Overview

- Relaxation
  - Properties of Shortest Paths and Relaxation
  - Bellman-Ford
  - Single-source Shortest Paths in DAG

Arbitrage (Question 24-3)

Arbitrage is the use of discrepancies in currency exchange rates to turn a profit.

Arbitrage is the use of discrepancies in currency exchange rates to turn a profit.

Arbitrage (Question 24-3)
In single-source algorithms, a path $p$'s $\pi$ only revised by relax function.

\begin{align*}
\text{Initialize Single-Source}(G, s) & : \\
\forall v' \in V, \pi(v') & = a, \\
\pi(s) & = 0, \\
L & = \infty, \\
\text{for each vertex } & a \in V \in G.
\end{align*}

- $p$.\pi$ is a lower bound on the weight of a shortest path from source $s$ to $a$.

Strategy: Start with an upper bound and keep revising it when you find a lower cost.

\begin{align*}
0 = p'.s \\
\infty = p'.a \\
L & = \pi(v) \\
\text{for each vertex } & a \in V \in G.
\end{align*}
Properties of Shortest Paths and Relaxation

- Triangle Inequality
  - For any edge \((u, v) \in E\), \(\delta(s, v) \leq \delta(s, u) + w(u, v)\)

- Upper-bound Property
  - \(v.d \geq \delta(s, v)\) for all \(v \in V\).
  - \(v.d\) only decreases in value

- No-path property
  - If there is no path from \(s\) to \(v\) then we always have \(v.d = \delta(s, v) = \infty\)

- Convergence Property
  - If \(s \rightarrow u \rightarrow v\) is a shortest path in \(G\) for some \(u, v \in V\) and if \(u.d = \delta(s, u)\) at any time prior to relaxing edge \((u, v)\)
  - then \(v.d = \delta(s, v)\)

Overview

- Relaxation
- Properties of Shortest Paths and Relaxation
- Dijkstra's Algorithm
- Single-source Shortest Paths in DAG
- Bellman-Ford
Housekeeping

- Once we know if edge exists, we know its weight in \(O(1)\) time.
- Weights stored in adjacency-list representation.
- Graph stored in adjacency-list representation.

\[
\begin{align*}
\infty + a &= \infty + a = \infty \quad \text{and} \\
\infty - a &= \infty - a = \infty
\end{align*}
\]

- Let \(a\) be a real number (so \(a \neq -\infty, \infty\))
- Arithmetic with infinity

Graph stored in adjacency-list representation

- Weights stored in adjacency-list.
- Once we know if edge exists, we know its weight in \(O(1)\) time.

Continued

- Path-relaxation Property
  - If \(p = \langle v_0, v_1, \ldots, v_k \rangle\) is a shortest path from \(s = v_0\) to \(v_k\) and we relax the edges of \(p\) in the order \((v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)\) then \(v_k.d = \delta(s, v_k)\). This holds even if other relaxation steps occur.
- Predecessor-subgraph property
  - Remember, there is a negative cycle, there is no shortest path.
  - This holds even if other relaxation steps occur.
  - Then we relax the edges of the order \(d\) in the order \(v_0, v_1, \ldots, v_{k-1}\) to get \(\langle v_0, v_1, \ldots, v_k \rangle = d\) if \(v_0 = s\), and \(v_0 = \infty\) otherwise.

Path-relaxation Property

Comparison

Comparison

Comparison
Bellman-Ford Algorithm

Bellman-Ford Algorithm

Overview

• Relaxation
• Properties of Shortest Paths and Relaxation

⇒ Bellman-Ford

• Single-source Shortest Paths in DAG

Bellman-Ford ⇐ Properties of Shortest Paths and Relaxation

• Relaxation

Dijkstra's Algorithm

Bellman-Ford Algorithm

Bellman-Ford Algorithm

Single-source Shortest Paths in DAG

Bellman-Ford ⇐ Properties of Shortest Paths and Relaxation

Relaxation

Bellman-Ford Algorithm

Bellman-Ford Algorithm

Single-source Shortest Paths in DAG

Bellman-Ford ⇐ Properties of Shortest Paths and Relaxation

Relaxation

Bellman-Ford Algorithm

Bellman-Ford Algorithm

Single-source Shortest Paths in DAG

Bellman-Ford ⇐ Properties of Shortest Paths and Relaxation

Relaxation
**Time Complexity**

- **How it Works**
  - Progressively decreases the estimate $v.d$
  - After $V - 1$ rounds, each vertex will reach its minimum value
  - Can work with negative edges
    - If it finds a negative cycle, returns False
    - Knows if there is a negative edge, if $v.d$ can be further reduced
  - Used in Reinforcement Learning
    - To update estimates of what sequence of actions to perform to finish a task
    - Can work with negative edges
    - After $V - 1$ rounds, each vertex will reach its minimum value
    - Progressively decreases the estimate $v.d$
Overview

- Dijkstra's Algorithm
- Bellman-Ford
- Relaxation

Properties of Shortest Paths and Relaxation

Example

Diagram showing a network with nodes and edges labeled with weights.
Faster with a DAG!

For a DAG can do this in $\Theta(V + E)$ time.

- Can be negative edges, but no cycles, so no negative weight cycles.
- Start with a topological sort of edges $(u, v)$ means $u$ precedes $v$.
- This will allow path relaxation property to be more efficient.

**For a DAG can do this in $\Theta(V + E)$ time**.
Proof:

How to we phrase the theorem?

Proof of Correctness (Theorem 24.5)

Code and Example

Running Time?
Overview

- Relaxation
- Properties of Shortest Paths and Relaxation
- Bellman-Ford
- Single-source Shortest Paths in DAG
- Dijkstra's Algorithm

Summary So Far

- Weighted directed graph
- Relaxation procedure
- Basis of all algorithms
- Bellman-Ford
- Can work with graphs with cycles and negative edges
- DAG shortest path
- Restricted to DAGs: much faster!
- Time: $\Theta(V + E)$

- Can detect negative cycles
- Time: $\Theta(V^2 E)$
Dijkstra's Algorithm

- Weight directed graph with no negative edges
  - Can have cycles, but no negative weight cycles

Breadth-first search

- Orders vertices by distance from source
- Can have cycles, but no negative weight cycles

Weighted directed graph with no negative edges

When are keys of min priority queue being updated?
- Lowered or increased?

Where is this like best-first? Where is the frontier?

What is the role of $S$?

Where are keys of min priority queue being updated?

```
RELAX(G, u)

for each vertex $v \in V - G \setminus \{n\} 
  \{n\} \cup S = S
  \{n\} \setminus S = \emptyset

while $\emptyset \neq \{n\}$
  $G = \{n\}$
  $S = \emptyset$

INITIALIZE-SINGLE-SOURCE(G, S)

DIJKSTRA(G, u)
```

How is this like best-first? Where is the frontier?

- Lowered or increased?

What are keys of min priority queue being updated?

Code
Continued

When \( x \) is added to \( S \), edge \((x, y)\) is relaxed, and so \( y.d \) is set to \( \delta(s,u) \) by convergence property.

But \( u \) is chosen, so \( u \) has minimum weight.

So the upper bound property is satisfied, since \( u \) is on a shortest path from \( s \) to \( u \).

When \( u \) is chosen, \( u \) has minimum weight.

But \( y.d \) is set to \( \delta(s,u) \), so \( p' \) is chosen over \( p \). By convergence property, \( \delta(s,y) \leq \delta(s,u) \) since \( y \) is on \( u \)'s shortest path.

By the upper bound property, \( u.d \geq \delta(s,u) \).

So \( u.d = \delta(s,u) \).

Contradiction.

Proof of Correctness

Claim: when we take a vertex \( v \) out of min priority queue (and into \( S \)), it will have its final weight: \( \delta(s,v) \).

Proof by Contradiction:

Let \( u \) be first vertex taken out of min priority queue (and into \( S \)).

In \( S \), the sum of \( u \) is correct (and into \( S \)).

Proof by Contradiction:

\( (n',s) \notin p \cdot n \).

By Lemma 24.1, subpath of \( p \cdot d \) from \( s \) to \( n \) is a shortest path.

Let be a shortest path from \( s \) to \( n \).

Proof by Contradiction:

\( (a',s) \notin p' \cdot n \) because no path property tells us that \( n \) would be \( \infty \).

Claim: when we take a vertex \( v \) out of min priority queue (and into \( S \)), it will have its final weight: \( \delta(s,v) \).
Greedy Algorithm

Timing
Summary

- All work on weighted directed graphs
- Relaxation procedure: Basis of all algorithms
  - Bellman-Ford
    + Can work with graphs with negative weight cycles
    + Can detect negative cycles
    + Time: $\Theta(VE)$
  - DAG Shortest Path
    + Dijkstra's algorithm
      + No negative edges, can have 0 weight edges
      + Can have cycles, but no negative weight cycles
      + Time: $\Theta(E \log V)$ (assuming all vertices are reachable)
    + Bellman-Ford
      + Can work with graphs with cycles and negative edges
      + All work on weighted directed graphs