Dijkstra’s Shortest-path Algorithm

Dijkstra algorithm is applied to solve *single-source shortest-path problems* on a *weighted, directed* graph $G=(V,E)$ whose edges are all *non-negative*. The algorithm as follows:

Algorithm: DIJKSTRA(G,s)

Initialize(G,s)

$S=\varnothing$

$Q=G.V$

while $Q\neq\varnothing$:

u=EXTRAMIN(Q)

$S=S\cup\{u\}$

for each vertex $v\in G.Adj[u]$

RELAX(u,v)

* Initialize(G,s) is a method to initialize an assistant array to store path information

* RELAX(u,v) is a method which is used to modify the path length between u and v like :

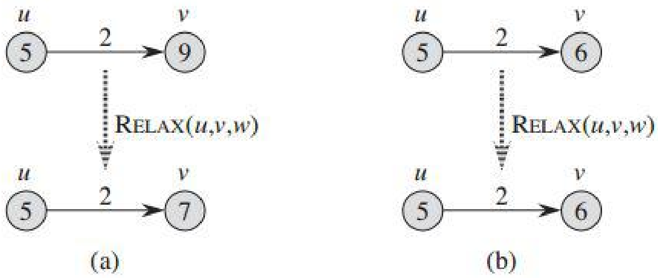

Figure 24.3 Relaxing an edge (u, v) with weight $w(u, v) = 2$. The shortest-path estimate of each vertex appears within the vertex. **(a)** Because $v.d > u.d + w(u, v)$ prior to relaxation, the value of $v.d$ decreases. **(b)** Here, $v.d \leq u.d + w(u, v)$ before relaxing the edge, and so the relaxation step leaves $v.d$ unchanged.

1. *Termination of Dijkstra's algorithm*

Correctness of Dijkstra's algorithm: *If Dijkstra's algorithm run on a non-negatively weighted and directed graph G with single source s, then for all* $u \in V$ *Dijkstra's algorithm terminates with* $\delta(s,u) = u.d$ *where* $\delta(s,u)$ *is the shortest path between* s *and* u*,* $u.d$ *is the distance between* s *and* u

2. *Analysis of Dijkstra's Algorithm*

(1) Dijkstra's Algorithm is a classical example of *greedy algorithm* which also resembles *breadth-first search*.

* *Generally speaking, greedy algorithm is an algorithm such that in every step, we regard the present result as the best until find a better one to replace it.*

(2) ***Running time***

Dijkstra's algorithm's running time depends on how to implement *min-priority queue*.

A. If we take advantage of the vertices numbered 1 to V, then the running time is $O(V^2)$

B. If the graph is *sufficiently sparse*(*the adjacent matrix have many many zero*), we implement the min-priority queue with min-heap, then the running time is $o(V^2 \log V)$

C. We can achieve running time of $o(V \log V + E)$ by using min-priority queue with *Fibonacci heap*.

* *Fibonacci heap and priority queue are both introduced briefly in my github repertory.* (http://github.com/markmakemate)

References

[1] Introduction to Algorithms(third edition), T. H.Cormen, *The MIT Press*, 2012.

[2] Algorithms(Fourth Edition), R.Sedgewick & K.Wayne, *Pearson Education*, 2011. .

[3] Data Structure and Algorithm Analysis in C (Second Edition), M.A.Weiss, *Addison Wesley Longman*, 1997