

Shortest Path Problem

- Given a directed graph $G = (V, E, w)$.
- Each edge $(u, v) \in E$ has associated real-valued weight $w(u, v)$.
- The weight $w(P)$ of a path P between vertices u and v in G is the sum of weights along P .
- Any path of minimum weight between u and v is called a shortest path from u to v .

Variants of Shortest Path Problem

- Single pair: shortest path from a given vertex u to a given vertex v .
- Single source: shortest paths from a single source vertex s to every vertex in G .
- Single target: shortest paths from every vertex in G to a single target vertex t .
- All pairs: shortest paths between all pairs of vertices in G .

Optimal Substructure Property

- Consider a shortest path P between vertices u and v in G .
- Let z be a vertex on P .
- The portion of P between u and z is a shortest path between u and z .
- The portion of P between z and v is a shortest path between z and v .

Cycles and Shortest Paths

- All edges have positive weights: Shortest paths cannot contain cycles.
- All edges have non-negative weights. 0-cycles can occur in shortest paths but can be removed without changing the length.
- Some edges have negative weights: If negative weight cycles are in the graph, shortest path problem is not well-defined. Otherwise, we are back in one of the above two cases.
- Conclusion: Shortest paths in graphs with n vertices have at most $n-1$ edges.

Triangle Inequality for Shortest Paths

- Claim: $\delta(s, v) \leq \delta(s, u) + w(u, v)$ for all $(u, v) \in E$
- Proof: The shortest path from s to u followed by the single edge path from u to v is a path from s to v .
- The length of this path must be at least equal to the length of the shortest path from s to v .

Shortest Path Algorithms - Intuition

- A path from s to every vertex v will be maintained.
 - $d[v]$ = weight of a path from s to v .
 - $p[v]$ = predecessor of v on a path from s to v .
- Initially
 - $d[s] = 0$,
 - $d[v] = \infty$ for all $v \neq s$,
 - $p[v] = \text{null}$ for all $v \in G$
- Suppose that an edge (u, v) exists such that
$$d[v] > d[u] + w(u, v)$$
- Relaxation: Update $p[v]$ and reduce $d[v]$. Repeat.

Initialization

- INITIALIZE-SINGLE-SOURCE(s)
 - $d[s] = 0$;
 - for each vertex v in $V - \{s\}$ do $d[v] = \infty$
 - for each vertex v in V do $p[v] = \text{null}$;

Relaxation

- RELAXATION(u, v)
 - if $d[v] > d[u] + w(u, v)$ then
 - $d[v] = d[u] + w(u, v)$
 - $p[v] = u$
 - else do nothing
- After relaxation attempt: $d[v] \leq d[u] + w(u, v)$

Algorithm

- INITIALIZE-SINGLE-SOURCE(s)
- RELAX as long as possible.

Upper Bound Property

- CLAIM: $d[v] \geq \delta(s, v)$ for all v in V at all times. Once $d[v] = \delta(s, v)$, $d[v]$ never changes.
- PROOF: By induction on the number of relaxations.
 - BASIS: Clearly true before the first relaxation.
 - INDUCTIVE STEP: Suppose that we are about to relax edge (u, v) . $d[u]$ will not change after the relaxation. By inductive hypothesis:
 - $d[v] = d[u] + w(u, v) \geq \delta(s, u) + w(u, v) \geq \delta(s, v)$
- NO-PATH PROPERTY: If $\delta(s, v) = \infty$ (no path from s to v) then $d[v] = \infty$. Follows directly from the upper bound property.

Convergence Property

- Let (u,v) be an edge in G .
 - Assume that the shortest path from s to v goes through (u,v) .
 - Assume that at some stage of the execution of the shortest path algorithm we have $d[u] = \delta(s,u)$, and we try to relax edge (u,v) .
 - After the relaxation attempt:
$$d[v] \leq d[u] + w(u,v) = \delta(s,u) + w(u,v) = \delta(s,v)$$
 - Upper bound property: $d[v] \geq \delta(s,v)$.

Bellman-Ford Algorithm

- INITIALIZE-SINGLE-SOURCE(G, s)
- for ($i = 1; i \leq n-1; i++$) do
 - for each edge $(u, v) \in G$ do relax (u, v) ;
- for each edge $(u, v) \in G$ do
 - if $d[v] > d[u] + w(u, v)$ then return FALSE;
- return TRUE;

- Worst case time complexity: $O(nm)$

Correctness of BF Algorithm

- Let $P = \{ v_0, v_1, \dots, v_k \}$ be a shortest path from $s=v_0$ to v_k .
- CLAIM: If G has no negative cycles then $d[v] = \delta(s,v)$ for all v after at most $n-1$ iterations.
- PROOF: Any path in G without cycles has at most $n-1$ edges. Therefore $k \leq n-1$.
 - Before the first round of relaxations, $d[v_0] = \delta(s, v_0)$. When (v_0, v_1) is relaxed in the first round, $d[v_1] = \delta(s, v_1)$.
 - Before the second round of relaxations, $d[v_1] = \delta(s, v_1)$. When (v_1, v_2) is relaxed in the second round, $d[v_2] = \delta(s, v_2)$.
 - Before the i -th round of relaxations, $d[v_{i-1}] = \delta(s, v_{i-1})$. When (v_{i-1}, v_i) is relaxed in the i -th round, $d[v_i] = \delta(s, v_i)$.

Correctness of BF Algorithm

- If G has no negative cycles then after $n-1$ iteration: $d[v] = \delta(s,v)$ for every $v \in G$.
- Let (u,v) be arbitrary edge in G .
 - $d[v] = \delta(s,v) \leq \delta(s,u) + w(u,v) = d[u] + w(u,v)$
- Bellman-Ford returns TRUE.

Correctness of BF Algorithm

- Suppose that G contains a negative cycle reachable from s .
- Assume for the purpose of contradiction that BF returns TRUE.
- Let $C = \{ v_0, v_1, \dots, v_k \}$, $v_0 = v_k$, be a negative cycle.
 - Since BF is assumed to return TRUE, we must have
 - $d[v_i] \leq d[v_{i-1}] + w(v_{i-1}, v_i)$, $i = 1, 2, \dots, k$.
 - Summing these inequalities around the cycle, we get

$$\sum_{i=1}^k d[v_i] \leq \sum_{i=1}^k (d[v_{i-1}] + w(v_{i-1}, v_i))$$

This leads to a contradiction since we get

$$0 \leq \sum_{i=1}^k w(v_{i-1}, v_i)$$

Shortest Paths in DAGs

- Topologically sort vertices of the DAG.
- Process vertices in topological order. For each vertex, relax its outgoing edges.

Correctness of the Shortest Paths Algorithm for DAGs

- If v is not reachable from s , then the algorithm returns $d[v] = \infty$ (no edge into v is ever relaxed).
- If v is reachable from s , consider the shortest path $\{v_0, v_1, \dots, v_k\}$ with $v_0 = s$ and $v_k = v$.
- The edges of this path: $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$ are relaxed in this order since the vertices are processed in topological order.
- As a consequence, shortest paths in a DAG are computed after one relaxation pass of all its edges.
- Worst-case time complexity of the algorithm is $O(m+n)$.

Dijkstra's Algorithm

- Non-negative edge weights.
- Let S be a set of vertices whose shortest paths from s have been determined. Initially $S = \emptyset$, $d[s] = 0$, $d[u] = \infty$ for all $u \neq s$.
- At each iteration, select a vertex u with the smallest $d[u]$ and not in S .
 - Relax along all edges going out from u .
 - Add u to S .
- How to implement Dijkstra's algorithm?

Correctness of Dijkstra's Algorithm

- We need to show that when a vertex u is added to S , then $d[u] = \delta(s, u)$.
- For the purpose of contradiction, let u be the first vertex added to S for which $d[u] > \delta(s, u)$. Note that $u \neq s$.
- Let P be the shortest path from s to this u .
- Let x be the last vertex on P that is in S . Note that it is possible that $x = s$.
- Let (x, y) be the edge on P . Note that it is possible that $y = u$.
- When x was added to S , the edge (x, y) was relaxed, Therefore
$$d[y] = \delta(s, x) + w(x, y) = \delta(s, y) \leq \delta(s, u) < d[u]$$
- Neither u nor y are in S . Since u was added to S , we have $d[u] \leq d[y]$, a contradiction.

All-Pairs Shortest Path Problem

- Find shortest paths between all pairs of vertices.
- Focus on finding minimum weights of paths. Shortest paths itself can be reconstructed if an appropriate predecessor matrix is used.
- Recall optimal substructure property of shortest paths.

Shortest Paths and Matrix Multiplication

- Enumerate the vertices from 1 to n .
- d_{ij}^m = weight of the shortest path from i to j with at most m edges, $m = 1, 2, \dots, n-1$.
- $d_{ij}^1 = w(i,j)$
- $d_{ij}^m = \min \{ d_{ij}^{m-1}, \min_{1 \leq k \leq n} \{ d_{ik}^{m-1} + w(k,j) \} \} =$
 $\min_{1 \leq k \leq n} \{ d_{ik}^{m-1} + w(k,j) \}$
- If you know how to multiply matrices, you will see that matrix D_m can be obtained by matrix "multiplication" of D_{m-1} with W
- Since $D_1 = W$, it follows that $D_m = W^m$

Floyd-Warshall Algorithm

- Enumerate the vertices from 1 to n .
- d_{ij}^k = weight of the shortest path from i to j with intermediate vertices (if any) having indices less than or equal to k .
- $d_{ij}^0 = w(i,j)$
- $d_{ij}^k = \min \{ d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1} \}$