Adjacency Matrix

- Most of the algorithms in this chapter use adjacency matrix

\[ W = (w_{ij}) \]

\[ w_{ij} = \begin{cases} 
0 & \text{if } i = j \\
\text{weight of directed edge } (i, j) & \text{if } i \neq j \text{ and } (i, j) \in E \\
\infty & \text{otherwise}
\end{cases} \]

- Output will be an \( n \times n \) array \( D = (d_{ij}) \)

\[ d_{ij} \] will be shortest-path weight from \( i \) to \( j \)

- Allowing negative weights doesn't run into negative cycles

\( \frac{(f_p)}{\Pi} \) = \( D \) key array \( u \times u \times n \)

- Output will be a predecessor matrix \( \Pi = (\pi_{ij}) \)

\[ \pi_{ij} = \text{nil if } i = j \text{ or not path from } i \text{ to } j \]

\[ \text{otherwise } \pi_{ij} = \text{ predecessor of } j \text{ on some shortest path from } i \]

- Most of the algorithms in this chapter use adjacency matrix

---

All-Pairs Shortest Paths (Chapter 25)

- Weighted Directed Graph

- Could run a single-source shortest paths algorithm \( |A| \)

- Let's do better! Many applications depend on this

\[ (A \oplus \lambda A)\Theta + \]

\[ \text{All vertices } \Theta + \]

\[ \text{Dijkstra's algorithm runs in } \Theta + \]

\[ (\lambda A \oplus \lambda F)\Theta \]

\[ \text{If graph has no negative cycles (like for route finding) } \]

\[ (\lambda A)\Theta \]

\[ \text{If graph has only negative cycles and edges } \]

\[ \text{Could run a single-source shortest paths algorithm } \]

---

All-Pairs Shortest Paths (Chapter 25)
Directed graph, negative edges, but no negative cycles

Step 1 of dynamic programming:
- Characterize the optimal solution
- Can use these optimal subpaths over and over again!
- For any \( j > i \), we have \( d_{ij} \) is shortest path for \( i \) to \( j \).

For any \( j \), path \( \langle v_0, v_1, \ldots, v_{j-1}, v_j \rangle \) is shortest for \( v_0 \) to \( v_j \) (Lemma 24.1)

\( d_{ij} = d \) if \( d_{ij} < d \)
- Say \( d \) is shortest path from \( i \) to \( j \) and \( d \) is shortest path from \( i \) to \( a \)
- \( d \) is shortest path from \( i \) to \( j \) (Lemma 24.1)

- Can use these optimal subpaths over and over again!
- But how?
- For any \( i, j \) if we know that the last step goes from \( k \) to \( j \), overall path
- But is optimal path from \( i \) to \( k \) any simpler?
- It will have one less edge than path from \( i \) to \( j \)

\( d_{ij} \) is shortest path from \( i \) to \( j \)

\( \langle v_0, v_1, \ldots, v_{i-1}, v_i \rangle \) is shortest path for \( v_0 \) to \( v_i \) (Lemma 24.1)

\( d_{ij} \) is shortest path for \( v_0 \) to \( v_j \)

\( d_{ij} \) is shortest path for \( v_0 \) to \( v_k \) plus edge \( (k, j) \)

\( d_{ij} \) is shortest path for \( v_0 \) to \( v_i \) plus edge \( (k, j) \)

\( d_{ij} \) is shortest path for \( v_0 \) to \( v_j \) plus edge \( (k, j) \)

Thus, if \( d_{ij} < d \), then \( d_{ij} \) is shortest path for \( i \) to \( j \).
Shortest Path Weights

- In each $g_i$, $g_i = g_i'$
- If graph has no negative weight cycles
  - If $j$ is reachable from $i$, shortest path exists from $i$ to $j$ will have at most $n-1$ edges
  - $\delta(i, j) = l_{ij}(n-1)$ since we can just pad on $w_{jj}$
  - In fact $\delta(i, j) = l_{ij}(n) = l_{ij}(n+1) = \ldots$

\[ \ldots (g_i') = (g_i) = (g_i') \]

Recursively Define Value of an Optimal Solution

- Consider shortest paths up to length $m$.
- $\ell(i, j)_{m}$ is 0 if $i = j$ and $\ell(i, j) = \infty$ if $i \neq j$.
- Be minimum weight of any path from $i$ to $j$ that contains at most $m$ edges.
- $\ell(i, j)_{m} = \min \{ \ell(i, k)_{m-1} + w_{kj}, 1 \leq k \leq n \}$ since can just add on $w_{jj}$ which is 0.
Computing Shortest-path Bottom-up

• Can view this as: ESP(...)ESP(ESP(W,W),W)...

\[ (M) \]

\textbf{Return} \( L^{(1)} \)

\textbf{EXTEND-SHORTEST-PATHS} \( (T) \) \( (w) \)

1. Let \( L^{(w)} \) be a new \( u \times u \) matrix
2. For \( m \) \( 1 \) to \( n \) - 1
3. \( M = (M)^{m-1} \)
4. \( M.' \)ROWS = \( u \) \( 1 \)

.......

\textbf{Rest of Code}

\section*{Time complexity?}

\[ (T) \]

\textbf{Return} \( L^{(1)} \)

\textbf{EXTEND-SHORTEST-PATHS} \( (T) \) \( (w) \)

1. Let \( L^{(w)} \) be a new \( u \times u \) matrix
2. For \( m \) \( 1 \) to \( n \)
3. \( M = (M)^{m-1} \)
4. \( M.' \)ROWS = \( u \) \( 1 \)

\section*{Computing Shortest-path Bottom-up
Overview

- Shortest Paths
- Floyd-Warshall Algorithm
- Shortest Paths and Matrix Multiplication

Example

- $L(1)$ is just $W$
- $L(2)$ is like $W$ but for hops of at most length 2

\[
\begin{pmatrix}
0 & 6 & 8 & 2 & 3 \\
6 & 0 & 5 & 1 & 4 \\
8 & 5 & 0 & 4 & 3 \\
2 & 1 & 4 & 0 & 7 \\
3 & 4 & 3 & 7 & 0
\end{pmatrix}
= (\text{INIT})
\]

\[
\begin{pmatrix}
0 & 6 & 8 & 2 & 3 \\
6 & 0 & 5 & 1 & 4 \\
8 & 5 & 0 & 4 & 3 \\
2 & 1 & 4 & 0 & 7 \\
3 & 4 & 3 & 7 & 0
\end{pmatrix}
= (\text{INIT})
\]

\[
\begin{pmatrix}
0 & 6 & 8 & 2 & 3 \\
6 & 0 & 5 & 1 & 4 \\
8 & 5 & 0 & 4 & 3 \\
2 & 1 & 4 & 0 & 7 \\
3 & 4 & 3 & 7 & 0
\end{pmatrix}
= (\text{INIT})
\]

\[
\begin{pmatrix}
0 & 6 & 8 & 2 & 3 \\
6 & 0 & 5 & 1 & 4 \\
8 & 5 & 0 & 4 & 3 \\
2 & 1 & 4 & 0 & 7 \\
3 & 4 & 3 & 7 & 0
\end{pmatrix}
= (\text{INIT})
\]
• Overall time is $O(3n^2 \log n)$ steps.

Can compute $T^w(u,v)$ in $O(n^2)$ steps.

To compute $L(1)$, can call routine on $T^w(u,v)$ and $T^w(v,u)$.

To compute $L(4)$, can call routine on $T^w(u,v)$ and $T^w(v,u)$.

To compute $L(8)$, can call routine on $T^w(u,v)$ and $T^w(v,u)$.

Can compute $L(n-1)$ in $O(\sqrt{n} \log n)$ steps.

Overall time is $O(V^3 \log(V))$.

Towards a Faster Implementation

Shortest Paths is like Matrix Multiplication

EXTEND-SHORTEST-PATHS($L$, $W$) = EXTENSION($L$) + $L$

In fact, just as matrix $\times$ is associative, so is EXTEND-SHORTEST-PATHS.

Pretty similar.

Pretty similar.

EXTEND-SHORTEST-PATHS($L$, $W$) = EXTENSION($L$) + $L$

In fact, just as matrix $\times$ is associative, so is EXTEND-SHORTEST-PATHS.

Pretty similar.

Pretty similar.
Structure of a Shortest Path

Previously, characterized the optimal substructure for a shortest path from \( s \to u \to v \), consider paths of shorter and shorter lengths. Applied dynamic programming in bottom-up approach.
Recursive Solution

Let $d(k)_{ij}$ be the weight of a shortest path from $i$ to $j$ for which all intermediate vertices are in $V_k$

- $d(0)_{ij} = w_{ij}$ since cannot have any intermediate vertices

- Can at most have one edge: $\langle i, j \rangle$ if it is in $E$

- If no edge $\langle i, j \rangle$, $w_{ij} = \infty$

- $d(k)_{ij} = \min(d(k-1)_{ij}, d(k-1)_{ik} + d(k-1)_{kj})$

WOW!

Different Optimal Substructure Approach

- Say $G$ has $n$ vertices: $\{1, ..., n\}$

- Consider subset $\{1, ..., k\} = V_k$

- For any pair of vertices $u, v$ in $\{1, ..., k\}$

- Consider paths whose intermediate vertices are in $\{1, ..., k\}$
Determining the Paths

- Need to keep track of the predecessors

\[ \Pi(i) \] corresponds to \[ D(i) \] for \( 0 \leq i \leq n \)

- \( \pi(i) \) predecessor of \( j \) on shortest path from vertex \( i \) with all intermediate vertices in \( V_k \)

- \( \Pi(0) \) is ?

How do we modify code?

```c

FLOYD-WARSHALL

1

D:

2 n

for k from 1 to n

3 let D[k][i][j] be a new n x n matrix

4 for i from 1 to n

5 for j from 1 to n

6 d[k][i][j] = min

7 d[k][i][j] = \( \Pi(k) \) or \( d[k-1][i][j] \)

8 return D

```

Time complexity:

\[
\left( (1 - \gamma)^p + (1 - \gamma)^p \cdot (1 - \gamma)^p \right) \text{ min } = (1 - \gamma)^p
\]

\[ u \rightarrow 1 \]

For \( u \rightarrow 1 \)

For \( u \rightarrow 1 \) for a new matrix \( u \times u \)

M = \( (D) \)

M = rows

FLOYD-WARSHALL(M)