SAPIENZA
UNIVERSITÀ DI ROMA

"Sapienza" University of Rome
Faculty of Information Engineering,
Informatics and Statistics
Department of Computer Science

Network Algorithms

Author
Simone Bianco

September 24, 2024

Contents

Information and Contacts

Personal notes and summaries collected as part of the *Network Algorithms* course offered by the degree in Computer Science of the University of Rome "La Sapienza".

Further information and notes can be found at the following link:
https://github.com/Exyss/university-notes. Anyone can feel free to report inaccuracies, improvements or requests through the Issue system provided by GitHub itself or by contacting the author privately:

- Email: **bianco.simone@outlook.it**
- LinkedIn: **Simone Bianco**

The notes are constantly being updated, so please check if the changes have already been made in the most recent version.

Suggested prerequisites:

Sufficient knowledge of computer networks, graph theory and algorithms design

Licence:

1 Wired network problems

1.1 Introduction on graph

In many network applications, graphs are used as a natural model. In other applications, the graph model may be less obvious, but appears to be anyway very useful. Graph algorithms are useful instruments to solve important and living problems. We will see a number of advanced techniques for efficient algorithm design to solve problems from networks and graphs.

Definition 1: Graph

A **graph** is a mathematical structure $G = (V, E)$ made of a set V called the *vertex set* (or *node set*), and a set $E \subseteq V \times V$ called *edge set*.

Graphs are usually represented through circles and lines, were each line between two vertices u, v represents the edge (u, v). We will assume to be working with *simple graphs*, a type of graph that doesn't allow loop edges, i.e. edges from a node to itself, or a multiple number of edges between two vertices.

The edges of a graph can also be *directed* or *undirected*. In the former, the two edges (u, v) and (v, u) are considered two distinct edges while in the latter they are considered as the same edge. A directed graph is usually also referred to as **digraph**.

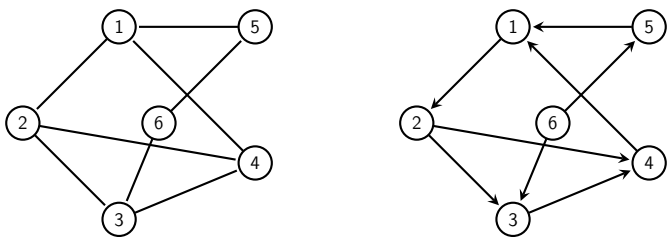

Figure 1.1: On the left: a simple graph. On the right: a simple digraph

Graphs were born in 1736, when Euler used them formalize and solve the famous *Seven Bridges of Königsberg* problem: is there a way to walk through all the bridges of the town and end up on the starting point?

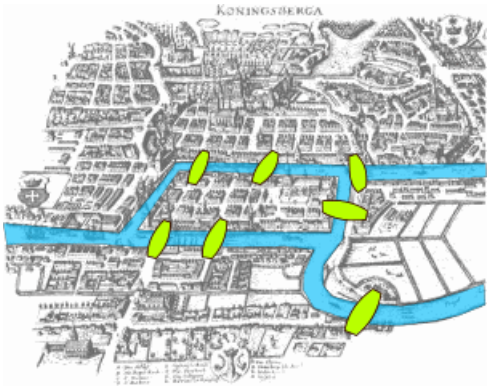

Figure 1.2: The city of Königsberg and its seven bridges

To solve the problem, Euler represented the problem as the following *multi-graph*, i.e. a non-simple graph that allows multiple edges between two vertices. Euler proved that the answer to the question is negative: a walk that passes through all the edges of such graph while also returning to the starting node cannot exist.

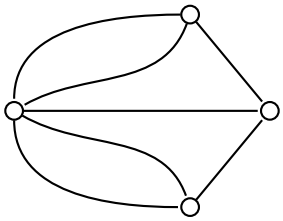

Figure 1.3: The multi-graph representing the Seven Bridges of Königsberg problem

In general, a **walk** on a graph G is given by a sequence of nodes $v_1, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E(G)$. A **path** is walk whose vertices are all distinct. As we'll see in the following sections, walks and paths are the basis of graph theory.

1.2 The routing problem

When packets are sent from a computer to another through a network, each computer has to route data on a path passing through intermediate computers. This problem is usually referred to as the **routing problem**.

By modelling the network as a graph whose vertices correspond to the computers and its edges correspond to the links between them, such problem is reduced to the concept of a path from an initial node to an arrival node.

Based on the required conditions, the routing reduces to a specific type of path problem:

1. In **non-adaptive routing**, the routing algorithm must minimize the number of intermediate computers on the route. This problem reduces to the *shortest path problem*, i.e. finding the path that passes through the lowest amount of edges from node s to node t. This type of routing gives good results with consistent topology and traffic conditions, but performs poorly in case of congestion.
2. In **adaptive routing**, the routing algorithm must take into account the traffic conditions: if a route is congested, we want to avoid it in. This problem reduces to the *least cost path problem*, i.e. finding the path with the least cost from node d to node t. This type of routing gives good results with high network workload, but routes must be computed frequently in order to perform well.
3. In **fault-sensitive routing**, the routing algorithm must consider the possibility of a link failing: we want the route with the highest probability of working.

Each of these problems can be modeled as a graph. In particular, adaptive routing and fault-sensitive routing need an additional *weight function* $w : E(G) \to \mathbb{R}$ such that $w(e)$ represents the weight of an edge $e \in E(G)$. The **weight (or cost) of a path** P, written as $w(P)$, is the sum of the edges that compose it.

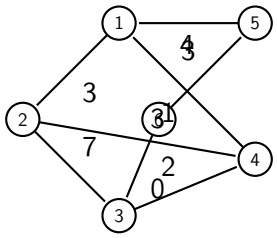

Figure 1.4: The path $P = 1, 2, 4, 3$ has weight $w(P) = 8$

The weight measure varies based on the context. In adaptive routing the traffic acts as the weight, while in fault-sensitive routing the probability acts as the weight. In particular, let $p(u, v)$ be the probability that an edge $(u, v) \in E(G)$ doesn't fail. Under the not-so-realistic assumption that edge failures occur independently of each other, we get that the probability that a path $P = v_1, \ldots, v_k$ doesn't fail is given by $p(v_1, v_2) \cdot \ldots p(v_{k-1}, v_k)$.

By setting each weight $w(u, v)$ equal to $-\log(p(u, v))$, we get that the product $p(v_1, v_2) \cdot \ldots p(v_{k-1}, v_k)$ reaches its maximum when the sum $w(v_1, v_2) + \ldots + w(v_{k-1}, v_k)$ reaches

its minimum. Through this weight function, fault-sensitive routing is also reduces to the least cost path problem.

Similarly, the shortest path problem can also be reduced to the least cost path problem by setting $w(u, v)$ equal to 1 for each edge. One problem to rule them all!

Definition 2: Distance

Let $G = (V, E)$ be a graph. Given two nodes $u, v \in V(G)$, the **distance** between u and v, written as $\text{dist}(u, v)$, is the minimum weight of all the paths $u \to v$ of G.

On digraphs the concept of distance is non-symmetrical: the distance $\text{dist}(u, v)$ may be different from the distance $\text{dist}(v, u)$. Moreover, when there is no path $u \to v$, we assume that $\text{dist}(u, v) = +\infty$.