Shortest Paths in Weighted Graphs

**Recall:** Last time we showed that BFS finds shortest paths in unweighted graphs.

**Today:**
1. Dijkstra’s Algorithm: Positive Weights
2. Bellman-Ford Algorithm: Arbitrary Weights
3. Detecting Negative Cycles
4. Shortest Paths in DAGs

1. \( G = (V, E) \)

\[ L : E \rightarrow \mathbb{R} \]

**Example:** Find shortest paths from A.

![Graph](image.png)
k - the set of vertices for which we know the shortest path.

How to extend k?
Dijkstra(G, s):

- dist[s] = 0
- \forall v \neq s \quad \text{dist}[v] = \infty.

- U = V \quad \text{Initialize a priority queue } Q \leftarrow V \text{ with keys } = \text{dist}.

- \text{while } u \neq \emptyset:
  - u \leftarrow \text{DeleteMin}(Q)

\text{update} \left\{\begin{array}{l}
\text{for } (u,v) \in E: \\
\quad \text{dist}[v] = \min(\text{dist}[v], \text{dist}[u] + l(u,v)).
\end{array}\right.

\text{DecreaseKey}(Q, v, \text{dist}[v]).

\text{How to implement?}

- Binary Heap \leq \text{DecreaseKey Insert} \text{ O}(\log |V|).
- Fibonacci Heap
Running Time

# of Operations:

- Make Queue: once. $O(|V|)$ or $O(|V|\log |V|)$.
- Delete Min: $|V|$. complicated. $O(|V|\log |V|)$.
- Decrease Key: at most $|E|$ times.

Overall Runtime: $O((|V|+|E|)\log |V|)$.

$O(|V|\log |V| + |E|)$ using Binary Heap.

$O(|V|\log |V|)$ using Fibonacci Heaps.
Claim: At any point in time, \( \forall v \in V \quad \text{dist}[v] = d(s, v). \)

Proof: By Induction.

Base case: Trivial.

In the second step \( k = \sum s^2 \) trivial.

Step: Let \( v \) be the vertex with smallest dist number. We show that \( \text{dist}[v] = d(s, v). \)

\[ \text{s} \in K \rightarrow \text{a} \rightarrow \text{k} \rightarrow \text{b} \rightarrow \text{v} \]

\[ \forall (a, v) \in E \]

\[ \text{dist}[v'] = \min(\text{dist}[v], \text{dist}[a] + l(a, v)) \]

\[ \text{dist}[v'] \leq \text{dist}[a] + l(a, v). \]

\[ \text{dist}[a] = d(s, a) \]

\[ \text{dist}[b] \leq \text{dist}[a] + l(a, b) \]

\[ \leq d(s, a) + l(a, b) \]

\[ = d(s, b). \]

\[ b \Rightarrow \text{dist}[b] < \text{dist}[v] \quad \text{contradiction}. \]
Dijkstra's tree.
Shortest Path with Arbitrary Lengths

Update $(u,v)$:

$$\text{update}(u,v): \quad \text{dist}[v] = \min(\text{dist}[v], \text{dist}[u] + l(u,v))$$

1. Update is Safe. \(\forall v: \text{dist}[v] \geq d(s,v)\).
2. If shortest path from $s$ to $v$ looks like that:

\[ s \rightarrow \cdots \rightarrow u \rightarrow v \]

$\text{dist}[u]$ is correct
$\text{update}(u,v)$ is correct.
Bellman Ford:

For $i = 1, \ldots, |V| - 1$:
- Update all edges (for all $(u, v) \in E$, update $(u, v)$)

Running Time: $O(|V| \cdot |E|)$ steps.
Shortest Paths in DAGs.

can we find shortest paths in DAGs using Bellman-Ford faster?

1. Find topological order on G.
2. For all edges (sorted according to the topological order)
   update(c_{uv}).

O(|V|+|E|)

Detect Negative Cycles?

No negative cycles \Rightarrow running Bellman-Ford for one more iteration would not change any dist.
Assume that in the last iteration there were no updates.

- \( \text{dist}[a_1] \leq \text{dist}[a_k] + l(a_i, a_j) \)
- \( \text{dist}[a_2] \leq \text{dist}[a_1] + l(a_i, a_j) \)
- \( \text{dist}[a_3] \leq \text{dist}[a_2] + l(a_i, a_j) \)
- \( \vdots \)
- \( \text{dist}[a_k] \leq \text{dist}[a_{k-1}] + l(a_i, a_j) \)

\[ \text{dist}[a_i] + \text{dist}[a_j] + \cdots + \text{dist}[a_k] \]

\[ + l(a_i, a_j) + l(a_i, a_j) + \cdots + l(a_i, a_j) \]

\( \Rightarrow \) every cycle is non-negative.