Dijkstra's Algorithm

Usage

- Dijkstra's algorithm works correctly for graphs whose edges are undirected, non-negative, non-parallel and weighted, and the graph doesn't contain self-loops.
- In addition to that, the algorithm needs a distinguished start vertex.
- If all these prerequisites are given, the algorithm will find the shortest path from the start vertex to any other vertex in the graph.
- "shortest path" = minimum sum of the weights of the edges that we've passed through to reach a target vertex (starting from the start vertex)

Pseudocode

See the next page for a complete description of the algorithm in pseudocode.

Complexity

- The time complexity highly depends on the implementation of the priority queue. For any implementation, the running time is in $O(E \cdot T_{dk} + V \cdot T_{em})$.
- With a binary heap or a self-balancing binary search tree the complexity becomes $O((E+V) \cdot \log(V))$.
- With a Fibonacci heap, the complexity improves to $O(E + V \cdot \log(V))$.

E: number of edges

V: number of vertices

T_{dk}: complexity of the decrease-key operation

T_{em}: complexity of the extract-minimum operation