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

# Network Algorithms

Lecture notes integrated with the book TODO

*Author*
Alessio Bandiera

September 29, 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/aflaag-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: **alessio.bandiera02@gmail.com**
- LinkedIn: **Alessio Bandiera**


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

**Suggested prerequisites:**

- Progettazione di Algorithmi

**Licence:**

# 1

# TODO

## 1.1 TODO

### 1.1.1 Classical solutions

**Algoritmo 1.1.1.1** *Bellman-Ford*: TODO

```
function BELLMANFORD(G)
    TODO
end function
```

**Algoritmo 1.1.1.2** *Dijkstra*: TODO

```
function DIJKSTRA(G)
    TODO
end function
```

**Algoritmo 1.1.1.3** *Floyd-Warshall*: Given a directed graph $G$, and an unconstrained weight function $w$ for the edges, the algorithms returns a matrix `dist` such that `dist`$[u][v]$ is the weight of the least-cost path from $u$ to $v$.

1: **function** FLOYDWARSHALL($G$, $w$)<br>
2:     Let `dist`$[n][n]$ be an $n \times n$ matrix, initialized with every cell at $+\infty$<br>
3:     **for** $u \in V(G)$ **do**<br>
4:         `dist`$[u][u] = 0$<br>
5:     **end for**<br>
6:     **for** $(u, v) \in E(G)$ **do**<br>
7:         `dist`$[u][v] = w(u, v)$<br>
8:     **end for**<br>
9:     **for** $k \in V(G)$ **do**<br>
10:         **for** $u \in V(G)$ **do**<br>
11:             **for** $v \in V(G)$ **do**<br>
12:                 `dist`$[u][v] = \min($`dist`$[u][k]$, `dist`$[k][v])$<br>
13:             **end for**<br>
14:         **end for**<br>
15:     **end for**<br>
16: **end function**

*Idea.* The core concept of the algorithm is to construct a matrix using a dynamic programming approach, that evaluates all possible paths between every pair of vertices. Specifically, to determine the shortest path from a vertex $u$ to a vertex $v$, the algorithm considers two options: either traveling directly from $u$ to $v$, or passing through an intermediate vertex $k$, potentially improving the path.

*Cost analysis.* The `for` loop in line 3 has cost $\Theta(n)$, the `for` loop in line 6 has cost $\Theta(m) = \Theta(n^2)$ and the cost of the triple nested `for` loop is simply $\Theta(n^3)$. Therefore, the cost of the algorithm is

$$\Theta(n) + \Theta(n^2) + \Theta(n^3) = \Theta(n^3)$$