Dijkstra Single Source Shortest Path Algorithm

The Single Source Shortest Path (*SSSP*) algorithm proposed by Dijkstra (*Dijkstra_SSSP*) computes all shortest paths from a single node. The idea of this algorithm is to start from a source node s and include iteratively in the route the nodes with the lowest costs from s. As shown in Algorithm 1, S is the set of nodes for which shortest paths have not been found, and d_u for node u is the shortest known distance from the source node s to node u. The algorithm starts by setting $S = V$ and $d_u = \infty$ for each node u except the source node s, which has $d_s = 0$. At each iteration, the vertex v that has the minimum distance value to the source is deleted from S, and each neighbor u of v is investigated to find if a path through v provides a shorter path to s than the current distance d_u.

Algorithm 1. *Dijkstra_SSSP*

```
1:  d_s ← 0                                          ▷ initialize distances
2:  for all i ≠ s do
3:      d_i ← ∞
4:  end for
5:  S ← V
6:  while S ≠ ∅ do
7:      find v_m ∈ S with minimum d                  ▷ find node v_m with minimum distance
8:      for all {v_m, u} do                          ▷ update each neighbor distance of v_m
9:          if d_u > d_vm + length({u, v_m}) then
10:             d_u ← d_vm + length({u, v_m})
11:         end if
12:     end for
13:     T ← T ∪ {v_m}                                ▷ include new node in the shortest path
14:     S ← S \ {v_m}                                ▷ remove new node from searched
15: end while
```

An example execution of *Dijkstra_SSSP* is shown in a directed graph G in Figure 1, where node a is the source node from which the shortest paths are to be computed. The node with the lowest distance is node a itself as all others have infinite distances initially. Nodes b and e that are neighbors of a are marked with distances 8 and 2, respectively, and a is added to T, removed from S in the first iteration of the loop. In the second iteration of the *for* loop, node b has the lowest distance in S, its neighbors e and c are marked with distances 3 and 8, and the previous distance 8 of node e is changed to 3 in this iteration. The algorithm proceeds similarly, adding a vertex to the already decided list of vertices T in each iteration, and finally, $T = \{a, b, e, d, c\}$ in sequence is obtained.

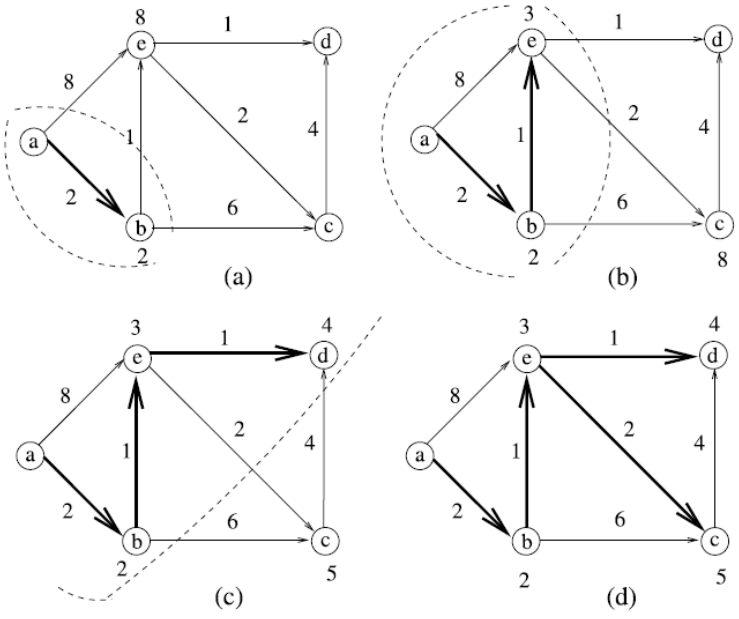

Figure 1 *Dijkstra_SSSP* execution example. **(a)** The graph G. **(b)** First iteration. **(c)** Second iteration. **(d)** Final iteration.