Finding shortest paths in a graph

- Input: a directed graph $G = (V, E)$, with nonnegative cost $c_e \geq 0$ for each edge $e \in E$. A node $s \in V$.
- Output: for each node $v \in V$, a minimum-cost path from s to v.

Finding shortest paths in a graph

- Input: a directed graph $G = (V, E)$, with nonnegative cost $c_e \geq 0$ for each edge $e \in E$. A node $s \in V$.
- Output: for each node $v \in V$, a minimum-cost path from s to v.
- Dijkstra's algorithm: a greedy approach

Finding shortest paths in a graph

- Input: a directed graph $G = (V, E)$, with nonnegative cost $c_e \geq 0$ for each edge $e \in E$. A node $s \in V$.
- Output: for each node $v \in V$, a minimum-cost path from s to v.
- Dijkstra's algorithm: a greedy approach
- Initialize: for each $v \in V$, if $(s, v) \in E$, let $d(v) \leftarrow c_{(s,v)}$, $p(v) \leftarrow s$, otherwise $d(v) \leftarrow \infty$, $p(v) \leftarrow \perp$. Let S be $\{s\}$.

Finding shortest paths in a graph

- Input: a directed graph $G = (V, E)$, with nonnegative cost $c_e \geq 0$ for each edge $e \in E$. A node $s \in V$.
- Output: for each node $v \in V$, a minimum-cost path from s to v.
- Dijkstra's algorithm: a greedy approach
- Initialize: for each $v \in V$, if $(s, v) \in E$, let $d(v) \leftarrow c_{(s,v)}$, $p(v) \leftarrow s$, otherwise $d(v) \leftarrow \infty$, $p(v) \leftarrow \perp$. Let S be $\{s\}$.
- Iterate: while $S \neq V$ and there exists $v \in V \setminus S$ such that $d(v) \neq \infty$:
  - let u be the minimizer of $d(u)$ among nodes not in S;
  - add u to S
  - for each $v \in V \setminus S$ such that $(u, v) \in E$, if $d(v) > d(u) + c_{(u,v)}$, update $d(v) \leftarrow d(u) + c_{(u,v)}$ and $p(v) \leftarrow u$.
- Output: for each $v \in S$, trace the path back to s using $p(\cdot)$.

Proof of Correctness

- Proof by induction.

Proof of Correctness

- Proof by induction.
- Induction hypothesis: at each stage of the algorithm, for any node $u \in S$, $d(u)$ is the cost of the minimum-cost path from s to u.

Proof of Correctness

- Proof by induction.
- Induction hypothesis: at each stage of the algorithm, for any node $u \in S$, $d(u)$ is the cost of the minimum-cost path from s to u.
- Base case when $S = \{s\}$ is trivial.

Proof of Correctness

- Proof by induction.
- Induction hypothesis: at each stage of the algorithm, for any node $u \in S$, $d(u)$ is the cost of the minimum-cost path from s to u.
- Base case when $S = \{s\}$ is trivial.
- (Introduce notation) Denote by P_v the path output by the algorithm for node v.

Proof of Correctness

- Proof by induction.
- Induction hypothesis: at each stage of the algorithm, for any node $u \in S$, $d(u)$ is the cost of the minimum-cost path from s to u.
- Base case when $S = \{s\}$ is trivial.
- (Introduce notation) Denote by P_v the path output by the algorithm for node v.
- When a node u is added to S with $p(u) = v$, show:
  - Among all paths within S, P_u has the minimum cost.

Proof of Correctness

- Proof by induction.
- Induction hypothesis: at each stage of the algorithm, for any node $u \in S$, $d(u)$ is the cost of the minimum-cost path from s to u.
- Base case when $S = \{s\}$ is trivial.
- (Introduce notation) Denote by P_v the path output by the algorithm for node v.
- When a node u is added to S with $p(u) = v$, show:
  - Among all paths within S, P_u has the minimum cost.
  - P_u has no more cost than any path that leaves S at some point.

Proof of Correctness

- Proof by induction.
- Induction hypothesis: at each stage of the algorithm, for any node $u \in S$, $d(u)$ is the cost of the minimum-cost path from s to u.
- Base case when $S = \{s\}$ is trivial.
- (Introduce notation) Denote by P_v the path output by the algorithm for node v.
- When a node u is added to S with $p(u) = v$, show:
  - Among all paths within S, P_u has the minimum cost.
  - P_u has no more cost than any path that leaves S at some point.
- Where did we use the condition $c_e \geq 0$?

Dealing with negative costs

- Input: a directed graph $G = (V, E)$, with cost $c_e \geq 0$ for each edge $e \in E$, and no negative cycle. A node $s \in V$.
- Output: for each node $t \in V$, the cost of a minimum-cost path from s to t.

Dealing with negative costs

- Input: a directed graph $G = (V, E)$, with cost $c_e \geq 0$ for each edge $e \in E$, and no negative cycle. A node $s \in V$.
- Output: for each node $t \in V$, the cost of a minimum-cost path from s to t.
- What goes wrong when there is a negative cycle?

Dealing with negative costs

- Input: a directed graph $G = (V, E)$, with cost $c_e \geq 0$ for each edge $e \in E$, and no negative cycle. A node $s \in V$.
- Output: for each node $t \in V$, the cost of a minimum-cost path from s to t.
- What goes wrong when there is a negative cycle?
- Bellman-Ford Algorithm:

Dealing with negative costs

- Input: a directed graph $G = (V, E)$, with cost $c_e \geq 0$ for each edge $e \in E$, and no negative cycle. A node $s \in V$.
- Output: for each node $t \in V$, the cost of a minimum-cost path from s to t.
- What goes wrong when there is a negative cycle?
- Bellman-Ford Algorithm:
  - Initialize: for each $v \in V$, if $(s, v) \in E$, let $d(v) \leftarrow c_{(s,v)}$, otherwise $d(v) \leftarrow \infty$.

Dealing with negative costs

- Input: a directed graph $G = (V, E)$, with cost $c_e \geq 0$ for each edge $e \in E$, and no negative cycle. A node $s \in V$.
- Output: for each node $t \in V$, the cost of a minimum-cost path from s to t.
- What goes wrong when there is a negative cycle?
- Bellman-Ford Algorithm:
  - Initialize: for each $v \in V$, if $(s, v) \in E$, let $d(v) \leftarrow c_{(s,v)}$, otherwise $d(v) \leftarrow \infty$.
  - Iterate: for each node $v \in V$, for each u such that $(u, v) \in E$, $d(v) \leftarrow \min\{d(v), d(u) + c_{(u,v)}\}$. If no update occurs in an iteration, terminate.

Dealing with negative costs

- Input: a directed graph $G = (V, E)$, with cost $c_e \geq 0$ for each edge $e \in E$, and no negative cycle. A node $s \in V$.
- Output: for each node $t \in V$, the cost of a minimum-cost path from s to t.
- What goes wrong when there is a negative cycle?
- Bellman-Ford Algorithm:
  - Initialize: for each $v \in V$, if $(s, v) \in E$, let $d(v) \leftarrow c_{(s,v)}$, otherwise $d(v) \leftarrow \infty$.
  - Iterate: for each node $v \in V$, for each u such that $(u, v) \in E$, $d(v) \leftarrow \min\{d(v), d(u) + c_{(u,v)}\}$. If no update occurs in an iteration, terminate.
  - If the program does not terminate after $n - 1$ rounds, report error (a negative cycle is found).

Dealing with negative costs

- Input: a directed graph $G = (V, E)$, with cost $c_e \geq 0$ for each edge $e \in E$, and no negative cycle. A node $s \in V$.
- Output: for each node $t \in V$, the cost of a minimum-cost path from s to t.
- What goes wrong when there is a negative cycle?
- Bellman-Ford Algorithm:
  - Initialize: for each $v \in V$, if $(s, v) \in E$, let $d(v) \leftarrow c_{(s,v)}$, otherwise $d(v) \leftarrow \infty$.
  - Iterate: for each node $v \in V$, for each u such that $(u, v) \in E$, $d(v) \leftarrow \min\{d(v), d(u) + c_{(u,v)}\}$. If no update occurs in an iteration, terminate.
  - If the program does not terminate after $n - 1$ rounds, report error (a negative cycle is found).
  - Output $d(v)$ for each $v \in V$.

Proof of Correctness

Lemma

In a graph containing no negative cycles, between any two nodes there is a minimum-cost path consisting of at most $n - 1$ edges.

Proof of Correctness

Lemma

In a graph containing no negative cycles, between any two nodes there is a minimum-cost path consisting of at most $n - 1$ edges.

Proof.

Take any minimum-cost path from s to t. If the path passes a node v twice, the part of the path between these is a cycle. Removing this cycle will not increase the total cost. Repeat this procedure until the path passes each node at most once. $\square$

Proof of Correctness

- By induction.

Proof of Correctness

- By induction.
- (Introduce additional notation for clarity:) let $d_i(v)$ be the value of $d(v)$ after the i-th iteration of the algorithm.

Proof of Correctness

- By induction.
- (Introduce additional notation for clarity:) let $d_i(v)$ be the value of $d(v)$ after the i-th iteration of the algorithm.
- Induction hypothesis: for each $v \in V$, $d_i(v)$ is no larger than the cost of a minimum-cost path from s to v using at most $i + 1$ edges.

Proof of Correctness

- By induction.
- (Introduce additional notation for clarity:) let $d_i(v)$ be the value of $d(v)$ after the i-th iteration of the algorithm.
- Induction hypothesis: for each $v \in V$, $d_i(v)$ is no larger than the cost of a minimum-cost path from s to v using at most $i + 1$ edges.
- For all i, $d_i(v)$ is the cost of an actual path from s to v.

Proof of Correctness

- By induction.
- (Introduce additional notation for clarity:) let $d_i(v)$ be the value of $d(v)$ after the i-th iteration of the algorithm.
- Induction hypothesis: for each $v \in V$, $d_i(v)$ is no larger than the cost of a minimum-cost path from s to v using at most $i+1$ edges.
- For all i, $d_i(v)$ is the cost of an actual path from s to v.
- Combining the lemma, when there is no negative cycle, after at most $n-1$ steps, $d(v)$ is the cost of a minimum-cost path.

Proof of Correctness

- By induction.
- (Introduce additional notation for clarity:) let $d_i(v)$ be the value of $d(v)$ after the i-th iteration of the algorithm.
- Induction hypothesis: for each $v \in V$, $d_i(v)$ is no larger than the cost of a minimum-cost path from s to v using at most $i + 1$ edges.
- For all i, $d_i(v)$ is the cost of an actual path from s to v.
- Combining the lemma, when there is no negative cycle, after at most $n - 1$ steps, $d(v)$ is the cost of a minimum-cost path.
- Running time: $O(mn)$, i.e., $O(|V| \cdot |E|)$.

Implementation of Dijkstra

- Implementation of Dijkstra using priority queue

Implementation of Dijkstra

- Implementation of Dijkstra using priority queue
- Running time: $O(m \log n)$.