Dynamic Programming
Part Three
Problem Set Five due right now, or due Wednesday with a late period.

Problem Set Six out, due next Monday.
  - Explore dynamic programming across different application domains!
  - Get a feel for how to structure DP solutions!
  - You may use a late day on Problem Set Six, but be aware this will overlap with the final project.

Handout: “Guide to Dynamic Programming” also available.
Final Project Logistics

- Final project will go out next Monday and be due on **Saturday, August 17** at **12:15PM** (note the different time).

- Format: Three algorithms questions, each of which combine two or more different techniques from the quarter.
  - No collaboration permitted with other students.
  - No outside sources may be consulted.
  - Course staff will only answer clarifying questions about the problems.
Please evaluate this course on Axess.

Your feedback really makes a difference.
Outline for Today

- **Shortest Paths Revisited**
  - What if the edge weights are negative?

- **The Bellman-Ford Algorithm**
  - A simple and elegant algorithm for finding shortest paths.

- **The Floyd-Warshall Algorithm**
  - Finding shortest paths between all pairs of points.
Negative Edge Weights
The Recurrence

- **Idea:** Find paths of lengths at most 0, 1, 2, ..., \( n \).
- Let \( w(u, v) \) denote the weight of edge \((u, v)\).
- Let \( s \) be our start node. Let \( \text{OPT}(v, i) \) be the length of the shortest \( s - v \) path whose length is at most \( i \), or \( \infty \) if no path exists.
- **Claim:** \( \text{OPT}(v, i) \) satisfies the following recurrence:

\[
\text{OPT}(v, i) = \begin{cases} 
0 & \text{if } i = 0 \text{ and } v = s \\
\infty & \text{if } i = 0 \text{ and } v \neq s \\
\min \left\{ \text{OPT}(v, i-1), \min_{(u,v) \in E} \{ \text{OPT}(u, i-1) + w(u, v) \} \right\} & \text{otherwise}
\end{cases}
\]
The Bellman-Ford Algorithm

- The **Bellman-Ford algorithm** evaluates this recurrence bottom-up:
  - Create a table DP of size $n \times n$.
  - Set $DP[v][0] = \infty$ for all $v \neq s$.
  - Set $DP[s][0] = 0$
  - For $i = 1$ to $n - 1$, for all $v \in V$:
    - Set $DP[v][i] = \min \{ DP[v][i - 1], \min \{ DP[u][i - 1] + w(u, v) \} \}$ (where $(u, v) \in E$)
  - Return row $n$ of DP.
\[ \text{OPT}(v,i) = \begin{cases} 
0 & \text{if } i=0 \text{ and } v=s \\
\infty & \text{if } i=0 \text{ and } v \neq s \\
\min \left\{ \text{OPT}(v,i-1), \min_{(u,v) \in E} \left\{ \text{OPT}(u,i-1)+w(u,v) \right\} \right\} & \text{otherwise} 
\end{cases} \]
Analyzing Time Complexity

- What is the time complexity of this algorithm?
- Create a table DP of size $n \times n$.
- Set $DP[v][0] = \infty$ for all $v \neq s$.
- Set $DP[s][0] = 0$
- For $i = 1$ to $n - 1$, for all $v \in V$:
  - Set $DP[v][i] = 
    \min \{ 
      DP[v][i - 1], 
      \min \{ 
        DP[u][i - 1] + w(u, v) \} \text{ (where } (u, v) \in E) 
    \}
- Return row $n$ of DP.

Answer: $O(mn)$, i you reverse $G$ prior to running the algorithm.
Analyzing Space Complexity

- What is the space complexity of this algorithm?
  - Create a table DP of size $n \times n$.
  - Set $DP[v][0] = \infty$ for all $v \neq s$.
  - Set $DP[s][0] = 0$
  - For $i = 1$ to $n - 1$, for all $v \in V$:
    - Set $DP[v][i] = \min \{ DP[v][i - 1], \min \{ DP[u][i - 1] + w(u, v) \} \}$ (where $(u, v) \in E$)
  - Return row $n$ of DP.

- Answer: $O(n^2)$. (Can we reduce this?)
All-Pairs Shortest Paths
Shortest Paths

- Dijkstra's algorithm and the Bellman-Ford algorithm solve the *single-source shortest paths problem* in which we want shortest paths starting from a single node.

- The *all-pairs shortest paths problem* asks how to find the shortest paths between all possible pairs of nodes.

- Can we already solve this problem?

- How efficient is our solution?
Intermediary Nodes

- A path between $u$ and $v$ starts at $u$, passes through some set of intermediary nodes, and ends at $v$.
- If there are no negative cycles, there is some shortest path from $u$ to $v$ where no nodes will be revisited. (*Why?*)
**Intermediary Nodes**

- Number all nodes $v_1, v_2, \ldots, v_n$.
- What does a shortest path from $u$ to $v$ look like if no intermediary nodes are allowed?
- What does a shortest path from $u$ to $v$ look like if only node $v_1$ can be an intermediary node?
- What does a shortest path from $u$ to $v$ look like if only nodes $v_1$ and $v_2$ can be intermediary nodes?
- What does a shortest path from $u$ to $v$ look like if only nodes $v_1$, $v_2$, and $v_3$ can be intermediary nodes?
The Recurrence

Let \( \text{OPT}(i, j, k) \) be the length of the shortest path from \( i \) to \( j \) where the only permitted internal nodes are \( v_1, v_2, \ldots, v_k \).

**Claim:** \( \text{OPT}(i, j, k) \) satisfies this recurrence:

\[
\text{OPT}(i, j, k) = \begin{cases} 
0 & \text{if } i = j \text{ and } k = 0 \\
\infty & \text{otherwise if } k = 0 \\
w(v_i, v_j) & \text{if } (v_i, v_j) \in E \text{ and } k = 0 \\
\min \left\{ \begin{array}{l}
\text{OPT}(i, j, k-1), \\
\text{OPT}(i, k, k-1) + \\
\text{OPT}(k, j, k-1)
\end{array} \right\} & \text{if } k \neq 0
\end{cases}
\]
The Floyd-Warshall Algorithm

- Let DP be an $n \times n \times (n + 1)$ table.
- For $i$ from 1 to $n$, $j$ from 1 to $n$:
  - Set $DP[i][j][0] = 0$ if $i = j$.
  - Set $DP[i][j][0] = w(v_i, v_j)$ if $i \neq j$ and $(u, v) \in E$.
  - Set $DP[i][j][0] = \infty$ if $i \neq j$ and $(u, v) \notin E$.
- For $k$ from 1 to $n$, $i$ from 1 to $n$, $j$ from 1 to $n$:
  - Set $DP[i][j][k] = \min\{DP[i][j][k - 1], DP[i][k][k - 1] + DP[k][j][k - 1]\}$
- Return row $k$ of DP.
Time and Space Complexity

- What is the time complexity of this algorithm?
  - $O(n^3)$

- What is the space complexity of this algorithm?
  - $O(n^3)$

- Interestingly, no dependence on the number of edges!
Further Algorithms

- **Johnson's Algorithm** combines Dijkstra's algorithm and Bellman-Ford together to solve the all-pairs shortest paths problem in arbitrary graphs with no negative cycles.
- Runtime is $O(mn + n^2 \log n)$ when implemented with appropriate data structures.
- How does that compare to Floyd-Warshall?
- Come talk to me after lecture for details!
Next Time

- Intractable Problems
- \textbf{NP}-Hardness
- Pseudopolynomial Algorithms