Shortest Paths

Recall BFS:

- **BFS** finds shortest paths (from node $s$) in **unweighted** graphs.

Today: **Weighted** graphs.

**Starting point:**

- A recursive description of problem:
  - \( \text{dist}(u) \) be shortest path from $s$ to $u$.

- \( \text{dist}(u) = \min \{ \text{dist}(v) + w(v,u) \} \), where $v \in \text{in}(u)$.

- **Recursive** but not a **recurrence** (due to smaller path lengths).

- **Negative weight?**

- **Cycles:**
  - Special case: **DAG**.
    - Original recursive definition is a **recurrence relation**.
      - \( \text{dist}(u) \) depends on \( \text{dist}(v) \), where \( v \in \text{in}(u) \).
    - DAGs have a topological order \( \rightarrow \).

- **Bellman-Ford**：“hop” counter:

  - \( \text{dist}(i, u) \): distance of $i$ to $u$ taking at most $i$ hops ($G$ is unreachable in $i$ hops).

  - \( \text{dist}(s, a) \): distance of $s$ to $a$.

  - \( \text{dist}(a, b) \): distance of $a$ to $b$.

  - \( \text{dist}(s, b) \): distance of $s$ to $b$.

  - \( \text{dist}(s, a) = 6 \) if $a$ is unreachable in $i$ hops.
\[
\text{dist}(i, v) = \begin{cases} 
0 & \text{if } v = i, \text{ or else} \\
\min\left\{ \min_{w \in \text{in}(v)} \text{dist}(i-1, w) + \text{dist}(w, v) \right\} 
\end{cases}
\]

Lemma: If there is a cycle, \(s \xrightarrow{a} u \xrightarrow{a} u \) then there is a shorter (or equal) path.

\[
\text{dist}(i, v) = \begin{cases} 
0 & \text{if } v = i, \text{ or else} \\
\min\left\{ \min_{w \in \text{in}(v)} \text{dist}(i-1, w) + \text{dist}(w, v) \right\} 
\end{cases}
\]

Lemma: Any path from \(s \xrightarrow{a} u\), with no cycles, takes \(n\) or fewer hops (regardless of \(n\)).

Proof: pigeonhole.

Dijkstra: Assume no negative edges.

Performance vs. B-F

Main idea: Find the shortest paths for each node, in increasing order by shortest path length.
Lemma: The node to $s$ is exactly one hop away (adjacent) to one of the closest nodes.

Proof: Suppose $s$, $u_i$, and $u_{i+1}$ is the shortest path from $s$ to $u_{i+1}$ and $x$ is the closest node to $u_{i+1}$.

Let $u_1, u_2, \ldots, u_{i+1}$ be the closest nodes to $s$.

Rewrite: $\text{dist}(i) = \text{distance to } i^{th} \text{ closest node}$.

$$\text{dist}(i) = \min_{0 \leq j \leq i} \left( \min_{u} \left( \text{dist}(j) + w(u, v) \right) \right)$$

$u_i$ is $i^{th}$ closest node.

Running time is $n \times (n \times m)$.

First approach:
- Store all $(\text{dist}(j) + w(u, v))$ values in a BST.

For $T$ elements in the data stream:
- In each iteration, take the minimum $(\log T)$.
- \( u_1 \rightarrow v \rightarrow u_i \rightarrow u_{i+1} \)
- removing all \( u_i \) \( u_{i+1} \)
- and add \( u \rightarrow v \rightarrow u_{i+1} \) \( O(m \log n) \)
- \( O(n + m \log m) \)

**Second:** only store one edge for each dist

**Third:** Amortized data structure (better overall performance)

- (Fibonacci heap) even if each insert/delete is \( O(log n) \) worst case is same

Reach \( O(M + n \log n) \) when no neg. edges.

---

**Shortest Path Tree**

**Lemma:** There is a tree rooted at \( S \) that contains a shortest path from

- \( S \) to each node \( u \),
- only one parent
- no cycles.

**Proof:**

Suppose for contradiction:

\( S \rightarrow u \rightarrow v \rightarrow a \rightarrow \) some path

\( \leq b \) but, \( w \)

Suppose \( S \rightarrow u \rightarrow x \rightarrow v \rightarrow a \)
Log, suppose $w_1 \leq w_2$.

Then $\leq w_1 \rightarrow x \rightarrow b$ is a shorter core path to $b$. $w_1 + w \leq w_2 + w$.

Because of this, can represent paths with a point array

**Parent $E_i$** is point or node in the shortest path tree.

**Negative Cycle detection.**

\[ S \rightarrow a \rightarrow b \rightarrow c \]
\[ \text{BF: } \text{dist}(i,V) \text{ SP to } V \text{ taking } i \text{ or some edges} \]

If in no cycles, then $\exists i \leq n$. 

**Lemma:** $\text{dist}(i, V) \leq \text{dist}(n+1, V)$

**Then there is a neg. cycle**
Case I

\[
\frac{w}{m} \overline{u} \\
\frac{w}{m} \overline{u} \quad n+1 \text{ or less}
\]

\[
\frac{w}{m} \overline{u} \quad \text{then } \frac{w}{m} \overline{u} \text{ has } n + 1 \text{ hops.}
\]

By pigeonhole, has a cycle.

By each-lemma, possible cycles are now in \( S \).

So this cycle must be removed.

Need to show this can happen.

Case II

\[
\text{dist}(i+1, u) \leq n \\
\text{depends on } \text{non}
\]

\[
\text{dist}(i, u) \leq n \\
\Rightarrow \text{dist}(i+1, u) = \text{dist}(i, u)
\]

\[
\Rightarrow \text{dist}(i+2, u) = \text{dist}(i+1, u)
\]