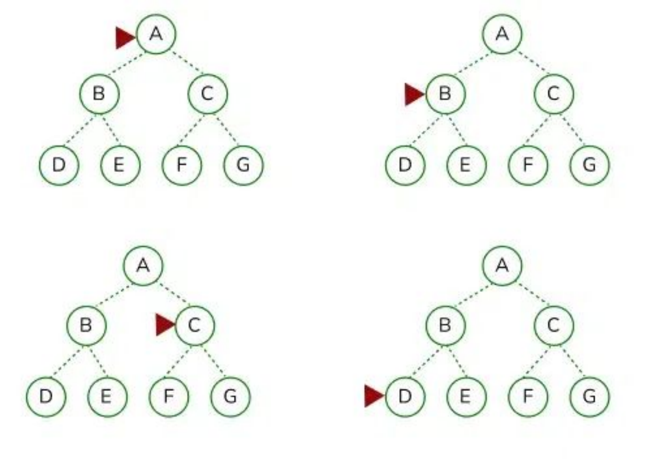
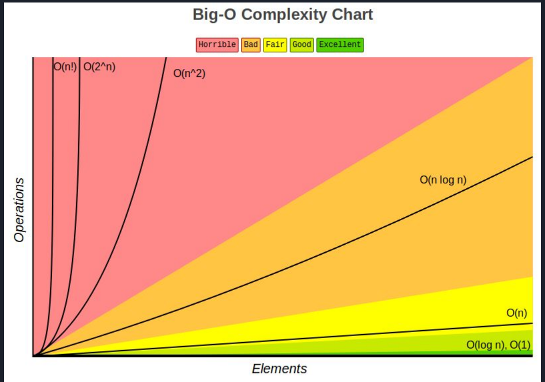
GRAPH-SEARCHING ALGORITHMS: BREADTH-FIRST SEARCH

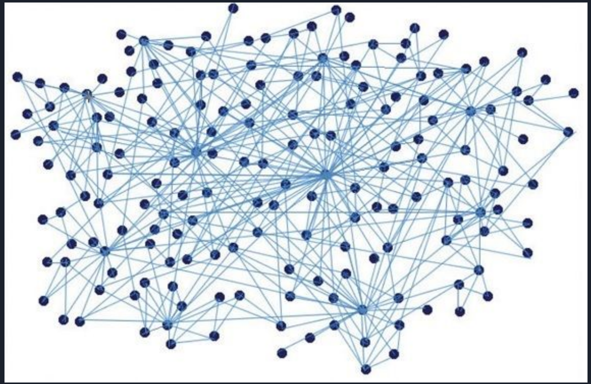


Leandro Jorge Fernández Vega

NOTATION AND RELEVANT VOCABULARY

- A graph is a set of vertices connected by edges.
- Mathematically, a graph is described as G=(V,E,f), where:
  - V is a set of vertices.
  - E is a set of edges.
  - f is an incidence function, indicating which two vertices are connected through a certain edge.
- Big-O notation, O(f(n)): For a size of problem n, Big-O notation indicates that, for the worst case, an algorithm will have a time complexity function of order f(n).

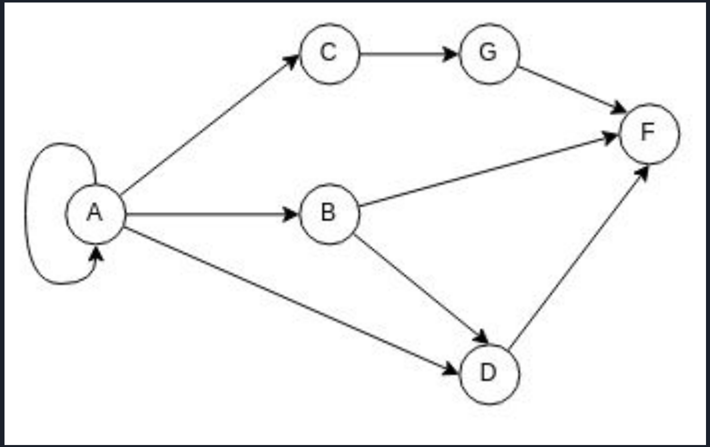




INTRODUCTION

We consider the problem of travelling from one node to another on a graph, passing through the least number of intermediate nodes, or equivalently, travelling through the least number of edges.

This problem is solved with the so-called Breadth-First Search Algorithm.



ALGORITHM

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

Where:

- u.color := color of current node node
- u.d := distance from origin to current node
- u.π := parent of current node
- Q := queue to store nodes
- Adj[u] := array containing all nodes connected to current node through edges

ALGORITHM
(PYTHON IMPLEMENTATION)

```
def bfs_shortest_path(G, start, goal):
        queue = deque([(start, [start])])
        visited = set()

        while queue:
            current, path = queue.popleft()

            if current == goal:
                return path

            if current not in visited:
                visited.add(current)

                for neighbor in G.neighbors(current):
                    if neighbor not in visited:
                        queue.append((neighbor, path + [neighbor]))

        return None  # No path found
```

PROPERTIES

- Completeness: BFS finds a solution if it exists and the ramification factor for every node is finite.
- Correctness: The optimal solution regarding number of actions is always reached, as the following theorem states:

Theorem: Let $G=(V,E)$ be a directed or undirected graph, and suppose that BFS is run on G from a given source vertex $s \in V$. Then, during its execution, BFS discovers every vertex $v \in V$ that is reachable from the source s, and upon termination, $v.d=\delta(s,v)$ for all $v \in V$, where $\delta(s,v)$ is the shortest distance from s to v. Moreover, for any vertex $v \neq s$ that is reachable from s, one of the shortest paths from s to v is a shortest path from s to $v.\pi$ followed by the edge $(v.\pi,v)$.

- Efficiency: $O(V)$ memory-wise and $O(V+E)$ time-wise.
- Others: The algorithm is used for problems where weights in the edges are irrelevant or every edge has the same weight.

BREADTH-FIRST TREES

For a graph $G=(V,E)$ with source s, we define the predecessor subgraph of G as

$G_\pi = (V_\pi, E_\pi)$, where

$V_\pi = \{v \in V : v.\pi \neq \text{NIL}\}$ UNION $\{s\}$

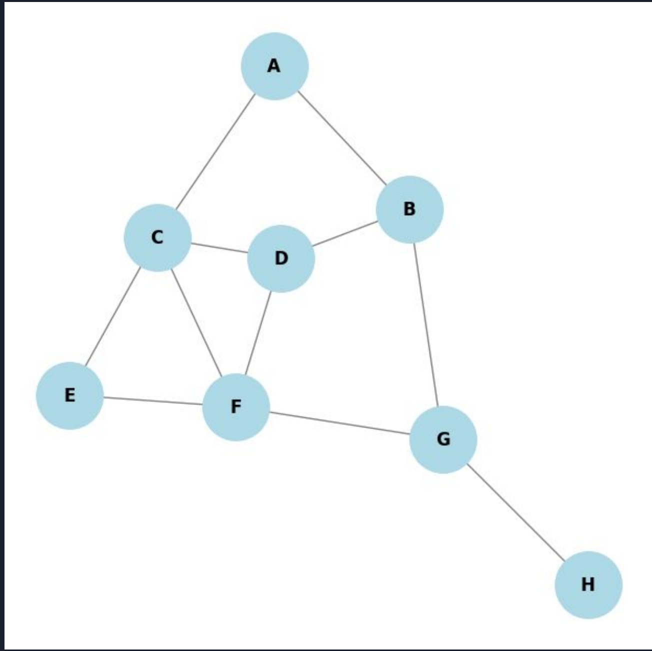
$E_\pi = \{(v.\pi, v) : v \in V_\pi - \{s\}\}$

The predecessor subgraph G_π is a breadth-first tree if V_π consists of the vertices reachable from s and, for all $v \in V_\pi$, the subgraph G_π contains a unique simple path from s to v, that is also a shortest path from s to v in G.

Lemma: When applied to a directed or undirected graph $G=(V,E)$, procedure BFS constructs π so that the predecessor subgraph $G_\pi = (V_\pi, E_\pi)$ is a breadth-first tree.

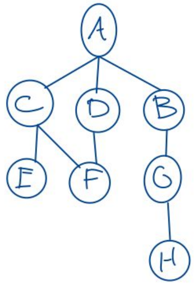
EXAMPLE

Find the shortest path between nodes A and H using BFS Algorithm and draw the Breadth-First Tree.



EXAMPLE (SOLUTION)

Find the shortest path between nodes A and H using BFS Algorithm and draw the Breadth-First Tree.

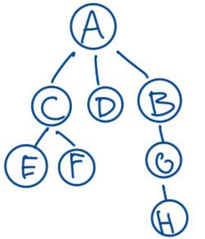


Node	Frontier	Visited
A	~~A~~	A
C	C-A	C
D	D-A	D
B	B-A	B
E	E-C-A	E
F	F-C-A	F
	F-D-A	
G	G-B-A	G
H	H-G-B-A	H

Path: H-G-B-A

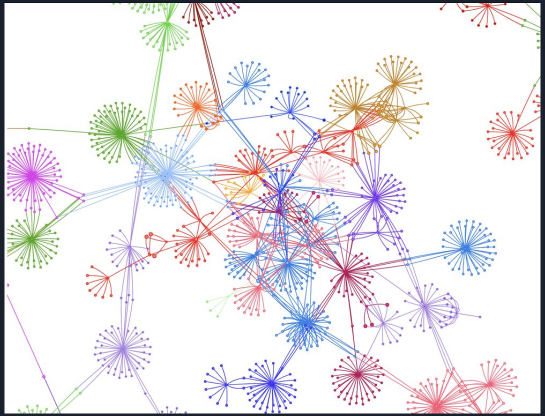
$$V_\pi = \{v \in V / \exists\, v.\pi\} \cup \{A\} = \{A, B, C, D, E, F, G, H\} = V$$

$$E_\pi = \{(v.\pi, v) / v \in V_\pi \setminus \{A\}\}$$



NETWORKX
(SEE NOTEBOOK)

NetworkX
Network Analysis in Python



THE END


APPROVED
APPROVED
APPROVED
Adobe Stock | #299939713


THANK YOU
Adobe Stock | #543408719