I. Common Algorithms for Single Source Shortest Path

These algorithms occur for all of the given single source shortest path algorithms, so Cormen presents them to shorten the other algorithms. This is fine if you remember what they are, but in case you have problems the algorithms are presented both ways in this handout.

Algorithm 1 Initialize-Single-Source(G, s)

1: procedure INITIALIZE-SINGLE-SOURCE((G[], s))
2: for each vertex \( v \in G.V \) do
3: \( v.d = \infty \)
4: \( v.\pi = \text{null} \)
5: end for
6: \( s.d = 0 \)
7: end procedure

Algorithm 2 Relax(u, v, w)

1: procedure RELAX((u,v,w[]))
2: if \( v.d > u.d + w(u,v) \) then
3: \( v.d = u.d + w(u,v) \)
4: \( v.\pi = u \)
5: end if
6: end procedure
II. Bellman-Ford Algorithm

This is Cormen’s short algorithm on page 651 and an expanded version. The Bellman-Ford algorithm returns true if there is a negative weight cycle and returns the shortest path solution if there are no negative weight cycles.

Algorithm 3 Bellman-Ford Single Source Shortest Path
1: procedure BELLMAN-FORD((G[],w[],s))
2: INITIALIZE-SINGLE-SOURCE(G, s)
3: for $i = 1$ to $|G.V| - 1$ do
4:   for each edge $(u,v) \in G.E$ do
5:     RELAX($u,v,w$)
6:   end for
7: end for
8: for each edge $(u,v) \in G.E$ do
9:   if $v.d > u.d + w(u,v)$ then
10:      return FALSE
11: end if
12: end for
13: return TRUE
14: end procedure

Algorithm 4 Bellman-Ford Single Source Shortest Path
1: procedure BELLMAN-FORD((G[],w[],s))
2: for each vertex $v \in G.V$ do
3:   $v.d = \infty$
4:   $v.\pi = \text{null}$
5: end for
6: $s.d = 0$
7: for $i = 1$ to $|G.V| - 1$ do
8:   for each edge $(u,v) \in G.E$ do
9:     if $v.d > u.d + w(u,v)$ then
10:        $v.d = u.d + w(u,v)$
11:        $v.\pi = u$
12:     end if
13:   end for
14: end for
15: for each edge $(u,v) \in G.E$ do
16:   if $v.d > u.d + w(u,v)$ then
17:      return FALSE
18: end if
19: end for
20: return TRUE
21: end procedure

Time Complexity: Bellman-Ford runs in $O(|V||E|)$
III. SINGLE SOURCE SHORTEST PATH IN A DAG

This is a generic algorithm to find the single source shortest path in a DAG which is based upon the earlier algorithm for a topological sort of the DAG.

Algorithm 5 DAG-Shortest-Paths(G[, w[, s]])
1: procedure DAG-SHORTEST-PATHS((G[, w[, s]]))
2:   topologically sort the vertices of G
3:   INITIALIZE-SINGLE-SOURCE(G, s)
4:   for each vertex u, in topologically sorted order do
5:     for each vertext v ∈ G.Adj[u] do
6:       RELAX(u, v, w)
7:     end for
8:   end for
9: end procedure

Algorithm 6 DAG-Shortest-Paths(G[, w[, s]])
1: procedure DAG-SHORTEST-PATHS((G[, w[, s]]))
2:   topologically sort the vertices of G
3:   for each vertex v ∈ G.V do
4:     v.d = ∞
5:     v.π = null
6:   end for
7:   s.d = 0
8:   for each vertex u, in topologically sorted order do
9:     for each vertext v ∈ G.Adj[u] do
10:    if v.d > u.d + w(u, v) then
11:       v.d = u.d + w(u, v)
12:       v.π = u
13:    end if
14:   end for
15: end for
16: end procedure

Time Complexity: O(V + E)
IV. DIJKSTRA’S ALGORITHM

This is Cormen’s short algorithm on page 658 and an expanded version. Dijkstra’s algorithm runs in less time than the others given in Cormen, but has an extra constraint:

**Constraint 1.** For each edge $(u, v) \in E$, $weight(u, v) \geq 0$

The time complexity of Dijkstra’s algorithm depends upon the implementation of the priority queue, $Q$. The best case known so far uses a Fibonacci heap (see Chapter 19).

**Algorithm 7** Dijkstra Single Source Shortest Path

1: procedure Dijkstra$(\langle G\rangle, w[,], s)$
2: \hspace{1em} Initialize-Single-Source$(G, s)$
3: \hspace{1em} $S = \emptyset$
4: \hspace{1em} $Q = G.v$
5: \hspace{1em} while $Q \neq \emptyset$ do
6: \hspace{2em} $u = \text{Extract-Min}(Q)$
7: \hspace{2em} $S = S \cup \{u\}$
8: \hspace{2em} for each vertex $v \in G.Adj[u]$ do
9: \hspace{3em} Relax$(u, v, w)$
10: \hspace{2em} end for
11: \hspace{1em} end while
12: end procedure

**Algorithm 8** Dijkstra Single Source Shortest Path

1: procedure Dijkstra$(\langle G\rangle, w[,], s)$
2: \hspace{1em} for each vertex $v \in G.V$ do
3: \hspace{2em} $v.d = \infty$
4: \hspace{2em} $v.\pi = \text{null}$
5: \hspace{1em} end for
6: \hspace{1em} $s.d = 0$
7: \hspace{1em} $S = \emptyset$
8: \hspace{1em} $Q = G.v$
9: \hspace{1em} while $Q \neq \emptyset$ do
10: \hspace{2em} $u = \text{Extract-Min}(Q)$
11: \hspace{2em} $S = S \cup \{u\}$
12: \hspace{2em} for each vertex $v \in G.Adj[u]$ do
13: \hspace{3em} if $v.d > u.d + w(u, v)$ then
14: \hspace{4em} $v.d = u.d + w(u, v)$
15: \hspace{4em} $v.\pi = u$
16: \hspace{3em} end if
17: \hspace{2em} end for
18: \hspace{1em} end while
19: end procedure

*Time Complexity:* $O(V \lg V + E)$