Shortest Spanning Tree (Prim algorithm)

procedure SHORTEST_SPANNING_TREE:
begin
  $W := \{v_1\}$; $E' := \varnothing$;
  comment: $b(v) =$ vertex $\in W : w(v, b(v)) = \min_{r \in W}\{w(v, r)\}$;
  for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1$;
  while $W \neq V$ **do**
    begin
      find $\overline{v} \in V \setminus W \; : w(\overline{v}, b(\overline{v})) = \min_{v \in V \setminus W}\{w(v, b(v))\}$;
      $W := W \cup \{\overline{v}\}$; $E' := E' \cup \{(\overline{v}, b(\overline{v}))\}$;
      for each $v \in V \setminus W$ **do**
        if $w(v, \overline{v}) < w(v, b(v))$ **then** $b(v) := \overline{v}$
    end
end.

Shortest Paths (Dijkstra algorithm)

procedure SHORTEST_PATHS:
begin
  $S := \{s\}$; $\ell(s) := 0$; $p(s) := \varnothing$;
  comment: $S =$ set of the vertices reached by a shortest path;
  comment: $\ell(v) =$ length of the shortest path from s to v that only
        passes through vertices $\in S$;
  comment: $p(v) =$ predecessor of v in the path of length $\ell(v)$;
  for each $v \in V \setminus \{s\}$ **do**
    begin
      $\ell(v) := w(s, v)$;
      $p(v) := s$
    end;
  while $S \neq V$ **do**
    begin
      find $\overline{v} \in V \setminus S \; : \ell(\overline{v}) = \min_{v \in V \setminus S}\{\ell(v)\}$;
      $S := S \cup \{\overline{v}\}$;
      for each $v \in V \setminus S$ **do**
        if $\ell(\overline{v}) + w(\overline{v}, v) < \ell(v)$ **then**
          begin
            $\ell(v) := \ell(\overline{v}) + w(\overline{v}, v)$;
            $p(v) := \overline{v}$
          end
    end
end.

Maximum Flow (Ford-Fulkerson algorithm)

```
procedure MAX_FLOW:
begin
    for i := 1 to n do  for j := 1 to n do ξ_ij := 0;
    comment: Vertex labels: a vertex can be in 3 states:
                 1. unlabeled;
                 2. labeled (if it belongs the vertex set V_1 that contains s) and non-explored;
                 3. labeled and explored (if the arcs emanating from it have been scanned);
    comment: Meaning of the vertex labels: the label of a vertex v_i has the structure:
                          { [+v_k, δ]  ⇔  ξ_ki can be increased, or
                          { [−v_k, δ]  ⇔  ξ_ik can be decreased,
                 where δ = maximum additional flow that can be sent from s to v_i.
    opt := false;
    while opt = false do
        begin
            label s with [+s, +∞];
            repeat
                let v_i be a vertex labeled [±v_k, δ(v_i)] and non-explored;
                for each unlabeled v_j ∈ {v_k ∈ V : (v_i, v_k) ∈ A and ξ_ik < q_ik} do
                    label v_j with [+v_i, min(δ(v_i), q_ij − ξ_ij)];
                for each unlabeled v_j ∈ {v_k ∈ V : (v_k, v_i) ∈ A and ξ_ki > 0} do
                    label v_j with [−v_i, min(δ(v_i), ξ_ji)];
                mark v_i as explored
            until t is labeled or no further vertex can be labeled;
            if t is unlabeled then opt := true
            else
                begin
                    δ* := δ(t);  x := t;
                    repeat
                        if the label of x is [+y, δ(x)] then ξ_yx := ξ_yx + δ*
                        else (i.e., [−y, δ(x)]) ξ_xy := ξ_xy − δ*;
                        x := y
                    until x = s;
                    cancel all labels
                end
        end;
    comment: Minimum cut: (V_1 = {v_j ∈ V : v_j is labeled }, V_2 = V \ V_1)
end.
```

Critical Path (CPM algorithm)

procedure CPM:
begin
  comment: Step 1. Number the vertices so that $i < j$ $\forall$ arc $(v_i, v_j) \in A$;
  add to G fictitious vertices v_0 and v_{n+1}, and the corresponding arcs;
  $B := A$;
  $k := 0$;
  while $k \leq n+1$ **do**
    begin
      select a non-numbered vertex v : $\{(v_i, v) \in B\} = \varnothing$;
      number v as v_k;
      $B := B \setminus \{(v, v_i) \in B\}$;
      $k := k + 1$
    end;
  comment: Step 2. Determine the makespan = length of the longest path;
  comment: $TMIN_k$ = minimum time instant at which event v_k can occur
    without violating the precedence relationships;
  $TMIN_0 := 0$;
  for $k := 1$ **to** $n+1$ **do**
    $TMIN_k := \max_{i:(v_i,v_k)\in A} \{TMIN_i + d(v_i, v_k)\}$;
  comment: $TMAX_k$ = maximum time instant at which event v_k can occur
    without delaying the project completion time, $TMIN_{n+1}$;
  $TMAX_{n+1} := TMIN_{n+1}$;
  for $k := n$ **downto** 0 **do**
    $TMAX_k := \min_{i:(v_k,v_i)\in A} \{TMAX_i - d(v_k, v_i)\}$
  comment: Step 3. Determine the basic parameters for each activity $a_h = (v_i, v_j)$:
    $EST(a_h)$ = earliest time instant at which a_h can start;
    $LST(a_h)$ = latest time instant at which a_h can start;
    $S(a_h)$ = slack time between earliest and latest start time;
  for each $a_h = (v_i, v_j) \in A$ **do**
    begin
      $EST(a_h) = TMIN_i$;
      $LST(a_h) = TMAX_j - d(v_i, v_j)$;
      $S(a_h) = LST(a_h) - EST(a_h)$
    end;
  comment: a critical activity is an activity a_h : $LST(a_h) = EST(a_h)$;
  comment: a critical path is a path from v_0 to v_{n+1} composed by critical activities
end.