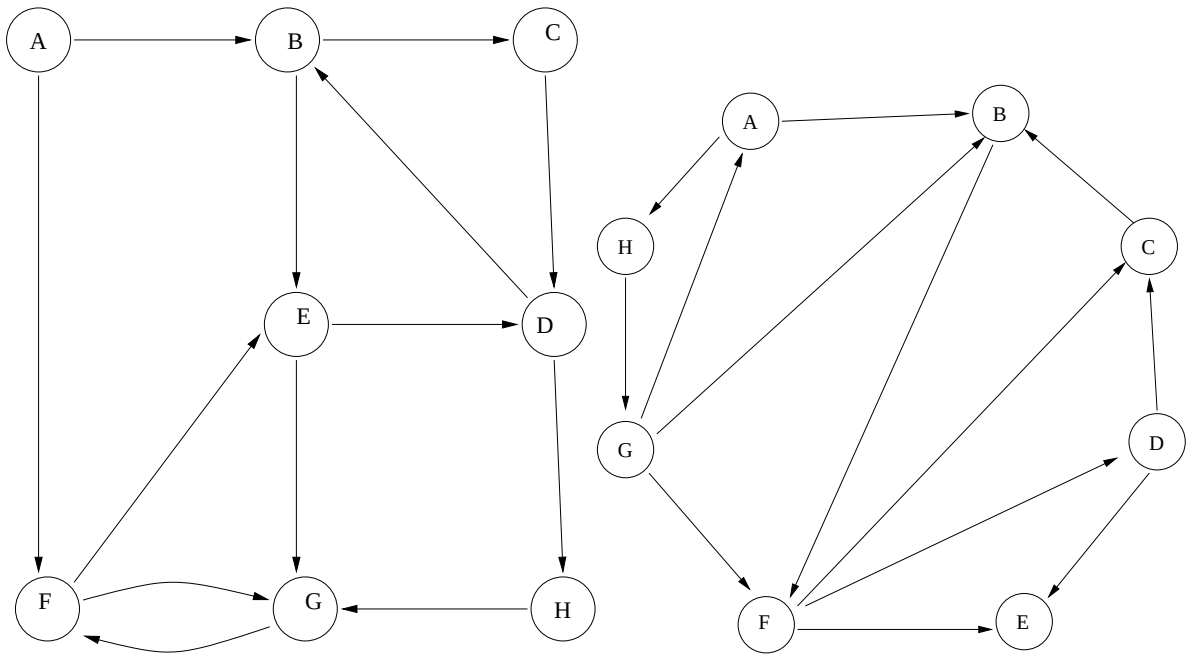
Practice Problems

1. Depth-First Search (DFS)
2. Shortest Path (Dijkstra's algorithm)
3. Minimum-Spanning Tree (MST)

Depth-First Search

- **DPV 3.2.**

Perform depth-first search on each of the following graphs; whenever there's a choice of vertices, pick the one that is alphabetically first.



Depth-First Search

- **DPV 3.11.**

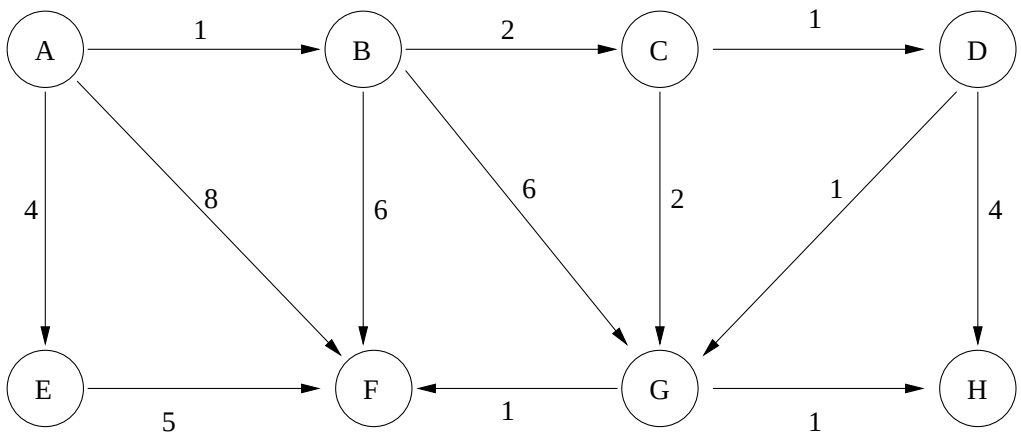
Design a linear-time algorithm which, given an undirected graph G and a particular edge e in it, determines whether G has a cycle containing e.

Shortest Path (Dijkstra's algorithm)

- **DPV 4.1.**

Suppose Dijstra's algorithm is run on the following graph, starting at node A.

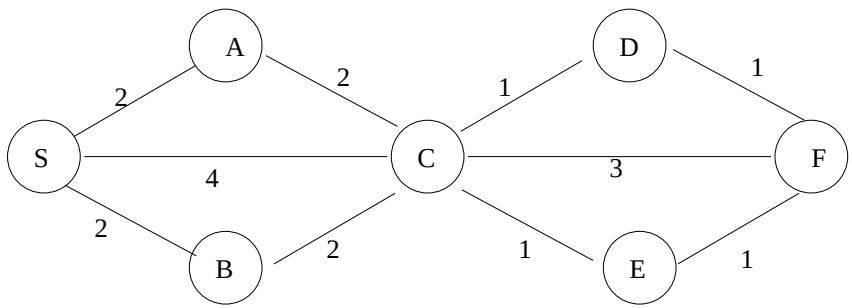
1. Draw a table showing the intermediate distance values of all the nodes at each iteration of the algorithm.
2. Show the final shortest-path tree.



Shortest Path (Dijkstra's algorithm)

- **DPV 4.18.**

In case where there are several different shortest paths between two nodes(and edges have varying lengths), the most convenient of these paths is often the one with fewest edges. For instance, if nodes represent cities and edges lengths represent costs of flying between cities, there might be many ways to get from city s to city t which all have the same cost. The most convenient of these alternatives is the one which involves the fewest stopovers. Accordingly, for a specific starting node s, define best[u] = minimum number of edges in a shortest path from s to u.
In the example below, the best values for nodes S, A, B, C, D, E, F are 0, 1, 1, 1, 2, 2, 3, respectively.



Give an efficient algorithm for the following problem.
Input: Graph $G = (V, E)$; positive edge lengths l_e; starting node $s \in V$.

Output: The values of best[u] should be set for all nodes $u \in V$.

Shortest Path (Dijkstra's algorithm)

- **DPV 4.18.**
- Solution

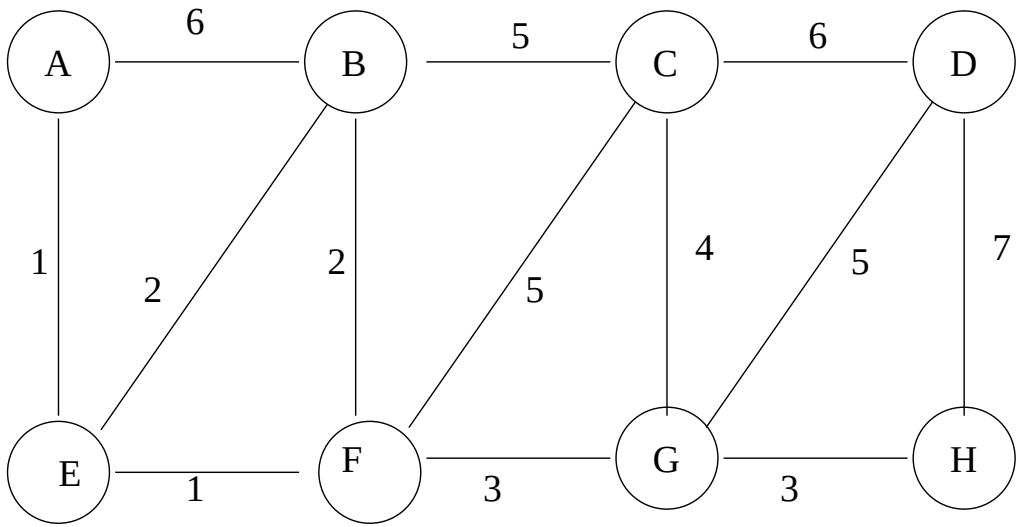
In the initialization loop, best(s) is set to 0 and all other entries of best are set to ∞. The main loop is modified as follows:

while Q is not empty **do**
  $u \leftarrow$ vertex in Q with smallest dist[]
  for all edges $(u, v) \in E$ **do**
    if dist(v)>dist(u)+$l(u, v)$ **then**
      dist(v)$\leftarrow$ dist(u)+$l(u, v)$
      best(v)$\leftarrow$ best(u)+1
    if dist(v)=dist(u)+$l(u, v)$ **then**
      if best(v)>best(u)+1 **then**
        best(v)$\leftarrow$ best(u)+1

Minimum-Spanning Tree

- **DPV 5.1.**

Consider the following graph. What is the cost of its minimum spanning tree?



Minimum-Spanning Tree

- **DPV 5.3.**

Design a linear-time algorithm for the following task.
Input: A connected, undirected graph G.
Question: Is there an edge you can remove from G while still leaving G connected?

Can you reduce the running time of your algorithm to $O(|V|)$?

Minimum-Spanning Tree

- **DPV 5.6.**

Let $G = (V, E)$ be an undirected graph. Prove that if all its edge weights are distinct, then it has a unique minimum spanning tree.

Minimum-Spanning Tree

- **DPV 5.9.**

The following statements may or may not be correct. In each case, either prove it (if it is correct) or give a counterexample (if it isn't correct). Always assume that the graph $G = (V, E)$ is undirected and connected. Do not assume that edge weights are distinct unless this is specifically stated.

1. If graph G has more than $|V| - 1$ edges, and there is a unique heaviest edge, then this edge cannot be part of a minimum spanning tree.
2. If G has a cycle with a unique heaviest edge e, then e cannot be part of any MST.
3. Let e be any edge of minimum weight in G. Then e must be part of some MST.
4. If the lightest edge in a graph is unique, then it must be part of every MST.
5. The shortest-path tree computed by Dijkstra's algorithm is necessarily an MST.
6. The shortest path between two nodes is necessarily part of some MST.
7. Prim's algorithm works correctly when there are negative edges.