Many network optimization models are actually special types of LP problems.

Four important kinds of network problems:

1. the shortest path problem
2. the minimum spanning tree problem
3. the maximum flow problem
4. the minimum cost flow problem

Network terminology:

**Nodes**

**Arcs:** directed arcs, undirected arcs (links)

**Networks:** directed networks, undirected networks

**Path:** directed path, undirected path

**Cycle**

**Connected:** connected nodes, connected network

**Spanning Tree:** a connected network with no undirected cycles

Corresponds to BFS for network simplex method.

**Arc capacity**

**Supply node, demand node, transshipment node**
**DIAGON ALLEY:** The pedestrian’s map.

![Diagram of Diagon Alley]

E is the entrance to diagon alley, and there is an exit to Charing Cross Road at R. Hagrid currently faces three problems:

[1] Determine which route from the entrance to Charing Cross Road has the **smallest total distance**.

[2] Floo network paths must be installed under the roads to establish communication among all the stations. Where should the paths be laid to provide connection between every pair of stations while **minimizing** the total paths laid?

[3] In case of an attach from Lord Voldemort, Diagon Alley is part of an escape route. There is a limit on pedestrian throughput on each path. How should the escape routes be planned so as to maximize the total throughput from the entrance to Charing Cross Road?
The Shortest Path Problem

Consider an undirected and connected network with two special nodes called the origin and the destination. Associated with each of the links (undirected arcs) is a non-negative distance. The objective is to find the shortest path from the origin to the destination.

Algorithm:

At iteration \( i \),

Objective: find the \( i \)th nearest node to the origin.

Input: \( i - 1 \) nearest nodes to the origin, shortest path and distance.

Candidates for \( i \)-th nearest node: nearest unsolved node to one solved.

Calculation for \( i \)-th nearest node: for each new candidate, add new edge to previously found shortest path to the \( i - 1 \)-th nearest node. Among the new candidates, the node with shortest distance becomes the \( i \)-th nearest node.

<table>
<thead>
<tr>
<th>Iteration, ( i )</th>
<th>Solved Nodes Directly Connected to Unsolved Nodes</th>
<th>Closest Connected Unsolved Node</th>
<th>Total Distance Involved</th>
<th>( i )-th Nearest Node</th>
<th>Minimum Distance</th>
<th>Last Connection</th>
</tr>
</thead>
</table>