Rodolfo Ferro
ferro@cimat.mx
https://rodolfoferro.xyz

Desvelando la Ciencia

Agosto, 2025

Caminos, Grafos y Tragos

La ciencia de volver a casa

ferro@cimat.mx

Presentación

Caminos, Grafos y Tragos

Rodolfo Ferro (ferro@cimat.mx)

- **Especialista en Métodos Estadísticos** @ CIMAT
- **Sr. SWE** @ Bisonic (GC Studios)
- **Profesor de DS** @ Código Facilito y ENES UNAM León
- **Profesor de AI** @ EdgeHub School of Innovation
- **Prev.:** ML Engineer @ Vindoo.ai (España), Sherpa Digital en IA @ Microsoft México, AI Research Assistant @ CIMAT & AI Research Intern @ Harvard.

ferro@cimat.mx

Presentación

Caminos, Grafos y Tragos

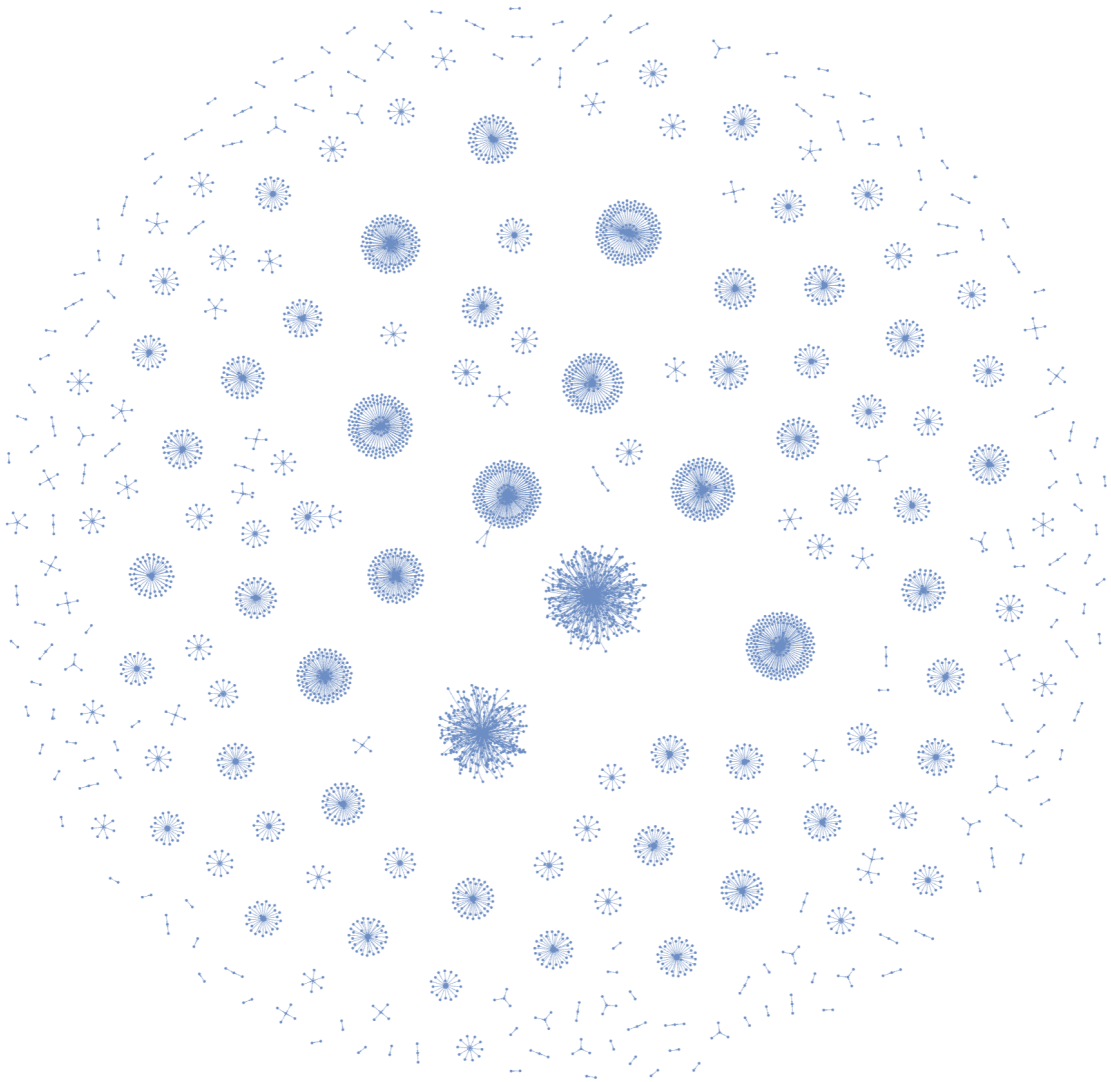
Temas de interés

- Computer Vision & Image Processing
- Deep Learning, Neural Networks (I'm more into discriminative AI than generative AI)
- Data Science & Machine Learning applied to industry problems
- Boardgames & TCGs

ferro@cimat.mx

Motivación

Caminos, Grafos y Tragos



Motivación

- Muchas aplicaciones computacionales involucran no solo un conjunto de elementos sino que también conexiones entre pares de elementos.

ferro@cimat.mx

Motivación

- Muchas aplicaciones computacionales involucran no solo un conjunto de elementos sino que también conexiones entre pares de elementos.
- Las relaciones que resultan de estas conexiones nos llevan a preguntas como:

ferro@cimat.mx

Motivación

- Muchas aplicaciones computacionales involucran no solo un conjunto de elementos sino que también conexiones entre pares de elementos.
- Las relaciones que resultan de estas conexiones nos llevan a preguntas como:
  - ¿Existen formas de recorrer las referencias (caminos de relaciones)? Es decir, ¿de llegar de un elemento inicial a uno final siguiendo las conexiones?

ferro@cimat.mx

Motivación

- Muchas aplicaciones computacionales involucran no solo un conjunto de elementos sino que también conexiones entre pares de elementos.
- Las relaciones que resultan de estas conexiones nos llevan a preguntas como:
  - ¿Existen formas de recorrer las referencias (caminos de relaciones)? Es decir, ¿de llegar de un elemento inicial a uno final siguiendo las conexiones?
  - ¿Cuáles y cuántos elementos se pueden alcanzar a partir de un elemento dado?

ferro@cimat.mx

Motivación

- Muchas aplicaciones computacionales involucran no solo un conjunto de elementos sino que también conexiones entre pares de elementos.
- Las relaciones que resultan de estas conexiones nos llevan a preguntas como:
  - ¿Existen formas de recorrer las referencias (caminos de relaciones)? Es decir, ¿de llegar de un elemento inicial a uno final siguiendo las conexiones?
  - ¿Cuáles y cuántos elementos se pueden alcanzar a partir de un elemento dado?
  - ¿Cuál es la mejor forma de llegar de un elemento a otro?

ferro@cimat.mx

Motivación

- Muchas aplicaciones computacionales involucran no solo un conjunto de elementos sino que también conexiones entre pares de elementos.
- Las relaciones que resultan de estas conexiones nos llevan a preguntas como:
  - ¿Existen formas de recorrer las referencias (caminos de relaciones)? Es decir, ¿de llegar de un elemento inicial a uno final siguiendo las conexiones?
  - ¿Cuáles y cuántos elementos se pueden alcanzar a partir de un elemento dado?
  - ¿Cuál es la mejor forma de llegar de un elemento a otro?
  - ¿Existen otras características relevantes que podamos identificar?

ferro@cimat.mx

SOLUCIÓN: GRAFOS

Pregunta: ¿Qué son los grafos?

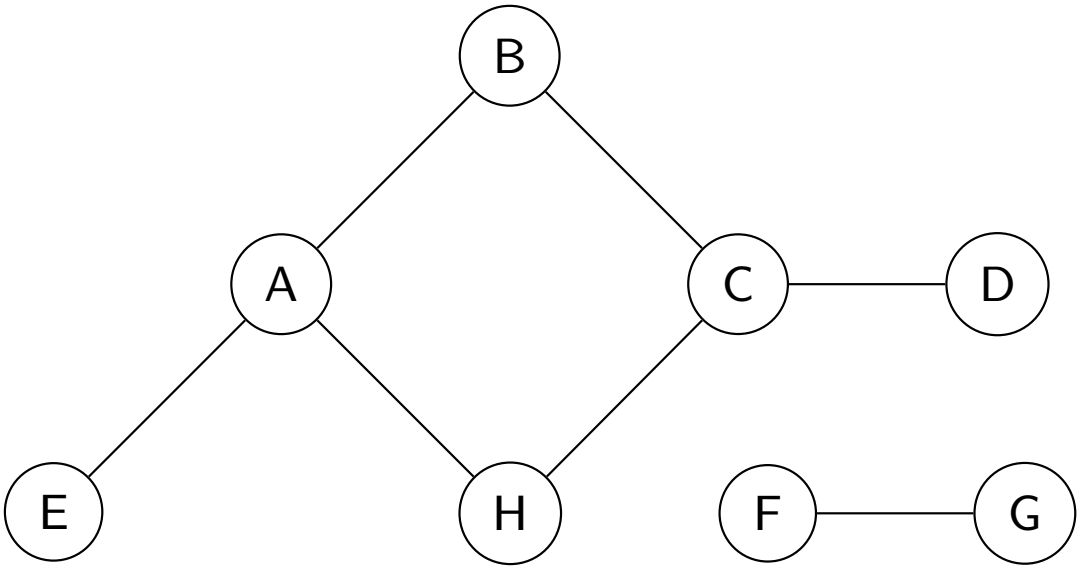
ferro@cimat.mx

Introducción a grafos 1

- Un **grafo** G es un par de conjuntos $G = (V, E)$.
- V es un conjunto de n objetos arbitrarios (usualmente denotados u, v) llamados **vértices** o **nodos**.
- E es un conjunto de **aristas**, **ejes** o **arcos** que unen pares de elementos $u, v \in V$.
- Las aristas son típicamente pares de vértices, $\{u, v\} \in E$ definiendo una **relación** entre el conjunto V con sí mismo: $E \subseteq V \times V$.

ferro@cimat.mx

Introducción a grafos 1



$G = (V, E)$

Vértices (V):
A, B, C, D, E, F, G, H

Aristas (E):
{A, B}, {A, H}, {B, C}, {C, D}, {C, H}, {E, A}, {F, G}

ferro@cimat.mx

Introducción a grafos 2

- En un **grafo no dirigido** las aristas son pares no ordenados, donde la relación $\{u, v\} = \{v, u\}$.
- En un **grafo dirigido**, o **digrafo**, las aristas son pares ordenados de vértices (están dirigidos) y $\{u, v\} \neq \{v, u\}$. (Y la conexión la representamos con una flechita $\longrightarrow$.)
- Si se asignan pesos o costos a las aristas, el grafo G es **ponderado** (o con pesos).
- Un grafo está conectado (es **conexo**) si existe un camino* de cualquier vértice u de V a cualquier otro vértice v de V. En caso contrario está desconectado (es **disconexo**).

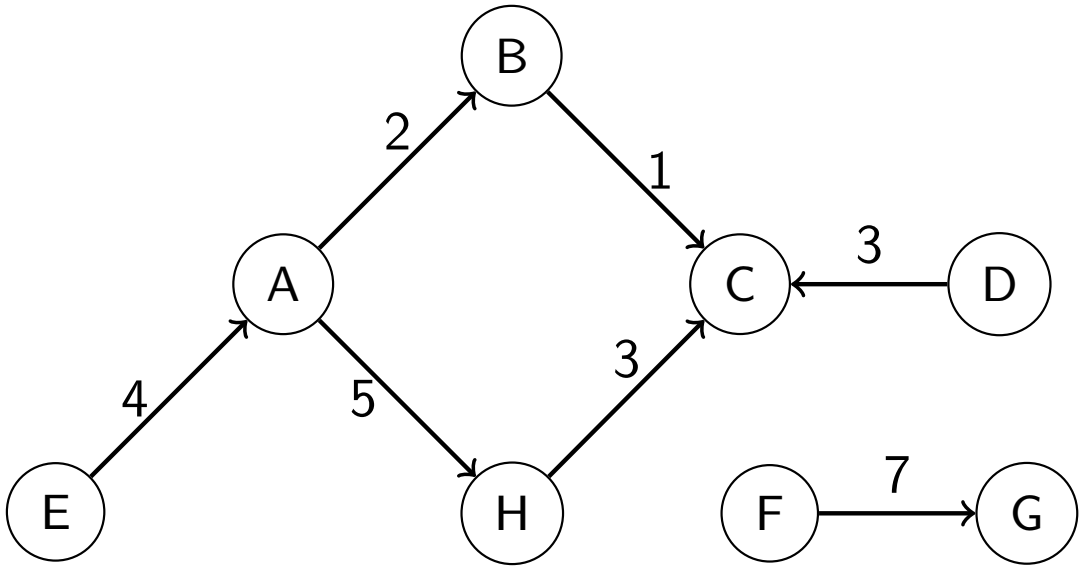
ferro@cimat.mx

Introducción a grafos 3

- Dos aristas $\{u_1, u_2\}$ y $\{v_1, v_2\}$ son **adyacentes** si tienen un vértice común.
- Una arista es **incidente** a un vértice si ésta lo une al vértice.
- Dos vértices u y v son **adyacentes** si una arista los une: $\{u, v\} \in E$.
- El **grado** $\delta(v)$ de un nodo $v \in V$ es el número de aristas de E que inciden en v (el número de vecinos).
- El **grado** $\Delta(G)$ de un grafo es el máximo de los grados de sus vértices: $max(\delta(v_i)), \forall v_i \in V$.

ferro@cimat.mx

Introducción a grafos 3



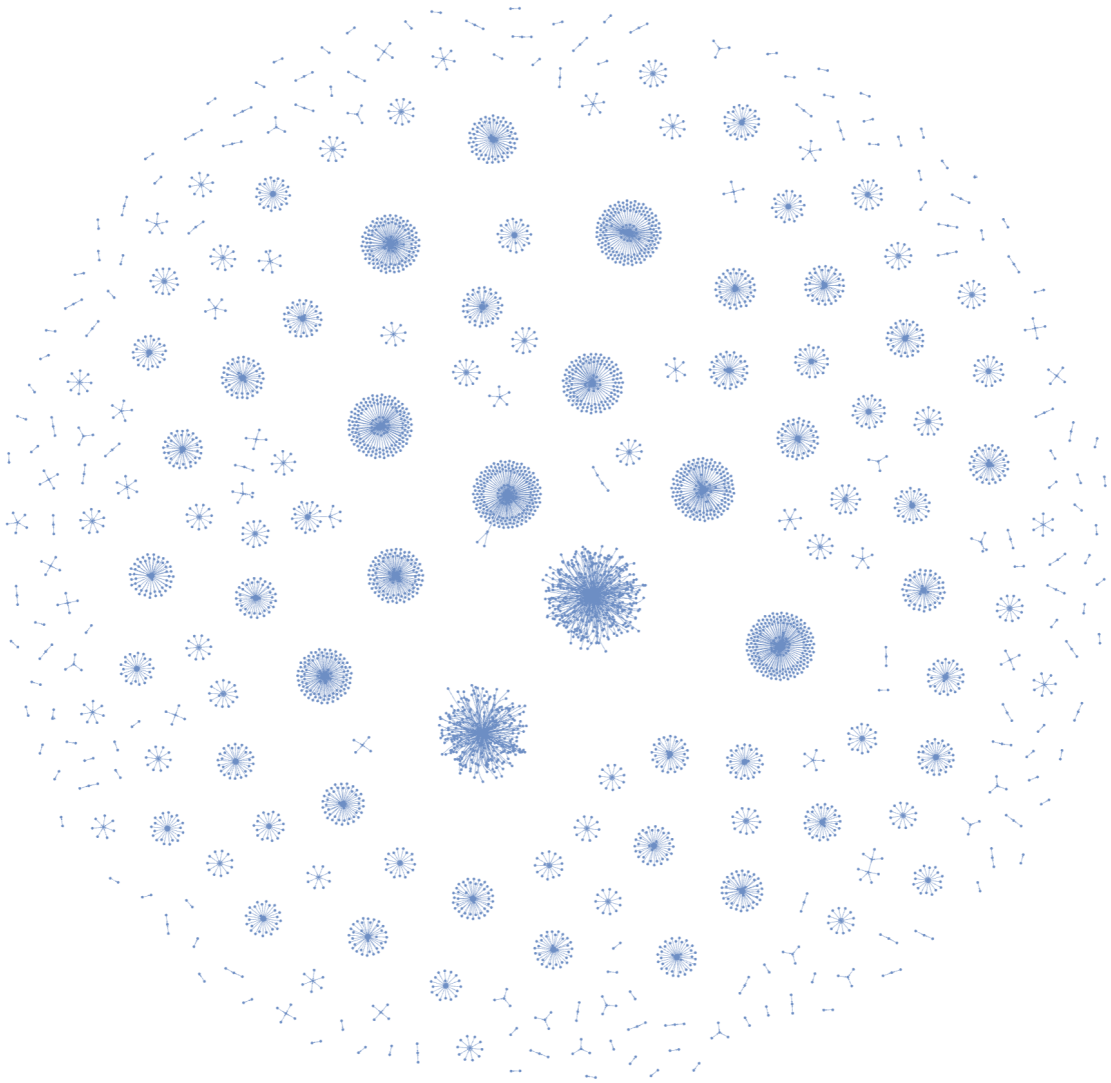
$G = (V, E)$

Vértices (V):
A, B, C, D, E, F, G, H

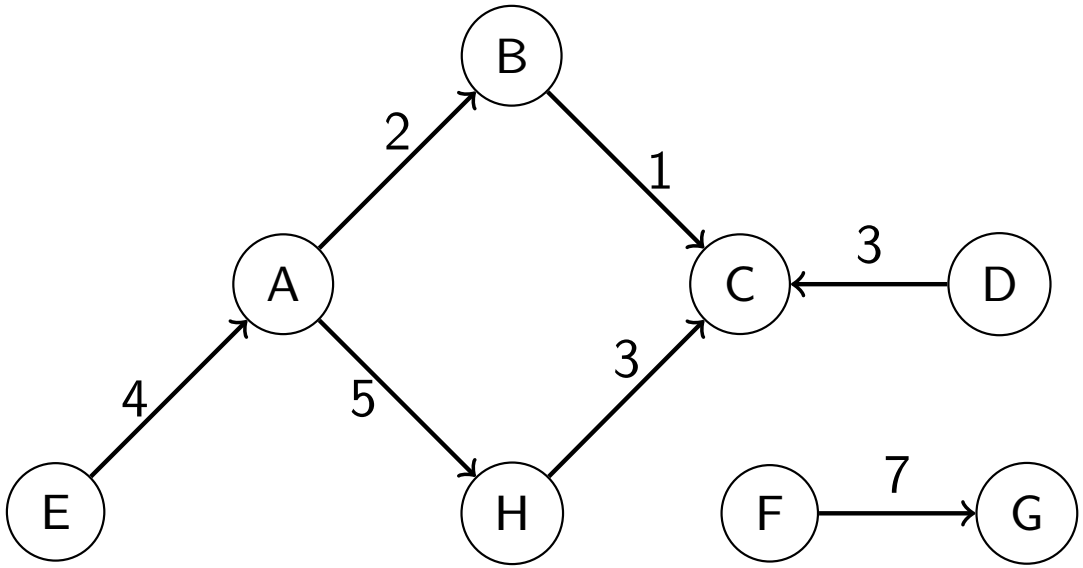
Aristas (E):
{A, B}, {A, H}, {B, C}, {D, C}, {H, C}, {E, A}, {F, G}

ferro@cimat.mx

Introducción a grafos 3



Introducción a grafos 3



$G = (V, E)$

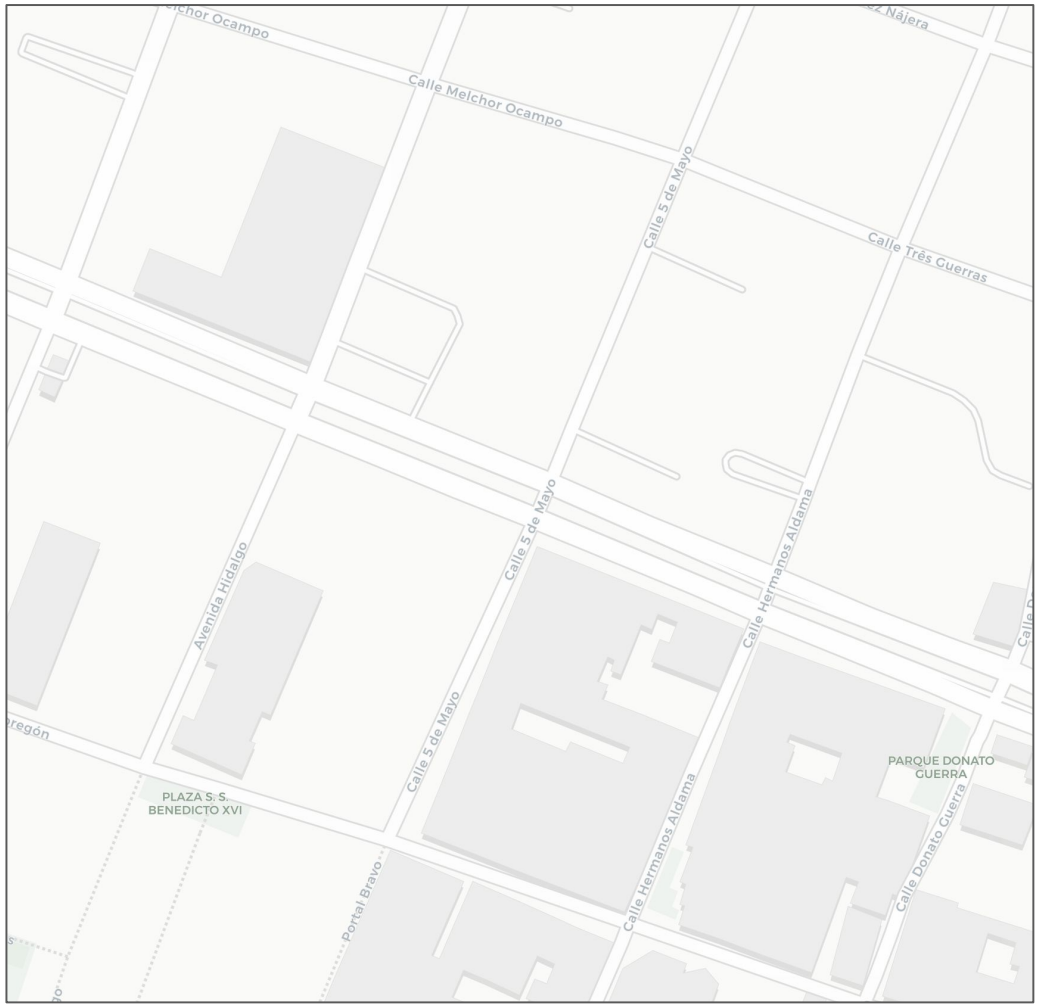
Vértices (V):
A, B, C, D, E, F, G, H

Aristas (E):
{A, B}, {A, H}, {B, C}, {D, C}, {H, C}, {E, A}, {F, G}

ferro@cimat.mx

Introducción a grafos 4

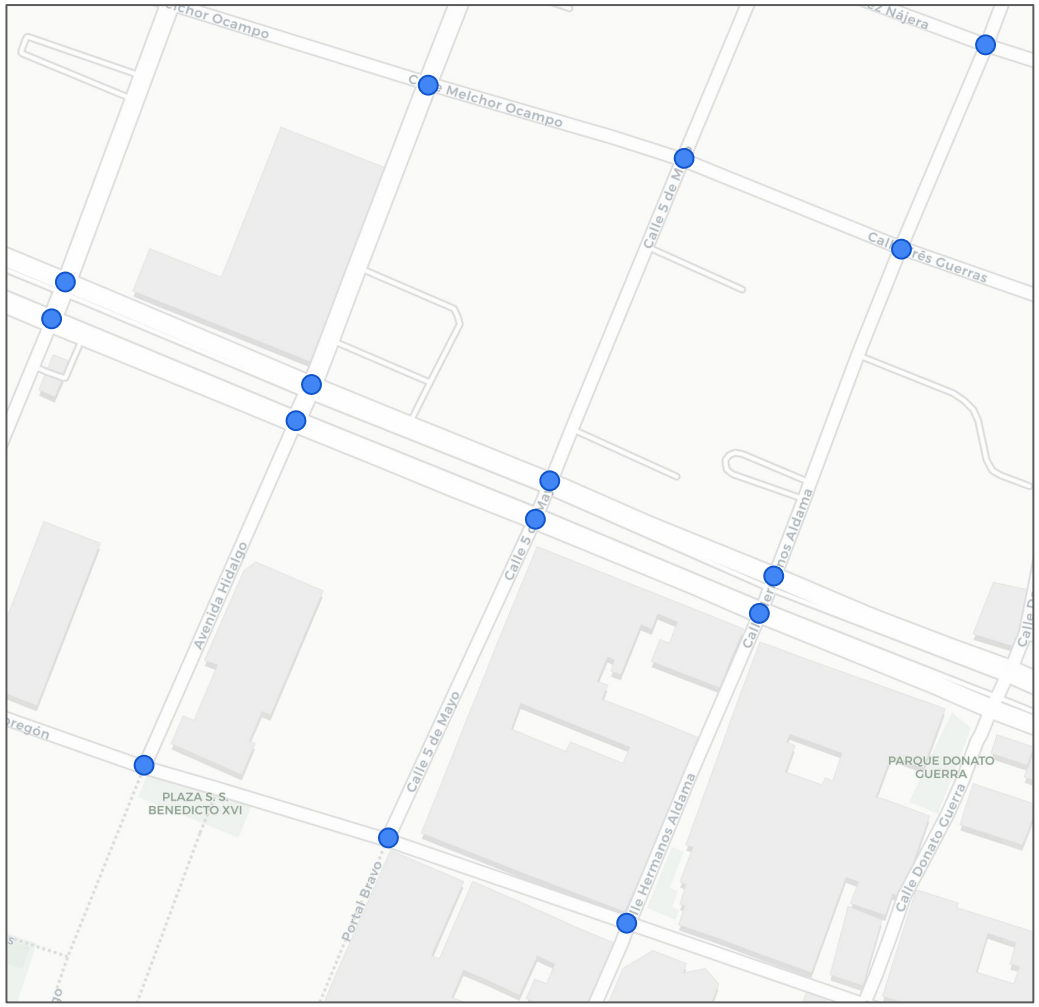
Llevándolo al contexto de una ciudad...
¡Podemos modelar la red de vialidades con grafos!



ferro@cimat.mx

Introducción a grafos 4

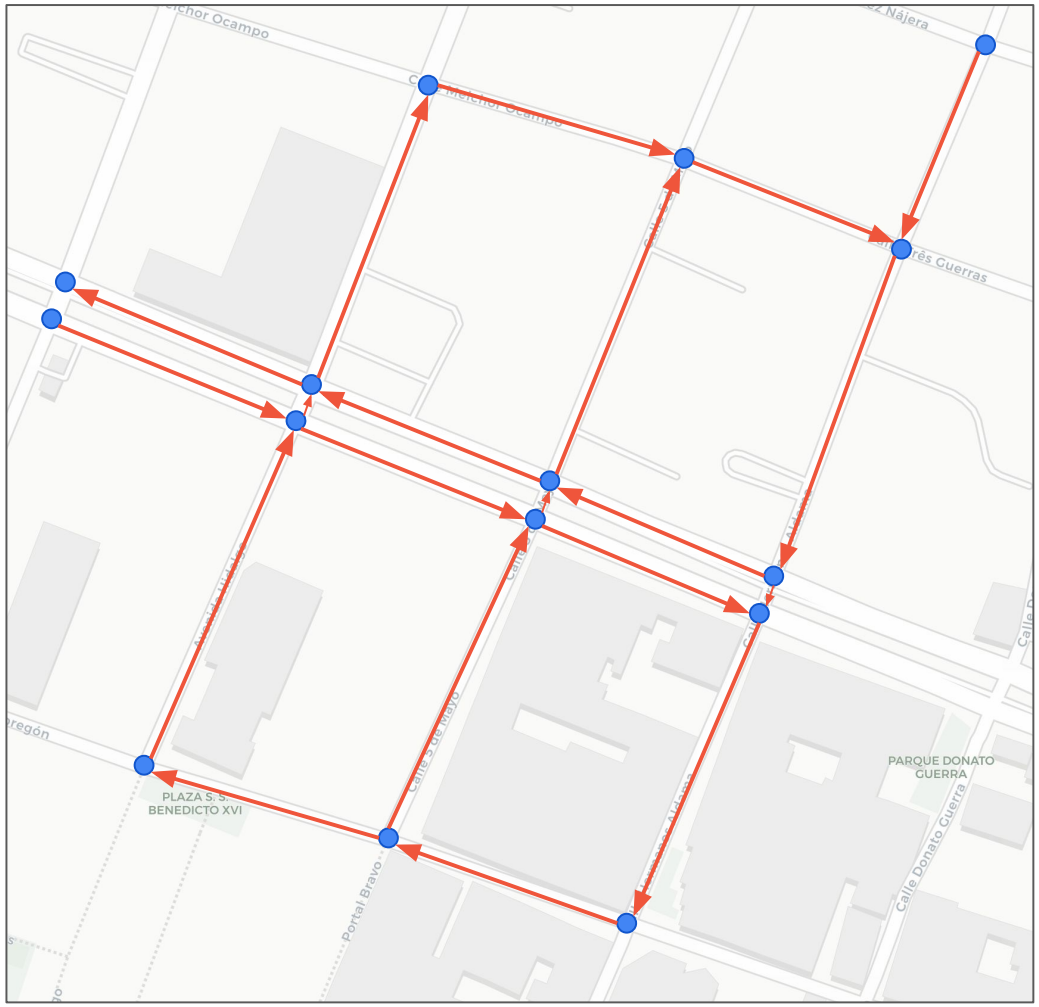
Llevándolo al contexto de una ciudad...
¡Podemos modelar la red de vialidades con grafos!



ferro@cimat.mx

Introducción a grafos 4

Llevándolo al contexto de una ciudad...
¡Podemos modelar la red de vialidades con grafos!

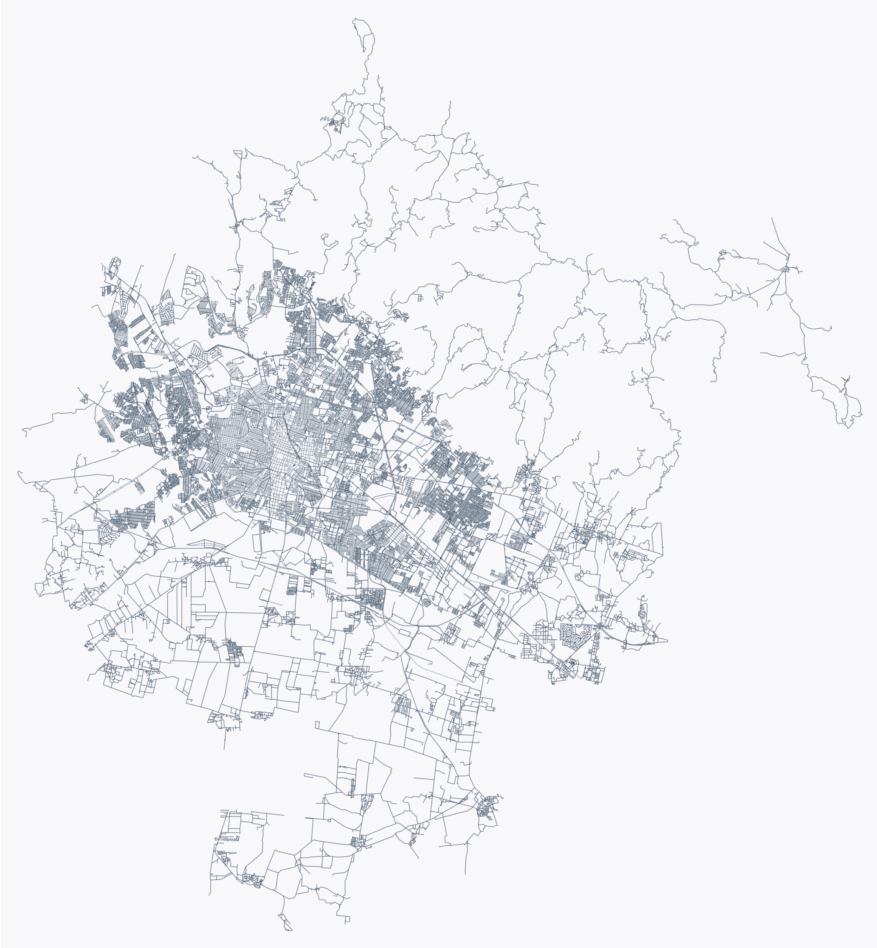


ferro@cimat.mx

Introducción a grafos 4

¡Podemos modelar la red de vialidades con grafos!

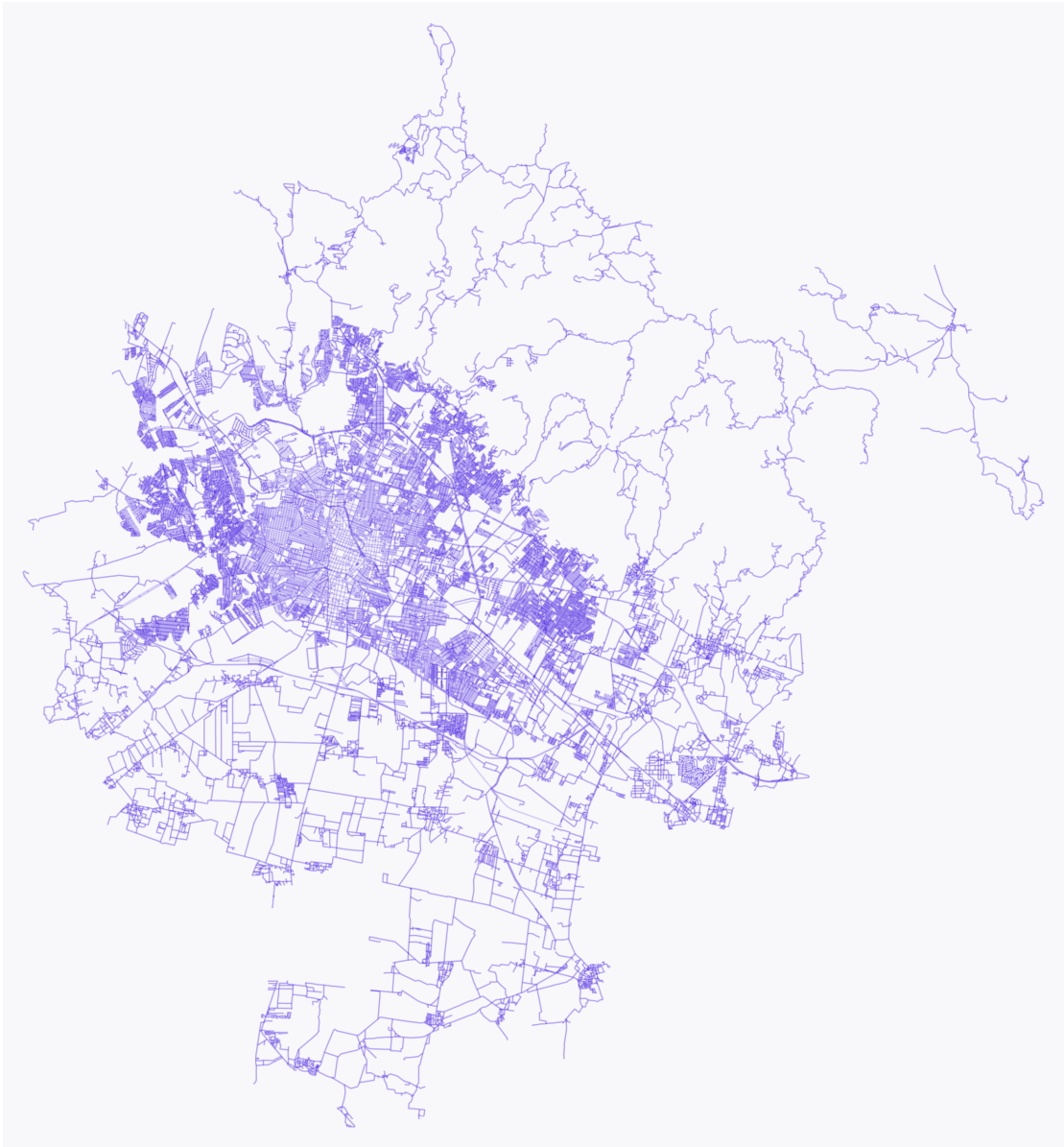
Caso: León, Guanajuato



ferro@cimat.mx

Introducción a grafos 4

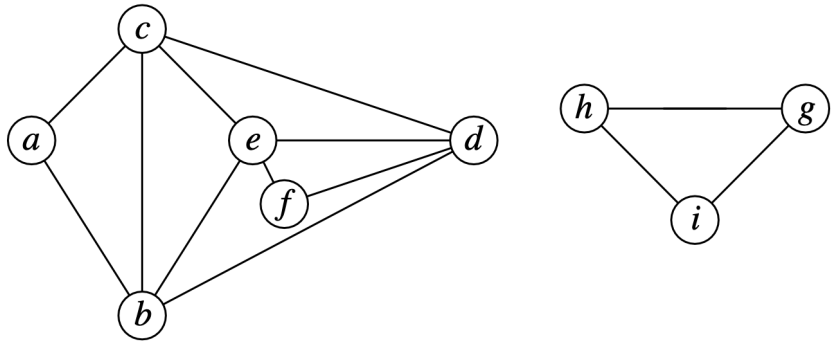
Caminos, Grafos y Tragos



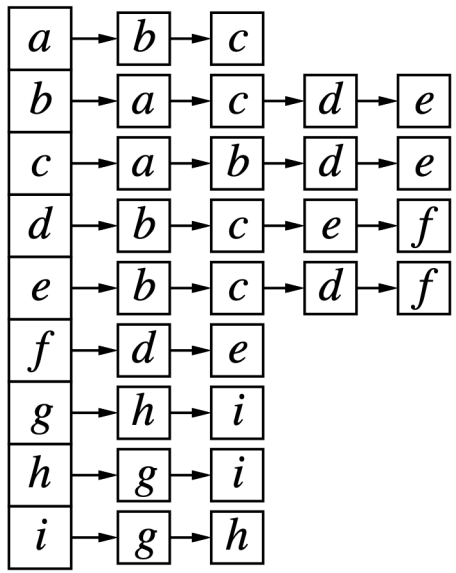
ferro@cimat.mx

Representación de grafos en la compu

Matrices de adyacencia, listas de adyacencia.



	a	b	c	d	e	f	g	h	i
a	0	1	1	0	0	0	0	0	0
b	1	0	1	1	1	0	0	0	0
c	1	1	0	1	1	0	0	0	0
d	0	1	1	0	1	1	0	0	0
e	0	1	1	1	0	1	0	0	0
f	0	0	0	1	1	0	0	0	0
g	0	0	0	0	0	0	0	1	0
h	0	0	0	0	0	0	1	0	1
i	0	0	0	0	0	0	1	1	0



ferro@cimat.mx

Problema de los puentes de Königsberg

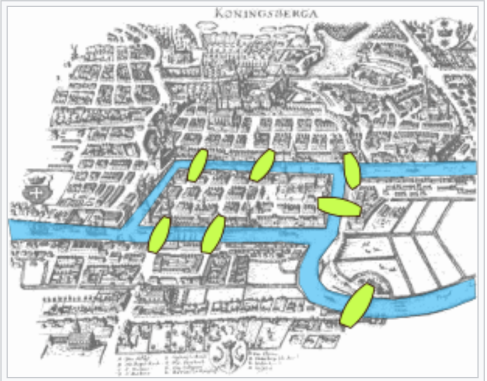
56 idiomas

Artículo Discusión Leer Editar Ver historial Herramientas

Coordenadas: 54°42′12″N 20°30′56″E (mapa)

El **problema de los puentes de Königsberg**, también llamado más específicamente **problema de los siete puentes de Königsberg**, es un célebre problema matemático resuelto por Leonhard Euler en 1736 y cuya resolución dio origen a la teoría de grafos.[1] Su nombre se debe a *Königsberg*, la ciudad de Prusia Oriental y luego de Alemania que desde 1945 se convirtió en la ciudad rusa de Kaliningrado.

Esta ciudad está atravesada por el río Pregolia. Este se bifurca y rodea con sus brazos a la isla Kneiphof,[2] de forma que el terreno queda dividido en cuatro regiones distintas, que entonces estaban unidas mediante siete puentes llamados puente del herrero, Puente Conector, Puente Verde, Puente del Mercado, Puente de Madera, Puente Alto y Puente de la Miel.[3] El problema se formuló en el siglo XVIII y consistía en encontrar un recorrido para cruzar a pie toda la ciudad pasando solo una vez por cada uno de los puentes y regresando al mismo punto de inicio.[4]



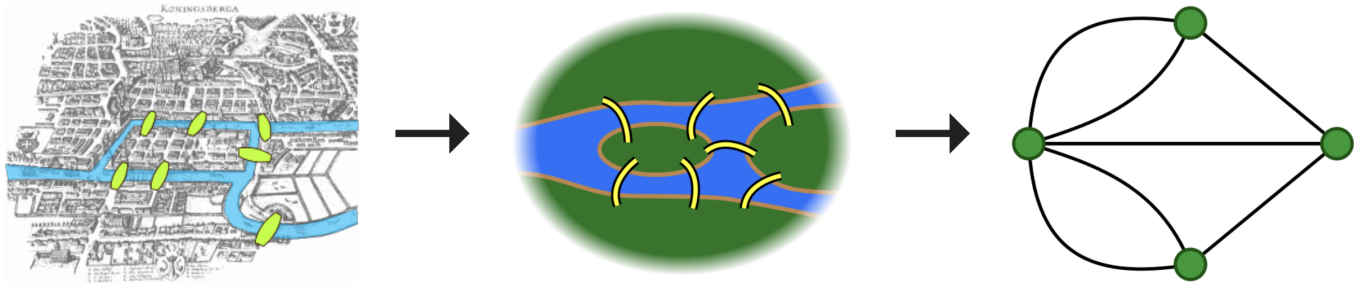
Mapa de Königsberg en la época de Leonhard Euler, que muestra dónde se encontraban los siete puentes (en verde claro) y las ramas del río (en celeste).

Caminos, Grafos y Tragos — Rodolfo Ferro (@rodo_ferro)

ferro@cimat.mx

Problema de los puentes de Königsberg. Wikipedia. https://w.wiki/F5YA

Recorridos en grafos 01



ferro@cimat.mx

En resumen...

Demostró que no se puede, pero hey, ¡dió origen a la teoría de grafos!

Problema de los puentes de Königsberg. Wikipedia. `https://w.wiki/F5YA`

Recorridos en grafos 02

- Un camino ϕ de tamaño n en una gráfica $G = (V, E)$ es una secuencia a $n+1$ vértices $u_i \in V$, para $0 \leq i \leq n$, tales que para todo $1 \leq i \leq n$, existe $e \in E$ tal que $\phi(e) = (u_{i-1}, u_i)$ o $\phi(e) = (u_i, u_{i-1})$.
- Dicho informalmente, todos los vértices después del primero son adyacentes a su predecesor.

ferro@cimat.mx

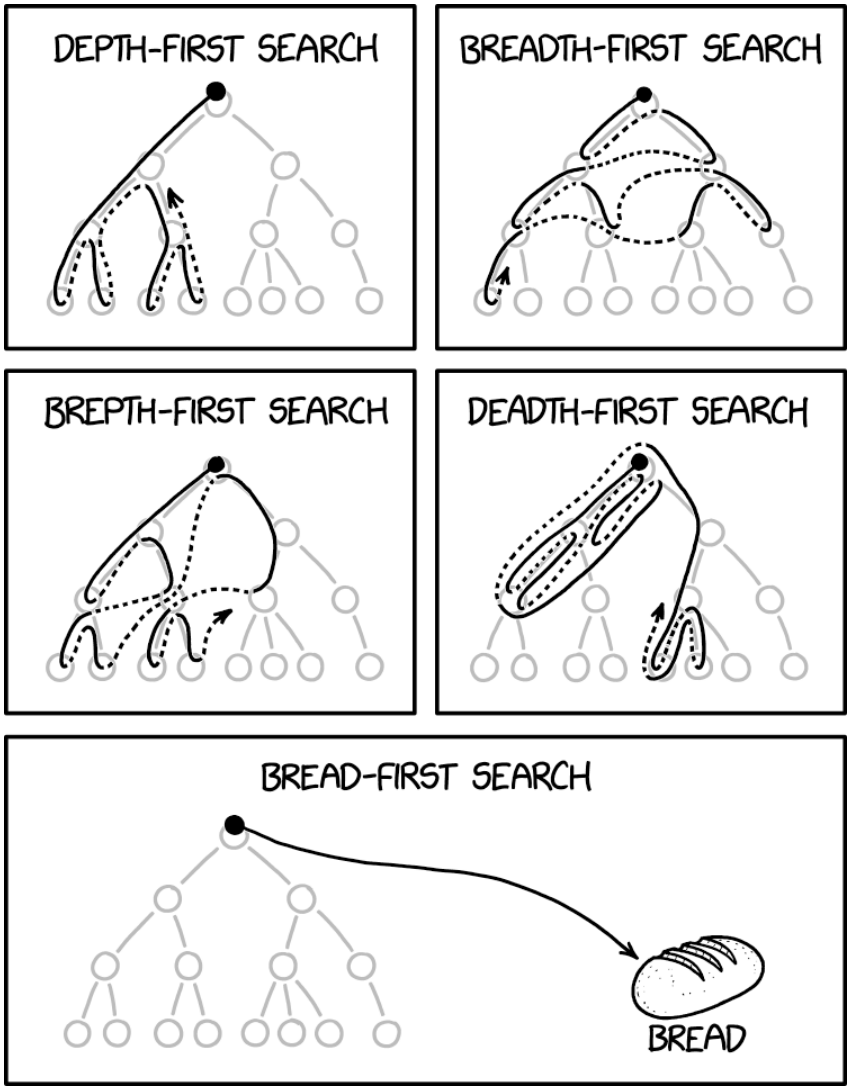
Recorridos en grafos 02

- Existen diferentes tipos de caminos:
  - **Camino simple:** todos los vértices y las aristas son distintos.
  - **Camino cíclico:** un camino simple excepto que el primer y el último vértice son el mismo.
  - **Camino Euleriano:** camino que pasa por cada arista una sola vez.
  - **Ciclo Euleriano:** camino cíclico que pasa por cada arista una sola vez, que inicia y termina en el mismo vértice.

ferro@cimat.mx

Recorridos en grafos 03

Caminos, Grafos y Tragos

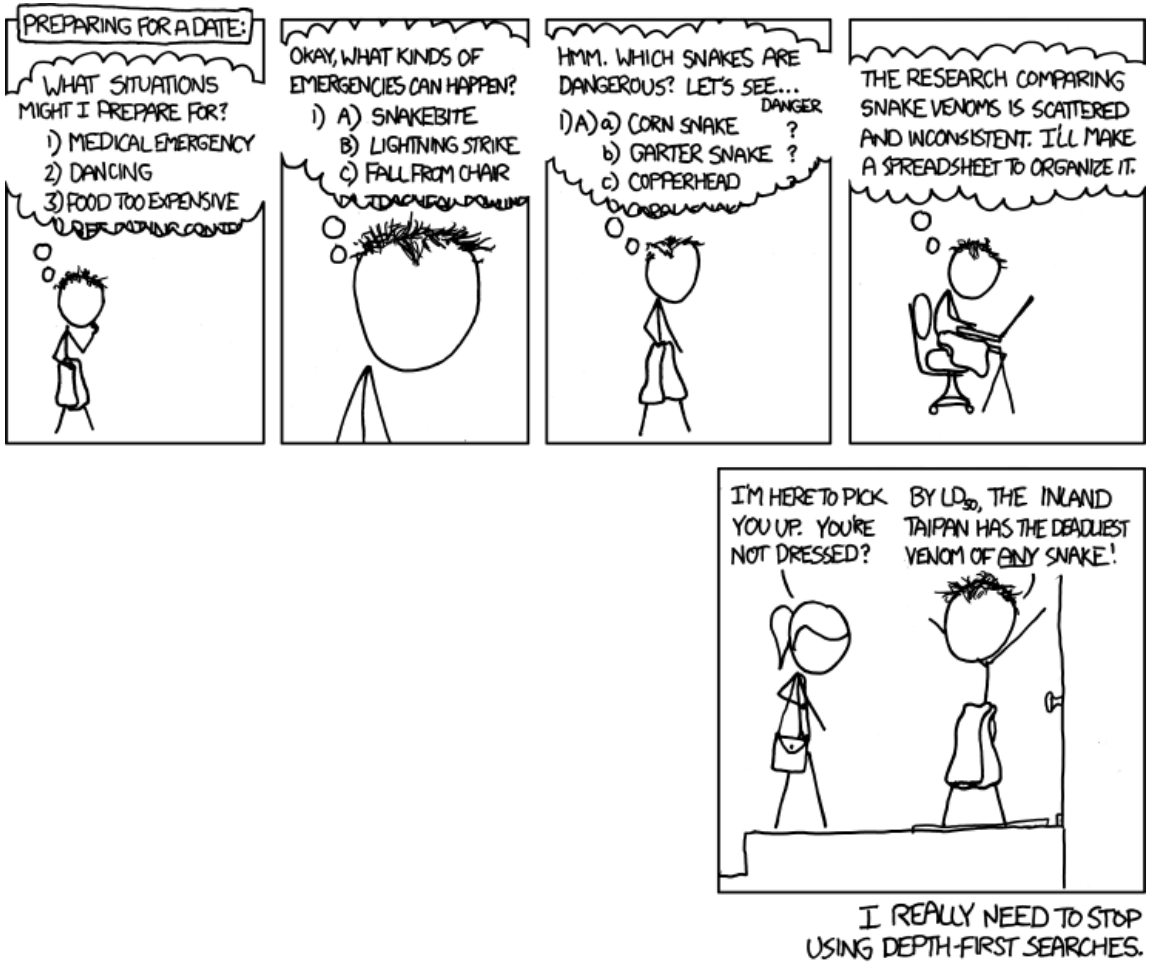


ferro@cimat.mx

Depth and Breadth. XKCD Comics. https://xkcd.com/2407/

Recorridos en grafos 04

Caminos, Grafos y Tragos



ferro@cimat.mx

Dijkstra

Caminos, Grafos y Tragos

Bae: Come over
Dijkstra: But there are so many routes to take and I don't know which one's the fastest
Bae: My parents aren't home
Dijkstra:

Dijkstra's algorithm

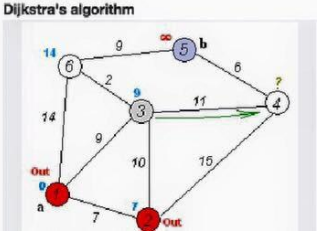
Graph search algorithm

Not to be confused with Dykstra's projection algorithm.

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.[1][2]

The algorithm exists in many variants; Dijkstra's original variant found the shortest path between two nodes,[2] but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree.

Dijkstra's algorithm



ferro@cimat.mx

Tomado de r/ProgrammerHumor

Dijkstra

- Edsger Wybe Dijkstra (1930-2002) fue pionero de la ciencia e industria de la computación. Matemático y físico teórico con PhD en CS. Turing Award en 1972.
- Fue de los primeros en opinar que la lógica matemática es indispensable para construir programas computacionales adecuados.
- Líder del movimiento de abolición de la operación `GOTO` de la programación (1968): `http://david.tribble.com/text/goto.html`

ferro@cimat.mx

Algoritmo de Dijkstra

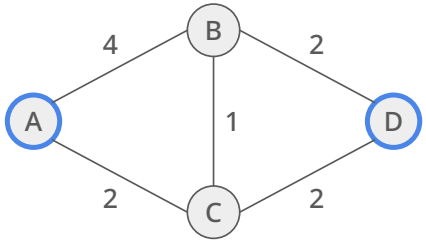
- Adecuado para algoritmos que tienen aristas con pesos **no negativos** (en caso contrario, otros como Bellman-Ford funcionan bien para ello).
- Es un algoritmo **glotón**, porque en cada paso toma la decisión que parece mejor en ese momento, sin esperar a ver todo el panorama.
- El algoritmo garantiza que una vez elegido un nodo con la distancia mínima, ya encontramos el camino más corto hacia él.

ferro@cimat.mx

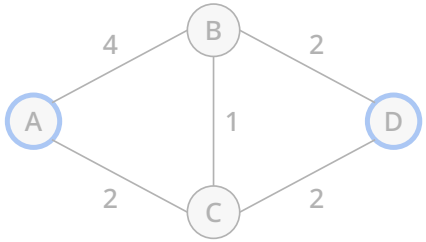
Algoritmo de Dijkstra

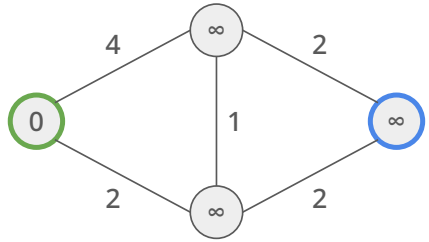
- El algoritmo selecciona el nodo no visitado con la distancia más pequeña conocida desde el origen.
- Una vez elegido, lo "marca" como ya resuelto.
- Esta estrategia de "escoger lo más barato disponible ahora" es lo que define a los algoritmos glotones.

ferro@cimat.mx

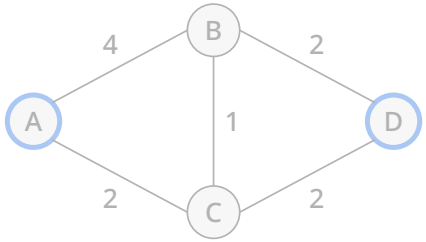


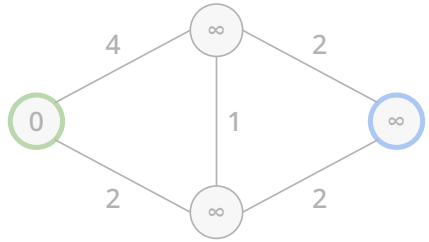
ferro@cimat.mx

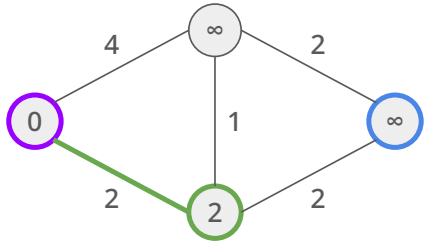




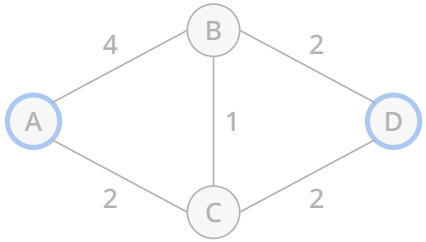
ferro@cimat.mx

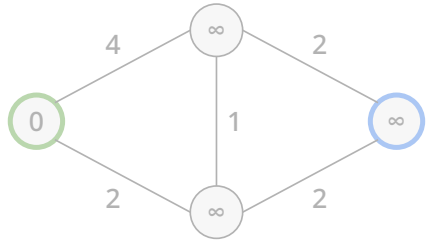


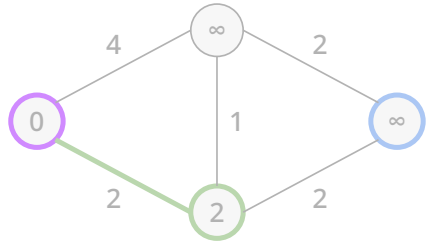


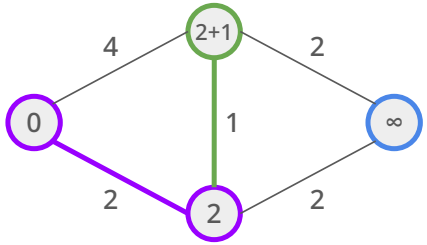


ferro@cimat.mx

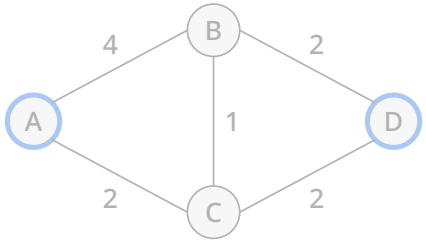


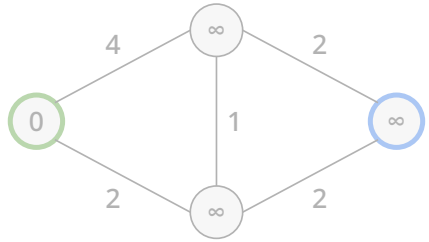


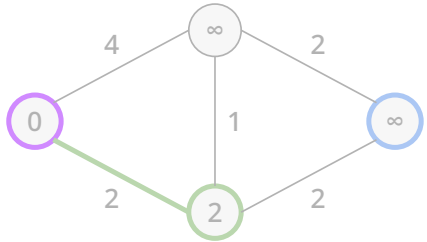


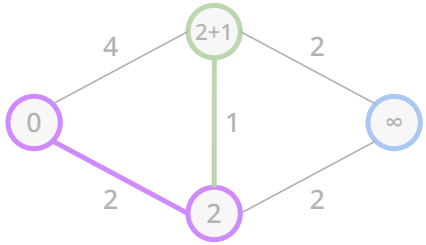


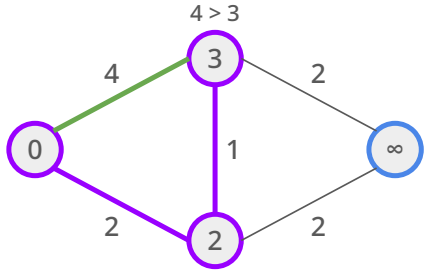
ferro@cimat.mx



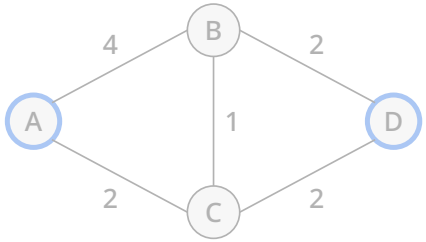


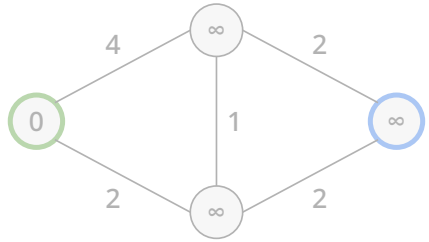


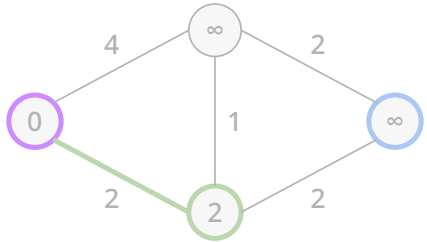


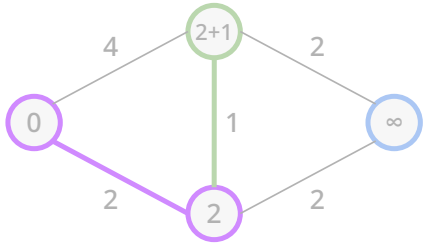


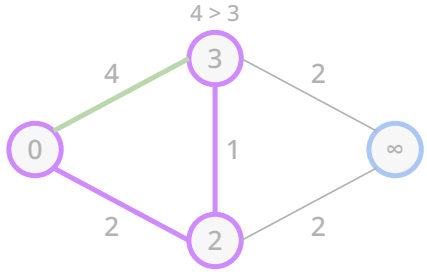
ferro@cimat.mx

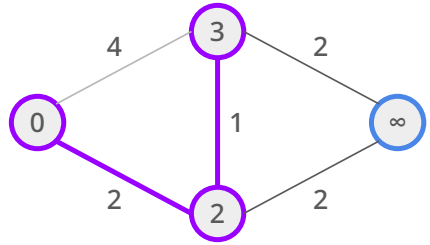




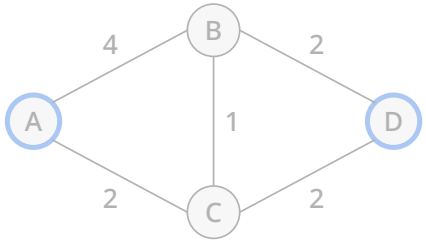


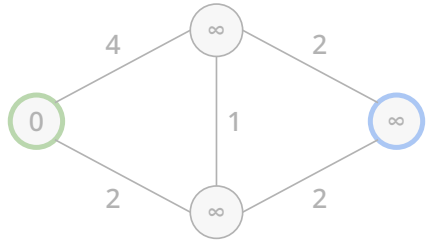


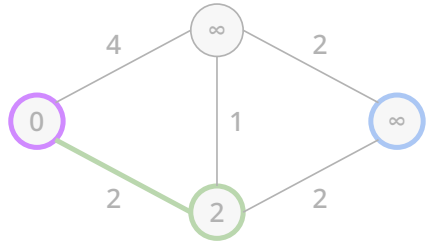


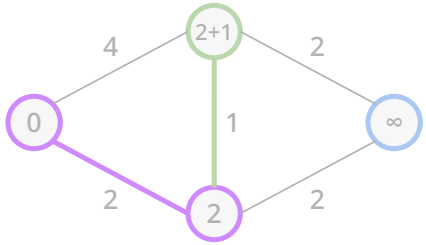


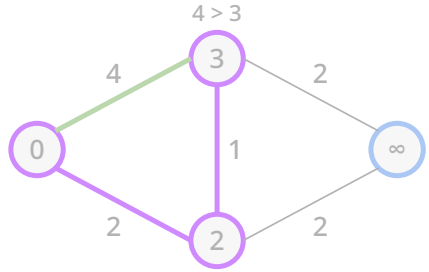
ferro@cimat.mx

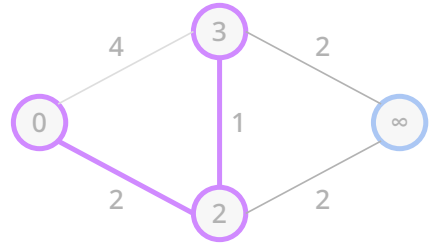


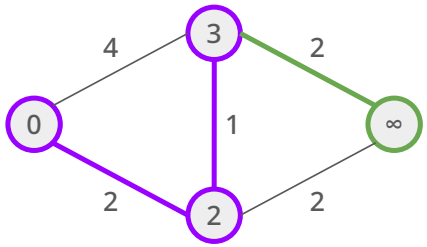




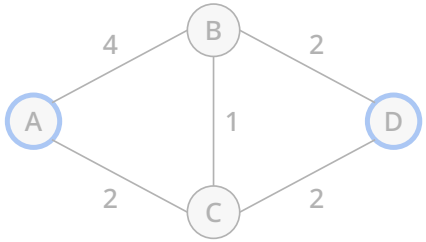


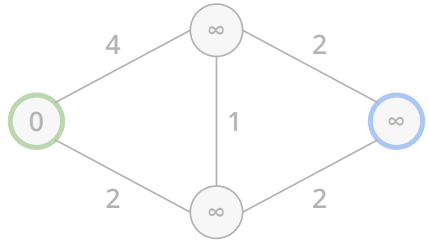


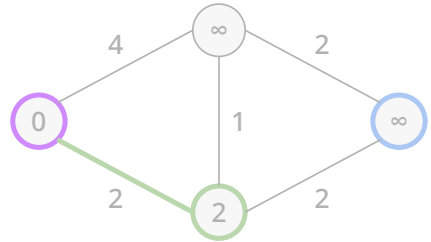


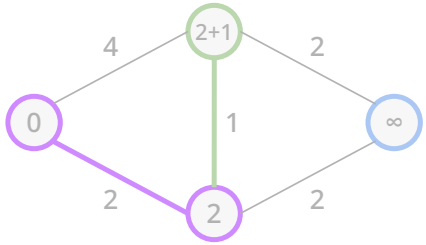


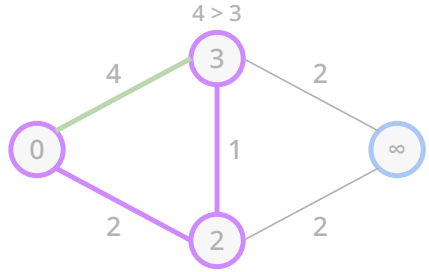
ferro@cimat.mx

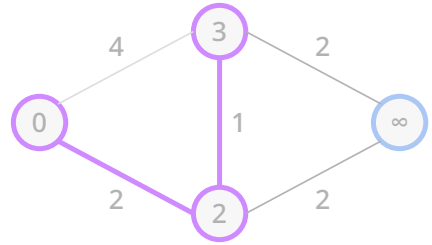


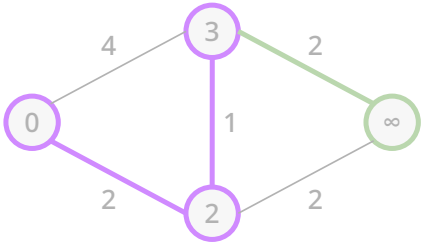


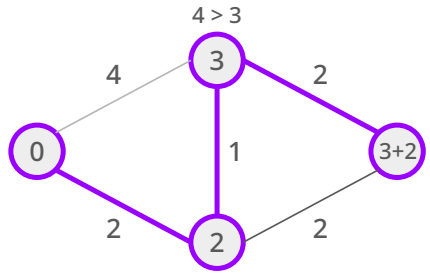




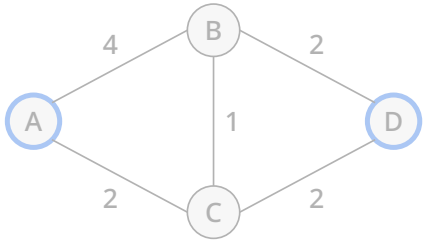


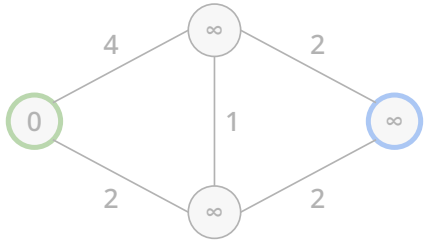


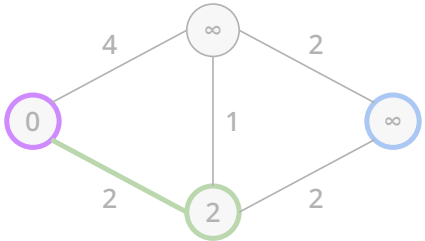


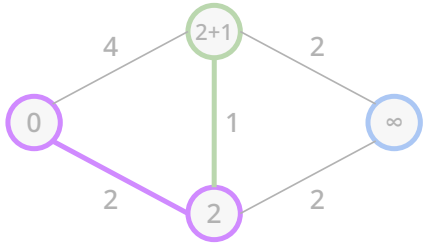


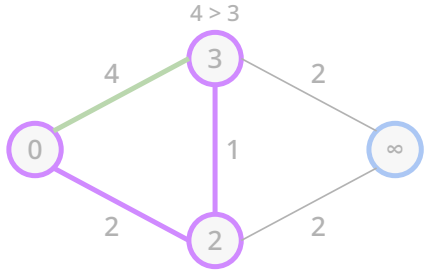
ferro@cimat.mx

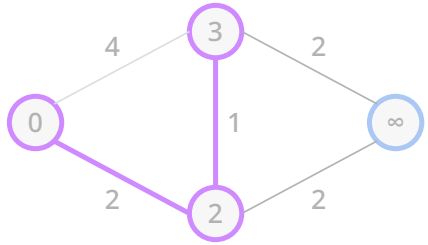


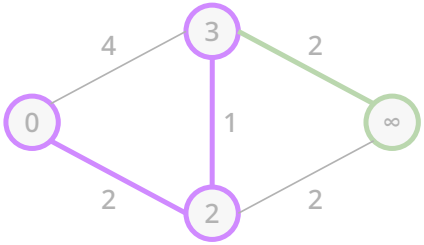


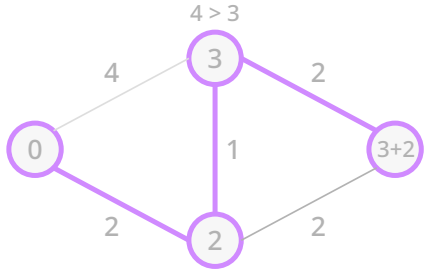


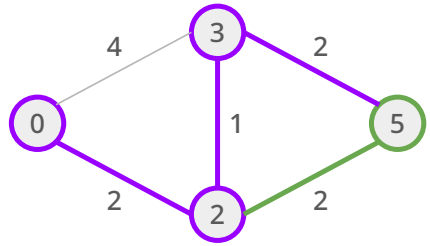




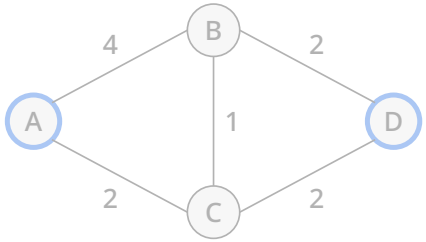


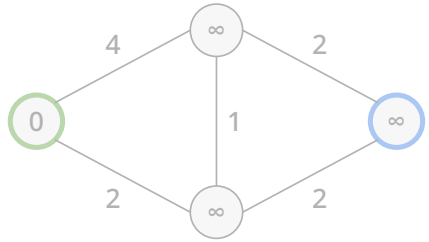


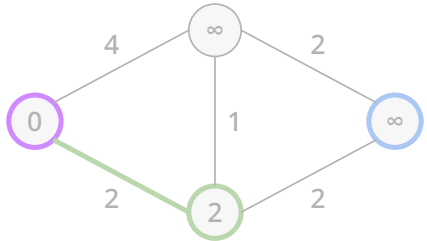


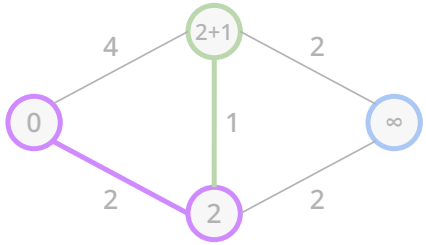


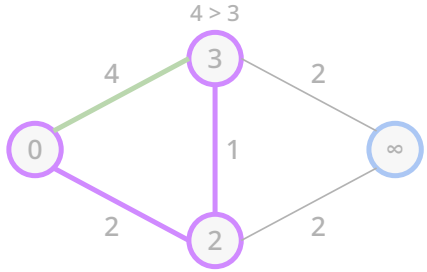
ferro@cimat.mx

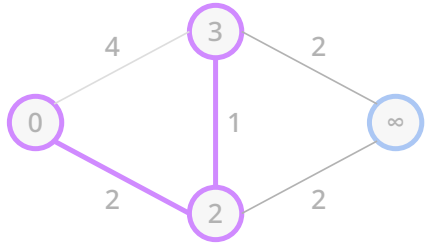


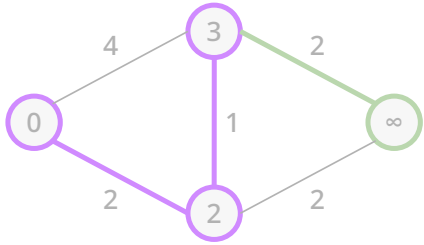


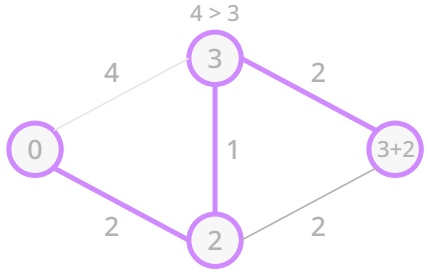


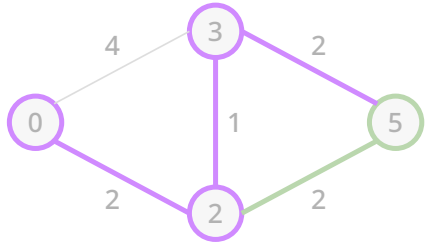


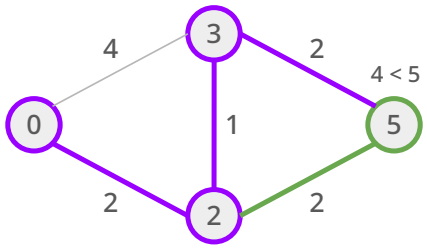




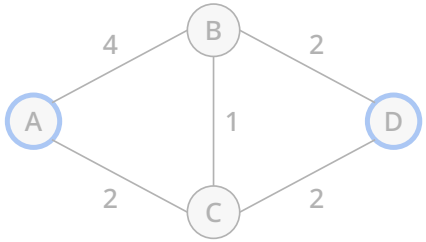


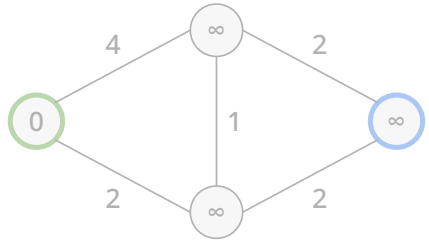


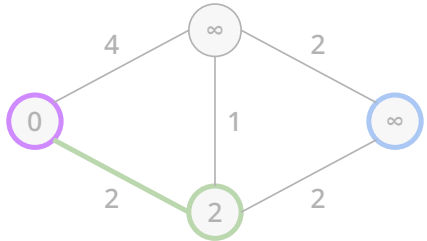


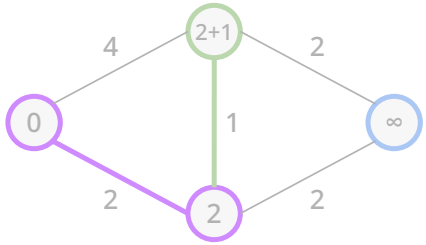


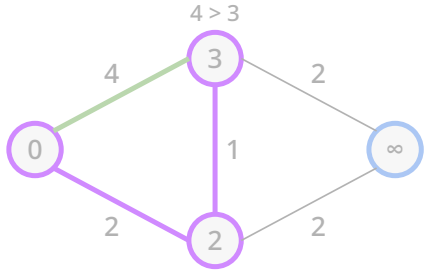
ferro@cimat.mx

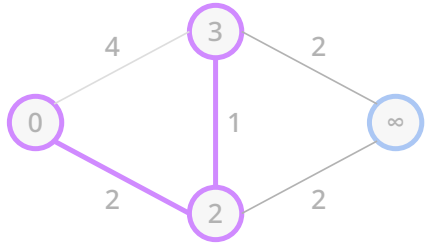


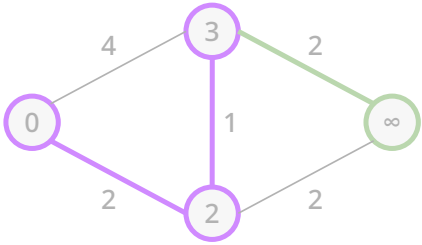


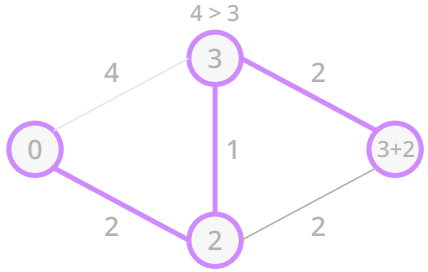


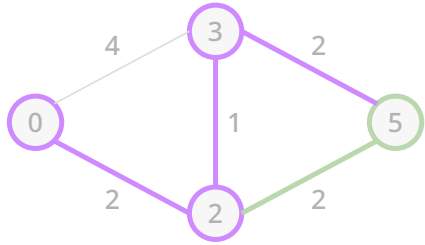


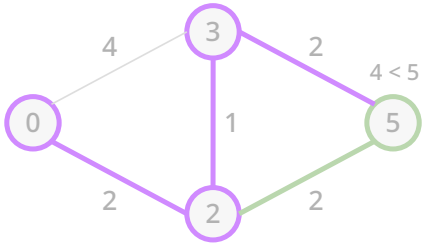


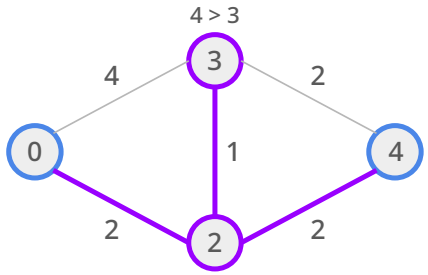












ferro@cimat.mx

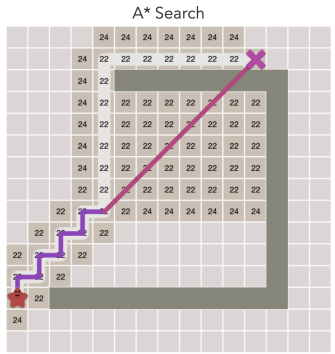
Algoritmo A^*

- Algoritmo descrito en 1968 por Peter Hart, Nils Nilsson y Bertram Raphael.
- En 1964, Nilsson inventó un método que utiliza una heurística para mejorar el tiempo de ejecución del algoritmo de Dijkstra: algoritmo A1.
- En 1967, Raphael mejoró este algoritmo pero no pudo probar si era óptimo: algoritmo A2.
- En 1968, Hart probó que A2 es óptimo cuando la heurística cumple ciertas características. Llamó al algoritmo A* (todos las posibles versiones del algoritmo A)
- Esta heurística es particularmente útil cuando no se conoce el grafo.

ferro@cimat.mx

Algoritmo A^*

- Se utiliza para encontrar el camino más corto entre un nodo origen s y un nodo destino t.
- Usa una función GUESS-DISTANCE(v, t) que da un estimado de la distancia de v a t.
- La diferencia entre Dijkstra y A* es que en cada vértice v, en lugar de almacenar $dist(v)$, almacena $dist(v) +$ GUESS-DISTANCE(v, t).



- Se puede usar para resolver juegos de rompecabezas (Minesweeper, 15-puzzle, Sokoban, Cubo de Rubik, Snake, etc.).

ferro@cimat.mx

¡Algoritmos en acción!

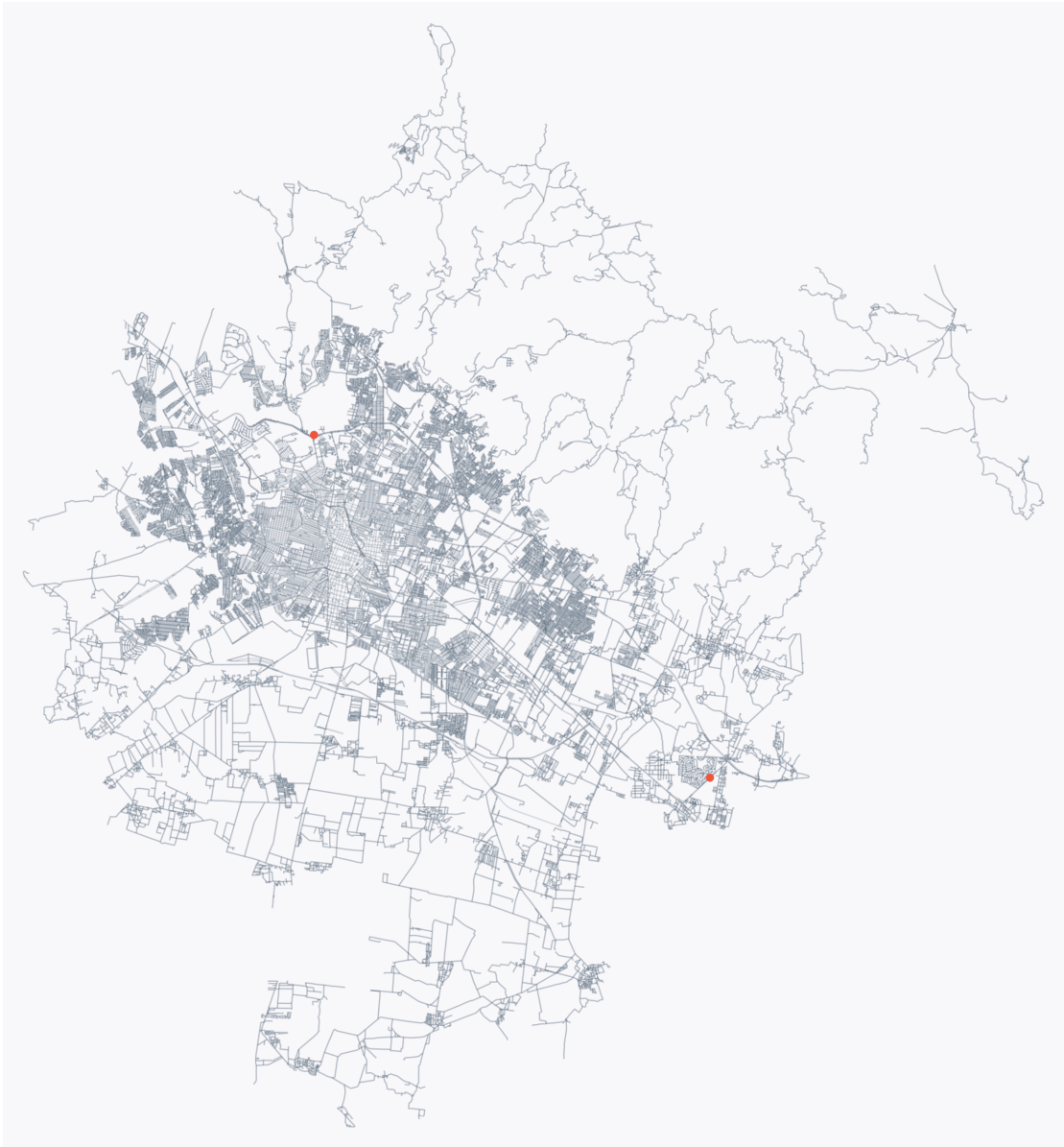
Exploremos cómo funcionan Dijkstra y A* en un laberinto.

Website: `https://algo-vz.netlify.app/`

ferro@cimat.mx

Encontrando caminos en León

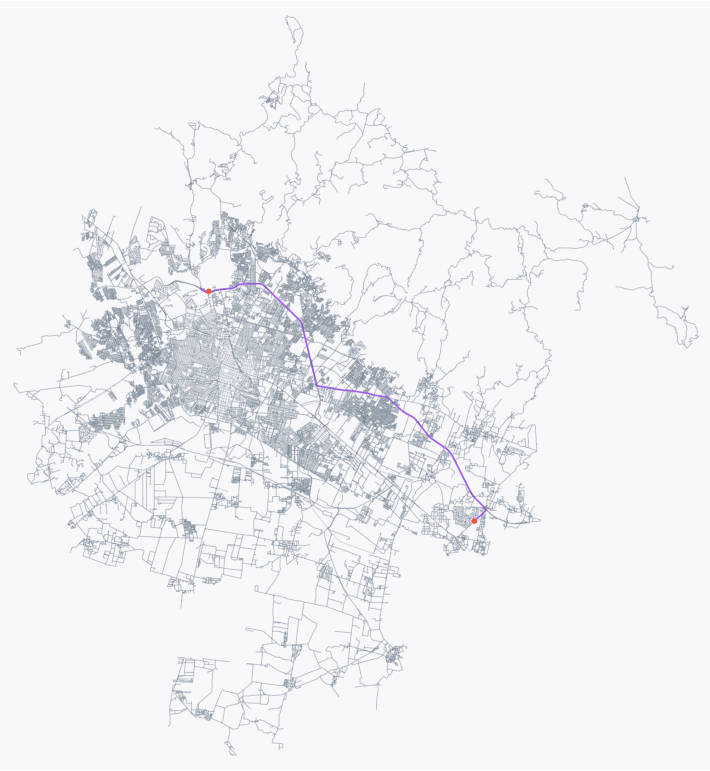
Caminos, Grafos y Tragos

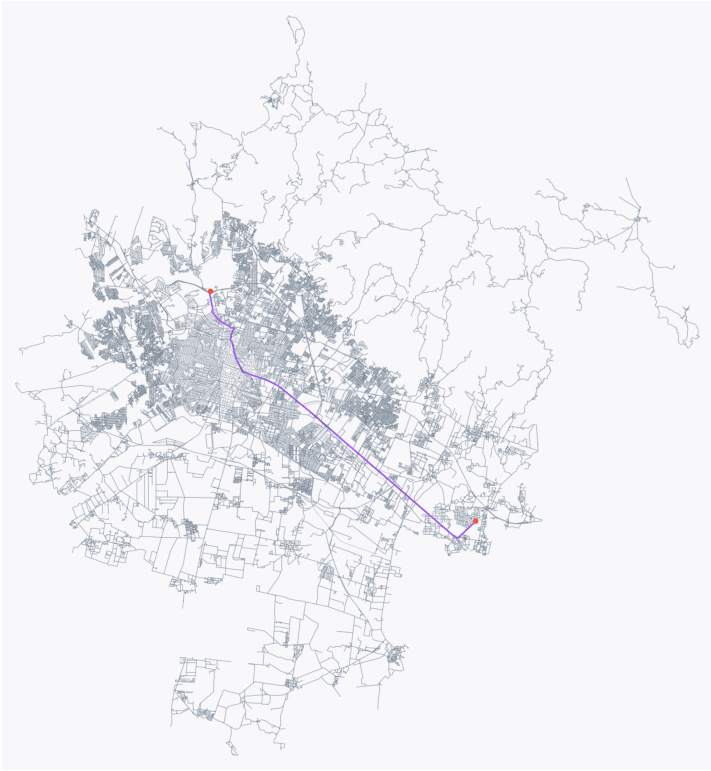


ferro@cimat.mx

Encontrando caminos en León

Caminos, Grafos y Tragos





Resultados de Dijkstra:

- Distancia: 28.32 km
- Velocidad promedio: 52.92 km/h
- Tiempo estimado: 32.11 min

Resultados de A*:

- Distancia: 25.84 km
- Velocidad promedio: 41.21 km/h
- Tiempo estimado: 37.62 min

ferro@cimat.mx

Aplicaciones 01

- El *pathfinding* se puede encontrar en muchas partes: videojuegos (*pathfinding* en NPCs o en un Snake), mapas (rutas óptimas), robots (planificación de rutas de desplazamiento)...
- Muchos problemas se pueden modelar con grafos: química (moléculas representadas como grafos), sociología (análisis de redes sociales)...
- ¡Hay un campo de IA que trabaja con información estructurada como grafos!

ferro@cimat.mx

Material de lectura

- Dijkstra's Algorithm Visualizer by Jan Smailbegovic: `https://www.davbyjan.com/`
- DSA Dijkstra's Algorithm by W3Schools: `https://www.w3schools.com/dsa/dsa_algo_graphs_dijkstra.php` (echar ojo a toda su sección de grafos)
- Introduction to the A* Algorithm from Red Blob Games: `https://www.redblobgames.com/pathfinding/a-star/introduction.html`
- Introduction to graphs and `tf.function`: `https://www.tensorflow.org/guide/intro_to_graphs`
- A Gentle Introduction to Graph Neural Networks, by Sanchez-Lengeling, et al.: `https://distill.pub/2021/gnn-intro/`

ferro@cimat.mx

¡GRACIAS!

¿Preguntas?

Continuemos el cotorreo:

`@rodo_ferro` & `ferro@cimat.mx`

Link a presentación:

`https://rodolfoferro.xyz/assets/talks/desvelando-grafos.pdf`

Libro recomendado:

Classic Computer Science Problems in Python

ferro@cimat.mx