Floyd-Warshall Algorithm For Solving the All-Pairs Shortest Path Problem

https://cs.pomona.edu/classes/cs140/
Outline

Topics and Learning Objectives
• Discuss and analyze the Floyd-Warshall Algorithm

Exercise
• None
All-Pairs Shortest Path Problem

Compute the shortest path from every vertex to every other vertex

• Input: a weighted graph
• Output:
  • Shortest path from $u \rightarrow v$ for all values of $u$ and $v$
  • Report that a negative cycle has been discovered

• Can we solve this problem with what we know already?
How do we turn a solution to the single-source shortest path (SSSP) problem into a solution for the all-pairs shortest path (APSP) problem?

• This is called a reduction!

• How many times do we need to run a SSSP procedure for APSP?
  
  a. 1  
  b. $n - 1$  
  c. $n$  
  d. $n^2$
What SSSP algorithms do we know?

Running time of APSP if we don’t allow negative edges?
• \( n \times O(\text{Dijkstra’s Algorithm}) = O(n \, m \, \lg n) \)
• For sparse graphs: \( O(n^2 \, \lg n) \)
• For dense graphs: \( O(n^3 \, \lg n) \)

Running time of APSP if we do allow negative edges?
• \( n \times O(\text{Bellman-Ford}) = O(n^2 \, m) \)
• For sparse graphs: \( O(n^3) \leftarrow \)
• For dense graphs: \( O(n^4) \leftarrow \)
Consider APSP on dense graphs.

- How many values are we going to output? $n^2$
- What is the potential length of a shortest path? $n - 1$
- What is the lower bound on the running time of APSP?
- It is tempting to say that the lower bound is $n^3$
- However, this lower bound has yet to be determined
- Consider the matrix multiplication procedure developed by Strassen
Specialized APSP Algorithm

• Although we can use Bellman-Ford and Dijkstra’s algorithms, there are, in fact, specialized APSP algorithms

• The Floyd-Warshall algorithm solves the APSP problem deterministically in $O(n^3)$ on all types of graph

• It works with negative edge lengths

• Meaning that is as good as Bellman-Ford for sparse graphs,

• And much better than Bellman-Ford for dense graphs.
**Question**

<table>
<thead>
<tr>
<th></th>
<th>Sparse Graphs</th>
<th>Dense Graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dijkstra’s</td>
<td>$O(n^2 \lg n)$</td>
<td>$O(n^3 \lg n)$</td>
</tr>
<tr>
<td>n times</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bellman-Ford</td>
<td>$O(n^3)$</td>
<td>$O(n^4)$</td>
</tr>
<tr>
<td>n times</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Floyd-Warshall</td>
<td>$O(n^3)$</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- What algorithm would you choose for sparse graphs?
  - Dijkstra’s *n times* if there are no negative edges, Floyd-Warshall otherwise

- What algorithm would you choose for dense graphs?
  - Always Floyd-Warshall
Optimal Substructure for APSP

Key concept:
- label the vertices 1 though n (giving them an arbitrary order),
- and then introduce the notation $V^{(k)} = \{1, 2, \ldots, k\}$

Optimal Substructure Lemma:
- Assume, for now, that the graph does not include a negative cycle
- Fix a source vertex $i$, a destination vertex $j$, and a value for $k$
- Then let $P$ be the shortest $i \to j$ path with \textit{internal} nodes from $V^{(k)}$
Example Substructure

Optimal Substructure Lemma:

• Fix a source vertex $i$, a destination vertex $j$, and a value for $k$
• Then let $P$ be the shortest $i \rightarrow j$ path with \texttt{internal} nodes from $V^{(k)}$
Example Substructure

Optimal Substructure Lemma:

• Fix a source vertex $i$, a destination vertex $j$, and a value for $k$
• Then let $P$ be the shortest $i \rightarrow j$ path with internal nodes from $V^{(k)}$

$i = 17$

$j = 10$
Example Substructure

Optimal Substructure Lemma:
• Fix a source vertex $i$, a destination vertex $j$, and a value for $k$
• Then let $P$ be the shortest $i \rightarrow j$ path with internal nodes from $V^{(k)}$

$i = 17$
$j = 10$
Example Substructure

Optimal Substructure Lemma:
• Fix a source vertex $i$, a destination vertex $j$, and a value for $k$.
• Then let $P$ be the shortest $i \rightarrow j$ path with internal nodes from $V^{(k)}$.

What is the value of the shortest path found by FW?

$i = 17$
$j = 10$
Example Substructure

Optimal Substructure Lemma:
• Fix a source vertex \( i \), a destination vertex \( j \), and a value for \( k \)
• Then let \( P \) be the shortest \( i \rightarrow j \) path with \textit{internal} nodes from \( V^{(k)} \)

What is the value of the shortest path found by FW?
Optimal Substructure Lemma

Suppose that G has no negative cycles. Let $P$ be the shortest (cycle-free) path $i \to j$, where all internal nodes come from $V^{(k)}$. Then:

- **Case 1**: if $k$ is not internal to $P$, then $P$ is also a shortest path $i \to j$ with all internal nodes from $V^{(k-1)}$.

- **Case 2**: if $k$ is internal to $P$, then:
  - Let $P_1$ = the shortest $i \to k$ path with nodes from $V^{(k-1)}$ and $V^{(k-1)}$.
  - Let $P_2$ = the shortest $k \to j$ path with nodes from $V^{(k-1)}$.
  - Effectively, $k$ splits the path into two optimal subproblems.
Picture of our cases

Case 1

Case 2
Floyd-Warshall Algorithm Base Cases

Let $A = 3D$ array, where $A[i, j, k] = \text{the length of the shortest } i \rightarrow j \text{ path with all internal nodes from } \{1, 2, \ldots, k\}$

• Which index ($i$, $j$, or $k$) do you think represents our base case?

What is the value of $A[i, j, 0]$ when...

• $i = j$? $0$
• there is a direct edge from $i$ to $j$ $c_{ij}$
• there is no edge directly connecting $i$ to $j$ $\infty$
FUNCTION FloydWarshall(graph)
    # Base 1 indexing for vertices labeled 1 through n
    pathLengths = [n by n by (n + 1) array]

    # Base case
    FOR vFrom IN [1 ..= n]
        FOR vTo IN [1 ..= n]
            IF i == j
                length = 0
            ELSE IF graph.hasEdge(vFrom, vTo)
                length = graph.edges[vFrom][vTo].weight
            ELSE
                length = INFINITY
            pathLengths[vFrom][vTo][0] = length

    # Table building
    continued next slide...
**FUNCTION** FloydWarshall(graph)

# Base 1 indexing for vertices labeled 1 through n

\[ \text{pathLengths} = [n \text{ by } n \text{ by } (n + 1) \text{ array}] \]

# Base case

cut from previous slide...

# Table building

\[
\text{FOR } k \text{ IN } [1 \ldots n] \\
\quad \text{FOR } vFrom \text{ IN } [1 \ldots n] \\
\quad \quad \text{FOR } vTo \text{ IN } [1 \ldots n] \\
\]

# Case 1

\[
\text{withoutK} = \text{pathLengths}[vFrom][vTo][k - 1] \\
\]

# Case 2

\[
\text{withKSubPathA} = \text{pathLengths}[vFrom][k][k - 1] \\
\text{withKSubPathB} = \text{pathLengths}[k][vTo][k - 1] \\
\]

\[
\text{pathLengths}[vFrom][vTo][k] = \min( \\
\quad \text{withoutK}, \\
\quad \text{withKSubPathA + withKSubPathB} \\
) \\
\]
Floyd-Warshall Algorithm

Running time?
• $O(n^3)$

Correctness?
• Substructure lemma

• Where are the final answers?
• How does it handle negative cycles?
• Reconstruction is similar to other dynamic programming problems.