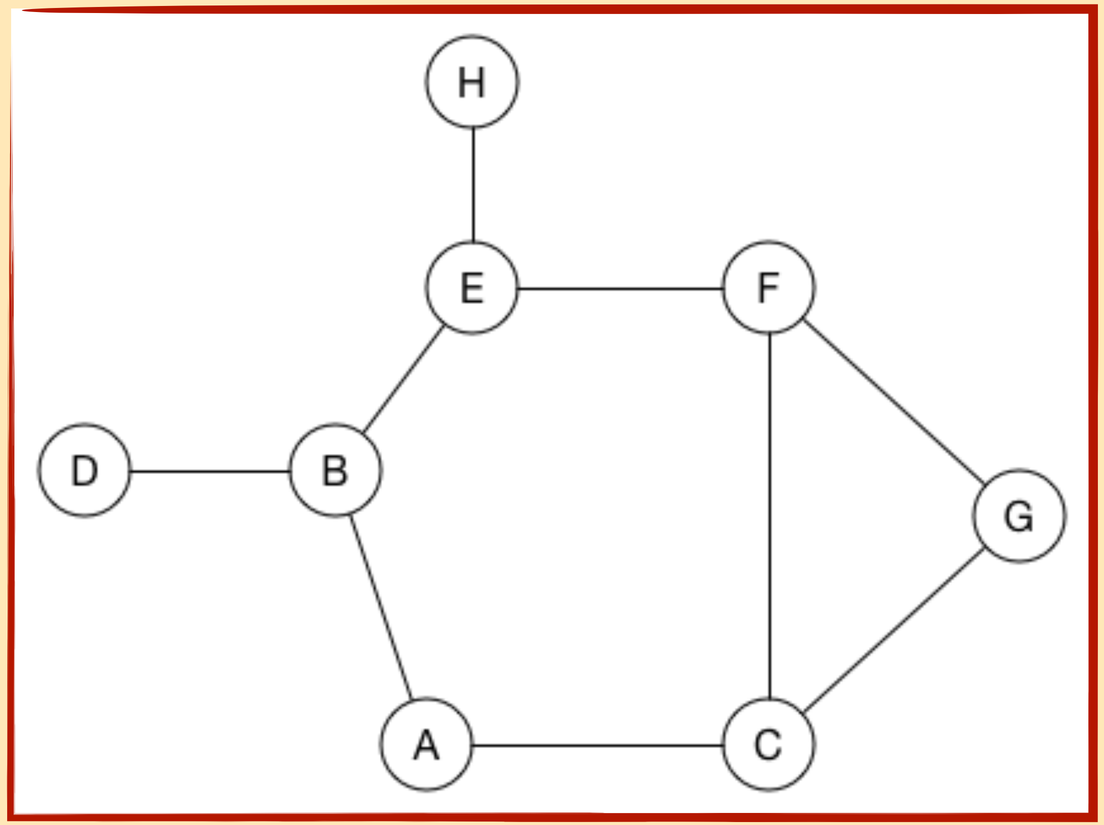
Graph Theory II

Seminar 13.

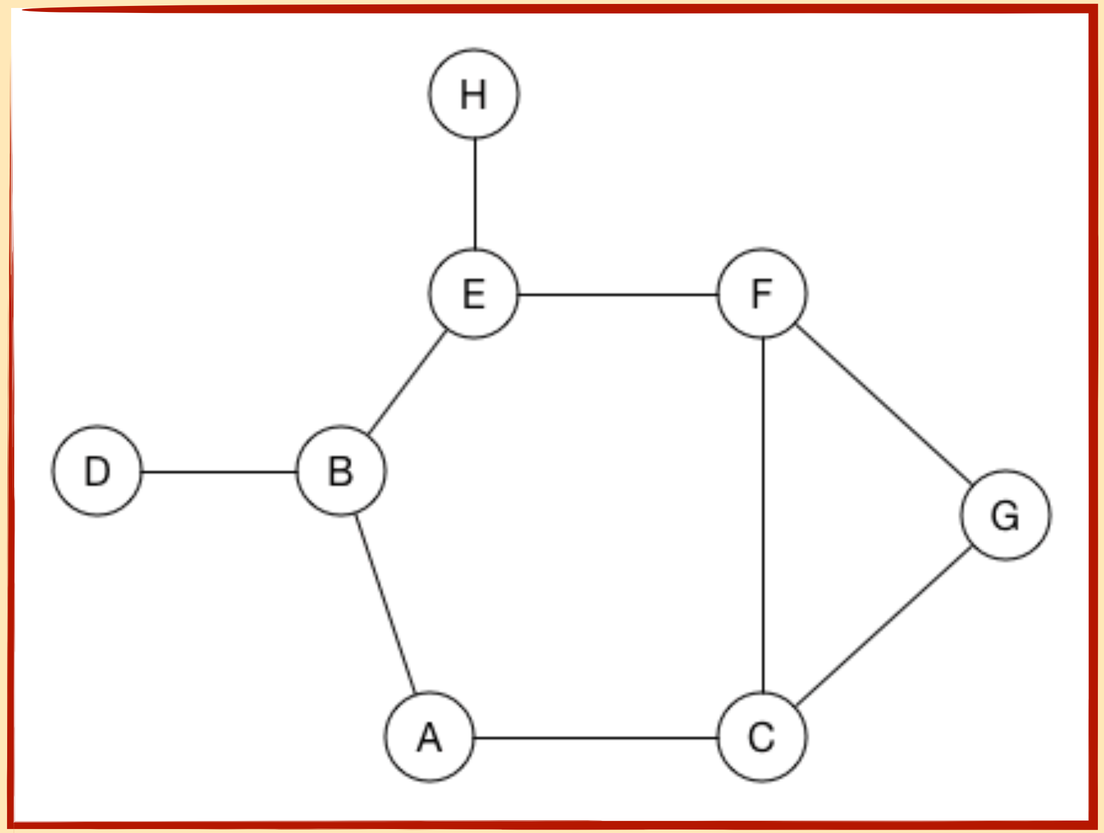
Graph Search algorithms

Breadth First Search



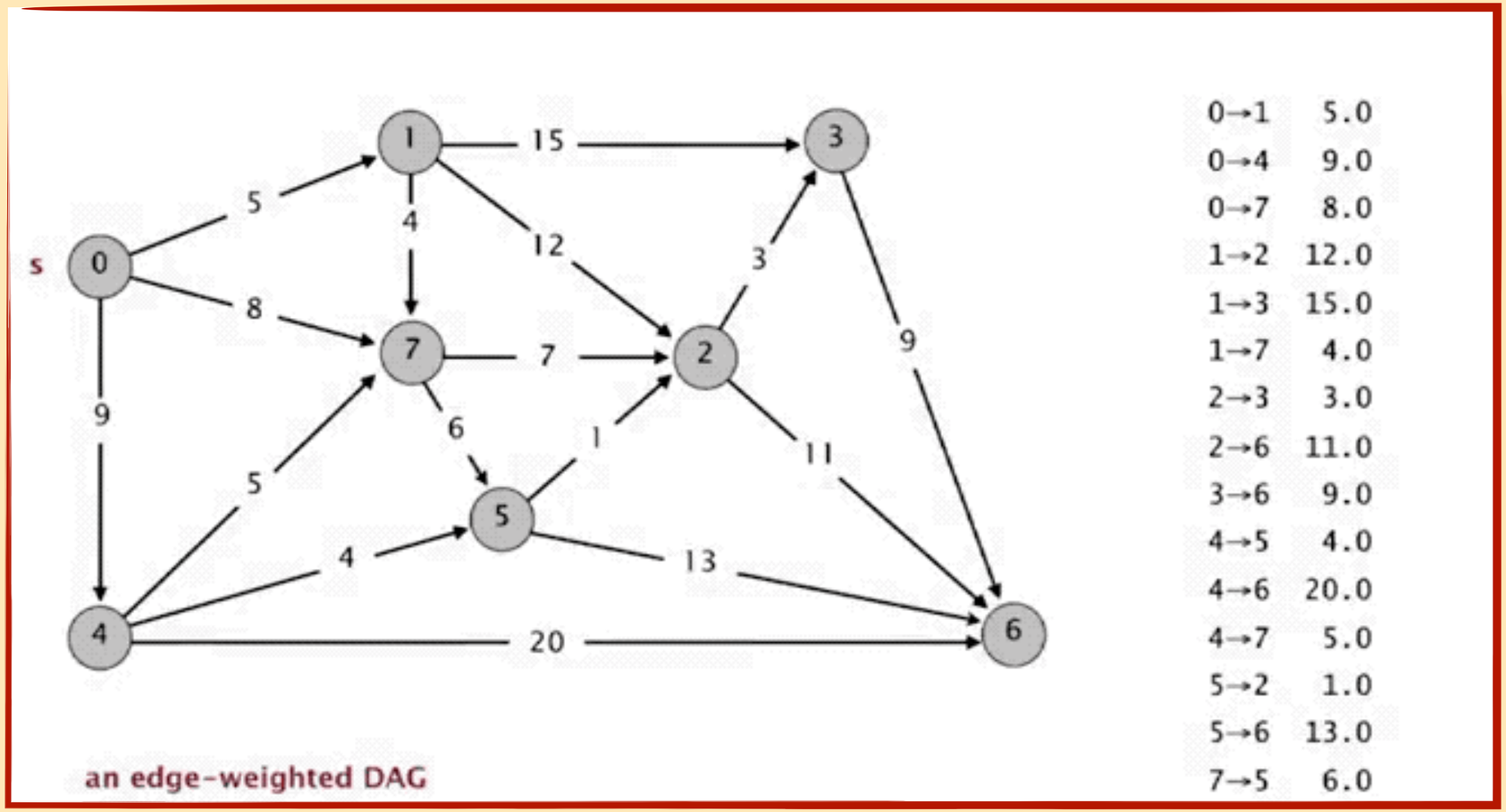
Graph Search algorithms

Depth First Search



Minimum distances

Dijkstra algorithm



Minimum distances

Matrix multiplication

- ***Task I:*** № of paths of fixed length
- ***Task II:*** All pairs shortest path

g – adjacency matrix

Minimum distances

Matrix multiplication

- ***Task I:*** № of paths of fixed length $\longrightarrow$ $d_{k+1}[i][j] = \sum_{p} d_k[i][p] \cdot g[p][j]$
- ***Task II:*** All pairs shortest path

g – adjacency matrix

Minimum distances

Matrix multiplication

- ***Task II:*** All pairs shortest path → ameliorate?

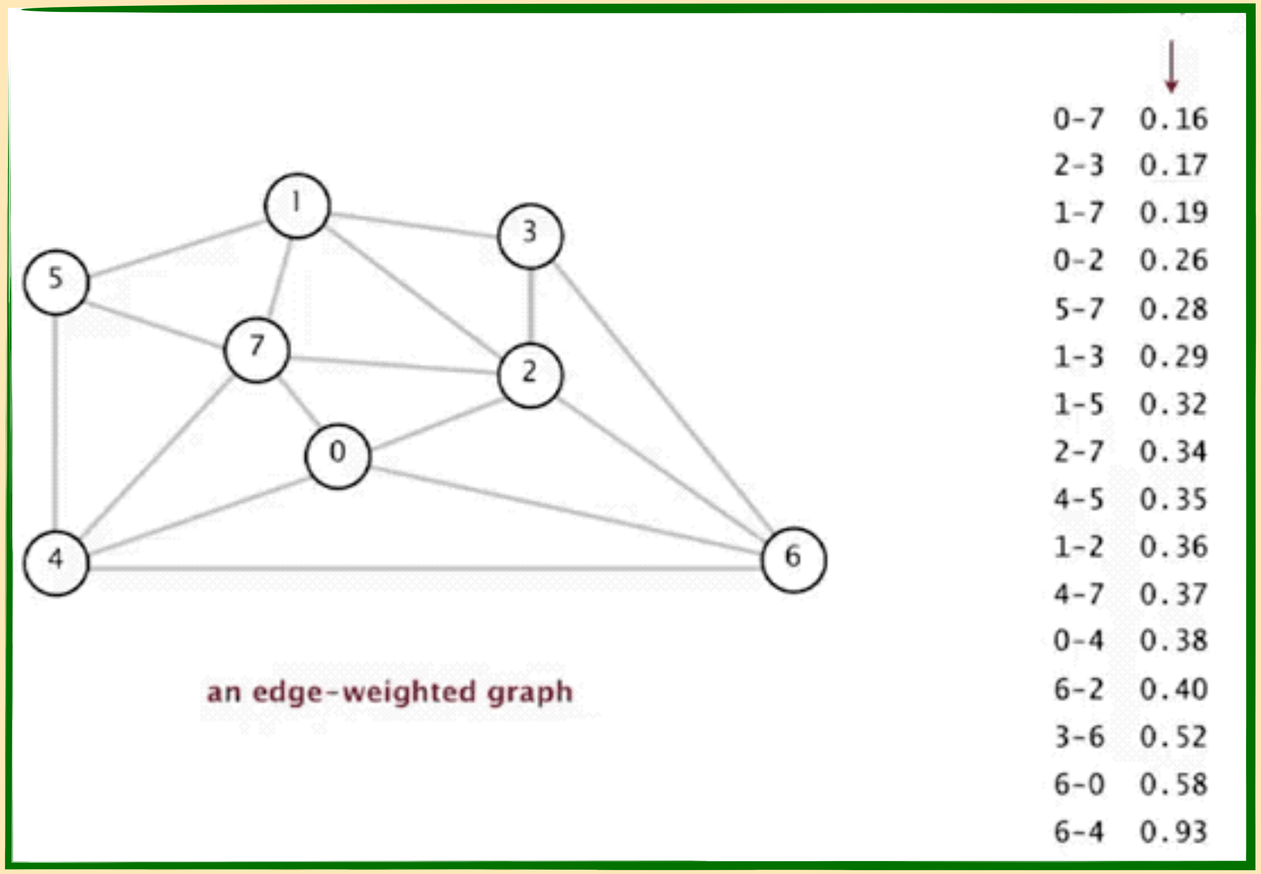
The algorithm

Consider a matrix D with $d_{ij} = c_{ij}$ if there exists an arc from i to j, ∞ otherwise. Let $d_{ii} = 0$ for all i. Then we define the "dot" operation as
$D^{(2)}_{ij} = (D \odot D^T)_{ij} = \min\{d_{i1} + d_{1j}, \ldots, d_{in} + d_{nj}\}$.

$D^{(2)}_{ij}$ represents the shortest path with 2 or fewer edges from i to j .
$D^{(3)}_{ij} = (D \odot D^{(2)}T)_{ij}$ = shortest path with number of edges ≤ 3 from i to j
$\vdots$
$D^{(n)}_{ij} = (D \odot D^{(n-1)T})_{ij}$ = shortest path with number of edges $\leq n$ from i to j

Minimum Spanning Tree

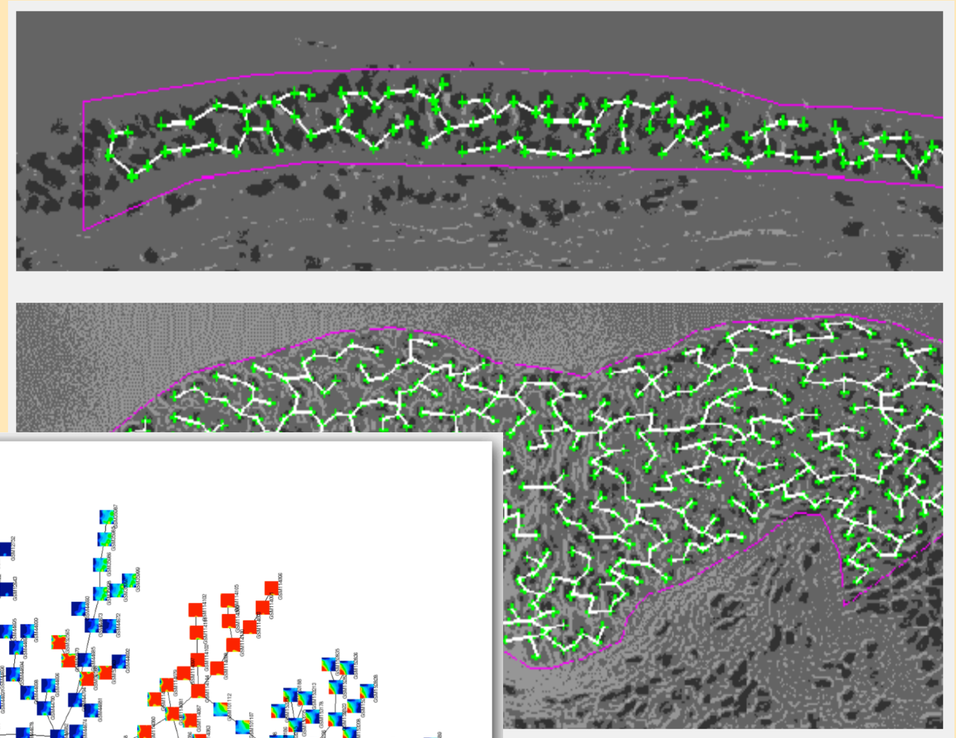
- ***Tree***: acyclic graph with *n* nodes and *n-1* edges with only one path existing between nodes
- ***Minimum spanning:*** tree with minimum overall weight

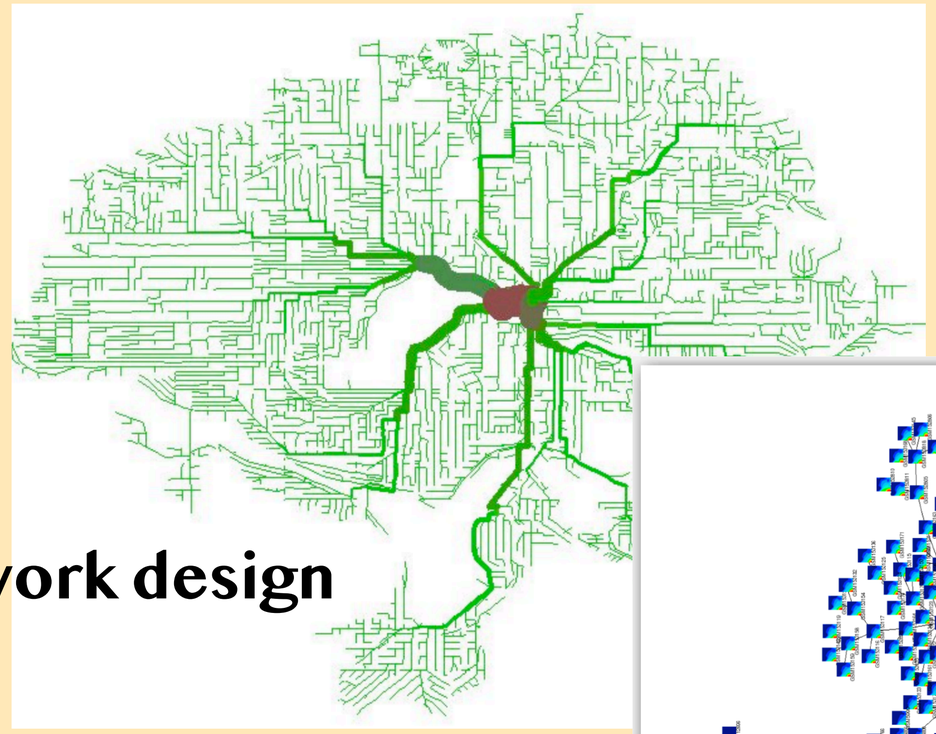


Minimum Spanning Tree

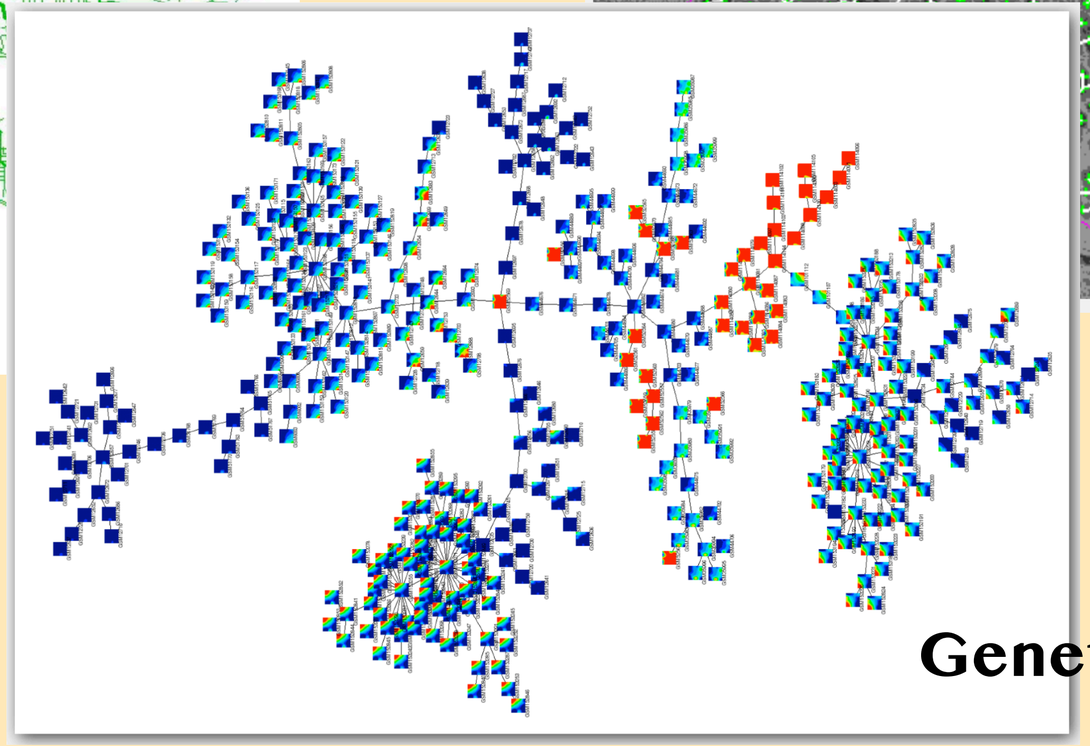
Applications

Medical images





Network design



Genetics

Minimum Spanning Tree

Prim's algorithm

- Choose source vertex
- Launch identical to Dijkstra
- Add the minimum edge to MST

minKey() – find minimum distance value unvisited neighbour

prim() – choose neighbour - update the following's distances and ancestors

Minimum Spanning Tree

Kruskal's algorithm

- Iterate through edges in non-decreasing order
- Check for safe edge => Add to the MST subgraph if TRUE

find() – identify the root index of particular node

union() – join two subsets

kruskal() – take edges - check for cycles - unite

Minimum Spanning Tree

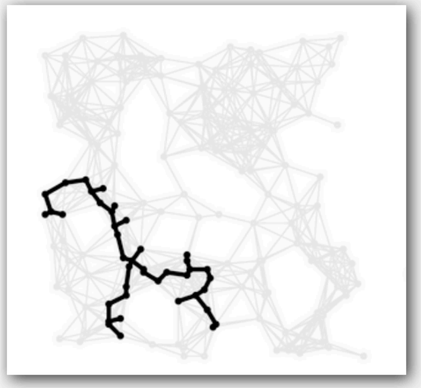
Algorithms' comparison

Prim's Algorithm	Kruskal's Algorithm
It starts to build the Minimum Spanning Tree from any vertex in the graph.	It starts to build the Minimum Spanning Tree from the vertex carrying minimum weight in the graph.
It traverses one node more than one time to get the minimum distance.	It traverses one node only once.
Prim's algorithm has a time complexity of $O(V^2)$, V being the number of vertices and can be improved up to O(E log V) using Fibonacci heaps.	Kruskal's algorithm's time complexity is O(E log V), V being the number of vertices.
Prim's algorithm gives connected component as well as it works only on connected graph.	Kruskal's algorithm can generate forest(disconnected components) at any instant as well as it can work on disconnected components

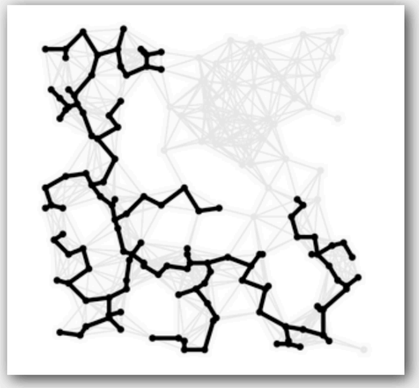
Minimum Spanning Tree

Algorithms' comparison

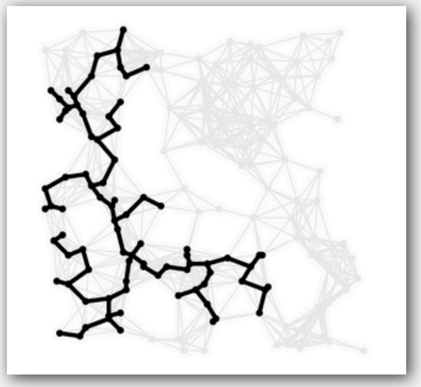
25%



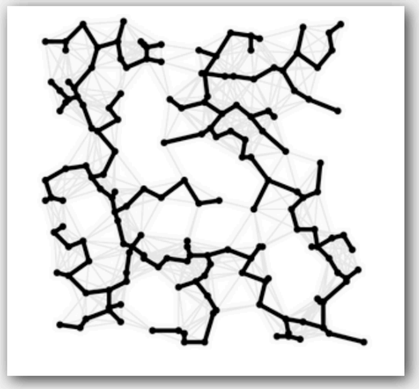
75%



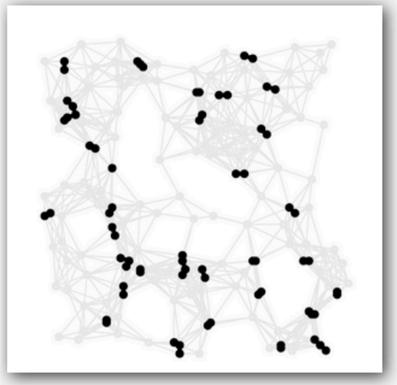
50%



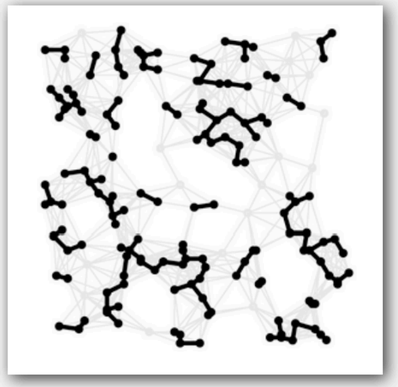
100%



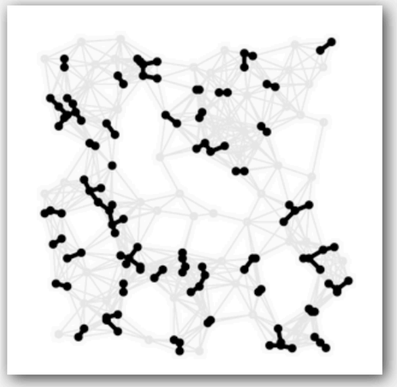
25%



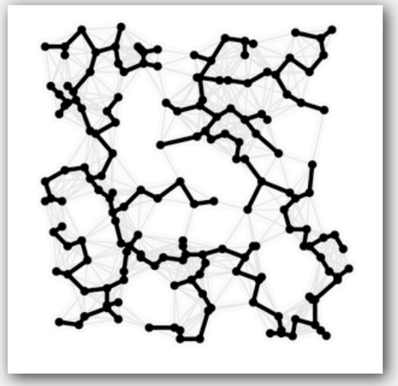
75%



50%

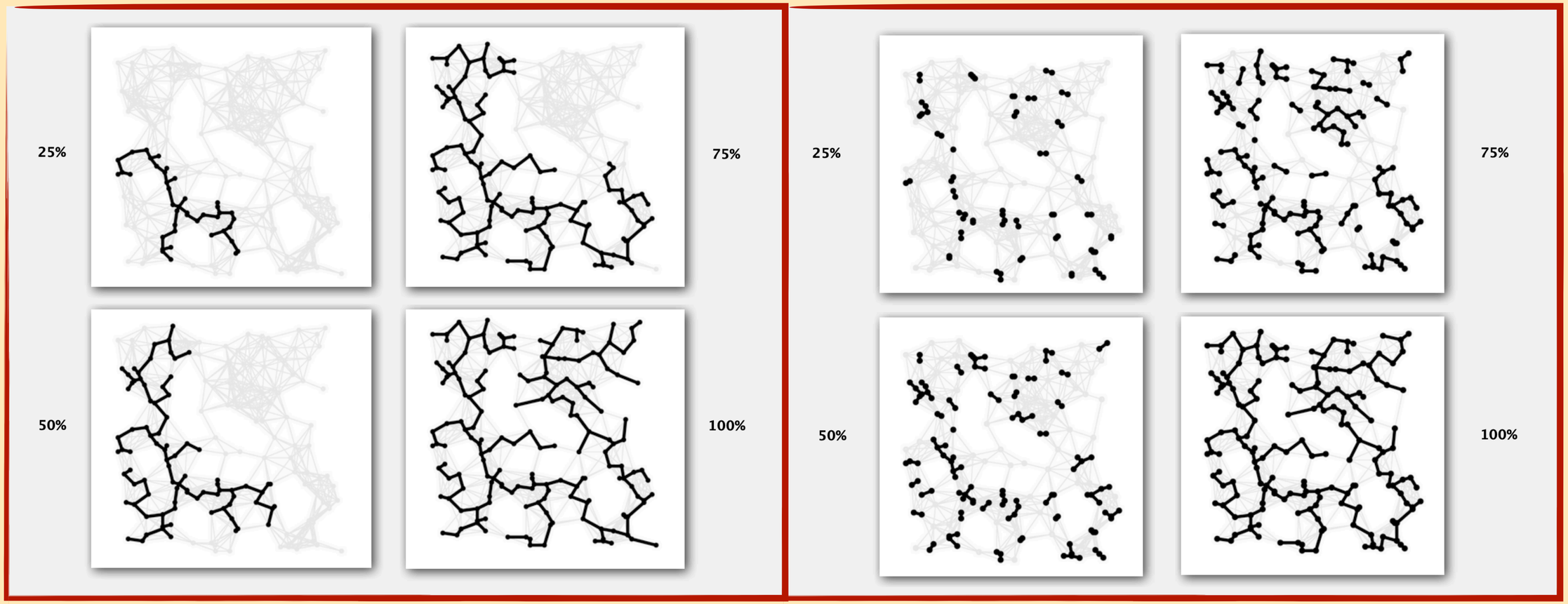


100%



Minimum Spanning Tree

Algorithms' comparison

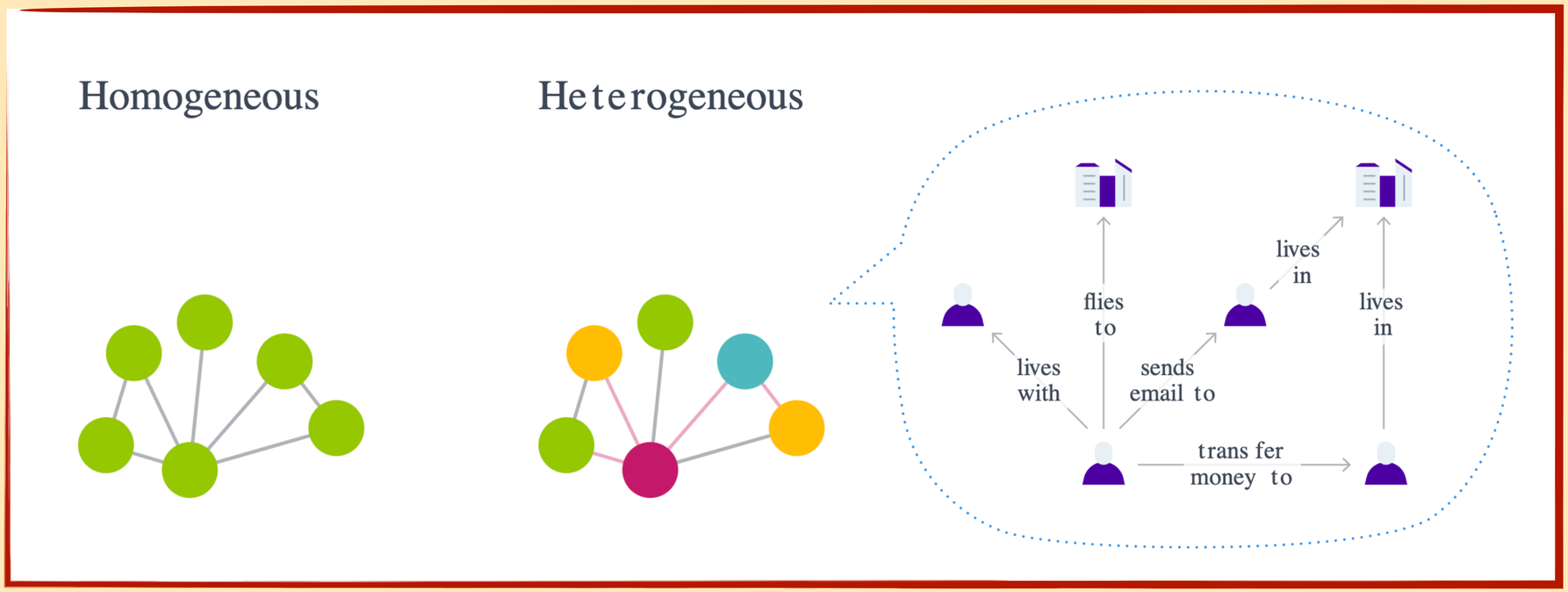


Prim

Kruskal

Graphs & ML

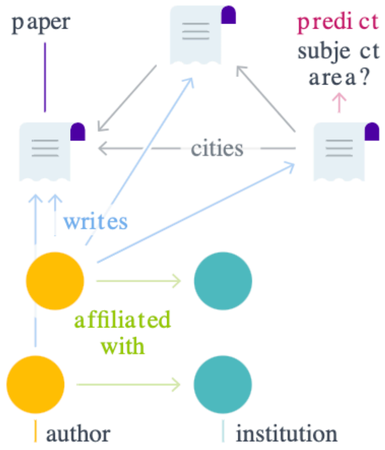
Types of graphs



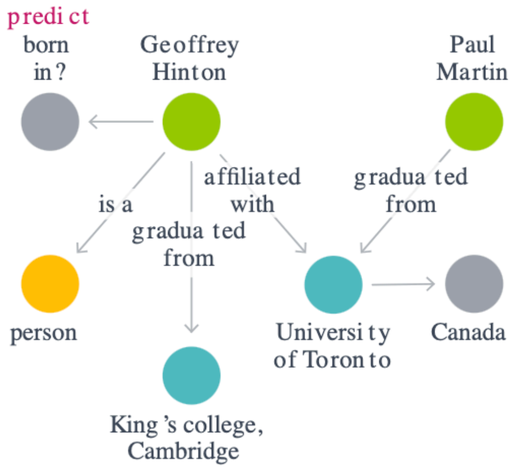
Graphs & ML

Types of tasks

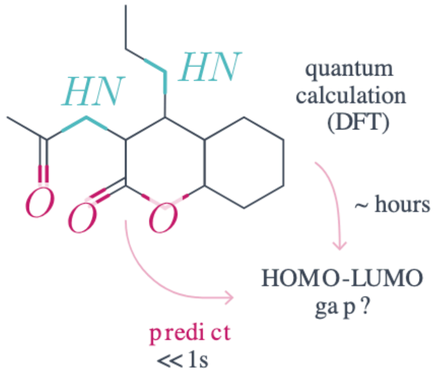
Node-level
MAG240M



Link-level
WikiKG90Mv2

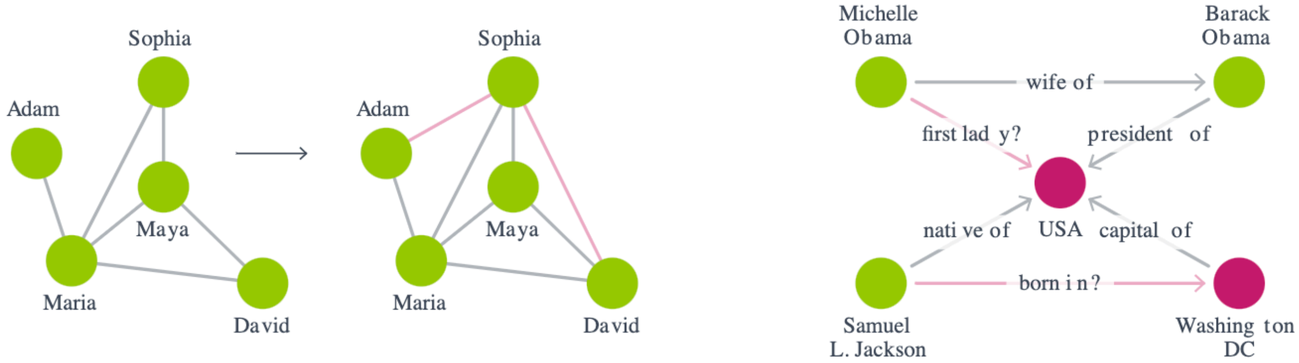


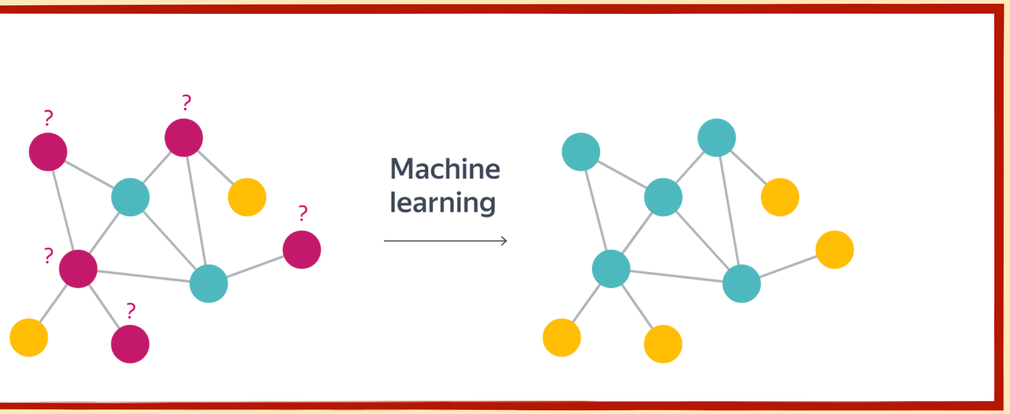
Graph-level
PCQM4Mv2

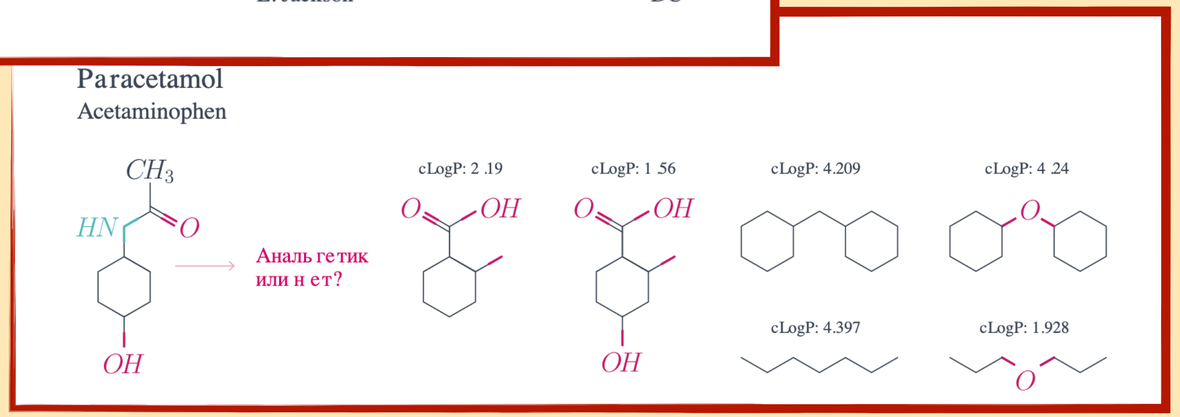


Graphs & ML

Types of tasks



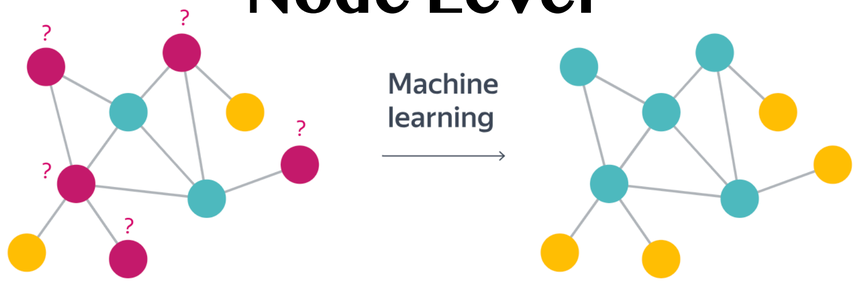




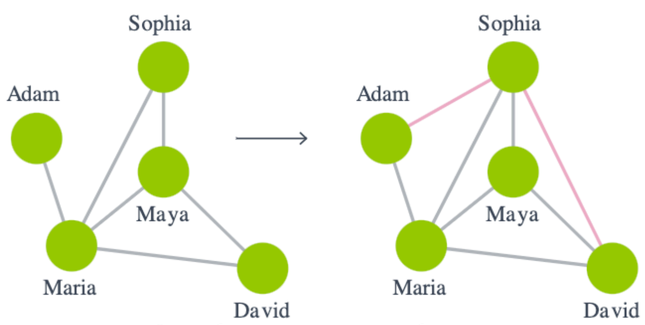
Graphs & ML

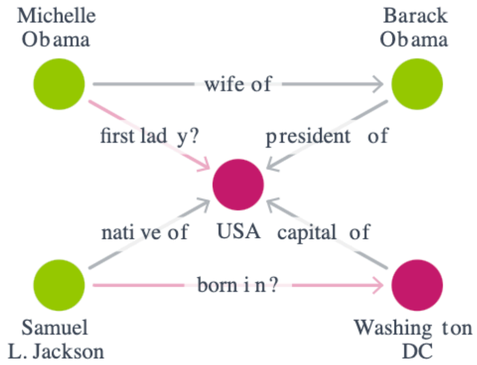
Types of tasks

Node Level

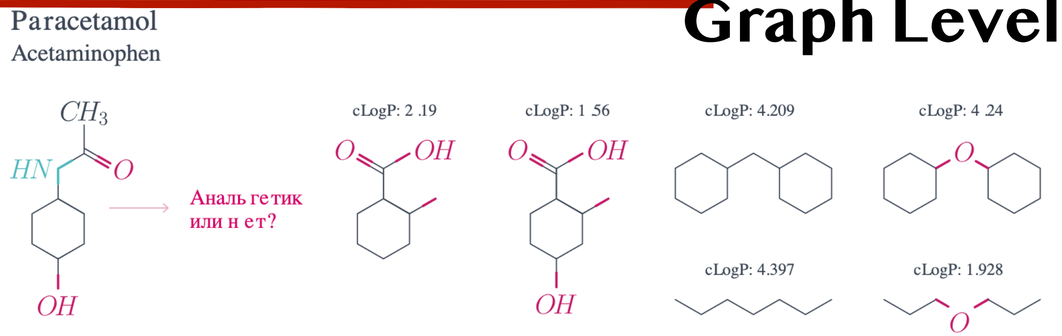


Link Level





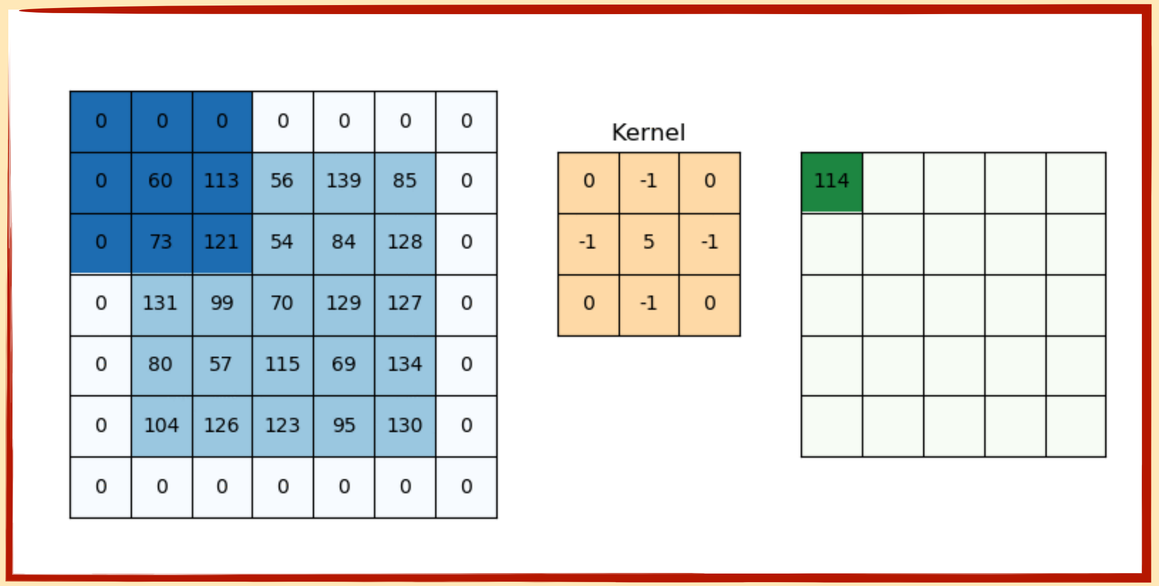
Graph Level



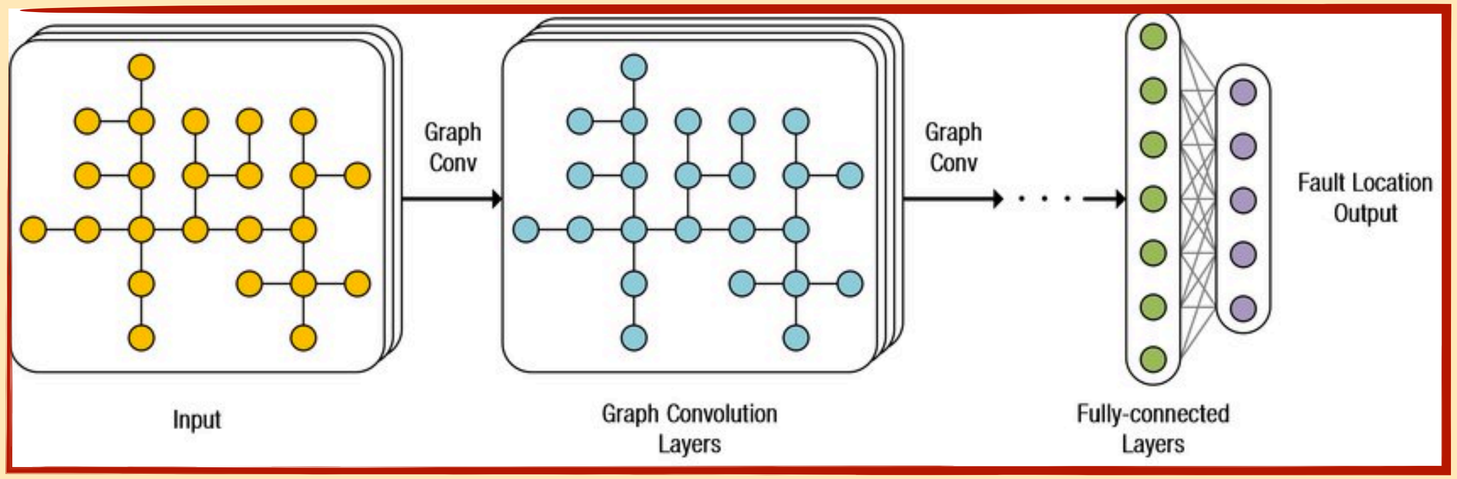
Graphs & ML

Neural Network: Arch

For images



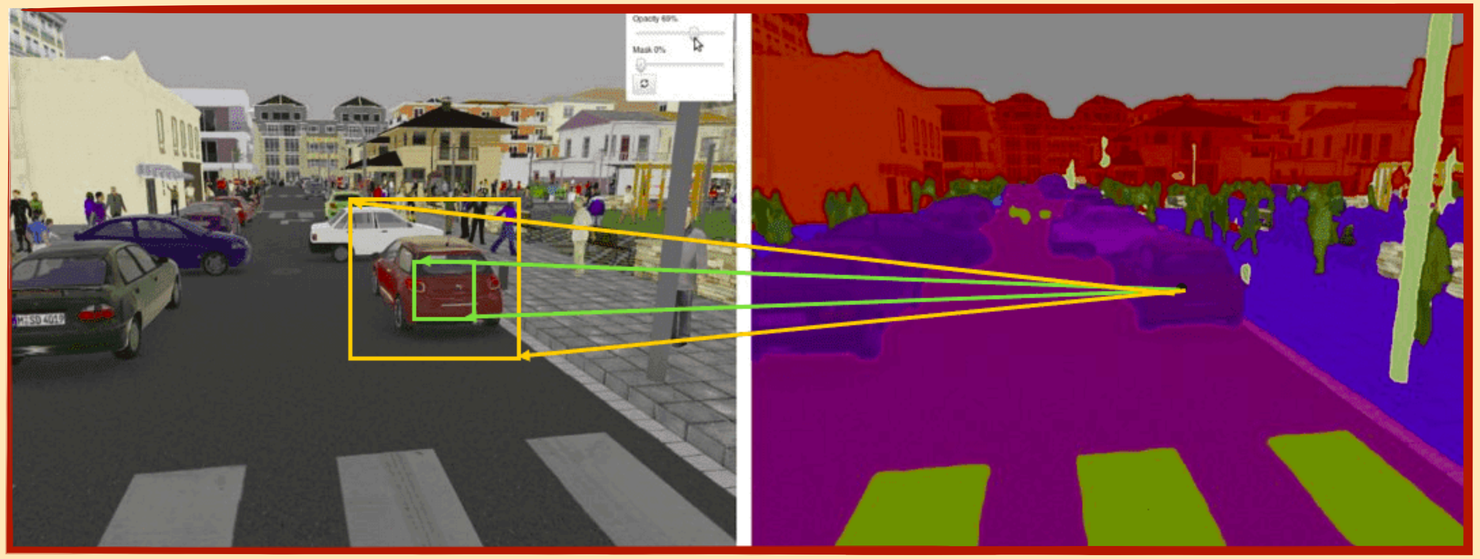
For graphs

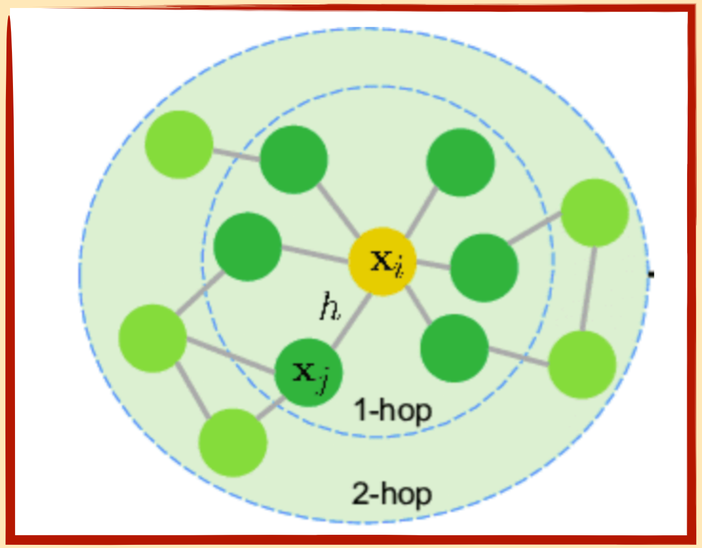


Graphs & ML

Neural Network: receptive field

For images





For graphs

Graphs & ML

Knowledge graphs & RL

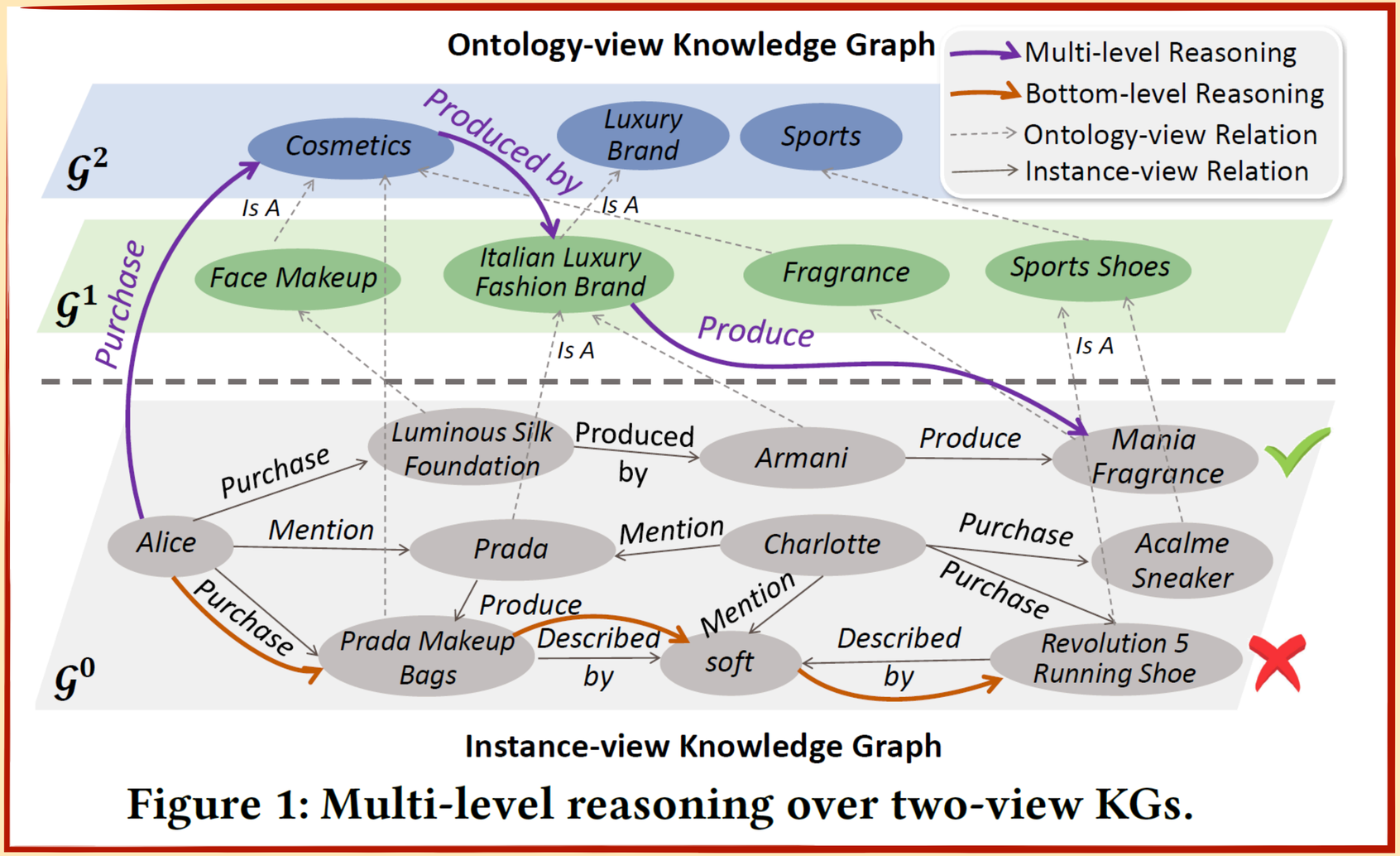


Figure 1: Multi-level reasoning over two-view KGs.