24.1-5 ⋆

Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \to \mathbf{R}$. Give an $O(VE)$-time algorithm to find, for each vertex $v \in V$, the value $\delta^*(v) = \min_{u \in V} \{\delta(u, v)\}$.

24.1-6 ⋆

Suppose that a weighted, directed graph $G = (V, E)$ has a negative-weight cycle. Give an efficient algorithm to list the vertices of one such cycle. Prove that your algorithm is correct.

24.2 Single-source shortest paths in directed acyclic graphs

By relaxing the edges of a weighted dag (directed acyclic graph) $G = (V, E)$ according to a topological sort of its vertices, we can compute shortest paths from a single source in $\Theta(V + E)$ time. Shortest paths are always well defined in a dag, since even if there are negative-weight edges, no negative-weight cycles can exist.

The algorithm starts by topologically sorting the dag (see Section 22.4) to impose a linear ordering on the vertices. If there is a path from vertex u to vertex v, then u precedes v in the topological sort. We make just one pass over the vertices in the topologically sorted order. As we process each vertex, we relax each edge that leaves the vertex.

```
DAG-SHORTEST-PATHS(G, w, s)
1  topologically sort the vertices of G
2  INITIALIZE-SINGLE-SOURCE(G, s)
3  for each vertex u, taken in topologically sorted order
4      do for each vertex v ∈ Adj[u]
5             do RELAX(u, v, w)
```

Figure 24.5 shows the execution of this algorithm.

The running time of this algorithm is easy to analyze. As shown in Section 22.4, the topological sort of line 1 can be performed in $\Theta(V + E)$ time. The call of INITIALIZE-SINGLE-SOURCE in line 2 takes $\Theta(V)$ time. There is one iteration per vertex in the **for** loop of lines 3–5. For each vertex, the edges that leave the vertex are each examined exactly once. Thus, there are a total of $|E|$ iterations of the inner **for** loop of lines 4–5. (We have used an aggregate analysis here.) Because each iteration of the inner **for** loop takes $\Theta(1)$ time, the total running time is $\Theta(V + E)$, which is linear in the size of an adjacency-list representation of the graph.

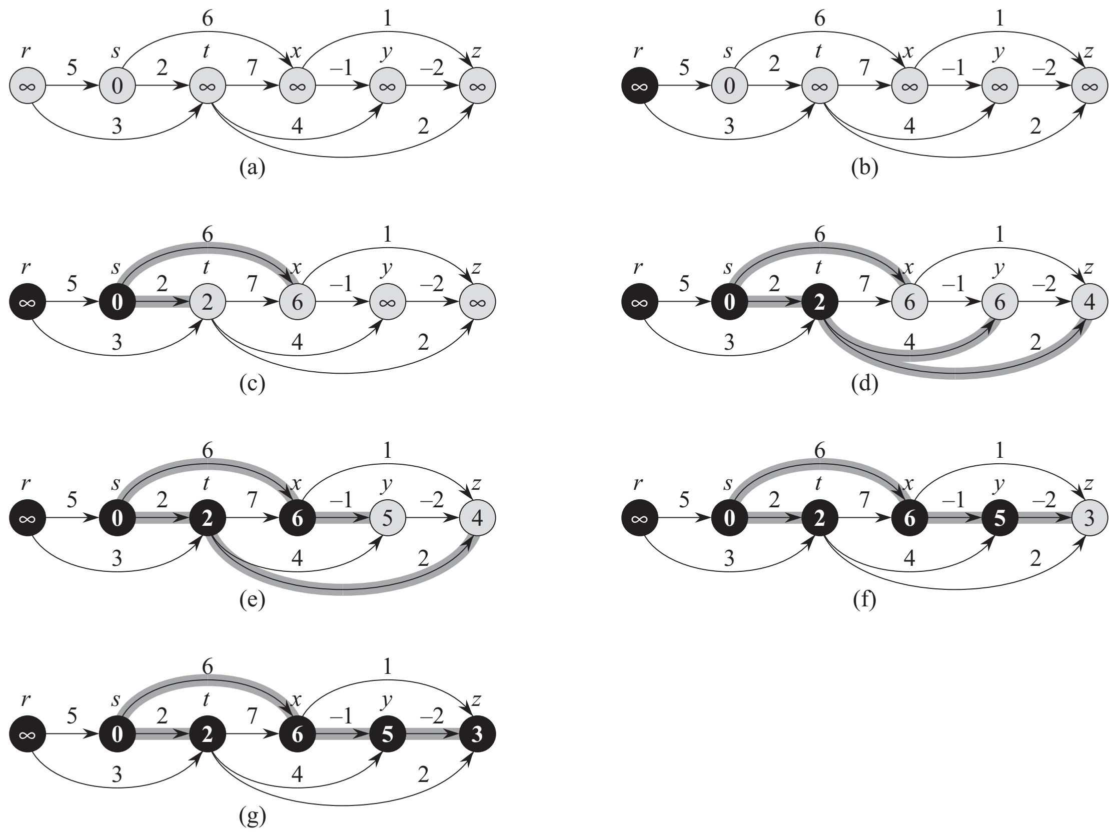

Figure 24.5 The execution of the algorithm for shortest paths in a directed acyclic graph. The vertices are topologically sorted from left to right. The source vertex is s. The d values are shown within the vertices, and shaded edges indicate the π values. **(a)** The situation before the first iteration of the **for** loop of lines 3–5. **(b)–(g)** The situation after each iteration of the **for** loop of lines 3–5. The newly blackened vertex in each iteration was used as u in that iteration. The values shown in part (g) are the final values.

The following theorem shows that the DAG-SHORTEST-PATHS procedure correctly computes the shortest paths.

Theorem 24.5

If a weighted, directed graph $G = (V, E)$ has source vertex s and no cycles, then at the termination of the DAG-SHORTEST-PATHS procedure, $d[v] = \delta(s, v)$ for all vertices $v \in V$, and the predecessor subgraph G_π is a shortest-paths tree.

Proof We first show that $d[v] = \delta(s, v)$ for all vertices $v \in V$ at termination. If v is not reachable from s, then $d[v] = \delta(s, v) = \infty$ by the no-path property. Now, suppose that v is reachable from s, so that there is a shortest path $p = \langle v_0, v_1, \ldots, v_k \rangle$, where $v_0 = s$ and $v_k = v$. Because we process the vertices in topologically sorted order, the edges on p are relaxed in the order $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$. The path-relaxation property implies that $d[v_i] = \delta(s, v_i)$ at termination for $i = 0, 1, \ldots, k$. Finally, by the predecessor-subgraph property, G_π is a shortest-paths tree. ■

An interesting application of this algorithm arises in determining critical paths in ***PERT chart***[2] analysis. Edges represent jobs to be performed, and edge weights represent the times required to perform particular jobs. If edge (u, v) enters vertex v and edge (v, x) leaves v, then job (u, v) must be performed prior to job (v, x). A path through this dag represents a sequence of jobs that must be performed in a particular order. A ***critical path*** is a *longest* path through the dag, corresponding to the longest time to perform an ordered sequence of jobs. The weight of a critical path is a lower bound on the total time to perform all the jobs. We can find a critical path by either

- negating the edge weights and running DAG-SHORTEST-PATHS, or
- running DAG-SHORTEST-PATHS, with the modification that we replace "∞" by "$-\infty$" in line 2 of INITIALIZE-SINGLE-SOURCE and "$>$" by "$<$" in the RELAX procedure.

Exercises

24.2-1

Run DAG-SHORTEST-PATHS on the directed graph of Figure 24.5, using vertex r as the source.

24.2-2

Suppose we change line 3 of DAG-SHORTEST-PATHS to read

3 **for** the first $|V| - 1$ vertices, taken in topologically sorted order

Show that the procedure would remain correct.

24.2-3

The PERT chart formulation given above is somewhat unnatural. It would be more natural for vertices to represent jobs and edges to represent sequencing con-

---

[2] "PERT" is an acronym for "program evaluation and review technique."