Shortest Paths Algorithms

The distance between two vertices u and v is the mimumum, over all paths starting at u and ending at v, of the weight of the path. The weight of a path is the sum of the weights of its (directed) edges. We sometimes use *length* instead of weight for edges and paths, and by shortest here we mean “of minimum length”. (the number of edges in a path is not relevant here, and we will explicitly say “number of edges” and not length if we refer to the number of edges).

Dijkstra’s algorithm for single-source shortest paths in directed graphs with non-negative weight

DIJKSTRA(G,w,r)

for each vertex $u \in V[G]$
  do $d[u] \longleftarrow \infty$
      $\pi[u] \longleftarrow NIL$
$d[r] \longleftarrow 0$
$Q \longleftarrow V[G]$
while $Q \neq \emptyset$
  do $u \longleftarrow$ EXTRACT-MIN(Q)
      for each $v \in Adj[u]$
        do if $d[u] + w(u, v) < d[v]$ // This is called “Relax($u, v, w()$)”
            then $\pi[v] \longleftarrow u$
                $d[v] \longleftarrow d[u] + w(u, v)$
                DECREASE-KEY($Q, v, d[v]$)

The implementation of Djikstra’s algorithm uses a min-priority queue Q, containing vertices v using $d[v]$ as the key-value. The running time of Djikstra’s algorithm depends on how we implement Q. At the end of the algorithm’s execution, $d[v]$ equals the total weight of a least-weight path from r to v. This path can be obtained, in reverse order, by following π links from v to NIL.

Similar to the analysis of BFS, we use $dist[v]$ to denote the weight of the shortest path from r to v in the input weighted graph G,w.

The correctness of Dijkstra’s algorithm relies on the following invariant of the **while** loop:

1. For each vertex v with $d[v] < \infty$, we have a path from r to v of length $d[v]$ which can be obtained, reversed, by taking $\pi[]$ pointers from v
2. For all $v \notin Q$, $d[v] = dist[v]$, while for all $v \in Q$ with $d[v] < \infty$, $d[v]$ is equal to the length of the shortest path from r to v that has, except for v, all the vertices not in Q.
3. For all $v \notin Q$ and all $v' \in Q$, we have $d[v] \leq d[v']$. The algorithm never modifies $d[v]$ and $\pi[v]$ for $v \notin Q$.

Running time: $O(|E|\log|V|)$ - since there are $O(|E|)$ DECREASE-KEY() operations and each can be done in $O(\log|V|)$ with binary heaps. Fibonnaci heaps achieve $O(|E|)$ running time for all the DECREASE-KEY() operations, and the running time becomes $O(|V|\log|V| + |E|)$, with $O(\log|V|)$ enough to do each of the $|V|$ EXTRACT-MIN(Q) operations.

FLOYD-WARSHALL(G,w) (here w is a $|V| \times |V|$ matrix)

1. $d^0 \longleftarrow w$ // (here d^k is a $|V| \times |V|$ matrix, for $k = 0, 1, \ldots, |V|$)
2. **for** $k \longleftarrow 1$ **to** $|V|$
3.   **for** $i \longleftarrow 1$ **to** $|V|$
4.   **for** $j \longleftarrow 1$ **to** $|V|$
5. $d^k[i,j] \longleftarrow \min\left(d^{k-1}[i,j], d^{k-1}[i,k] + d^{k-1}[k,j]\right)$

Explanation: $d^k[i,j]$ stands for the length of the shortest path from i to j that uses in its interior only vertices from $\{1, 2, \ldots, k\}$.

Running time: $O(|V|^3)$.