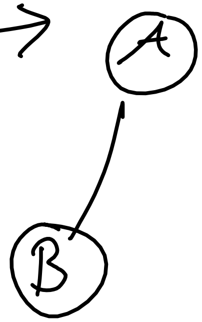
Graph: pair of two sets $V$ and $E$

vertices edges

vertex




A and B are adjacent

pair of vertices

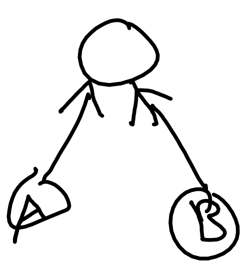
$\langle A, B \rangle \simeq \langle B, A \rangle$

defn. $\longrightarrow$ directed / undirected

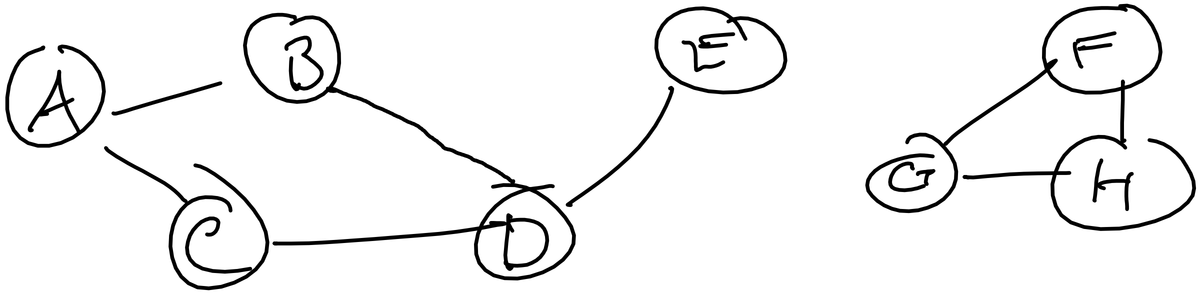
weighted / unweighted

$\langle V, V, W \rangle$

emergent $\longrightarrow$ connected : all vertices have some seq. edges that relate them



DFS (




, E, F )

Function DFS( graph, start, end)

s ← new stack

s.push(start)

visited ← new HashTable

visited.put(start....)

while stack not empty:

ns ← Find_neighbors(s.pop() . graph)   {A, B, C, D, E}

for each n in ns:

if n not a key in visited:

s.push(n)   visited.put(n, ?)

if n = end: return True

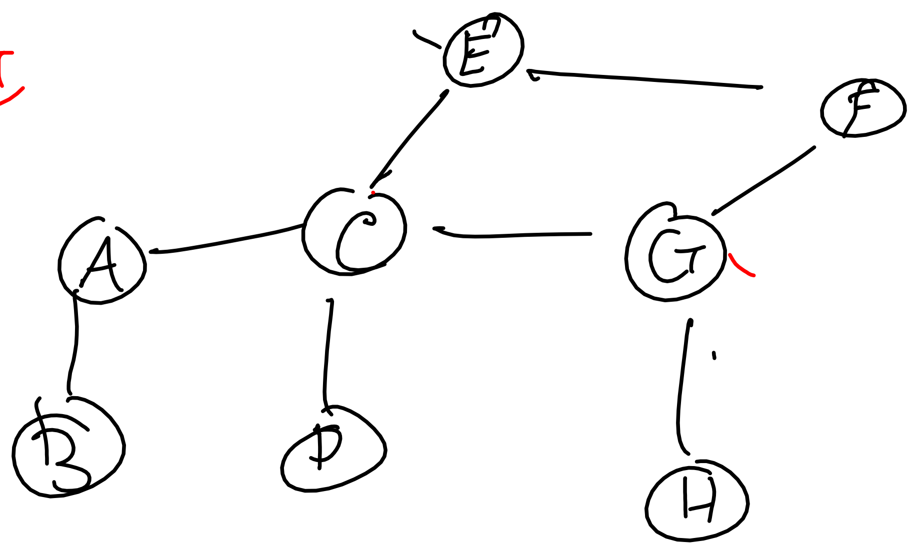
Return false

Function BFS(graph, start, end):

Like DFS but with queue

BFS: unweighted graph: guaranteed to give path w/ fewest edges

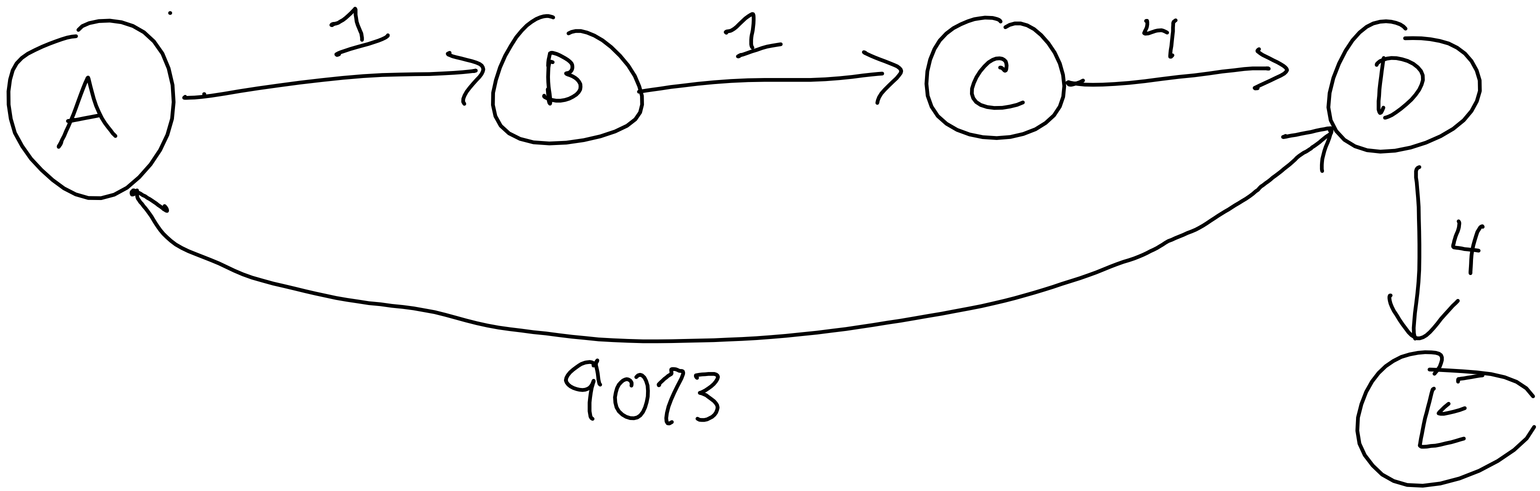
A ~> G



C / B

E / D / G

F / H




Right here is where lecture got really confusing. Instead of giving you the clunky, bad version of Dijkstra's Algorithm from class, we'll just go over it on Tuesday in a (hopefully clearer) fashion