Shortest paths, negative weights
All pairs
WHAT ABOUT NEGATIVE EDGE WEIGHTS?
WHERE DOES OLD ARGUMENT BREAK DOWN

\[ w(p) \geq d_u + \delta(y, u) \]
FIRST IDEAS:
$\text{SSSP}(G,s)$

$\text{SHORT}_{i,v} =$
\[ \text{SSSP}(G,s) \]

\[
\text{SHORT}_{i,v} = \begin{cases} 
\infty & \text{if } i = 0 \\
0 & \text{if } v = s \\
\min_{x \in V} & \text{if } \text{SHORT}_{i-1,v} \\
& \begin{cases} 
\text{SHORT}_{i-1,v} \\
\text{SHORT}_{i-1,x} + w(x, v) 
\end{cases} 
\end{cases}
\]
MAX LEN OF A SIMPLE PATH:
BELLMAN-FORD(G, s)
BELLMAN-FORD\((G, s)\)

1. \(\text{SHORT}_{0,s} \leftarrow 0\)
2. \(\forall v \in V - \{s\}, \text{SHORT}_{0,v} \leftarrow \infty\)
3. \(\text{for } i = 1, \ldots, V - 1\)
4. \(\quad \text{do } \text{for each } v \in V - \{s\}\)
5. \(\quad \quad \text{do } \text{SHORT}_{i,v} = \min_{x \in \text{Adj}(v)} \begin{cases} \text{SHORT}_{i-1,v} \\ w(x, v) + \text{SHORT}_{i-1,x} \end{cases}\)
BELLMAN-FORD\((G, s)\)

1. \text{SHORT}_{0,s} \leftarrow 0
2. \forall v \in V - \{s\}, \text{SHORT}_{0,v} \leftarrow \infty
3. \textbf{for} \ i = 1, \ldots, V - 1
4. \quad \textbf{do} \ \textbf{for} \ \text{each} \ \ v \in V - \{s\}
5. \quad \quad \textbf{do} \ \text{SHORT}_{i,v} = \min_{x \in \text{Adj}(v)} \left\{ \text{SHORT}_{i-1,v} \right\}
   \quad \quad \quad \frac{w(x, v)}{+ \text{SHORT}_{i-1,x}}
BELLMAN-FORD($G, s$)

1. $\text{SHORT}_{0,s} \leftarrow 0$
2. $\forall v \in V - \{s\}, \text{SHORT}_{0,v} \leftarrow \infty$
3. for $i = 1, \ldots, V - 1$
   4. do for each $e = (x, y) \in E$
   5. do $\text{SHORT}_{i,y} = \min \left\{ \begin{array}{l}
\text{SHORT}_{i-1,y} \\
\text{SHORT}_{i,y} \\
w(x, y) + \text{SHORT}_{i-1,x}
\end{array} \right\}$
\[ \text{SHORT}_{i,v} = \left\{ \begin{array}{ll} \infty & i = 0 \\ 0 & v = s \\ \min_{x \in V} \{ \text{SHORT}_{i-1,v} \} & \text{otherwise} \end{array} \right. \]

\[ \text{SHORT}_{i,v} + \text{SHORT}_{i-1,x} + w(x,v) \]
\[\text{short}_{i,v} = \begin{cases} \infty & i = 0 \\ 0 & v = s \\ \min_{x \in V} \left\{ \text{short}_{i-1,v} \right\} \\ \text{short}_{i-1,x} + w(x,v) \end{cases} \]
\[
\text{short} \ i,v = \begin{cases} 
\infty & i = 0 \\
0 & v = s \\
\min_{x \in V} \left\{ \text{short}_{i-1,v} \right\} & \text{short}_{i-1,x} + w(x,v) 
\end{cases}
\]

BF(G,d)
\[
\text{SHORT}_{i,v} = \begin{cases} 
\infty & i = 0 \\
0 & v = s \\
\min_{x \in V} \left\{ \text{SHORT}_{i-1,v} \right\} & \text{SHORT}_{i-1,x} + w(x,v) 
\end{cases}
\]

\[\begin{array}{c|cccccccc}
& 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
A & & & & & & 18 & & \\
B & & & 8 & 8 & & & & \\
C & & & & & & 4 & & \\
D & & & & & & 0 & 0 & 0 \\
E & & & 7 & 7 & & & & \\
F & & & & & & 4 & & \\
G & & & & & 5 & 5 & & \\
H & & & 5 & 5 & & & & \\
I & & & & & & 7 & & \\
\end{array}\]
$\text{short}_i,v = \begin{cases} \infty & i = 0 \\ 0 & v = s \\ \min_{x \in V} \{ \text{short}_{i-1,v} + w(x,v) \} & \end{cases}$
OPTIMIZATION

BELLMAN-FORD($G, s$)
1 \textbf{short}_{0,s} \leftarrow 0
2 \forall v \in V - \{s\}, \textbf{short}_{0,v} \leftarrow \infty
3 \textbf{for } i = 1, \ldots, V - 1
4 \quad \textbf{for } \text{each} \ e = (x, y) \in E
5 \quad \quad \textbf{do } \textbf{short}_{i,y} = \min \begin{cases} \textbf{short}_{i-1,y} \\ \textbf{short}_{i,y} \\ w(x, y) + \textbf{short}_{i-1,x} \end{cases}

BELLMAN-FORD($G, s$)
1 \ d_s \leftarrow 0
2 \forall v \in V - \{s\}, \ d_v \leftarrow \infty
3 \textbf{for } i = 1, \ldots, V - 1
4 \quad \textbf{for } \text{each} \ e = (x, y) \in E
5 \quad \quad \textbf{do } d_y \leftarrow \min \{ \ d_y, w(x, y) + d_x \ \}$
Bellman-Ford \((G, s)\)

1. \(d_s \leftarrow 0\)
2. \(\forall v \in V - \{s\}, \; d_v \leftarrow \infty\)
3. \(\text{for } i = 1, \ldots, V - 1\) do
4. \(\text{for each } e = (x, y) \in E\) do
5. \(d_y \leftarrow \min \{ d_y, w(x, y) + d_x \} \)
NEGATIVE CYCLES?
NEGATIVE CYCLES?

\[ s \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow t \]

\[-5\]

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>A</th>
<th>B</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
NEGATIVE CYCLES?

![Graph with nodes s, 2, 3, 1, t and edges with weights 2, 3, 1, -5]

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPLICATIONS OF BF
Figure 3: Lucent’s intranet as of 1 October 1999.
WHAT HAPPENS WHEN B CHANGES...
DISTANCE VECTOR

image: hurricane electric
ALL-PAIRS SHORTEST PATH

Graph with nodes a, b, h, k, j, i and edges with weights:
- a to b: 1
- a to k: -4
- b to h: 2
- b to j: 7
- h to i: 10
- k to i: -8
- k to j: 3
- j to i: 5

The graph shows the shortest paths between all pairs of nodes.
\text{ASHORT}_{i,j,k} =
$\text{ASHORT}_{i,j,k} =$
ASHORT_{i,j,k} =
\[
\text{ASHORT}_{i,j,k} = \begin{cases} 
    w_{i,j} & \text{k} = 0 \\
    \min \left\{ \begin{array}{l}
    \text{ASHORT}_{i,j,k-1} \\
    \text{ASHORT}_{i,k,k-1} + \text{ASHORT}_{k,j,k-1}
    \end{array} \right. & \text{k} \geq 1
\end{cases}
\]
FLOYD-WARSHALL(G, W)
INT GRAPH[128][128], N; // A WEIGHTED GRAPH AND ITS SIZE

void floydWarshall() {
  for (INT k = 0; k < N; k++)
    for (INT i = 0; i < N; i++)
      for (INT j = 0; j < N; j++)
        GRAPH[i][j] = min( GRAPH[i][j], GRAPH[i][k] + GRAPH[k][j] );
}

INT main {
  // INITIALIZE THE GRAPH[][] ADJACENCY MATRIX AND N
  // GRAPH[i][i] SHOULD BE ZERO FOR ALL I.
  // GRAPH[i][j] SHOULD BE "INFINITY" IF EDGE (I, J) DOES NOT EXIST
  // OTHERWISE, GRAPH[i][j] IS THE WEIGHT OF THE EDGE (I, J)
  floydWarshall();
  // NOW GRAPH[i][j] IS THE LENGTH OF THE SHORTEST PATH FROM I TO J
}
Max flow

Min Cut
“Consider a rail network connecting two cities by way of a number of intermediate cities, where each link of the network has a number assigned to it representing its capacity. Assuming a steady state condition, find a maximal flow from one given city to the other.”
We will finally describe a more recent and more peaceful application of flow methods to railways as used by Nederlandse Spoorwegen for Timetable 2007.

NS runs an hourly train service on its route Amsterdam-Rotterdam-Roosendaal-Vlissingen and vice versa with the timetable shown above. The trains have more stops but for our purposes only those given in the table are of interest since at the stations given train sections can be coupled or separated. For each of the stages of any scheduled trains NS has estimated the number of passengers as given in the table on the next page. All data concerns weekdays and 2nd class seats.

The problem to be solved is: What is the minimum amount of train stock necessary to perform this train service in such a way that at each stage there are enough seats?

In order to answer this question, one should know a number of further characteristics and constraints. In a first version of the problem considered, the train stock consisted of one type of two-way train units, each consisting of three carriages. Each unit has 60 seats.

Each unit has at both ends an engineer's cabin and units can be coupled together up to a certain maximum length, often 8 carriages, meaning in this case 8 train units. The train length can be changed by coupling or decoupling units at the terminal stations of the lines that is at Amsterdam and Vlissingen and en route at the intermediate stations Rotterdam and Roosendaal. Any train unit decoupled from a train arriving at place $p$ at time $t$ can be linked up to any other train departing from $p$ at any time later than $t$. The Amsterdam-Vlissingen schedule is such that in practice this gives enough time to make the necessary switchings.

A last condition is that for each place $p \in \{Amsterdam, Rotterdam, Roosendaal, Vlissingen\}$ the number of train units staying overnight at $p$ should be constant during the week but may vary for different places. This requirement is made to facilitate surveying the stock and to equalize at any place the load of overnight cleaning and maintenance throughout the week. It is not required that the same train units after a night in Roosendaal for example should return to Roosendaal at the end of the day. Only the number of units is of importance.

Given these problem data and characteristics, one may ask for the minimum number of train units that should be available to perform the daily cycle of train rides required. It is assumed that if there is sufficient stock for Monday till Friday then this should also be enough for the weekend services since at the weekend a few early trains are cancelled and on the remaining trains there is a smaller expected number of passengers. Moreover, it is not taken into consideration that stock can be exchanged during the day with other lines of the network. In practice this will happen but initially this possibility is ignored.

A network model

If only one type of railway stock is used, last
FLOW NETWORKS

\[ G = (V, E) \]

SOURCE + SINK:

CAPACITIES:
FLOW NETWORKS

\[ G = (V, E) \]

SOURCE + SINK: NODE S, AND T

CAPACITIES: \( c(u, v) \)

ASSUMED TO BE 0 IF NO \((u, v)\) EDGE
EXAMPLE
FLOW

MAP FROM EDGES TO NUMBERS:

CAPACITY CONSTRAINT:

FLOW CONSTRAINT:

\[ |f| = \]
MAX FLOW PROBLEM

GIVEN A GRAPH $G$, COMPUTE
GREEDY SOLUTION?