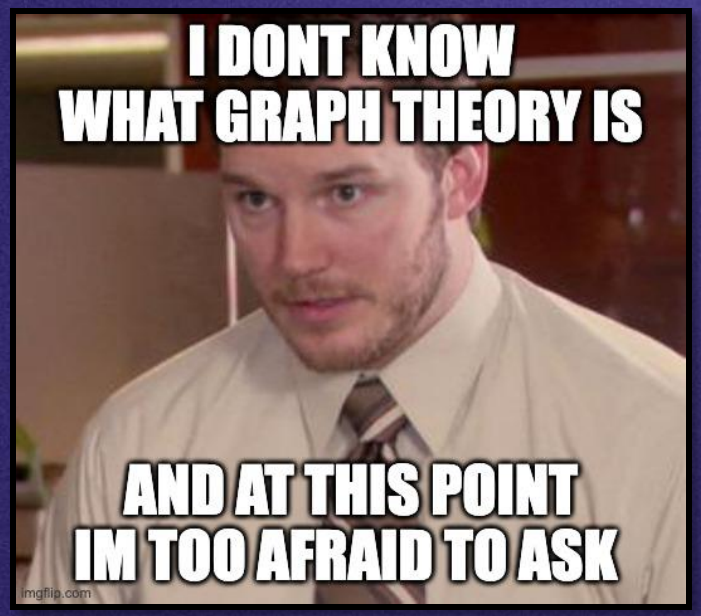


An introduction to graph theory, and its applications

Nicholas Gobern

Security Analyst | SpecterOps

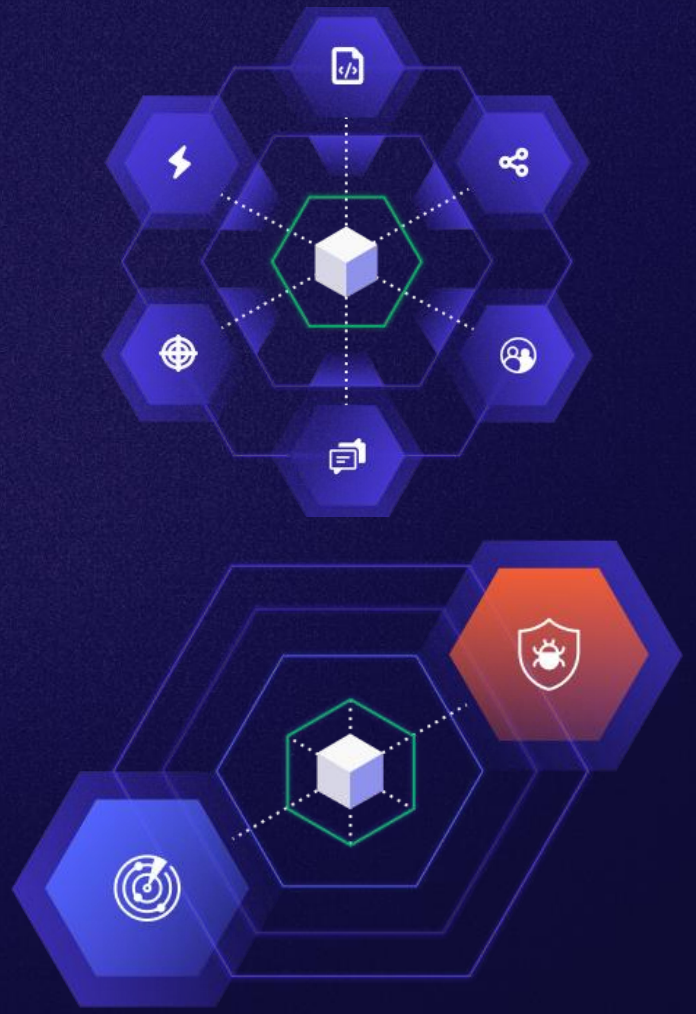
Nicholas Gobern

- Defensive Security Analyst
- Army DCO bubba
- OSCP, OSEP, OSWE
- Hampton University ‘17 Computer Science
- Photographer, DJ, Gamer, The reason the rum is always gone

SPECTEROPS
SO-CON
2025

Agenda

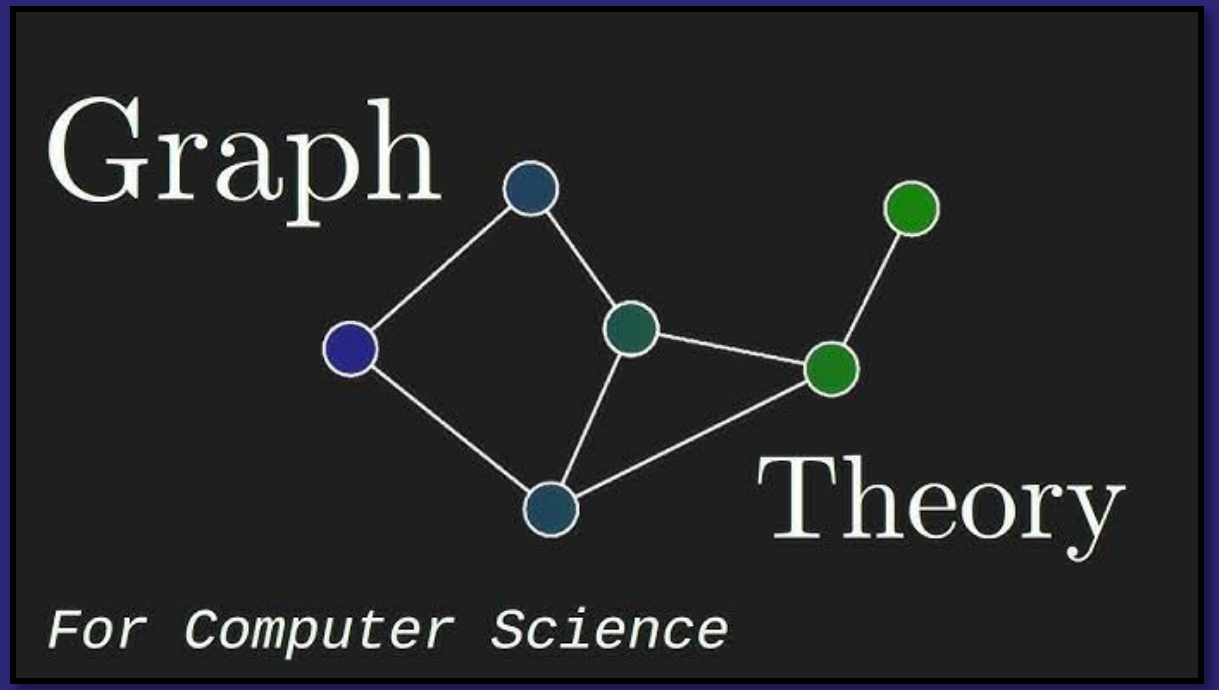
- What is Graph theory
- Graph theory “problem”
- Mathematical Applications
- Popular Algorithms
- Graph Theory Made Easy
- Graph Theory in Our World



What is graph theory

Graph Theory

The study of graphs, which
are mathematical structures used to model pairwise relations between objects. A graph in this context is made up of vertices (also called nodes or points) which are connected by edges (also called arcs, links or lines).



https://en.wikipedia.org/wiki/Graph_theory
(my favorite definition)

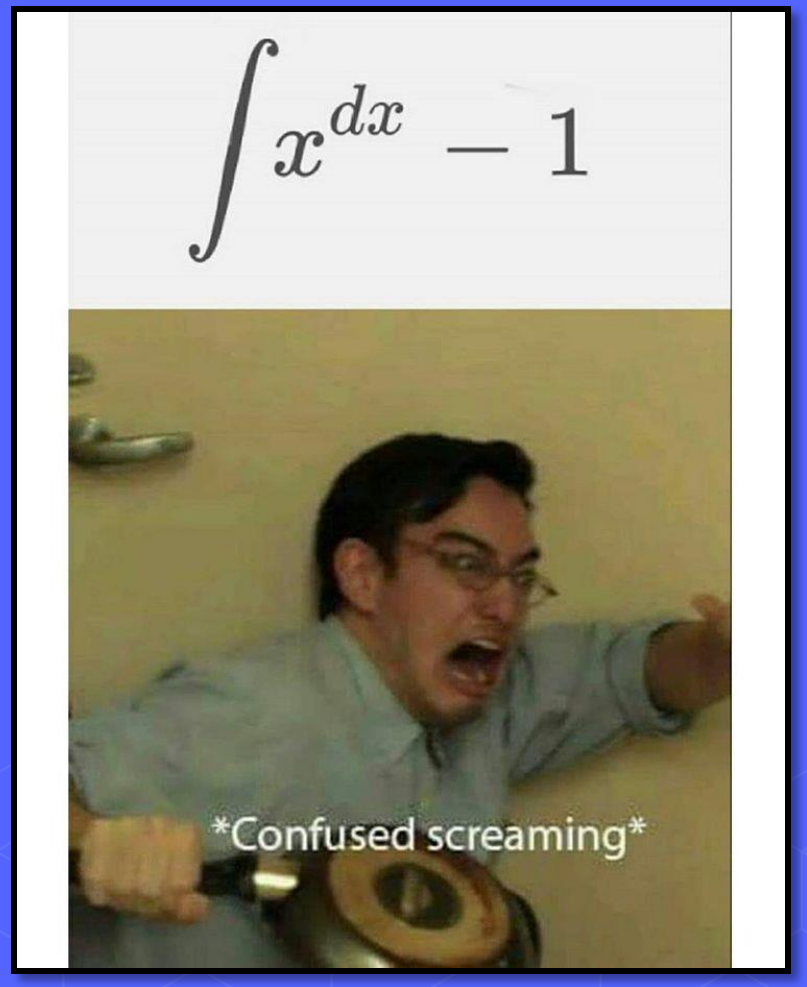
Why would I use a graph?

Uses for graphs

A way to visualize and analyze data, from business reports and weather forecasts to personal finance and social networks, aiding in understanding trends, making informed decisions, and communicating information effectively

This is represented by nodes as points, and edges as the connections. Nodes and edges can be translated to anything but can be thought of nouns and verbs.

Mathematical Applications



Definitions

Shared Terminology

Vertices or Nodes: Instance or unique value

Edges: The connection between two nodes

Graph: non-linear data structure consisting of vertices and edges

Tree: A connected Graph that has no cycles (linear)

Directed Graph: Graph where each edge has a one-way relationship

Weighted Graph: A graph where each edge is assigned a value

Definitions cont..

Breadth-first traversal: is when we visit the starting node and then on the first pass visit all the nodes directly connected to it. In the second pass, we visit the nodes that are two edges away from the starting node.

Depth-first traversal is when we visit the starting node and then proceed to follow links through the graph until we reach a dead end. When we reach a dead end, we back up along the path until we find an unvisited adjacent edge.

What does this look like

G - Graph
V - Vertices
E - Edge

G = (V,E) where V = { V1, V2, V3} & E= {(V1,V2), (V1,V3), (V2,V3) }

Graph Theory Questions

Seven Bridges of Königsberg

The city of Königsberg (now Kaliningrad, Russia) was situated on both sides of the Pregel River and included two islands connected to the mainland by seven bridges. It asked whether it was possible to walk through the city of Königsberg and cross each of its seven bridges exactly once.

Traveling Salesman

In this problem, a salesperson must visit several cities or towns. The goal is to determine a path to take between the cities such that you return to the starting city after visiting each city exactly once in the fastest way possible.

(Think of mail drivers)

Graphing Algorithms

algorithm

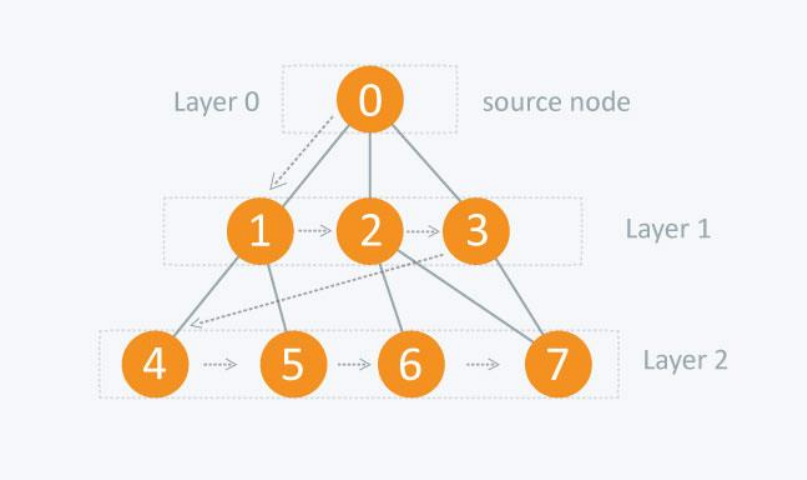
noun

Word used by programmers when they do not want to explain what they did.

THE BEST FUN SITE = 9GAG.COM

Breadth First Search

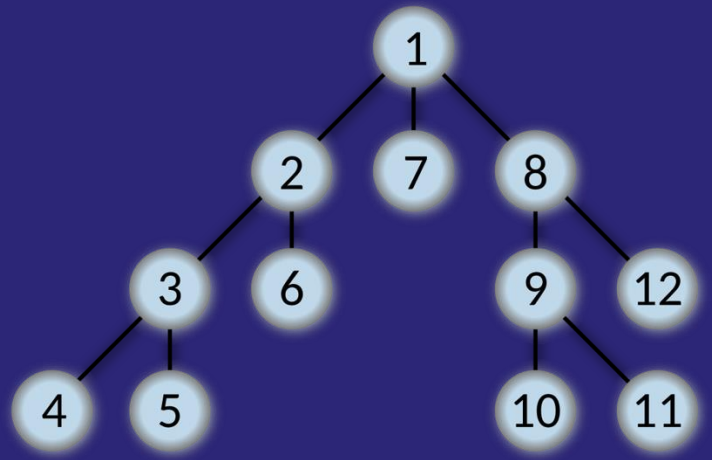
Starting from a selected node (source or starting node),traverse the graph one level at a time, Exploring the neighbor nodes (nodes which are directly connected to source node).




```
BFS (G, s)
//Where G is the graph and s is the source node
      let Q be queue.
      Q.enqueue( s )
//Inserting s in queue until all its neighbor vertices are
marked.

      mark s as visited.
      while ( Q is not empty)
//Removing that vertex from queue, whose neighbor will be
visited now
          v  =  Q.dequeue( )

          //processing all the neighbors of v
          for all neighbors w of v in Graph G
              if w is not visited
                          Q.enqueue( w )
//Stores w in Q to further visit its neighbor
                          mark w as visited.
```

Depth First Search

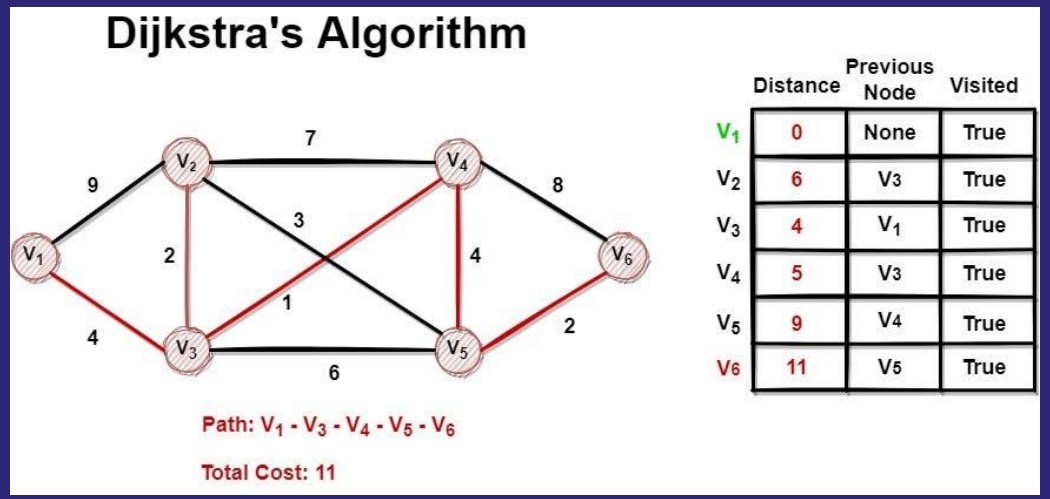
Explore as far as possible along each branch before backtracking, using a stack-like approach to manage the traversal




```
DFS(G,v)   //v is the vertex where the search starts
        Stack S := {};   //start with an empty stack
        for each vertex u, set visited[u] := false;
        push S, v;
        while (S is not empty) do
            u := pop S;
            if (not visited[u]) then
              visited[u] := true;
              for each unvisited neighbor w of u
                  push S, w;
            end if
        end while
```

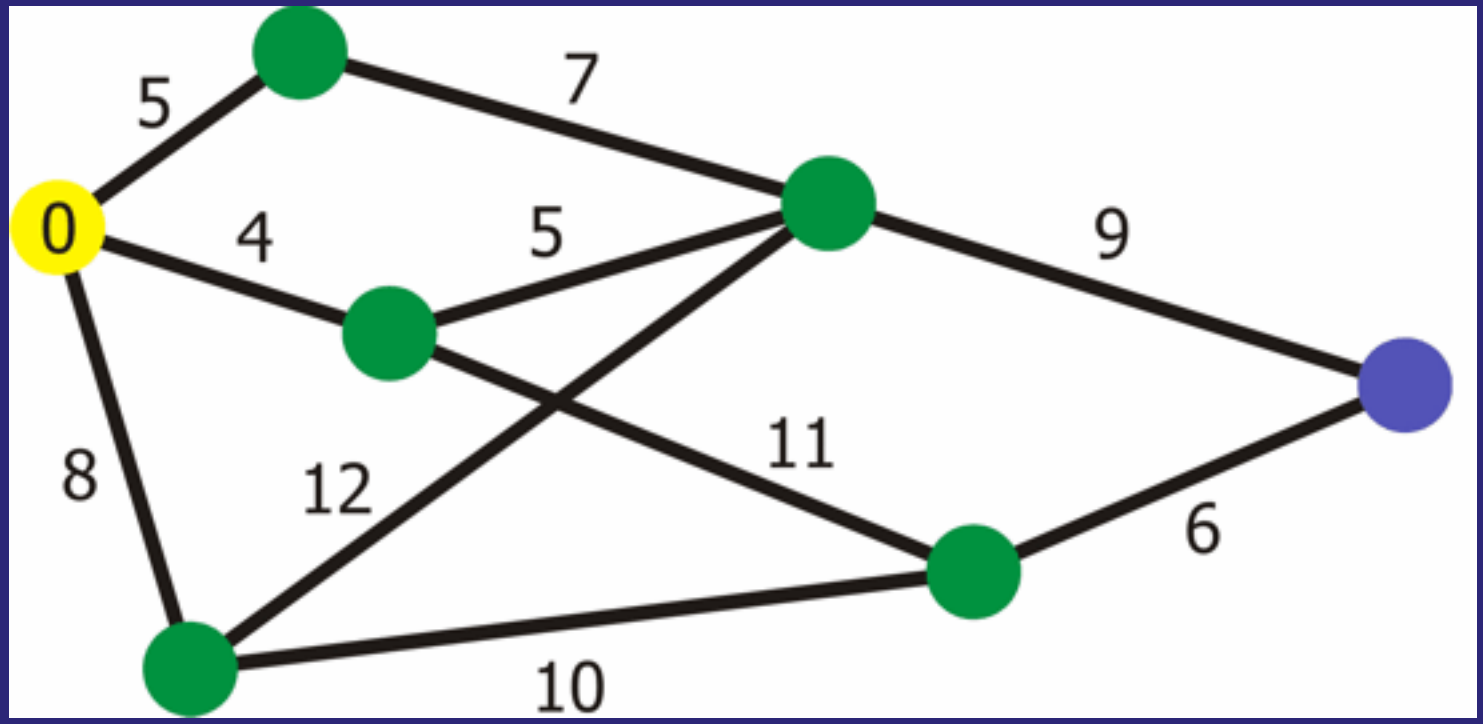
Dijkstras Shortest Path

Used to find the shortest path, having non-negative edge weight in the graphs, between two vertices on a graph.



	Distance	Previous Node	Visited
V1	0	None	True
V2	6	V3	True
V3	4	V1	True
V4	5	V3	True
V5	9	V4	True
V6	11	V5	True

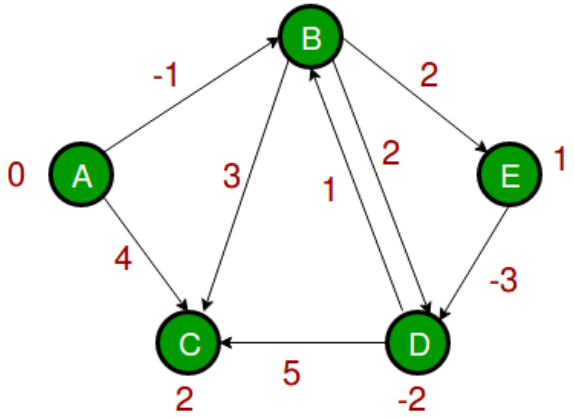
```
  Dijkstra(Graph, source):

      for each vertex v in Graph.Vertices:
          dist[v] ← INFINITY //infinity means dist is
unknown and acts as an arbitrary large num
          prev[v] ← UNDEFINED
          add v to Q
      dist[source] ← 0

      while Q is not empty:
          u ← vertex in Q with min dist[u]
          remove u from Q

        for each neighbor v of u still in Q:
            alt ← dist[u] + Graph.Edges(u, v)
              if alt < dist[v]:
                  dist[v] ← alt
                  prev[v] ← u

  return dist[], prev[]
```


5
7
0
4
5
9
11
8
12
6
10

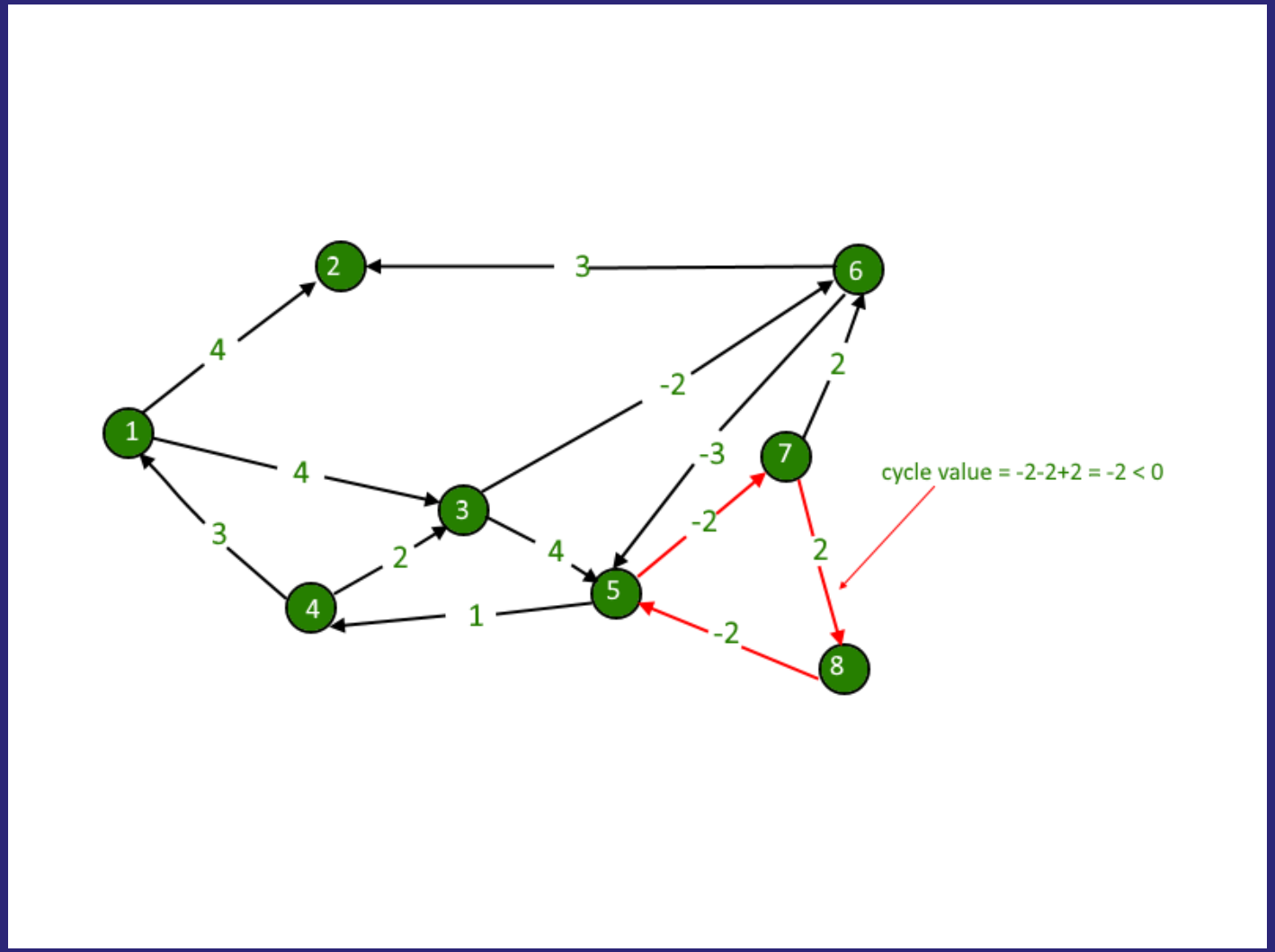
Bellman Ford Algorithm

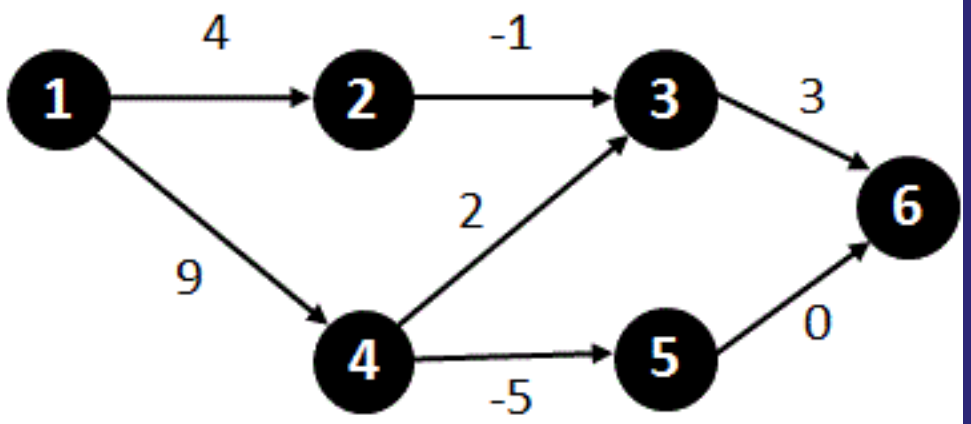
Dijkstra is not suitable when the graph consists of negative edges, so this algorithm takes that into account. The reason is, it doesn't revisit those nodes which have already been marked as visited. It also checks for a negative cycle, which is an instance where the distance becomes a negative value

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	∞	1
0	-1	2	1	1
0	-1	2	-2	1




```
bellmanFordAlgorithm(G, s) //G is the graph and s is the source
vertex
  for each vertex V in G
    dist[V] <- infinite // dist is distance
      prev[V] <- NULL // prev is previous
  dist[s] <- 0
  for each vertex V in G
    for each edge (u,v) in G
      temporaryDist <- dist[u] + edgeweight(u, v)
      if temporaryDist < dist[v]
        dist[v] <- temporaryDist
        prev[v] <- u
  for each edge (U,V) in G
    If dist[U] + edgeweight(U, V) < dist[V]
      Error: Negative Cycle Exist]
  return dist[], prev[]
e
```


1
2
3
4
5
6
7
8
4
3
4
-2
-3
2
4
3
2
-2
2
1
-2
cycle value = -2-2+2 = -2 < 0

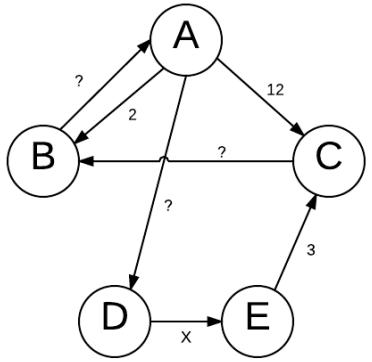


What is shortest path from 1 to 6 ?

IDeserve ©

Floyd-Warshall Algorithm

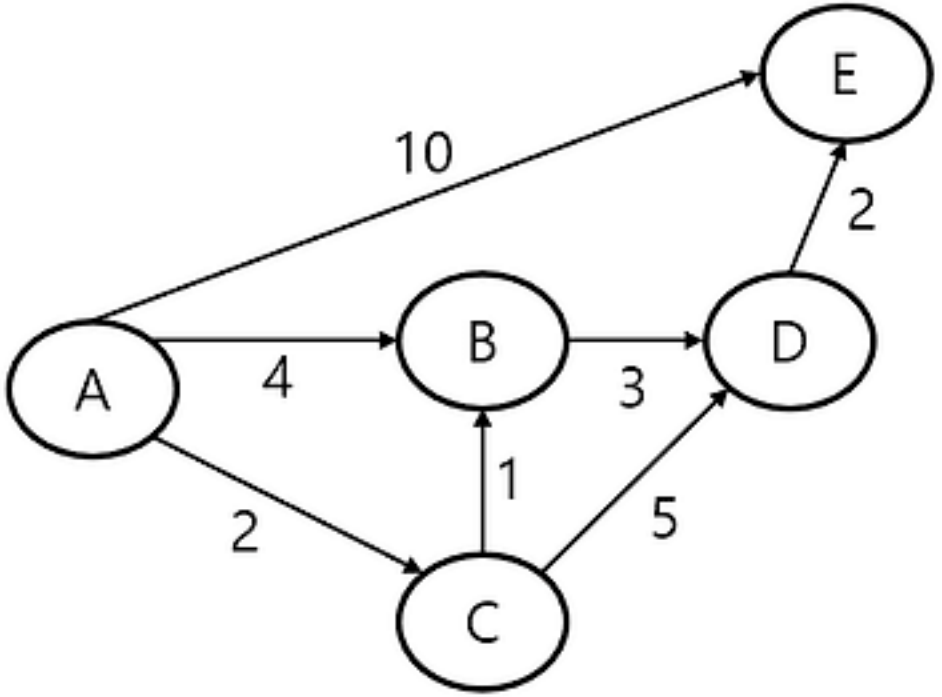
Floyd-Warshall is used to find the shortest path of all pairs. The idea is to iterate over all nodes, and for each node, iterate over all other nodes in a positive or negative weighted graph.



	A	B	C	D	E
A	0	2	10	5	7
B	3	0	13	8	10
C	7	4	0	12	14
D	12	9	5	0	X
E	10	7	3	15	0

```
Floyd-Warshall(w, n){ // w: weights, n: number of vertices
  for i = 1 to n do // initialize, D (0) = [wij]
      for j = 1 to n do{
        d[i, j] = w[i, j];
      }
      for k = 1 to n do // Compute D (k) from D (k-1)
        for i = 1 to n do
            for j = 1 to n do
              if (d[i, k] + d[k, j] < d[i, j]){
                  d[i, j] = d[i, k] + d[k, j];
              }
      return d[1..n, 1..n];
}
```

DP	A	B	C	D	E
A	0	4	2		10
B		0		3	
C		1	0	5	
D				0	2
E					0



Many more

- Johnson’s algorithm for All-pairs shortest paths
- Shortest Path in Directed Acyclic Graph
- Dial’s Algorithm
- Multistage Graph (Shortest Path)
- Shortest path in an unweighted graph
- Karp’s minimum mean (or average) weight cycle algorithm
- Etc…

Graph Theory made easy

Tools available

Programming

- Python Networkx
- C++ Boost Graph Library
- JavaScript D3.js
- Graph-Theoretic Programming language (type of FORTRAN)

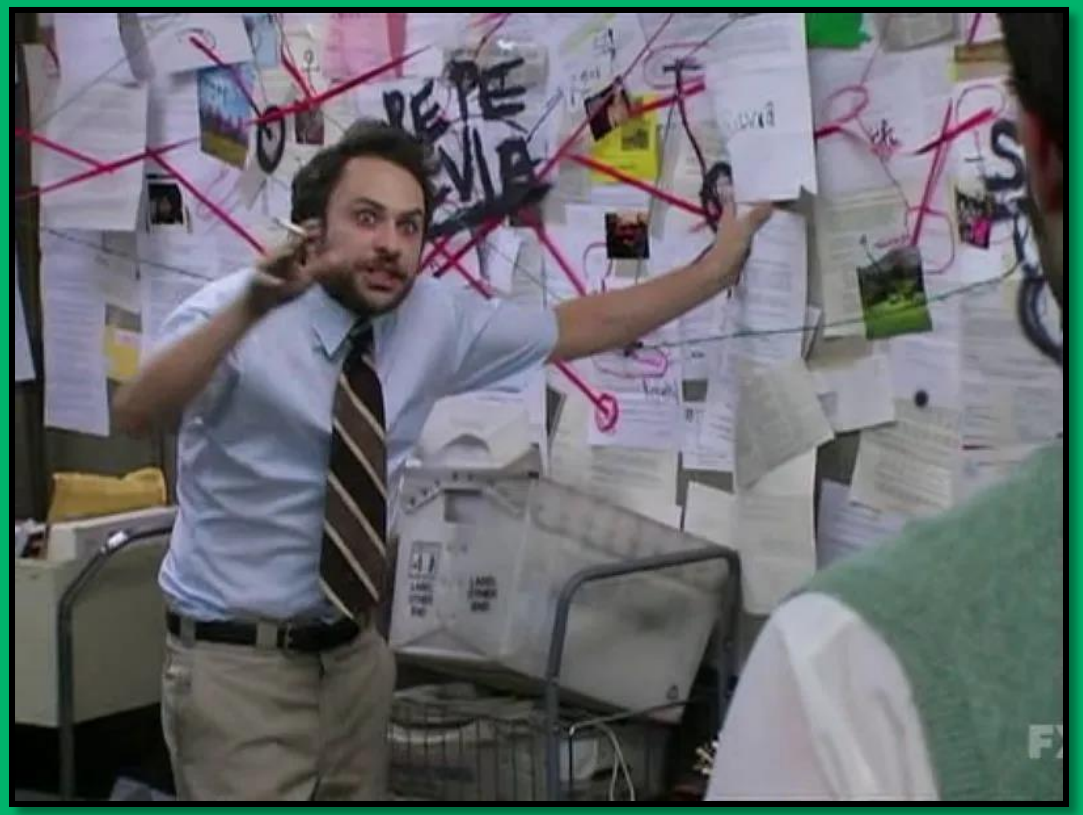
Databases

- Neo4j
- ArangoDB
- Cayley
- TerminusDB
- Nebula

Query Languages

- Cypher Query
- Gremlin
- SparQL
- Graph Query Language

How its Actually Used



I just like this meme

How its used in the world

Ontology: Nouns and verbs of a business to inform decisions

Formally defined as a “representation of knowledge that sets out the concepts and relationships within a particular domain”

Ontologies are typically represented using a graph model, where the nodes represent concepts or classes, and the edges represent relationships between them.

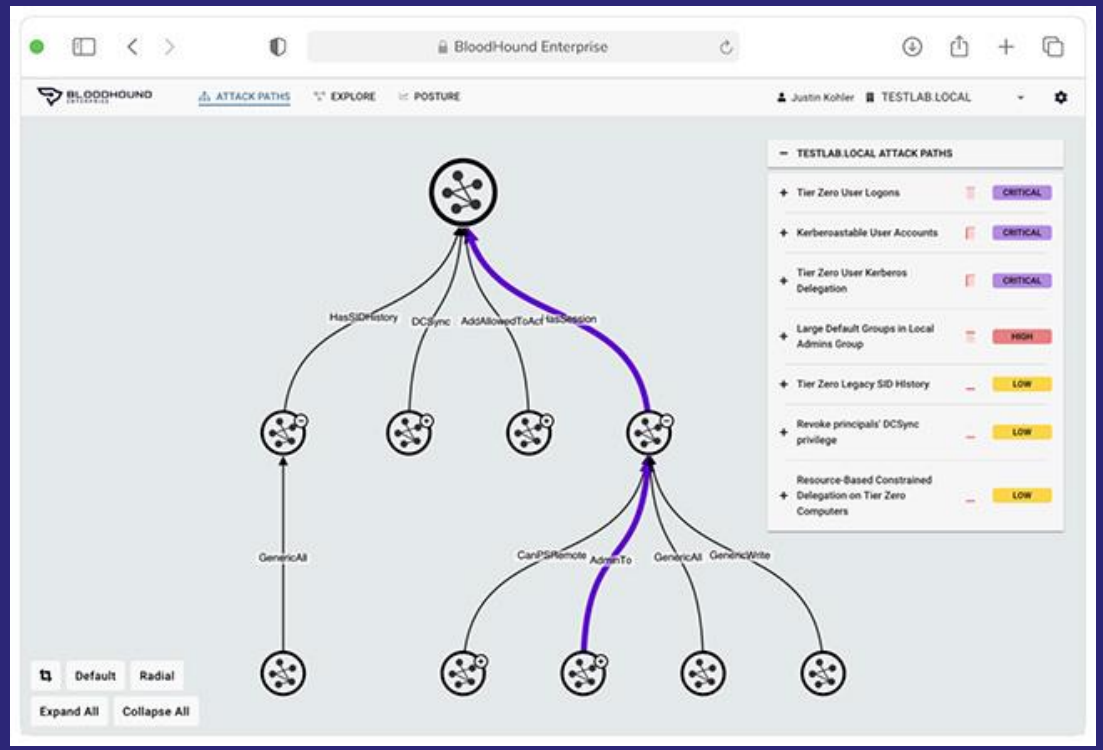
For example, in a medical ontology, the nodes might represent diseases, symptoms, and treatments, while the edges represent the relationships between them, such as "causes," "treats," or "diagnoses."

How its used in Cyber security

Attack Paths

Visualize the sequence of steps or vulnerabilities an attacker could exploit to gain access to a system or network, ultimately reaching a target

This can be used to calculate where you can accept risk within the organization

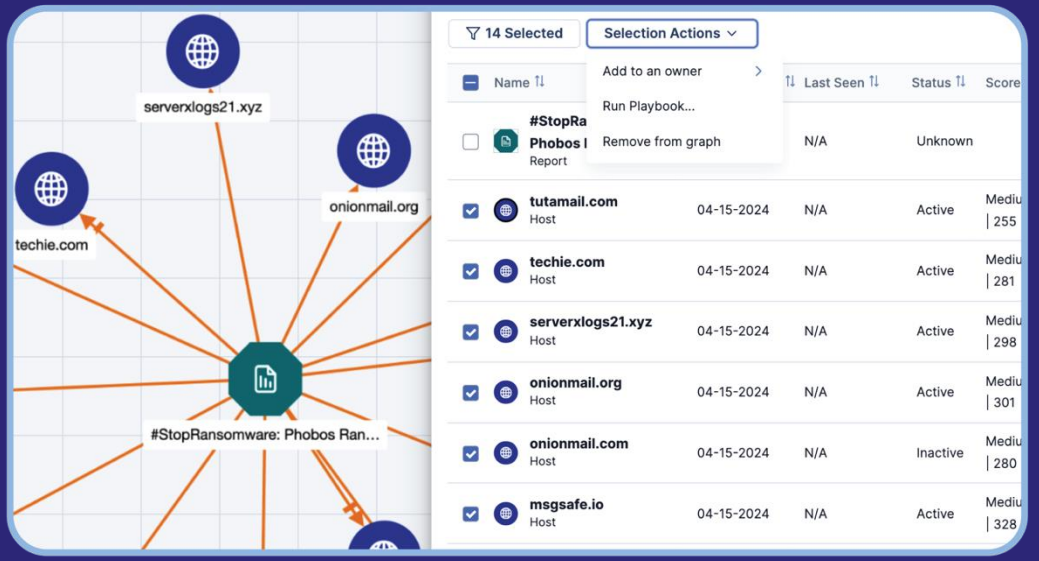


How its used in Cyber security

Threat Intelligence

Used to visualize vulnerabilities, assets, or other information that all stem from a single source

- APT actions
- Vulnerabilities by systems
- Malicious websites with their assets



How its used in Cyber security

Detecting Malicious Activity

Anomaly detection is the identification of data that diverges significantly from established norms

In Closing

Graphs are a fantastic way to assist in shrinking “complex” problems to “complicated” problems. It’s a way to condense data, identify relationships, and inject parameters to enhance a decision-making process.

SPECTEROPS

Thank you

Nicholas Gobern | ngobern@specterops.io