**Algorithm 1** Dijkstra's Algorithm

**Input:** Graph $G = (V, E)$, source $s \in V$, target $t \in V$
**Output:** Shortest path from $s$ to $t$

// Initialize weights
Construct a map *distance* whose keys are nodes and values are distances
Construct a map *previous* whose keys are nodes and values are nodes
**for** $v \in V$ **do**
    $distance[v] \leftarrow \infty$
    $previous[v] \leftarrow$ undefined
**end for**
$distance[s] \leftarrow 0$

// Initialize priority queue
Construct an empty priority queue $Q$
**for** $v \in V$ **do**
    Add $v$ to $Q$ with priority $distance[v]$
**end for**

// Calculate paths
**while** $|Q| > 0$ **do**
    Let $u = Q$.getMin()
    $Q$.deleteMin()
    **if** $distance[u] = \infty$ **then**
        // TODO: What should happen here?
    **end if**
    **for** $v \in$ neighbors($u$) **do**
        Let $altDistance = distance[u] +$ edgeWeight$(u, v)$
        **if** $altDistance < distance[v]$ **then**
            $distance[v] \leftarrow altDistance$
            $previous[v] \leftarrow u$;
            $Q$.updateWeight($v$,$distance[v]$)
        **end if**
    **end for**
**end while**

// Return shortest path from $s$ to $t$
Construct an empty stack of vertices called *path*
*path*.push($t$)
**while** *path*.top() $\neq s$ **do**
    *path*.push(*previous*[*path*.top()])
**end while**
**return** *path*