



SAPIENZA
UNIVERSITÀ DI ROMA

“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	1
1 TODO	2
1.1 TODO	2
1.1.1 Classical solutions	2

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 Algoritmi

Licence:

These documents are distributed under the [GNU Free Documentation License](#), a form of copyleft intended for use on a manual, textbook or other documents. Material licensed under the current version of the license can be used for any purpose, as long as the use meets certain conditions:

- All previous authors of the work must be **attributed**.
- All changes to the work must be **logged**.
- All derivative works must be **licensed under the same license**.
- The full text of the license, unmodified invariant sections as defined by the author if any, and any other added warranty disclaimers (such as a general disclaimer alerting readers that the document may not be accurate for example) and copyright notices from previous versions must be maintained.
- Technical measures such as DRM may not be used to control or obstruct distribution or editing of the document.

1

TODO

1.1 TODO

1.1.1 Classical solutions

Algoritmo 1.1.1.1 *Bellman-Ford*: TODO

```
1: function BELLMANFORD( $G$ )  
2:   TODO  
3: end function
```

Algoritmo 1.1.1.2 *Dijkstra*: TODO

```
1: function DIJKSTRA( $G$ )  
2:   TODO  
3: end function
```

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

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

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