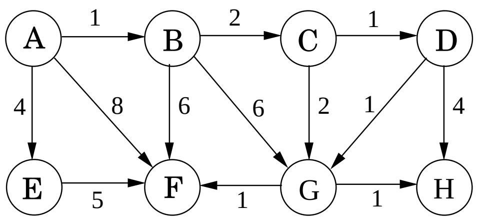
Exercises

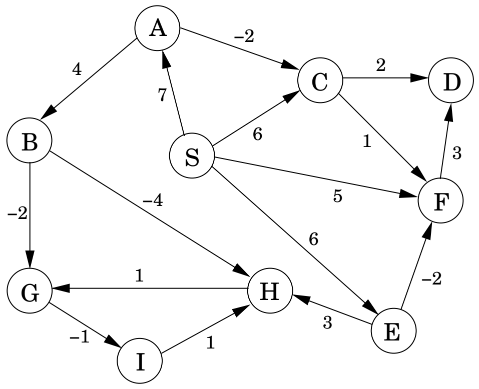
4.1. Suppose Dijkstra's algorithm is run on the following graph, starting at node A.



(a) Draw a table showing the intermediate distance values of all the nodes at each iteration of the algorithm.

(b) Show the final shortest-path tree.

4.2. Just like the previous problem, but this time with the Bellman-Ford algorithm.



4.3. *Squares.* Design and analyze an algorithm that takes as input an undirected graph $G = (V, E)$ and determines whether G contains a simple cycle (that is, a cycle which doesn't intersect itself) of length four. Its running time should be at most $O(|V|^3)$.

You may assume that the input graph is represented either as an adjacency matrix or with adjacency lists, whichever makes your algorithm simpler.

4.4. Here's a proposal for how to find the length of the shortest cycle in an undirected graph with unit edge lengths.

> When a back edge, say (v, w), is encountered during a depth-first search, it forms a cycle with the tree edges from w to v. The length of the cycle is level$[v]$ − level$[w] + 1$, where the level of a vertex is its distance in the DFS tree from the root vertex. This suggests the following algorithm:
>
> - Do a depth-first search, keeping track of the level of each vertex.
> - Each time a back edge is encountered, compute the cycle length and save it if it is smaller than the shortest one previously seen.

Show that this strategy does not always work by providing a counterexample as well as a brief (one or two sentence) explanation.

4.16. Section 4.5.2 describes a way of storing a complete binary tree of n nodes in an array indexed by $1, 2, \ldots, n$.

(a) Consider the node at position j of the array. Show that its parent is at position $\lfloor j/2 \rfloor$ and its children are at $2j$ and $2j+1$ (if these numbers are $\leq n$).

(b) What the corresponding indices when a complete d-ary tree is stored in an array?

Figure 4.16 shows pseudocode for a binary heap, modeled on an exposition by R.E. Tarjan.[2] The heap is stored as an array h, which is assumed to support two constant-time operations:

- $|h|$, which returns the number of elements currently in the array;
- h^{-1}, which returns the position of an element within the array.

The latter can always be achieved by maintaining the values of h^{-1} as an auxiliary array.

(c) Show that the `makeheap` procedure takes $O(n)$ time when called on a set of n elements. What is the worst-case input? (*Hint:* Start by showing that the running time is at most $\sum_{i=1}^{n} \log(n/i)$.)

(a) What needs to be changed to adapt this pseudocode to d-ary heaps?

4.17. Suppose we want to run Dijkstra's algorithm on a graph whose edge weights are integers in the range $0, 1, \ldots, W$, where W is a relatively small number.

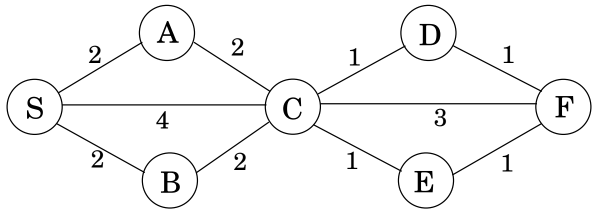
(a) Show how Dijkstra's algorithm can be made to run in time $O(W|V| + |E|)$.

(b) Show an alternative implementation that takes time just $O((|V| + |E|) \log W)$.

4.18. In cases 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 edge 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

$$\texttt{best}[u] = \text{minimum number of edges in a shortest path from } s \text{ to } u.$$

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.



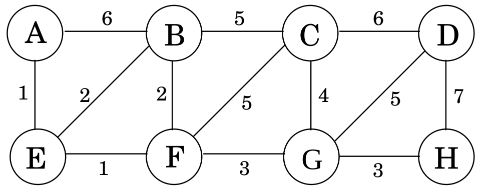
Give an efficient algorithm for the following problem.

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 $u \in V$.

[2]See: R. E. Tarjan, *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, 1983.

Exercises

5.1. Consider the following graph.

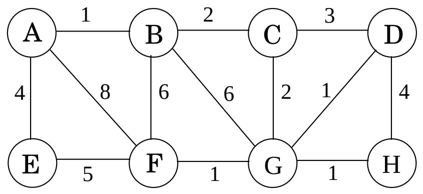


(a) What is the cost of its minimum spanning tree?

(b) How many minimum spanning trees does it have?

(c) Suppose Kruskal's algorithm is run on this graph. In what order are the edges added to the MST? For each edge in this sequence, give a cut that justifies its addition.

5.2. Suppose we want to find the minimum spanning tree of the following graph.



(a) Run Prim's algorithm; whenever there is a choice of nodes, always use alphabetic ordering (e.g., start from node A). Draw a table showing the intermediate values of the `cost` array.

(b) Run Kruskal's algorithm on the same graph. Show how the disjoint-sets data structure looks at every intermediate stage (including the structure of the directed trees), assuming path compression is used.

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 $O(|V|)$?

5.4. Show that if an undirected graph with n vertices has k connected components, then it has at least $n - k$ edges.

5.5. Consider an undirected graph $G = (V, E)$ with nonnegative edge weights $w_e \geq 0$. Suppose that you have computed a minimum spanning tree of G, and that you have also computed shortest paths to all nodes from a particular node $s \in V$.

Now suppose each edge weight is increased by 1: the new weights are $w'_e = w_e + 1$.

(a) Does the minimum spanning tree change? Give an example where it changes or prove it cannot change.

(b) Do the shortest paths change? Give an example where they change or prove they cannot change.

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.

5.7. Show how to find the *maximum* spanning tree of a graph, that is, the spanning tree of largest total weight.

5.8. Suppose you are given a weighted graph $G = (V, E)$ with a distinguished vertex s and where all edge weights are positive and distinct. Is it possible for a tree of shortest paths from s and a minimum spanning tree in G to not share any edges? If so, give an example. If not, give a reason.

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. Do not assume that edge weights are distinct unless this is specifically stated.

(a) 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.

(b) If G has a cycle with a unique heaviest edge e, then e cannot be part of any MST.

(c) Let e be any edge of minimum weight in G. Then e must be part of some MST.

(d) If the lightest edge in a graph is unique, then it must be part of every MST.

(e) If e is part of some MST of G, then it must be a lightest edge across some cut of G.

(f) If G has a cycle with a unique lightest edge e, then e must be part of every MST.

(g) The shortest-path tree computed by Dijkstra's algorithm is necessarily an MST.

(h) The shortest path between two nodes is necessarily part of some MST.

(i) Prim's algorithm works correctly when there are negative edges.

(j) (For any $r > 0$, define an *r-path* to be a path whose edges all have weight $< r$.) If G contains an r-path from node s to t, then every MST of G must also contain an r-path from node s to node t.

5.10. Let T be an MST of graph G. Given a connected subgraph H of G, show that $T \cap H$ is contained in some MST of H.

5.11. Give the state of the disjoint-sets data structure after the following sequence of operations, starting from singleton sets $\{1\}, \ldots, \{8\}$. Use path compression. In case of ties, always make the lower numbered root point to the higher numbered one.

$$\texttt{union}(1,2), \texttt{union}(3,4), \texttt{union}(5,6), \texttt{union}(7,8), \texttt{union}(1,4), \texttt{union}(6,7), \texttt{union}(4,5), \texttt{find}(1)$$

5.12. Suppose you implement the disjoint-sets data structure using union-by-rank but not path compression. Give a sequence of m `union` and `find` operations on n elements that take $\Omega(m \log n)$ time.

5.13. A long string consists of the four characters A, C, G, T; they appear with frequency $31\%, 20\%, 9\%$, and 40%, respectively. What is the Huffman encoding of these four characters?

5.14. Suppose the symbols a, b, c, d, e occur with frequencies $1/2, 1/4, 1/8, 1/16, 1/16$, respectively.

(a) What is the Huffman encoding of the alphabet?

(b) If this encoding is applied to a file consisting of $1{,}000{,}000$ characters with the given frequencies, what is the length of the encoded file in bits?