Algoritmos y estructuras de datos 2

PROYECTO UBER

Integrantes: Suden Paulina y Ruiz Joaquin

Link repositorio: https://github.com/JoacoR7/ProyectoUberAlgo2

PROBLEMAS ENCONTRADOS DURANTE EL DESARROLLO

- Cómo hacer la búsqueda de vehículos
- Implementar Dijkstra con esquinas no lineales
- Cómo manejar el path
- Adaptación a códigos ajenos (del grupo)
- Implementación CLI
- Implementación de grafo
- Adaptación a GitHub

SOLUCIONES A LOS DIFERENTES PROBLEMAS

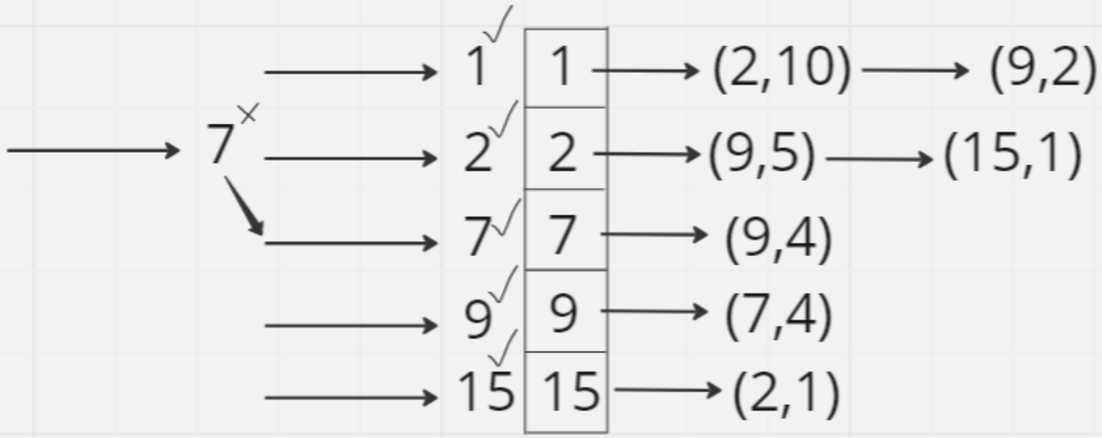
- **Hash table**: Ubicaciones fijas y móviles (autos y personas), precalcular las distancias a todas las esquinas (para encontrar autos), guardar los datos de la implementación de Dijkstra (distancias, visitados, padres, etc.),
- **Grafo** para guardar el mapa.
- **Linear probing** para evitar colisiones de esquinas en el mapa.
- **Dijkstra** para calcular el camino más corto entre 2 esquinas, si existe.
- **Matriz** para guardar los datos relevantes del mapa
- **Uso de librería** "sys" para CLI

SOLUCIONES A LOS DIFERENTES PROBLEMAS

- **Linear probing** para evitar colisiones de esquinas en el mapa.

Esquinas: (1,2,7,9,15)

Aristas: {(1,2,10),(1,9,2),(2,9,5),(15,2,1),(2,15,1),(7,9,4)}



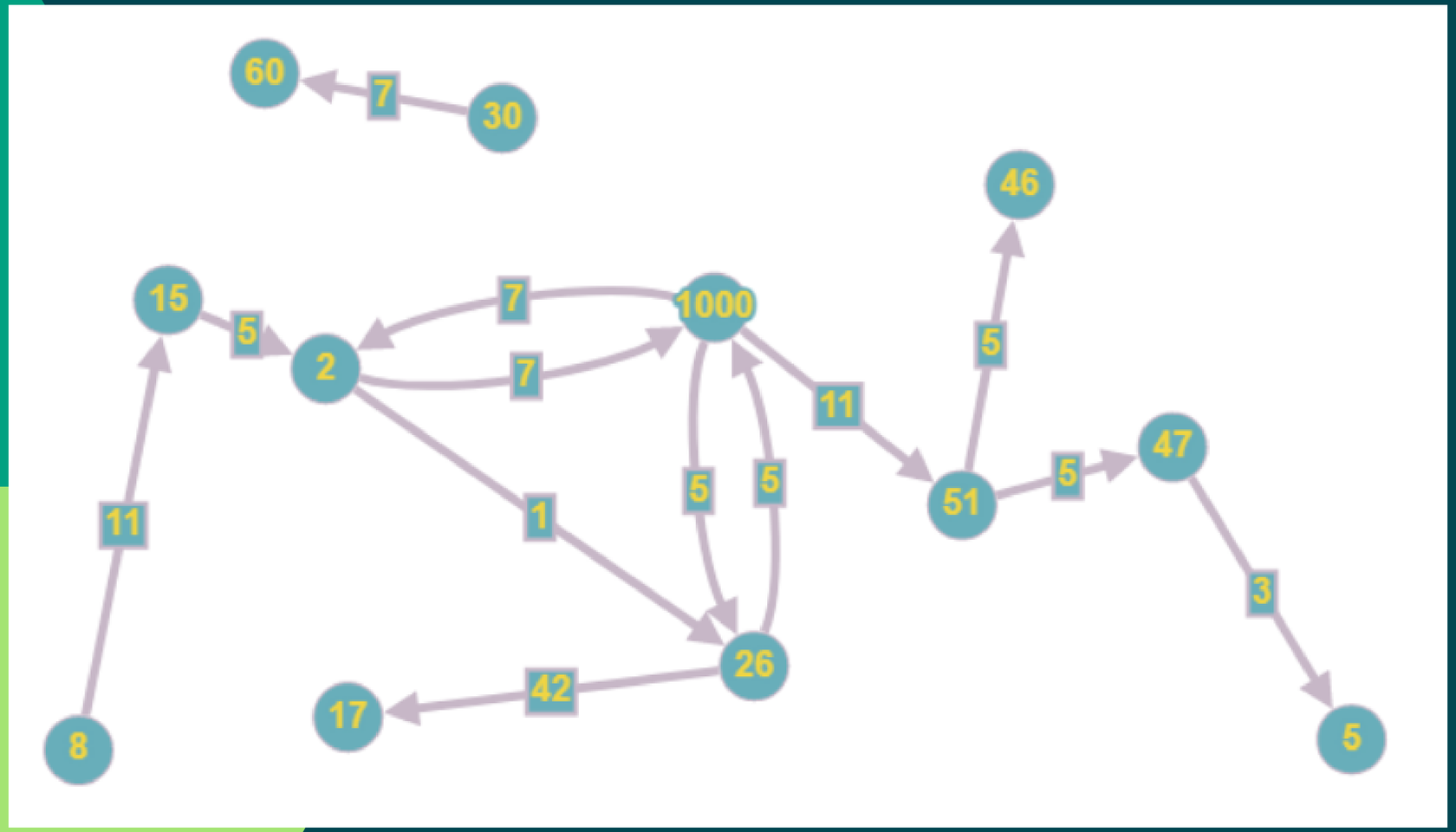
SOLUCIONES A LOS DIFERENTES PROBLEMAS

- **Matriz** para guardar los datos relevantes del mapa

1	{(2,10),(7,6),(9,2),(15,11)}
2	{(7,9),(9,5),(15,1)}
7	{(9,4)}
9	{(7,4)}
15	{(2,1),(7,10),(9,6)}

Par ordenado (e,d), donde e es la esquina donde puede llegar, y d la distancia mínima (calculada con Dijkstra)

MAPA DE PRUEBA (entre otros)



POSIBLES MEJORAS A LA SOLUCIÓN

- Contemplar más casos a la hora de calcular la distancias entre dos objetos.