

Single-Source Shortest-Paths

The single-source shortest-paths problem is finding the shortest path from a source vertex s to all other vertices in a weighted graph.

If all nonnegative weights, then Dijkstra's alg.

If some neg. weights, then Bellman-Ford alg.

Algs. work for directed and undirected graphs.

Notation:

Let $w(u, v)$ be the weight of edge (u, v) .

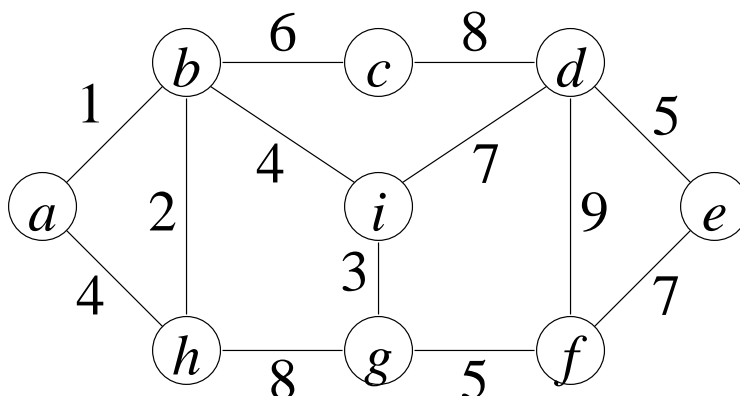
If p is a path, let $w(p) =$ sum of weights on p .

More Notation:

Let $\delta(s, v) =$ shortest distance from s to v .

Let $d[v] \geq \delta(s, v)$ with $d[s] = \delta(s, s) = 0$.

Let $s \overset{\text{opt}}{\rightsquigarrow} v$ stand for shortest path from s to v .



Key Properties

Triangle Inequality: If (u, v) is an edge, then $\delta(s, v) \leq \delta(s, u) + w(u, v)$. If p is a path from u to v , then $\delta(s, v) \leq \delta(s, u) + w(p)$.

Proof: $\delta(s, v) > \delta(s, u) + w(u, v)$ contradicts δ 's definition. So does $\delta(s, v) > \delta(s, u) + w(p)$.

Optimal Subpath Property: If u is on $s \overset{\text{opt}}{\rightsquigarrow} v$, then $\delta(s, v) = \delta(s, u) + \delta(u, v)$.

Proof: Let p_1 and p_2 be the subpaths from s to u and from u to v . $\delta(s, u) < w(p_1)$ or $\delta(u, v) < w(p_2)$ contradicts $\delta(s, v) = w(p_1) + w(p_2)$.

Convergence Property: If $s \rightsquigarrow u \rightarrow v$ is a shortest path, then $\delta(s, v) = \delta(s, u) + w(u, v)$.

Proof: Follows from the optimal subpath property.

Path-Relaxation Property: If $d[v] > \delta(s, v)$, then some edge (x, y) satisfies $d[x] = \delta(s, x)$ and $d[y] > d[x] + w(x, y)$.

Proof: Some edge (x, y) on $s \overset{\text{opt}}{\rightsquigarrow} v$ must be the first edge with $d[x] = \delta(s, x)$ and $d[y] > d[x] + w(x, y) = \delta(s, y)$.

These properties justify the following procedures.

INITIALIZE-SINGLE-SOURCE(G, s)

for each vertex v in G

do $d[v] \leftarrow \infty$

$\pi[v] \leftarrow \text{NIL}$

$d[s] \leftarrow 0$

Initializing to ∞ ensures that $d[v] \geq \delta(s, v)$ for all v .

$d[s] = 0$ ensures that $d[s] = \delta(s, s)$.

RELAX(u, v, w)

if $d[v] > d[u] + w(u, v)$

then $d[v] \leftarrow d[u] + w(u, v)$

 (optional) **DECREASE-KEY**($Q, v, d[v]$)

$\pi[v] \leftarrow u$

If $d[u] \geq \delta(s, u)$ and $d[v] \geq \delta(s, v)$ before **RELAX**, then they remain true afterwards.

Also, if $d[v] > \delta(s, v)$, then the Path Relaxation Property implies that some call to **RELAX** will improve d .

Dijkstra's Algorithm

This assumes weights are nonnegative.

DIJKSTRA(G, w, s)

 INITIALIZE-SINGLE-SOURCE(G, s)

$S \leftarrow \emptyset$

$Q \leftarrow$ vertices of G using $key[v] = d[v]$

while Q is not empty

do $u \leftarrow$ EXTRACT-MIN(Q)

$S \leftarrow S \cup \{u\}$

for each v adjacent from u

do RELAX(u, v, w)

Proof of Correctness:

Basis: s is assigned to u , $d[s] = 0 = \delta(s, s)$

Assume: For all $x \in S$, $d[x] = \delta(s, x)$

Show: $d[u] = \delta(s, u)$ when u is extracted.

Induction: Suppose $\delta(s, u) < d[u]$.

Some edge (x, y) in $s \overset{\text{opt}}{\rightsquigarrow} u$ goes from S to Q .

$d[y] = \delta(s, y)$ because (x, y) has been relaxed,

but $d[u] \leq d[y] = \delta(s, y) \leq \delta(s, u) < d[u]$

contradicts u being extracted instead of y .

DIJKSTRA is $O(E \lg V)$ with binary heaps.

Bellman-Ford Algorithm

This works with negative weights. FALSE is returned if a negative cycle is detected.

```

BELLMAN-FORD( $G, w, s$ )
  INITIALIZE-SINGLE-SOURCE( $G, s$ )
  for  $i \leftarrow 1$  to  $|V| - 1$ 
    do for each edge  $(u, v)$  in  $G$ 
      do RELAX( $u, v, w$ )
  for each edge  $(u, v)$  in  $G$ 
    do if  $d[v] > d[u] + w(u, v)$ 
      then return FALSE
  return TRUE

```

Proof of Correctness:

All finite shortest paths have $\leq V - 1$ edges. The i th iteration relaxes i th edge in the path. If shortest path is finite, then after $V - 1$ iterations, $d[v] = \delta(s, v)$.

If there is an infinite shortest path, then $d[v] > d[u] + w(u, v)$ for some edge (u, v) .

BELLMAN-FORD is $O(VE)$.

Difference Constraints

A difference constraint is $x_j - x_i \leq b_k$, where x_i and x_j are variables and b_k is a constant.

Difference constraints can represent time relationships between events. If x_2 must occur within 2 hours after x_1 , then $x_2 - 2 \leq x_1$. If x_2 must occur at least 2 hours after x_1 , then $x_2 - 2 \geq x_1$.

The triangle inequality for shortest paths is a difference constraint. $\delta(s, v) \leq \delta(s, u) + w(u, v)$ is equivalent to $\delta(s, v) - \delta(s, u) \leq w(u, v)$.

A set of difference equations $x_j - x_i \leq b_k$ can be *reduced* to a weighted graph by $w(v_i, v_j) = b_k$ and $w(s, v_j) = w(s, v_i) = 0$.

There is a solution for single-source shortest paths if and only if there is a solution to the difference equations.

All-Pairs Shortest Paths

The all-pairs shortest-paths problem is finding the shortest path for each pair of vertices. Algs. for this problem can be adapted for transitive closure.

Let $\delta(i, j, m)$ be the shortest length from i to j using $\leq m$ edges.

$$\delta(i, j, 1) = \begin{cases} 0 & \text{if } i = j \\ w(i, j) & \text{if } (i, j) \text{ is an edge} \\ \infty & \text{otherwise} \end{cases}$$

$$\delta(i, j, 2m) = \min_{k=1}^n (\delta(i, k, m) + \delta(k, j, m))$$

This recursive definition is correct.

Basis: Equation for $\delta(i, j, 1)$ is correct.

Assume: $\delta(i, j, m)$ is correct.

Show: Equation for $\delta(i, j, 2m)$ is correct.

Induction: Optimal path of $\leq 2m$ edges consists of two optimal paths of $\leq m$ edges.

All-Pairs Shortest Paths

ALL-PAIRS(G, w)

$n \leftarrow$ number of vertices in G

$D \leftarrow$ an $n \times n$ matrix initialized to $\delta(i, j, 1)$

$m \leftarrow 1$

while $m < n - 1$

do **EXTEND**(D)

$m \leftarrow 2m$

return D

EXTEND($D[1 \dots n, 1 \dots n]$)

for $i \leftarrow 1$ **to** n **do**

for $j \leftarrow 1$ **to** n **do**

for $k \leftarrow 1$ **to** n **do**

$D[i, j] \leftarrow \min(D[i, j], D[i, k] + D[k, j])$

ALL-PAIRS is $O(V^3 \lg V)$.

Floyd-Warshall Algorithm

FLOYD-WARSHALL(G, w)

$n \leftarrow$ number of vertices in G

$D \leftarrow$ an $n \times n$ matrix initialized to $\delta(i, j, 1)$

for $k \leftarrow 1$ **to** n **do**

for $i \leftarrow 1$ **to** n **do**

for $j \leftarrow 1$ **to** n **do**

$D[i, j] \leftarrow \min(D[i, j], D[i, k] + D[k, j])$

return D

Let $\lambda(i, j, k)$ be the shortest distance from i to j with all intermediate vertices $\leq k$.

Basis: $D[i, j] \leq \lambda(i, j, 0)$ due to initialization.

Assume: $D[i, j] \leq \lambda(i, j, k-1)$ before k th iteration.

Show: $D[i, j] \leq \lambda(i, j, k)$ after k th iteration.

Induction: Either $\lambda(i, j, k) = \lambda(i, j, k-1)$ or

$$\lambda(i, j, k) = \lambda(i, k, k-1) + \lambda(k, j, k-1)$$

FLOYD-WARSHALL is $O(V^3)$.

Johnson's Algorithm for Sparse Graphs

```

JOHNSON( $G, w, s$ )
  ( $G', w'$ )  $\leftarrow$  ( $G, w$ ) with new vertex  $s$ ,  $w'(s, v) = 0$ 
  if BELLMAN-FORD( $G', w', s$ ) = FALSE
    then return NULL
  for each edge  $(u, v)$  in  $G$ 
    do  $\widehat{w}(u, v) \leftarrow w(u, v) + \delta'(s, u) - \delta'(s, v)$ 
  for each vertex  $u$  in  $G$ 
    do DIJKSTRA( $G, \widehat{w}, u$ )
      for each vertex  $v$  in  $G$ 
        do  $D[u, v] \leftarrow \widehat{\delta}(u, v) - \delta'(s, u) + \delta'(s, v)$ 
  return  $D$ 

```

Triangle inequality, $\delta'(s, v) \leq \delta'(s, u) + w(u, v)$,
implies $0 \leq \delta'(s, u) + w(u, v) - \delta'(s, v)$.

Thus, $\widehat{w}(u, v) \geq 0$ for all edges (u, v) .

For any path p from u to v , $\widehat{w}(p) = w(p) + \delta'(s, u) - \delta'(s, v)$ because of the telescoping sum.

JOHNSON is $O(VE \lg V)$ if binary heaps are used for **DIJKSTRA**.