• Homework 5
  • Due Saturday at 2am (so, basically Friday night)
• Homework 6 won't be posted until next Friday
  • Enjoy your break!
• Problem A: Ferry Loading  
  • [http://acm.hust.edu.cn/vjudge/contest/view.action?cid=70745#problem/A](http://acm.hust.edu.cn/vjudge/contest/view.action?cid=70745#problem/A)

• Problem B: Winning Streak  
Problem: UVA 104: Arbitrage

Summary: Take advantage of currency fluctuations to make money.

Input: Table of exchange rates

Output: Shortest sequence of currencies to buy that will yield a profit of more than 1%
Dijkstra’s and Bellman-Ford give you single-source shortest path, what if you want to find all-pairs shortest path?

What happens if you want to find the shortest distance between all pairs of nodes?

- On a weighted, connected graph, use Floyd Warshall algorithm
- Implement in ~4 lines of code
- \(O(V^3)\) instead of \(V\) Dijkstra's algorithm, which would be \(O(V^3 \log V)\)
- Dynamic programming
// inside int main()
// precondition: m[i][j] contains the weight of edge (i, j)
// or INF if there is no such edge
// (m is an adjacency matrix)

for (int k = 0; k < V; k++)
    for (int i = 0; i < V; i++)
        for (int j = 0; j < V; j++)
            m[i][j] = min(m[i][j], m[i][k] + m[k][j]);

// common error: remember that loop order is k->i->j
• Intuition: gradually allow the usage of intermediate vertices [1..k]
• E.g., suppose we’re looking for the shortest path from 3 to 4:

![Diagram of a network with vertices 0, 1, 2, 3, 4 and edges between them.]

The current content of Adjacency Matrix $D$ at $k = -1$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>$\infty$</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>$\infty$</td>
<td>0</td>
<td>$\infty$</td>
<td>4</td>
<td>$\infty$</td>
</tr>
<tr>
<td>2</td>
<td>$\infty$</td>
<td>1</td>
<td>0</td>
<td>$\infty$</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>$\infty$</td>
<td>3</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>0</td>
</tr>
</tbody>
</table>

$\text{sp}(3, 2, -1) = 3$  $\text{sp}(2, 4, -1) = 1$  $\text{sp}(3, 4, -1) = 5$

We will monitor these two values.
• $k=0$ allows us to find shorter paths from $3\rightarrow 0\rightarrow 1$, $3\rightarrow 0\rightarrow 2$, $3\rightarrow 0\rightarrow 4$
• $k=2$ allows us to find shorter paths from $0 \rightarrow 2 \rightarrow 4$, $3 \rightarrow 2 \rightarrow 4$
• Recall that $3 \rightarrow 2 \rightarrow 4$ shortest path is actually $3 \rightarrow 0 \rightarrow 2 \rightarrow 4$

**Explanation of Floyd's**

**The current content of Adjacency Matrix $D$ at $k = 2$**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>$\infty$</td>
<td>0</td>
<td>$\infty$</td>
<td>4</td>
<td>$\infty$</td>
</tr>
<tr>
<td>2</td>
<td>$\infty$</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>0</td>
</tr>
</tbody>
</table>
**Exercise**: The diameter of a graph is the longest distance between any pair of vertices. Explain how to find the diameter of a graph.

**Exercise**: Explain how to find the strongly connected components of a graph in $O(V^3)$ time.
We are going to build a DP single source shortest path algorithm as follows (weighted directed graph with negative edges). The dp array `double[] dists` will store the length of the shortest path from the starting node \( v \) to vertex \( i \) using any walk of length \( k \) or less. The algorithm iterates \( k = 1, 2, \ldots \) updating dists at each step. Answer the following questions:

(a) How should we initialize the array dists?

(b) Explain how to update dists during each iteration.

(c) When can we stop iterating through \( k \)-values?

(d) How can we use the above algorithm to find negative cycles?
We are going to build a DP all pairs shortest path algorithm as follows (weighted directed graph with negative edges). The dp array `double[][] dists` will store the length of the shortest path from vertex `i` to vertex `j` only using vertices numbered less than `k`. The algorithm iterates `k = 1, 2, ...` updating `dists` at each step. Answer the following questions:

(a) How should we initialize the array `dists`?

(b) Explain how to update `dists` during each iteration.

(c) When can we stop iterating through `k`-values?

(d) How can we use the above algorithm to find negative cycles?