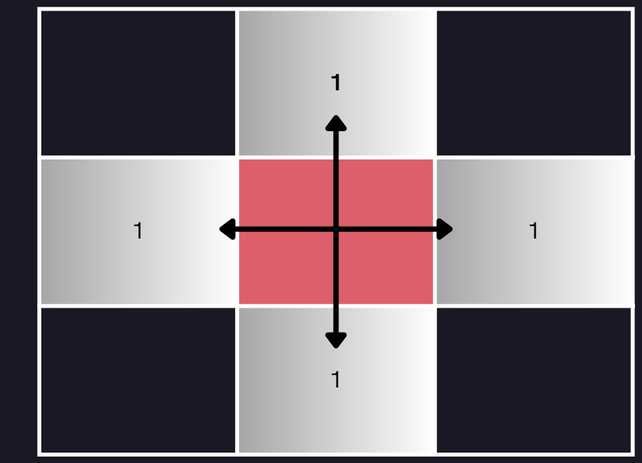
```
export class Graph {
    listAdj: { [key: string]: string[] };

    constructor() {
        this.listAdj = {};
    }
}
```

Como el laberinto es un grafo, se pueden utilizar algoritmos de grafos

REGLAS

1. Los laberintos tienen salida
2. Movimientos permitidos:
    a. Derecha (Costo 1)
    b. Izquierda (Costo 1)
    c. Arriba (Costo 1)
    d. Abajo (Costo 1)
    e. No se puede ir en diagonal
3. No se pueden atravesar paredes
4. El agente es de tamaño 1x1



ESTRATEGIAS DE RESOLUCION

06

ESTRATEGIAS POSIBLES

Podemos encontrar un camino que conduzca a la salida?

Podemos lograr que ese camino solucion sea el mas corto posible?

Podemos encontrar el camino mas corto posible visitando la menor cantidad de nodos posibles?

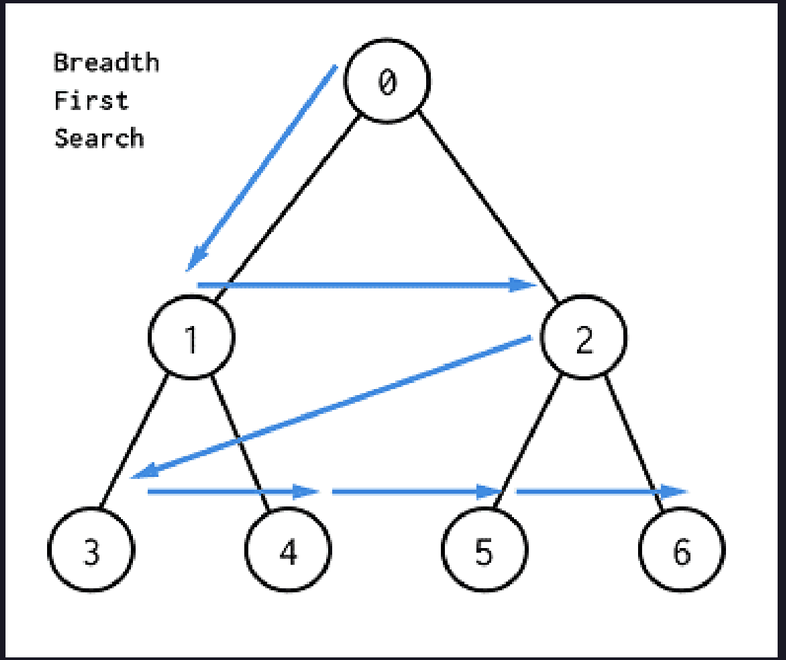
ESTRATEGIA:
ENCONTRAR UNA SALIDA
BFS/DFS

BFS

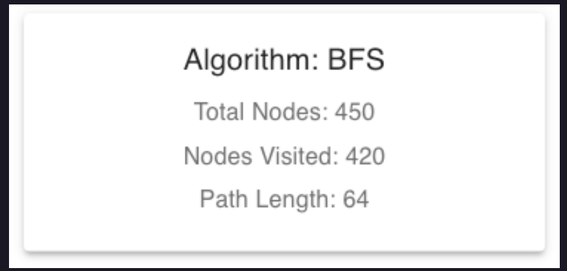
Explora el grafo por niveles, asegurando encontrar el camino más corto en términos de cantidad de nodos en grafos sin pesos.

Utiliza una cola FIFO donde los nodos se extraen por el inicio y se agregan por el final.

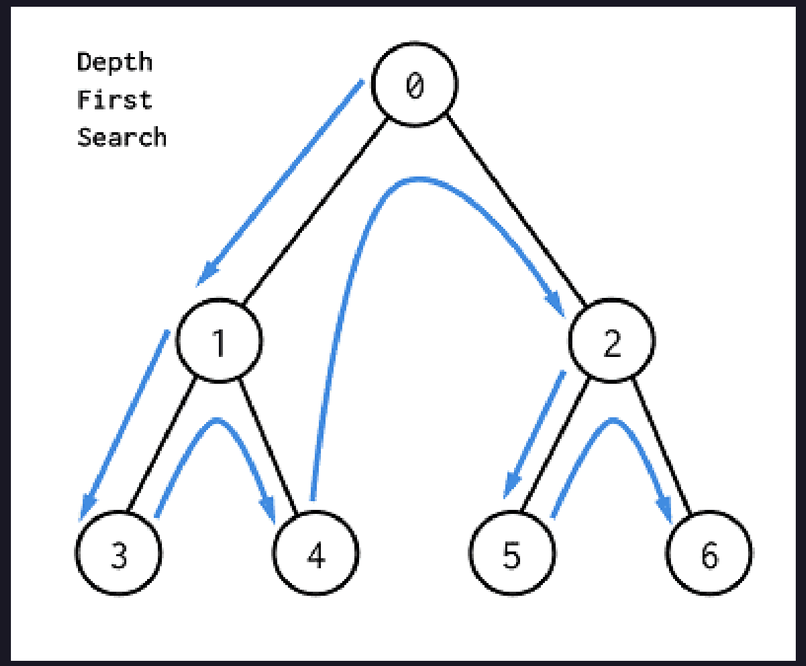
BFS garantiza encontrar una salida si existe, pero no optimiza el número de nodos visitados, lo que lo hace adecuado para resolver problemas de búsqueda amplia pero no necesariamente eficiente cuando el grafo es muy grande.



Datos demo



DFS



Explora el grafo o árbol avanzando tan profundamente como sea posible en una rama antes de retroceder.

Encuentra una salida pero no garantiza que sea la mas corta y puede generar soluciones arbitrarias y recorrer caminos innecesarios, dependiendo del orden de exploración.

Utiliza una pila (LIFO) que le permite priorizar caminos largos en lugar de caminos amplios.

Datos demo

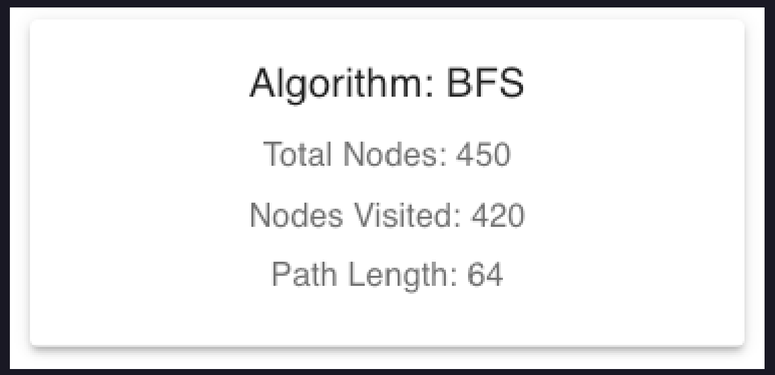
Algorithm: DFS

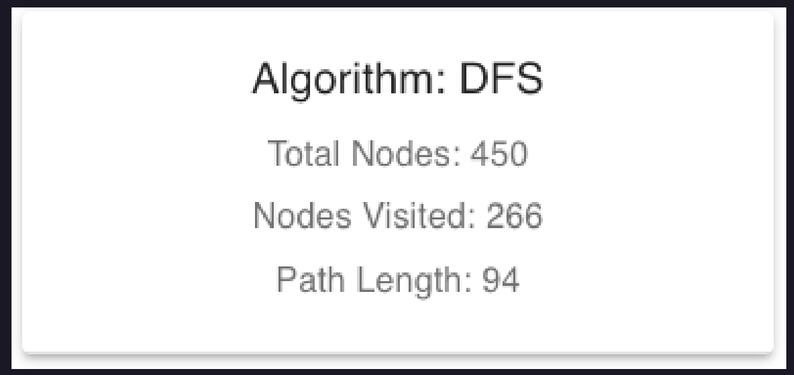
Total Nodes: 450

Nodes Visited: 266

Path Length: 94

BFS VS DFS





Ambos llegan a una solución, pero cual es mejor?

BFS da un menor costo, DFS visita muchos menos nodos (200 nodos menos, en comparación). Esto sera siempre así?

06

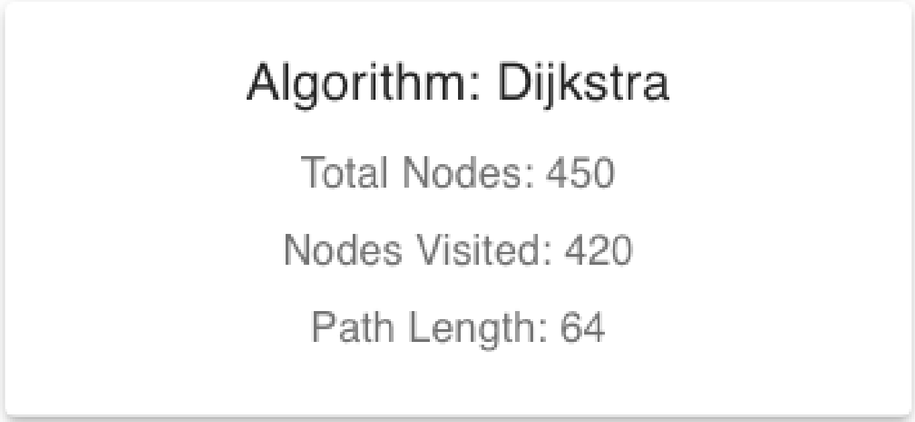
ESTRATEGIA: CAMINO MAS CORTO DIJKSTRA

Resuelve el problema de encontrar caminos de costo mínimo en un grafo con pesos no negativos. No solo encuentra una salida, sino que garantiza el camino más corto en términos de costo acumulado entre un nodo inicial y un nodo objetivo.

- Garantía de Óptimo: Siempre encuentra el camino de costo mínimo entre el nodo inicial y el objetivo.
- Versatilidad: Funciona en grafos dirigidos y no dirigidos, siempre que los pesos sean no negativos.
- Eficiencia Mejorada: Usando una cola de prioridad (heap), alcanza una complejidad orden de E logarítmico en V.

Aunque eficiente en grafos más pequeños o medianos, puede volverse costoso en términos de exploración de nodos en grafos densos o muy grandes, ya que visita todos los nodos alcanzables para garantizar la óptima solución

ESTRATEGIA: CAMINO MAS CORTO DIJKSTRA



Tenemos que el camino optimo tiene un total de 64 nodos, pero aun así seguimos visitando 420/450 nodos

PODEMOS ENCONTRAR EL CAMINO MAS CORTO MIENTRAS REDUCIMOS AÚN MAS EL NÚMERO DE NODOS VISITADOS?

ESTRATEGIA: CAMINO MAS CORTO Y REDUCIR LA CANTIDAD DE NODOS VISITADOS

A*

Combina la búsqueda de caminos mínimos con una heurística para guiar la exploración, logrando un equilibrio entre eficiencia y optimalidad

- Optimalidad Garantizada: Encuentra el camino de menor costo si h(v) es admisible.
- Eficiencia: Explora menos nodos que Dijkstra al usar la heurística para priorizar.
- Flexibilidad: Permite ajustar la heurística según el problema, mejorando el rendimiento en escenarios específicos.

La calidad de la solución depende de la calidad de la heurística.

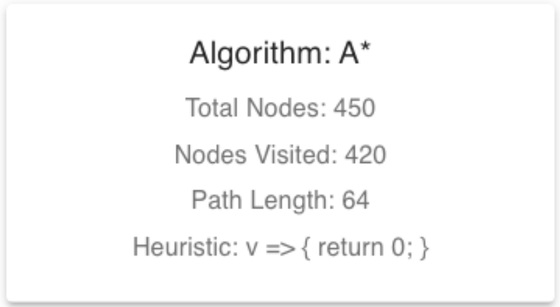
En grafos grandes o con heurísticas poco informativas, puede volverse computacionalmente costoso.

h(v)

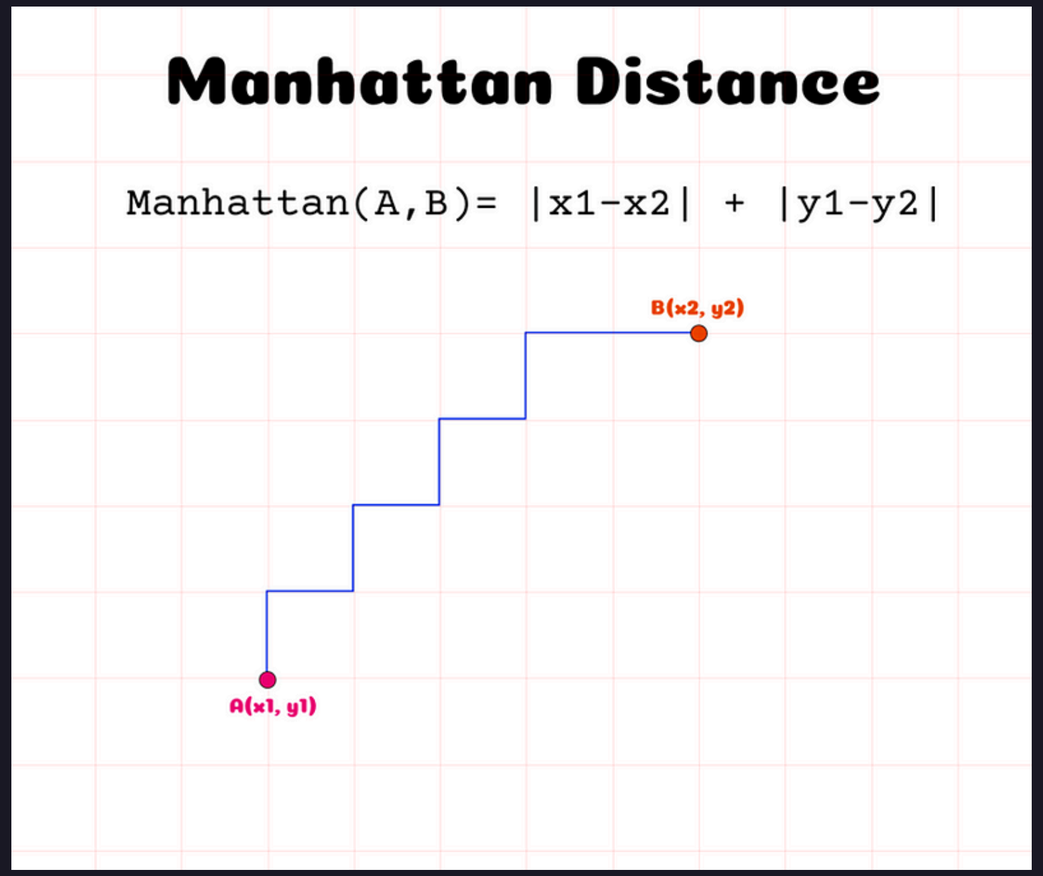
HEURISTICAS

h(V) = 0 (Sin heuristica)

si h(v) es 0, entonces sólo juega un papel g(n), y A* se convierte en el Algoritmo de Dijkstra, que tiene garantizado encontrar el camino más corto.



- Mismos resultados que Dijkstra (Esperable)
- Heuristica Admisible porque
  - h(v) = 0
  - 0 <= **ĥ(v) [Los costos de movimientos siempre son positivos]**



Algorithm: A*

Total Nodes: 450

Nodes Visited: 269

Path Length: 64

Heuristic: v => { const target = '14-29'; const [xV, yV] = v.split('-').map(Number); const [xT, yT] = target.split('-').map(Number); return Math.abs(xV - xT) + Math.abs(yV - yT); }

DISTANCIA DE MANHATTAM

Por que esta distancia?

Recordemos una de las reglas de nuestro problema era el siguiente:

- “**No se puede ir en diagonal”**

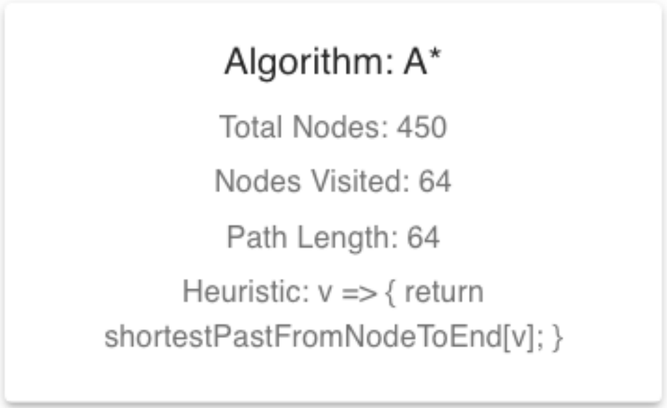
Por lo tanto la forma que representa mejor de calcular distancias en nuestro problema es mediante esta distancia.

Admisible porque: la distancia minima de ir de s a t es la determinada por Manhattam, por lo tanto siempre va a ser menor que la distancia en el grafo.

Problema de esta Heuristica: Puede sobreestimar la proximidad a la meta, prefiere caminos que parecen mas cortos pero no lo son (pues considera la distancia mas corta sin obstaculos).

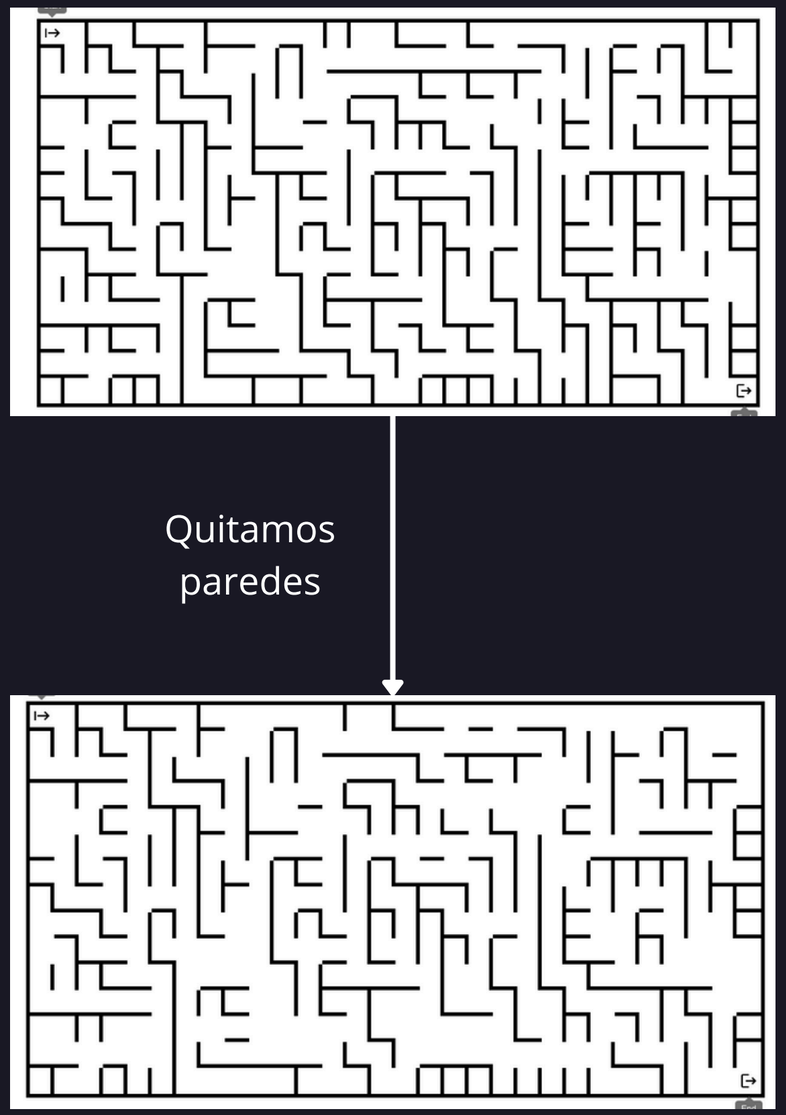
La Heuristica perfecta
h(v) = ħ(v)

Como h(v) es exactamente igual al coste de moverse de s al destino, A* sólo seguirá el mejor camino y nunca expandirá nada más, obteniendo la mejor optimizacion posible.
Como conclusion se llega a que mientras mas inforamcion se tenga sobre el gráfo mas optimo se logra.



- Los nodos visitados son los nodos que componen el camino (64 y 64)

No siempre vamos a tener tanta información sobre el grafo



HEURISTICA “PODAR” LABERINTO

Consiste en dos pasos:

Paso previo:

- Poda del laberinto complicado (quitamos paredes)

Ejecutamos A* en el laberinto complicado:

- La función heurística va a estar definida por:
  - **h(v) = "longitud del camino más corto de v a d en el laberinto simplificado iniciando en el punto v"**
  - Como en un laberinto mas sencillo, la solución optima siempre va a ser menor a la solución optima del problema complejo, esta heurística no sobreestima el costo verdadero, por lo tanto es admisible. (Lower bound del problema)

DEMO