VE281

Data Structures and Algorithms

Dynamic Programming: All-Pairs Shortest Paths

Final Exam

- Time: Dec. 13th, 10:00 am – 11:40 am.
- Location: See Canvas announcement
- A written exam.
  - Like our written assignments.
  - Pseudo-code OK (but make sure we can **understand** it!)
- Closed book and closed notes.
- No electronic devices are allowed, except basic calculators.
  - These include laptops and cell phones.
  - We will show a clock on the screen.
- Abide by the **Honor Code**!

Final Exam Topics

- Binary Search Tree
  - Insertion, removal, time complexity, other efficient operations
  - k-d tree
- Balanced Binary Search Tree
  - AVL Tree
  - Red-black tree
- Graph and Graph Algorithms
  - Graph search, topological sorting, MST, shortest path
- Dynamic Programming

Fibonacci heap not included

All-Pairs Shortest Paths

- Input: a weighted graph $G = (V, E)$
- Output: for every pair of nodes $v, u \in V$, a shortest path from u to v
- Straightforward solution: run a single-source all-destinations shortest-path algorithm $|V|$ times
  - Dijkstra's algorithm: each run $O(|V| \log |V| + |E|)$
  - Bellman-Ford algorithm: each run $O(|V|^2 + |V||E|)$
  - If the graph has no negative-weighted edges, Dijktra's algorithm is a good choice
  - If it has negative-weighted edges, Bellman-Ford should be applied. Time complexity is $O(|V|^2(|V| + |E|))$

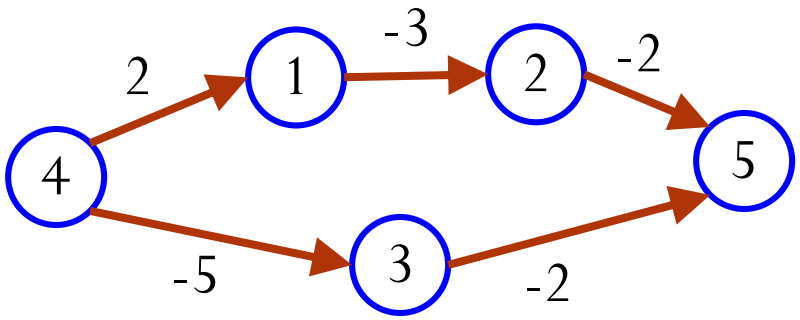
Can we do better?

Floyd-Warshall Algorithm

- Work for graph with negative edge weight
  - Assume there is **no negative cycle**
- A dynamic programming-based algorithm
- Time complexity: $O(|V|^3)$
  - Better than Bellman-Ford, especially for dense graph

Optimal Substructure

- Let $V = \{1,2,\dots,n\}$ and $V^{(k)} = \{1,2,\dots,k\}$
- Define general problem $Q_{ij}^{(k)}$: find a shortest path from node i to j with all intermediate nodes from set $V^{(k)}$
  - Note: without negative cycle, a shortest path is cycle-free
  - Define $d_{ij}^{(k)}$ as the length of such a shortest path
  - We ultimately want to solve $Q_{ij}^{(n)}$ for $1 \le i, j \le n$

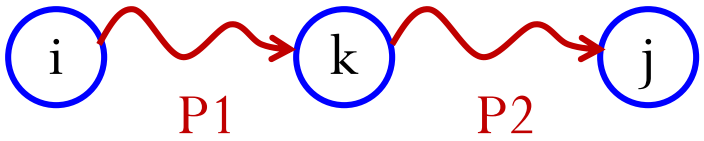


What is a solution to $Q_{45}^{(2)}$?

Path 4→1→2→5

Optimal Substructure

- Consider a shortest path for $Q_{ij}^{(k)}$
  - Case 1: Does not include k as an intermediate node
    - Claim: the path is also a shortest path for $Q_{ij}^{(k-1)}$
    - $d_{ij}^{(k)} = d_{ij}^{(k-1)}$
  - Case 2: has k as an intermediate node
    - Claim: P1 is a shortest (cycle-free) path for $Q_{ik}^{(k-1)}$ and P2 is a shortest (cycle-free) path for $Q_{kj}^{(k-1)}$
    - $d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$



- Recursion: $d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}$

Initial Situation

- What is $Q_{ij}^{(0)}$?
  - Find a shortest path from i to j with no intermediate nodes
  - Just the edge between node i and j (if exists)

- Thus, $d_{ij}^{(0)} = w_{ij}$
  - $w_{ij} = \begin{cases} 0 & i = j \\ edge\ weight\ of\ (i,j) & i \neq j, (i,j) \in E \\ +\infty & i \neq j, (i,j) \notin E \end{cases}$

Floyd-Warshall Algorithm

- Compute the shortest-path weights bottom up
- Let W be the weight matrix $(w_{ij})_{n\times n}$

```
FLOYD-WARSHALL(W)
1  n = W.rows
2  D^(0) = W
3  for k = 1 to n
4      let D^(k) = (d_ij^(k)) be a new n × n matrix
5      for i = 1 to n
6          for j = 1 to n
7              d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
8  return D^(n)
```

Time complexity? $O(n^3)$

Space complexity? $O(n^2)$

Reconstruct Shortest Paths

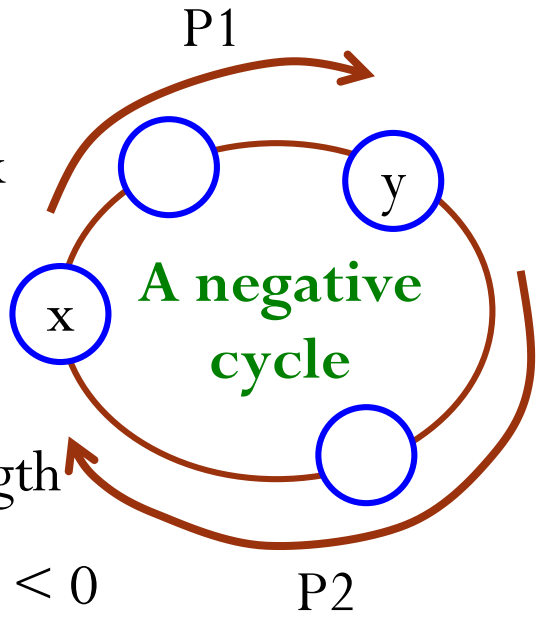
- Maintain a predecessor matrix $P^{(k)} = (p_{ij}^{(k)})$ during the Floyd-Warshall algorithm

- $$p_{ij}^{(0)} = \begin{cases} NULL, & i = j \text{ or } w_{ij} = +\infty \\ i, & i \neq j \text{ and } w_{ij} < +\infty \end{cases}$$

- $$p_{ij}^{(k)} = \begin{cases} p_{ij}^{(k-1)}, & d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ p_{kj}^{k-1}, & d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases}$$

Detect Negative Cycles

- Claim: if there is a negative cycle, then there exists a **negative diagonal** entry $d_{ii}^{(n)}$ in the final matrix
- Proof: consider a negative cycle
  - Arbitrarily choose a node x on the cycle
  - Let y be a remaining node with the highest index
  - Consider the y-th iteration in outer loop
  - $d_{xx}^{(y)} = \min\{d_{xx}^{(y-1)}, d_{xy}^{(y-1)} + d_{yx}^{(y-1)}\}$
  - Note: $d_{xy}^{(y-1)} \leq$ P1's length, $d_{yx}^{(y-1)} \leq$ P2's length
  - $d_{xx}^{(y)} \leq d_{xy}^{(y-1)} + d_{yx}^{(y-1)} \leq$ negative cycle cost < 0
  - $d_{xx}^{(n)} \leq d_{xx}^{(y)} < 0$



Transitive Closure of Directed Graph

- Given a graph $G = (V, E)$, we might want to determine whether G contains a path from i to j for all pairs of nodes i and j
- **Transitive closure** of G is another graph $G^* = (V, E^*)$, where $E^* = \{(i, j)$: there is a path from i to j in $G\}$

- One solution: assign edge weight 1 to each edge in E and run Floyd-Warshall
  - If there is a path from node i to j, then $d_{ij}^{(n)} < +\infty$
  - Otherwise, $d_{ij}^{(n)} = +\infty$
  - $O(n^3)$ time complexity

Transitive Closure of Directed Graph

- Another similar solution: substitute logical OR $\vee$ and logical AND $\wedge$ for min and $+$, respectively, in Floyd-Warshall
- Define $t_{ij}^{(k)}$ to be 1 if there exists a path from node i to j with all intermediate nodes from the set $\{1,2,\ldots,k\}$, and 0 otherwise
- Recurrence:
  - $t_{ij}^{(0)} = \begin{cases} 1, & i = j \text{ or } (i,j) \in E \\ 0, & i \neq j \text{ and } (i,j) \notin E \end{cases}$
  - $t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$
- This method is more time and space efficient