Dynamic Programming

Reading: Section 6.4 and 6.8
Bellman-Ford Algorithm

Shortest-Path($G, s, t$)

$n =$ number of nodes in $G$

Array $M[0...n-1, V]$

Define $M[0, t] = 0$ and $M[0, v] = \infty$ for all other $v \in V$

For $i = 1, \ldots, n-1$

For $v \in V$ in any order

Compute $M[i, v]$ using the recurrence

Endfor

Endfor

Return $M[n-1, s]$
Exercise (solution)

- Find cost shortest s-z path
Exercise (solution)

- Find cost shortest s-z path
Exercise (solution)

- Find cost shortest s-z path
Exercise (solution)

- Find cost shortest s-z path
Exercise (solution)

- Find cost shortest s-z path
Exercise (solution)

- Find cost shortest s-z path
Exercise (solution)

- Find cost shortest s-z path
Exercise (solution)

- Find cost shortest s-z path
Exercise (solution)

- Find cost shortest s-z path

![Graph with node labels and edge weights]

<table>
<thead>
<tr>
<th></th>
<th>s</th>
<th>t</th>
<th>y</th>
<th>x</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>i/v</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>∞</td>
<td>-4</td>
<td>9</td>
<td>∞</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>∞</td>
<td>∞</td>
<td>9</td>
<td>-6</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>-4</td>
<td>-9</td>
<td>-6</td>
<td>0</td>
</tr>
</tbody>
</table>
Exercise (solution)

- Find cost shortest s-z path
Exercise (solution)

- Find cost shortest s-z path
Network Flow

Reading: Section 7.1 and 7.2
Trucking Problem
Trucking Problem

How many “units” can you send from Vancouver to Winnipeg, given the restrictions?
Network Flow

- Direct applications:
  - commodities in networks
  - transporting food on the rail network
  - packets on the internet
  - gas through pipes

- Indirect application:
  - Matching graphs
  - Airline Scheduling
Definition

- **Flow network**
  - Directed graph
  - Source and target
  - Edge capacities $c(e)$ that are positive

- **Flow**
  - Capacity constraints
  - What goes in, has to go out

- **Max-flow problem**
  - Find a flow of maximum value
Example
Example with some Flow

- What is the flow in this graph?
- Can you think of a bottleneck?
How to solve this problem?

- Idea: Repeatedly choose paths and “augment” flow on those paths until we can no longer do so

- Does it work?
Residual Graph

- Residual graph: data structure to identify opportunities to push more flow on edges with leftover capacity or undo flow on edges already carrying flow.
Residual Graph

- Residual graph: data structure to identify opportunities to push more flow on edges with leftover capacity or undo flow on edges already carrying flow.

- Original edge $e = (u, v) \in E$
  - Flow $f(e)$
  - Capacity $c(e)$

- Forward residual edge
  - $e = (u, v)$
  - Residual capacity $c(e) - f(e)$

- Backward residual edge
  - If $f(e) > 0$, create edge $e_0 = (v, u)$
  - Residual capacity $f(e)$
Example with some Flow

- Capacity/Flow
Residual Graph
Ford-Fulkerson Algorithm (idea)

- Do until it is possible
  - Pick augmenting path in residual graph
  - Choose edge with lowest capacity
  - Increase the flow by that amount
  - Update residual graph
Find the Max-Flow