Lecture 18: Shortest Paths Revisited

- Find shortest path weight from s to v for all $v \in V$
- Observation: Subsets of shortest paths are shortest paths
- Try to find a recursive solution in terms of subproblems!
- Attempt 1:
  1. **Subproblems**
     - Try $\boxed{x(v) = \delta(s, v), \text{ shortest path weight from } s \text{ to } v}$
  2. **Relate**
     - $x(v) = \min\{x(u) + w(u, v) \mid (u, v) \in E\}$
     - Dependency graph is the transpose of the original graph (all edge directions reversed)!
     - If graph had cycles, subproblem dependencies can have cycles... :(
     - For now, **assume graph is acyclic**
     - Then we can compute subproblems in a topological sort order!
  3. **Base**
     - $x(s) = \delta(s, s) = 0$
     - $x(v) = \infty$ for any other $v \neq s$ without incoming edges
  4. **Solution**
     - Compute subproblems via top-down memoized recursion or bottom-up
     - Solution is subproblem $x(v)$, for each $v \in V$
     - Can keep track of parent pointers to subproblem that minimized recurrence
  5. **Time**
     - # subproblems: $|V|$
     - Work for subproblem $x(v)$: $O(1 + \deg_{\text{in}}(v))$

$$\sum_{v \in V} O(1 + \deg_{\text{in}}(v)) = O(|V| + |E|)$$

- This is just DAG Relaxation! What if graph contains cycles? Next time!

Single Source Shortest Paths Revisited (Attempt 2)

1. **Subproblems**

   - Increase subproblems to add information to make acyclic!
   - $\boxed{x(v, k)\text{, minimum weight of any edge path from } s \text{ to } v \in V \text{ using } \textit{at most } k \text{ edges}}$

2. **Relate**

   - $x(v, k) = \min\{\{x(u, k-1) + w(u, v) \mid (u, v) \in E\} \cup \{x(v, k-1)\}\}$
   - Subproblems only depend on subproblems with strictly smaller k, so no cycles!

3. **Base**

   - $x(s, 0) = 0$ and $x(v, 0) = \infty$ for $v \neq s$ (no edges)
   - (draw subproblem graph)

4. **Solution**

   - Compute subproblems via top-down memoization or bottom up (draw graph)
   - Can keep track of parent pointers to subproblem that minimized recurrence
   - Unlike normal Bellman-Ford, parent pointers always form a path back to s!
   - If has finite shortest path, than $\delta(s, v) = x(v, |V| - 1)$
   - Otherwise some $x(v, |V|) < x(v, |V| - 1)$, so path contains a negative weight cycle
   - Claim: All cycles along a parent path have negative weight
   - Proof: If not, removing cycle is path with fewer edges with no greater weight

5. **Time**

   - # subproblems: $|V| \times (|V| + 1)$
   - Work for subproblem $x(v, k)$: $O(\deg_{\text{in}}(v))$

$$\sum_{k=0}^{|V|} \sum_{v \in V} O(\deg_{\text{in}}(v)) = \sum_{k=0}^{|V|} O(|E|) = O(|V||E|)$$

   - Computing $\delta(s, v)$ takes $O(|V|)$ time per vertex, so $O(|V|^2)$ time in total
   - Running time $O(|V|(|V| + |E|))$. Can we make $O(|V||E|)$?
   - Only search on vertices reachable from s, then $|V| \leq |E| + 1 = O(|E|)$
   - Such vertices can be found via BFS or DFS in $O(|V| + |E|)$ time

This is just **Bellman-Ford**!

Dynamic Programs for APSP

This problem considers three dynamic-programming approaches to solve the All-Pairs Shortest Paths (APSP) problem: given a weighted directed graph $G = (V, E, w)$, with possibly negative weights but **no negative cycles**, compute $\delta(u, v)$ for all pairs of vertices $u, v \in V$. Assume vertices are identified by consecutive integers such that $V = \{1, 2, \dots, |V|\}$.

- An approach similar to Bellman–Ford uses subproblems $x(u, v, k)$: **the smallest weight of a path from vertex u to v having at most k edges**. Describe a dynamic-programming algorithm on these subproblems to solve APSP in $O(|V|^2|E|)$ time.

  2. **Relate:** $x(u, v, k) = \min\{x(u, y, k-1) + w(y, v) \mid (y, v) \in E\} \cup \{x(u, v, k-1)\}$ (only depends on smaller k so acyclic)
  3. **Base:** $x(u, u, 0) = 0$, $x(u, v, 0) = \infty$ for $u \neq v$
  4. **Solution:** $x(u, v, |V| - 1)$ for all $u, v \in V$
  5. **Time:** $O(|V|^3)$ subproblems, each $O(1 + \deg_{\text{in}}(v))$ work: $O(|V|^2|E|)$ in total

- Consider subproblems $z(u, v, k)$: **the smallest weight of a path from vertex u to v which only uses vertices from $\{1, 2, \dots, k\} \cup \{u, v\}$**. Describe a dynamic-programming algorithm on these subproblems to solve APSP in $O(|V|^3)$ time. This algorithm is known as **Floyd-Warshall**.

  2. **Relate:** $x(u, v, k) = \min\{x(u, k, k-1) + x(k, v, k-1), x(u, v, k-1)\}$ (only depends on smaller k so acyclic)
  3. **Base:** $x(u, u, 0) = 0$, $x(u, v, 0) = w(u, v)$ if $(u, v) \in E$, or ∞ otherwise
  4. **Solution:** $x(u, v, |V|)$ for all $u, v \in V$
  5. **Time:** $O(|V|^3)$ subproblems, each $O(1)$ work: $O(|V|^3)$ in total. The constant number of dependencies per subproblem brings the factor of $O(|E|)$ in the running time down to $O(|V|)$.