The amount of water flowing into node $x$ increases by 5. How much can the flow into adjacent nodes increase? The upper number $=$ the capacity; the lower $=$ the flow.

We are asking for potential individual increases; these increases are not simultaneously achievable.

$\textbf{Remaining capacity} = \text{edge capacity} - \text{edge flow}$. 
Suppose $x$ is labeled with a potential increase in the amount of water available to it. Suppose $y$ is an unlabeled node connected (in either direction) to $x$.

The potential increase at $y$ is:

- $x \to y$ edges: $\min(\text{potential increase at } x, \text{ remaining capacity})$.
- $y \to x$ edges: $\min(\text{potential increase at } x, \text{ flow on the edge})$.

Label $y$ with this increase if it is $>0$. Otherwise, leave $y$ unlabeled. Mark the edge with a check $\checkmark$. 
Max-flow algorithm. Don’t have to state it; be able to run it.

**Input:** A network.  
**Output:** A maximal flow and a minimal cut. 

**Comment:** At each step we label nodes with potential increases in flow. Then we either label edges with a new increased flow or make a minimum capacity cut.

- **Start with 0 flow along each edge.**

**A:** Label the source with $\infty$. (We now look for a path along which the current flow can increase.)

**B:** Pick an unlabeled node connected by an edge to a node with a positive potential increase. Label the unlabeled node with its potential increase. Put a check on the edge.

- If the sink is unlabeled and more nodes can be labeled, return to **B**.
- If the sink is labeled with an increase $d$, work backward to the source to get a path along the checked edges (forward or backward). For each forward edge along the path, increase the flow by $d$; for each backward edge, decrease its flow by $d$.
- Record the new flow (the changed flows along the path and the other unchanged flows); remove all potential increases on nodes and edge checks. Go to **A**.
- If no more nodes can be labeled and the sink is unlabeled, you have a maximal flow. Cut the edges which go from a labeled to an unlabeled node. This is a minimum cut. Stop.
Apply the max-flow algorithm to find a maximal flow and a minimum cut.
Apply the max-flow algorithm to find a maximal flow and a minimum cut.
Marriage problem.

There are 4 boys \( \{A, B, C, D\} \) and 5 girls \( \{M, N, O, P, Q\} \). “×” marks couples who will dance together.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>O</td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>P</td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>

Find a matching with the most dance couples.

Draw a graph which represents the possible choices. Then make this a network.
Marriage problem.

There are 4 boys \( \{A, B, C, D\} \) and 5 girls \( \{M, N, O, P, Q\} \). “×” marks couples who will dance together.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>O</td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>P</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>

Find a matching with the most dance couples.
MARRIAGE PROBLEM. Let the capacity of each edge be 1. Marriages correspond to flows in the network. A maximal flow will give a matching with the most couples.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>O</td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>P</td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>
**The shortest path problem**

**Given:** a connected undirected *weighted* graph with an *origin* node. Here, the weight of an edge is its length.

**Goal:** find the shortest path from each node to the origin. The *distance* between two nodes or between a node and an edge is the length of the shortest path which contains both of them, i.e., a path from the point to and including the edge.

- Find the distance between points a and b.
- Find the distance between point a and edge E.

![Graph diagram]

The distance of an edge to a node includes edge’s length.
Three edges leave the origin as shown below. Find the distances to the first three edges. Which node has a known distance? Which node distances are unknown? If the three red nodes shown are the closer to the origin than the other nodes, what are their distances?

When a node’s distance to the origin has been calculated, it becomes a *known* node. An edge between a known node and an unknown node is a *cut* edge. Its *estimated distance* is its known node’s distance plus its length.

The cut edges form a cut between the known nodes and the unknown nodes. Every path from the origin (which is a known node) to an unknown node must cross one of these *cut* edges.
All cut edges (edges between known and unknown nodes) are shown in red. Fill their boxes with their estimated distances. Find the correct distance for one of the unknown nodes. Put this distance in the circle. Mark an edge which gives this distance with an arrow.
**Lemma.**

- The distance from the origin to another node = the minimum of the distances to the node’s edges.
- If the estimated distance of a cut edge is ≤ the estimated distances of other cut edges, then the estimated distance is the true distance between the edge and the origin.

**Shortest path Algorithm**

**Input.** A weighted (distance) graph and an origin node. Put a box along each edge and a circle at each node.

**Goal.** Label each node with its distance to the origin. Add origin-pointing arrows to edges along minimal-length paths to the origin (cross off other edges).

**Invariants (facts that are true at each step).**

- Edges with numbers which have not been crossed off are cut edges (one node is known --has a number -- the other is not) Their numbers are their estimated distances.
- The number in a known node’s circle is its distance.
- Paths which follow the arrows back to the origin have minimal length.
Procedure.

- Put 0 in the origin circle and make it the initial **current node** (the newest known node).
- Loop:
  - If all nodes are known, stop. You are done.
  - For each edge between the current node and an unknown node (cut edge), put the current node’s distance plus the edge’s length in the edge’s box (these are estimated distances).
  - Pick an edge whose boxed number is minimal. The edge’s unknown node becomes the new current node.
    - Label this node with the edge’s boxed number.
    - Draw an arrow on the edge pointing back to the origin.
    - Cross off the numbers on cut edges leading to the current node.
- Repeat the loop: