Algoritmos y estructuras de datos 2

Proyecto Uber

Integrantes: Suden Paulina y Ruiz Joaquin

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

Soluciones: Estructuras a utilizar

- **Hash table**: Ubicaciones fijas y móviles (autos y personas), datos relevantes (distancias y posibilidad de acceso (autos)).
- **Grafo** para guardar el mapa (hash).
- **Linear probing** para evitar colisiones de esquinas en el mapa.
- **DFS** para determinar si existe camino entre esquinas.
- **Dijkstra** para calcular el camino más corto entre 2 esquinas.

Estado actual
Problemas

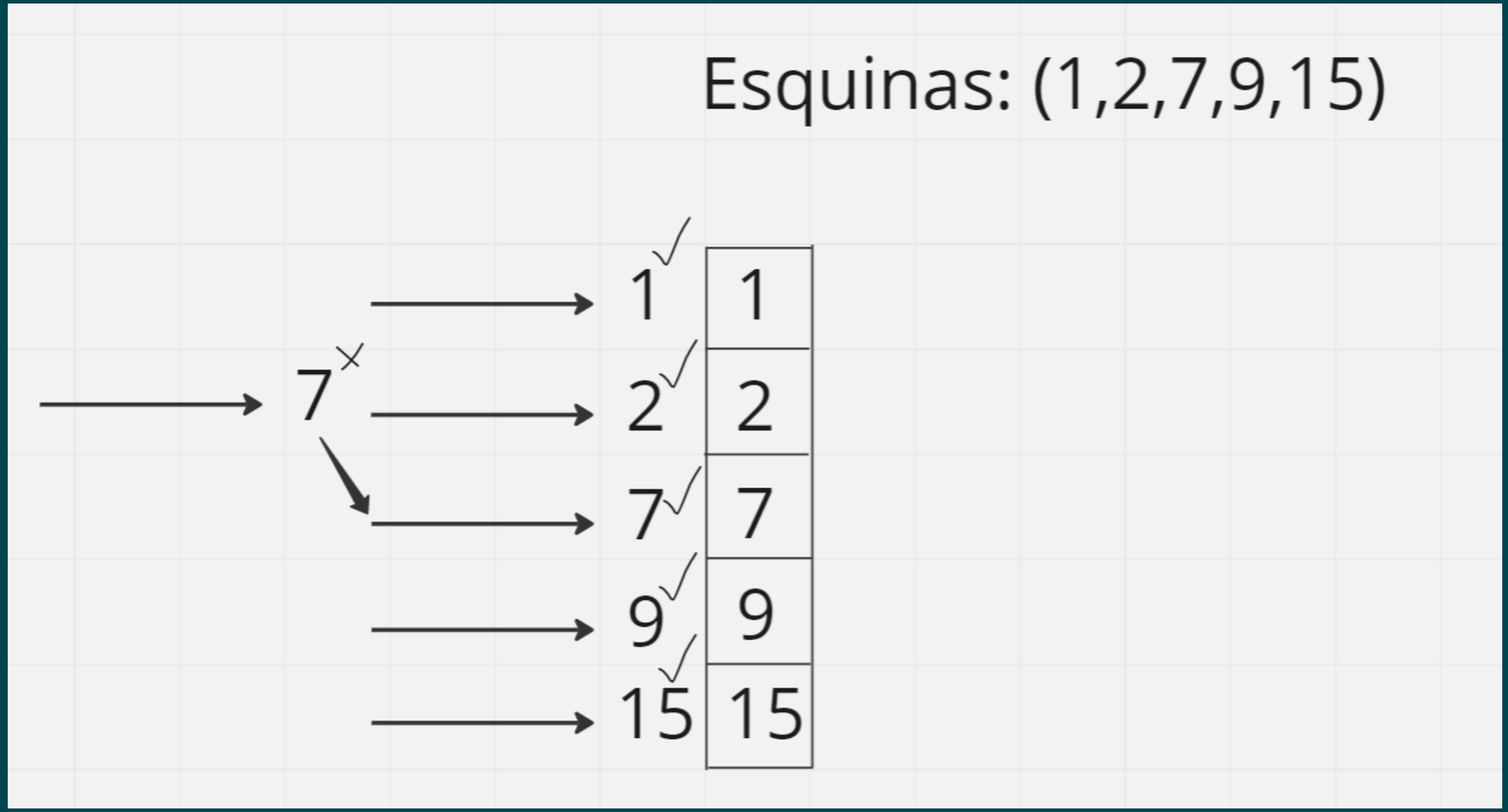
- Tenemos gran parte de las implementaciones pensadas y/o hechas, sólo que no tenemos cómo verificar si vamos bien o si hay algo que corregir.

- Encontrar la forma de que los datos coincidan con el vértice (ya que las esquinas no serán lineales).
- Crear mapa (nos falta un ejemplo concreto).
- ¿La carga de las direcciones de ubicaciones móviles o fijas tendrá la forma [e1, d1, e2, d2] o, se cargará dato por dato?

Plan para alcanzar el objetivo en fecha

- Pensamos que una vez que tengamos un ejemplo concreto del mapa, vamos a poder terminar todas las implementaciones, hacer las pruebas necesarias, probar qúe tan eficiente es nuestro programa, y obviamente, realizar las correcciones necesarias.

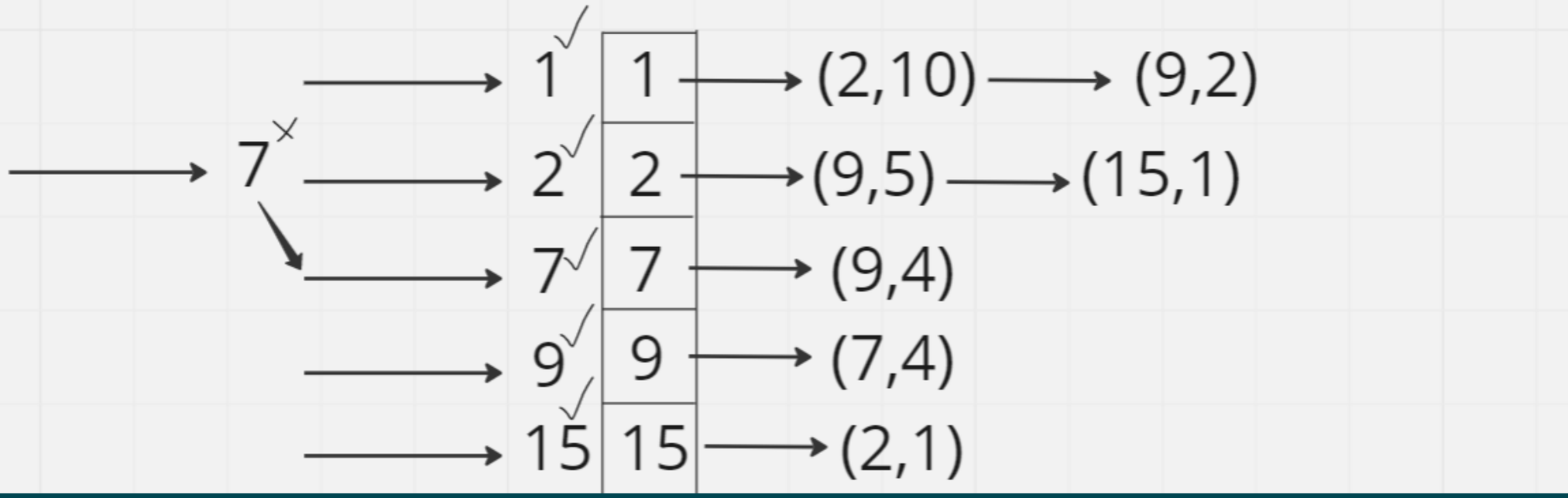
Linear probing



Ejemplo de carga

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)}



Gráfico

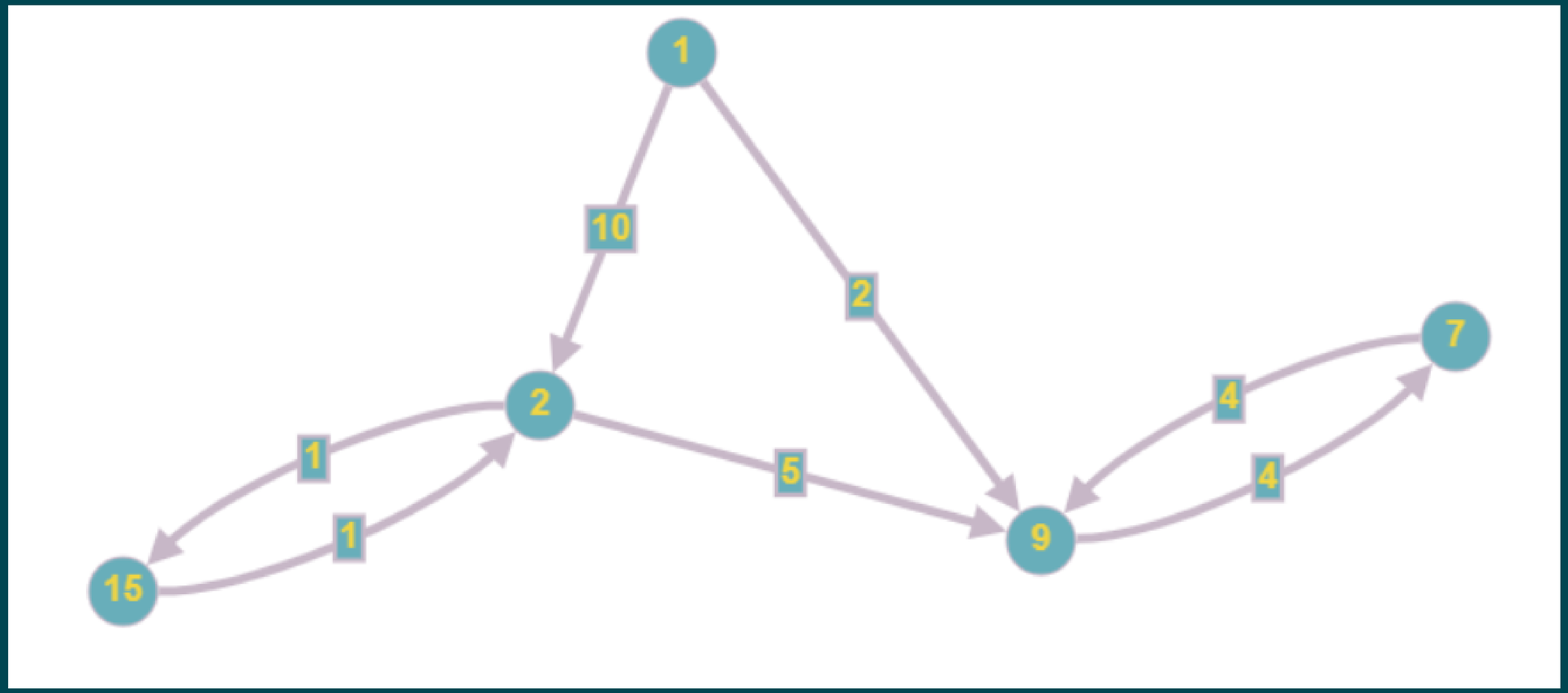


Tabla con datos relevantes

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)