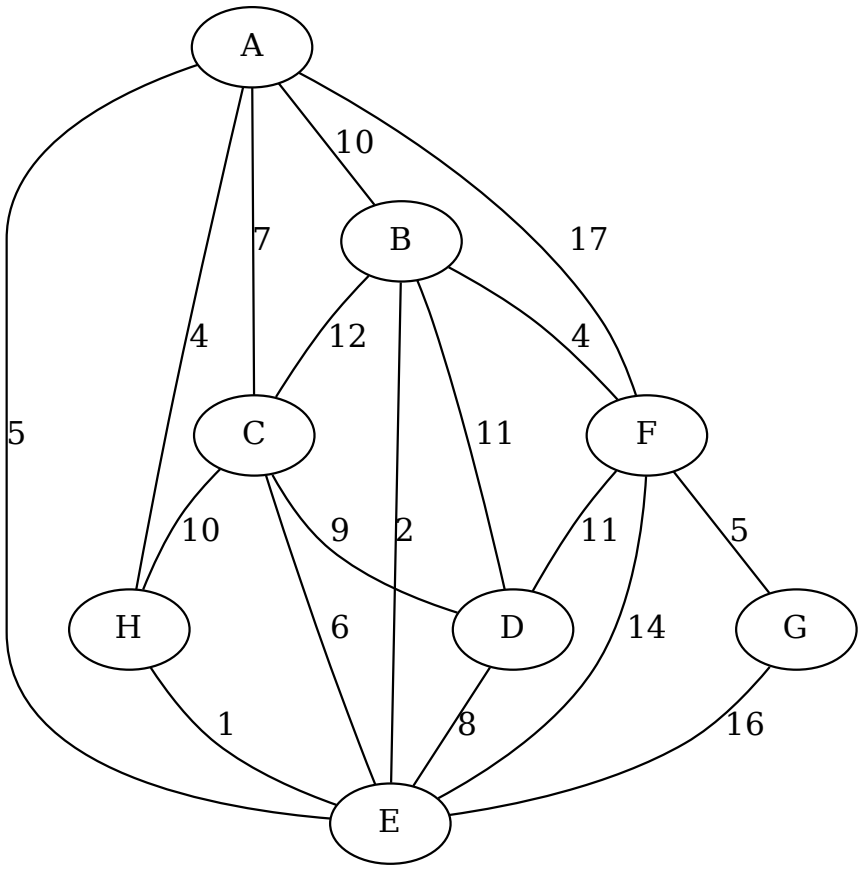
TD 7: Minimum Spanning Trees

1 Execute Prim's algorithm

Execute Prim's algorithm starting from A and show the resulting spanning tree. If there are ties, break them alphabetically.



2 Maximum Spanning Tree

Recall that unlike the problem of computing **shortest** paths, the problem of computing **longest** (simple) paths has no polynomial-time algorithm. In this exercise we observe that the same phenomenon does not manifest itself for spanning trees. Show an efficient algorithm which takes as input an edge-weighted undirected graph and outputs a spanning tree of **maximum** weight.

3 Unique Spanning Tree

Show that if all edge weights of a graph G are distinct, then G has a unique spanning tree of minimum weight.

4 Divide and Fail to Conquer

Consider the following algorithm for computing a minimum spanning tree: given graph $G = (V, E)$, partition V into two sets of equal size V_1, V_2 (or almost equal size, if $|V|$ is odd); solve the problem recursively on the graphs induced by V_1, V_2; add to the solution the cheapest edge with one endpoint in V_1 and the other in V_2. Is this algorithm correct?

5 Spanning Trees vs Shortest Path Trees

Recall that the shortest-path algorithms we have seen (for example, Dijkstra's algorithm) produce as output a shortest-path tree from the initial vertex. Since such an algorithm would naturally try to minimize the distances from the initial vertex, it may be tempting to attempt to use it to produce a minimum spanning tree. Conversely, it may be tempting to try to use a minimum spanning tree algorithm to compute shortest paths. The objective of this exercise is to realize that these heuristics fail badly.

1. Give an example of a graph on n vertices with minimum spanning tree cost t such that the shortest-path tree from any vertex s has weight at least $\Omega(t \cdot n)$.
2. Give an example of a graph on n vertices such that in all minimum spanning trees and for all vertices x there exists a vertex y such that the distance between x and y in the tree is $\Omega(n \cdot \mathrm{dist}(x, y))$.

The first example is supposed to demonstrate that on certain graphs, every shortest-path tree is significantly heavier than the minimum spanning tree. The second example that any minimum spanning tree will distort the shortest path distances to such an extent that it will fail to be close to a shortest-path tree for any initial vertex.

6 Boruvka's Algorithm

Boruvka's algorithm for MST is the following: we maintain a set of selected edges T (initially $T = \emptyset$). While $|T| < n - 1$ we do the following:

1. Compute the connected components of $G = (V, T)$ (using BFS/DFS)
2. For each component, we initialize its **best** edge as `NULL`.
3. For each edge $xy \in E$, if x, y are in the same component we skip this edge.
4. If, however, x, y are in distinct components C_1, C_2 we compare xy with the best edge of C_1, C_2 and update if xy has smaller weight (or has the same weight but comes first alphabetically). Every edge is considered better than `NULL`.
5. Add to T the best edge of each component.

Prove that this algorithm is correct and calculate its complexity. Assume that the input graph is connected.