DEMOCRATIC AND PEOPLE'S REPUBLIC OF ALGERIA
MINISTRYOF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH
MOHAMED BOUDIAF
UNIVERSITY - M'SILA

Faculty of Mathematics and Computer Science

Computer Science department

N°: ……

FIELD: Mathematics and Computer Science

SECTOR: Computer Science

OPTION: BUSINESS INTELLIGENCE AND OPTIMIZATION (IDO)

A thesis submitted for obtaining
Academic master

By: CHOUDER IMANE

Entitled

Search of The Optimum Itinerary on a Digitized Road Network

Defended on ...../2021

Board of Examiners:

……	M'sila University	President
Dr. GADRI Said	M'sila University	Supervisor
……	M'sila University	Examiner

Academic year: 2020 / 2021

DEMOCRATIC AND PEOPLE'S REPUBLIC OF ALGERIA
MINISTRYOF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH
MOHAMED BOUDIAF
UNIVERSITY - M'SILA

Faculty of Mathematics and Computer Science

Computer Science department

N°: ……

FIELD: Mathematics and Computer Science

SECTOR: Computer Science

OPTION: BUSINESS INTELLIGENCE AND OPTIMIZATION (IDO)

A thesis submitted for obtaining
Academic master

By: CHOUDER IMANE

Entitled

Search of The Optimum Itinerary on a Digitized Road Network

Defended on ...../2021

Board of Examiners:

……	M'sila University	President
Dr. GADRI Said	M'sila University	Supervisor
……	M'sila University	Examiner

Academic year: 2020 / 2021

Dedication

I dedicate this work to my parents who have always given me everything I need to be the person I am.

I dedicate it to my sister, my brothers, my extended family, to all of my friends and colleagues, and finally to everyone who contributed directly or indirectly to the completion of this work, even by supplication.

Thanks

First, I would like to thank God for giving me the strength, will, and patience to complete this humble work. After that, I would like to thank Dr Gadri Said, my supervisor, for his management of this work, with my best wishes for complete health, longevity, and lasting tenderness. I also would like to thank Dr.Hemmak Allaoua, and I also do not forget to thank all the professors of the Faculty of Mathematics and Computer Science at the University of M'sila in general, and Computer science professors in particular. I would like to thank all the people who helped me directly or indirectly in completing this work.

Table of content

List of tables

List of figures

List of Algorithms

List of abbreviations

- **Fig: Figure.**
- **Tab: Table.**
- **Algo: Algorithm.**
- **CSC: Search algorithm for a simply connected component.**
- **CFC: Algorithm for finding a strongly connected component.**
- **SP: Shortest Path.**
- **SPP: Shortest Path Problem.**
- **WPF: Windows Presentation Foundation.**
- **SQL: Structured Query Language.**
- **RDBMS: Relational Database Management System.**
- **SSMS: SQL Server Management Studio.**
- **BFS: Breadth Search First.**
- **LP: Linear Program.**
- **GA: Greedy algorithm.**
- **LS: Local Search.**
- **SA: Simulated Annealing.**
- **HC: Hill Climbing.**
- **TS: Tabu Search.**
- **SS: Scratcher Search.**
- **GA: Genetic Algorithm.**
- **ACO: Ant colony optimization.**
- **BCO: Bee colony optimization.**
- **PSO: particle swarm optimization.**

General Introduction

General Introduction

The road network is constantly growing as a result of reconstruction in complete To facilitate the study of road networks we digitize them, and satellites are an effective tool for capturing network updates. Medium-quality images can be obtained for free by some satellites such as LANDSAT, while there are high-resolution satellites but are very expensive.

We are in a time of speed and every saved second is important and beneficial, so as much as we care to continue our road and reach we care to reach quickly! That is why the science of graph theory presented to us what is known as the optimum Itinerary on a road network, and this is what we will focus our study on ! ..

The study of graphs, which are mathematical structures used to describe pairwise relationships between objects, is known as graph theory in mathematics. In this sense, a graph is made up of vertices (also known as nodes or points) linked by edges (also called links or lines). There is a difference drawn between directed and undirected graphs, wherein edges symmetrically connect two vertices, and directed networks, in which edges connect two vertices asymmetrically. Graphs are one of the most important topics in discrete mathematics.[1]

Our project consists to a comprehensive study of the graph theory. An in-depth study of SPP, and the creation of a desktop application named ‘SPP App’, this application requests of a Weighted directed graph information; Such as its name, size, set of arcs, and its direction from one vertex to another. The APP shows according to the user's request:

- ❖ SP by Dijkstra's algorithm.
- ❖ SP by Bellman's algorithm.
- ❖ List of vertices.
- ❖ List of Arcs.
- ❖ List of successors.
- ❖ List of predecessors.
- ❖ Adjacence matrix.
- ❖ Incidence matrix.

We divided our manuscript into several chapters:

- Starting with a general introduction.
- In chapter 1: we presented some important concepts and definitions, including graph theory, and some types and rules of graph theory.
- In chapter 2: we briefly described the area of our project, which is to find the optimum route in a digital network.
- Chapter 3: describes the programming environment.
- Chapter 4: we presented realized application.
- Finally, we concluded our work with a general conclusion.

Chapter 1: The Foundations of Graph Theory

1.1. Graph-theory

1.1.1. Definition of Graph theory

It is a sub-field of mathematics which deals with graphs, we can say that it is the study of lines and points, or it is the study of the relationship between edges and vertices.

1.1.2. History

Graph Theory dates back to 1735 and Euler's Seven Bridges of Konigsberg. The city of Konigsberg was a town with two islands, connected and to the mainland by seven bridges. The question set was whether it was possible to take a walk and cross each bridge exactly once. In the first demonstration of graph theory, Euler showed that it was not possible.[2]

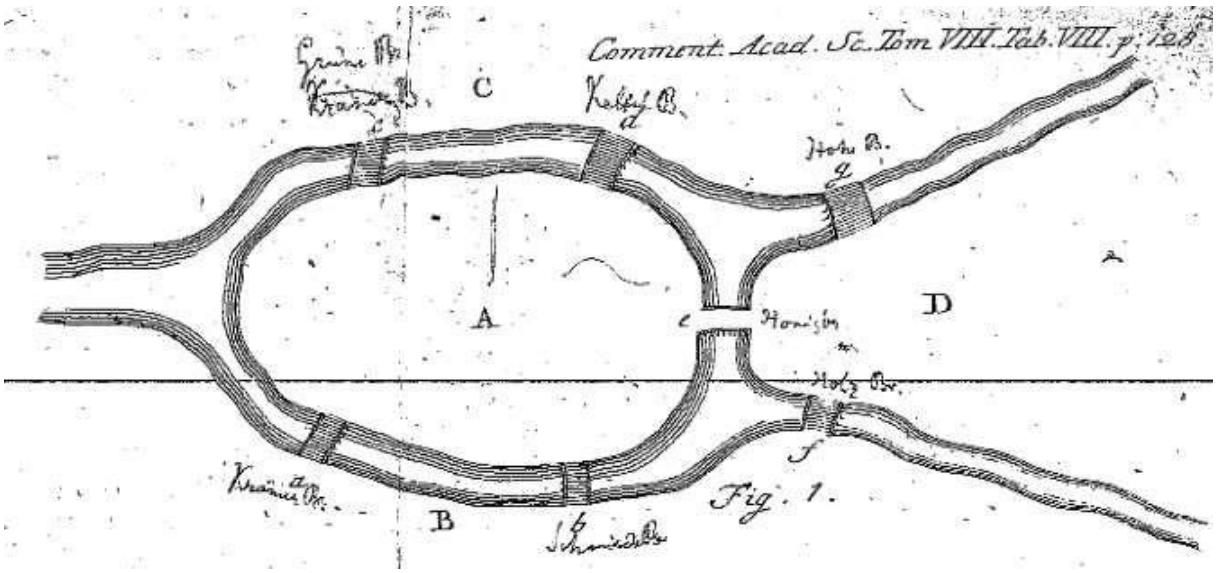


Fig1.1: Euler's Seven Bridges of Konigsberg[3]

1.1 Graph

1.2.1 Definition of Graph

The graph is a network or we say that the graph is used to designate representation mathematical of a network.

So generally: "Network" ≡ "Graph", As a more accurate definition:

a Graph is a collection of connected vertices by **edges** (in **Undirected graph= symmetric graph**) or **arcs**.(in **Directed graph= Di-graph**)[4]

We call a graph the pair G (X, U) such that:

- X ={x1, x2, ….., xn} is the set of vertices of the graph.
- U = {u1, u2,…., Un} is the set of arcs of the graph
- U □ XxX

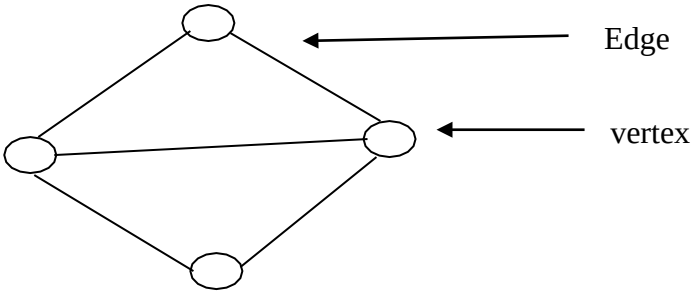


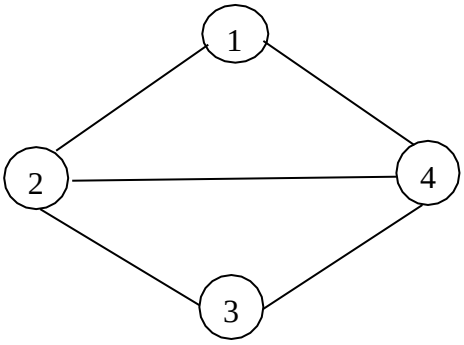
Figure 1.2.: Definition of a graph.

1.2.2. Properties of a graph

A graph G contains a vertex xi is represented by a point and arc U = (xi, xj) ∈ XxX is represented by an arrow or a line segment (depending on the type of oriented / non-oriented graph)

1.2.3. Order of the graph:

The order of the graph G is the number of vertices n = | X |.

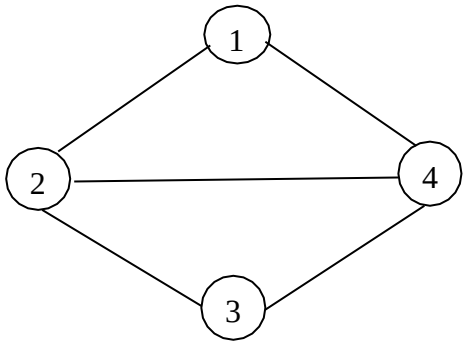


n=|X|=4

Fig 1.3.: Order of the graph.

1.2.4. size of the graph

the size of the graph G is the number of Edges/Arcs m = | U |.



m=|U|=5

Fig 1.4.: Size of the graph.

1.2.5. Definition of loop in Graph

If xi = xj ==> u = (xi, xj) is represented by a loop (i.e. the two vertices are combined).

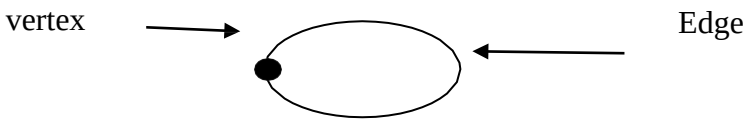
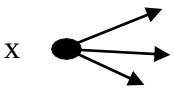


Fig 1.5.: Loop in graph.

1.2.6. The external degree of a vertex x

The external degree of the vertex x is the number of arcs of which x is the initial end or the number of outgoing arcs of x. We denote by d + (x).[1]

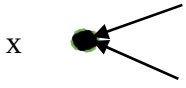


d + (x) = 3

Fig 1.6.: external degree of a vertex x.

1.2.7. The interior degree of a vertex x

The interior degree of the vertex x is the number of arcs of which x is the terminal end of the number of incoming arcs in x and We denote by d-(x).[1]

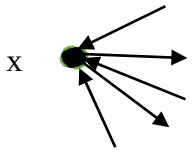


d - (x) = 2

Fig 1.7.: external degree of a vertex x.

1.2.8. The degree of a vertex x

The degree of the vertex x is the number of arcs of which x is the initial or terminal end , and we denote d(x) = d+ (x) + d- (x). [1]



d(x) = d + (x) + d- (x)
d(x) = 2 + 3
d(x) = 5

Fig 1.8.: The degree of a vertex x.

1.2.9. Predecessor and Successor

If (x, y) ∈ U, then x is said to be the predecessor of y and y is the successor of x.

Fig 1.9.: Predecessor and Successor.

Also, we call:

Γ+(x): the set of successors of x.
Γ-(x): the set of predecessors of x.

1.2.10. Source vertex and Sink vertex.

A source vertex s is a vertex which has no predecessor (Γ - (s)= ϕ / there are only outgoing arcs). But A sink vertex is a vertex which does not have successor (Γ + (x) = ϕ / there are only incoming arcs). [5]

Fig 1.10.: Source vertex and ending vertex.

1.2.11. Graph types

1.2.11.1. A valued graph

is a graph G (X, U, P) such that we associate with the graph G (X, U) a function

F defined as follows:

F (u) = P is called the weight of the arc u, and we denote by P(x, y) or P(u).[6] [1]

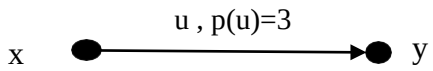


Fig 1.11.: A valued graph.

1.2.11.2.Complete graph

A graph G (X, U) is complete if and only if:

$\forall x, y \in X, (x, y) \in U$ and $(y, x) \in U$ [3][6] [1]

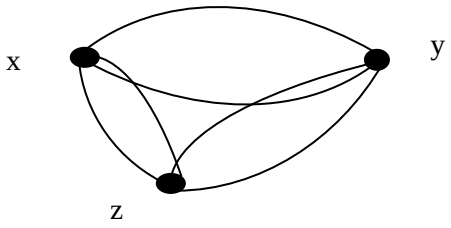


Fig 1.12.: Complete graph.

1.2.11.3. Simple graph

A graph G (X, U) is a simple graph if and only if:

- Contains no loops.
- $x, y \in X, \exists$ most $u= (x, y) \in U$.

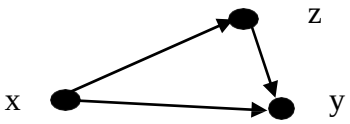


Fig 1.13.: Simple graph.

1.2.11.4Empty graph

A graph G (X, U) is empty graph if there is no vertex, arc or edge ($X=\phi$, $U =\phi$).

1.2.11.5Trivial graph

A graph G (X, U) is a trivial graph if there are vertices, but there is no arc or edge:

($U =\phi$)

Fig 1.14.: Trivial graph.

1.2.11.6Refletive graph

A graph G (X, U) is a reflexive graph, if there is a loop in each vertex x of G :

$\forall x \in X \Longrightarrow (x, x) \in U$.

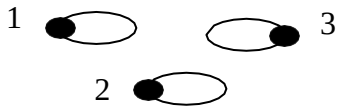


Fig 1.15.: Refletive graph.

1.2.11.7Transitive graph

A graph G (X, U) is a transitive graph if:

$\forall x, y, z \in X$, si $(x, y) \in U$ et $(y, z) \in U \Longrightarrow (x, z) \in U$

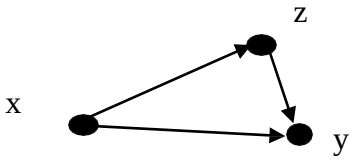


Fig 1.16.: Transitive graph.

1.2.11.8The inverse of a graph

The inverse of a graph G (X, U) is the graph G '(X, U') deduced of G by reversing the direction of its arcs U.

if the inverse of G is G' ===> the inverse of G' is G.

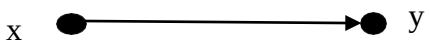


Fig 1.17.: a graph G.

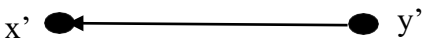


Fig 1.18.: G' the inverse of the graph G.

1.2.11.9Complementary graph

The complementary graph of G (X, U) is G'(X, U') such that:

$(x, y) \in U \Longrightarrow (x, y) \notin U'$[1]

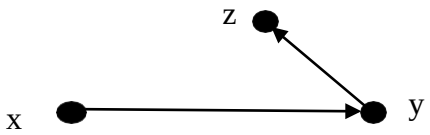


Fig 1.19.: A graph G.

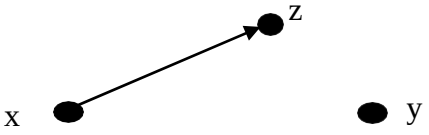


Fig 1.20.: G' Complementary graph of G.

1.2.11.10 Multigraph

in a directed graph G(X, U), x; y ∈ X if x; y are linked by two arcs u1,u2 of the same direction:

G is a multigraph and the arc (x, y) is a multiple arc or symmetric arc.

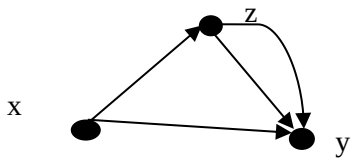


Fig 1.21.: Multigraph graph.

1.2.11.11 Bipartite graph

A bipartite graph (or bi-graph) is a graph of which the set of its vertices can be divided into two disjoint subsets of vertices X1 and X2 so that two vertices of the same subset are not adjacent, G(X, U) with:

$X = X1 \cup X2$ and $X1 \cap X2 = \phi$

and if u=(x1, x2) and $x1 \in X1$ ===> $x2 \in X2$. ; $x1 \in X2$ ===> $x2 \in X1$. [1]

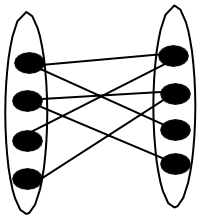


Fig 1.22.: Multigraph graph.

1.2.11.12 Planar graph

G is a planar graph if it is possible to draw it in the plane so that any two arcs do not intersect except at their ends.

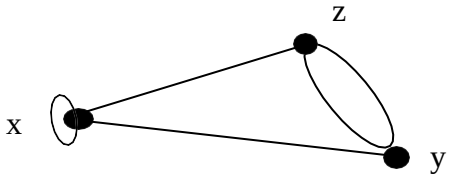


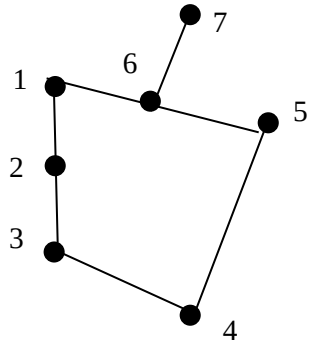
Fig 1.23.: Planar graph.

1.2.11.13 Isomorphic graphics

Two graphs G (X, U), G '(X', U ') are isomorphic if there is a bijection f: X→ X'

Such that: (x, y)∈U ⇔ (f(x), f(y)) ∈U'

if the two vertices x, y are linked by an arc in G ===> the image of x and the image of y by f are also linked by an arc in G '. [1]



F(1)=a
F(2)=h
F(3)=f
F(4)=e
F(5)=d
F(6)=b
F(7)=c

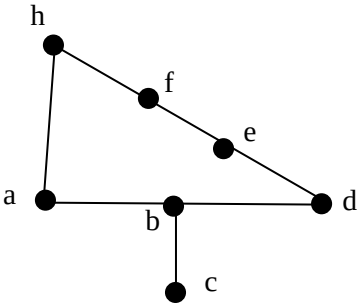
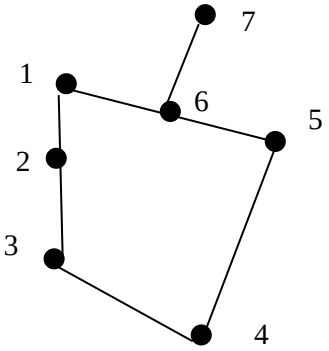


Fig 1.24.: Graph G. **Fig 1.25.: Isomorphic Graph G'.**

1.2.11.14 Subgraph

G '(X',U') is a subgraph of G means G' is G with removing some vertices and arcs , so X'⊂X and U '= {u ∈ U / both ends of u belong to X'}.



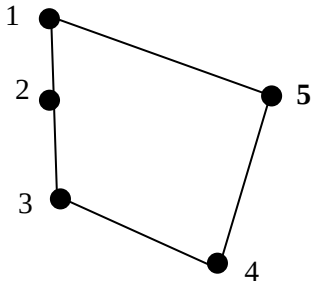


Fig 1.26.: Graph G. **Fig 1.27.: G' Subgraph of G**

1.2.11.15 Partial graph

G '(X', U ') is a partial graph of G(X, U), if X' = X, U'⊂U

so G' has the same set X of vertices with G, but a different set of arcs U'≠U.

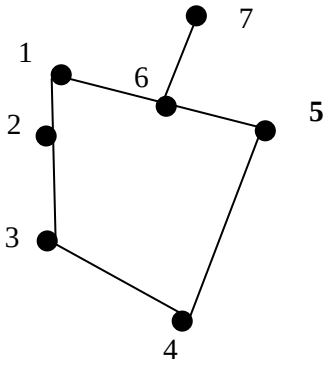


Fig 1.28.: Graph G.

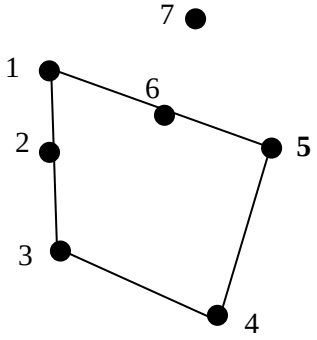


Fig 1.29.: G' partial graph of G.

1.2.11.16 Partial subgraph

G '(X', U ') is a partial subgraph of G (X, U), if G ' is the graph G with removing some vertices and some arcs from G.

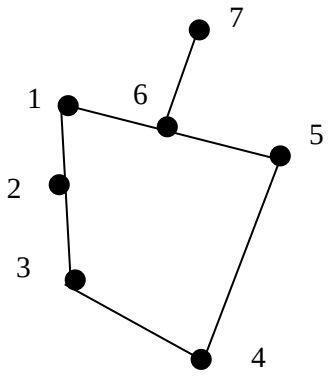


Fig 1.30.: Graph G.

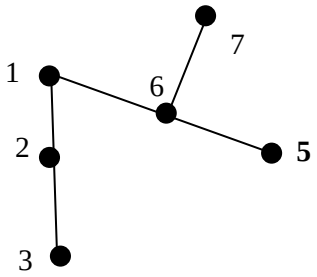


Fig 1.31.: G' partial subgraph of G

1.2.12. basic terms in Graph Theory

Term	**definition**
Path	It is a group of interconnected sequential vertices by arcs
Elementary path	It is a path that passes through all the arcs of the graph once and only once
Hamiltonian path	It is a path which passes through all the vertices of the graph
Circuit	It is a simple and closed path
Elementary circuit	It is a circuit which passes through all the arcs of the graph once and only once.
Hamiltonian circuit	It is a circuit which passes through all the vertices of the graph

Tab 1.1. : basic terms.

1.2.13. Graph representation methods

1.2.13.1. Static methods

these methods represent graphs using matrices:

1) Adjacency matrix (vertices - vertices) :

An adjacency matrix is a square matrix used to represent a finite graph in graph theory and computer science. The matrix's entries show whether two vertices in the graph are adjacent or not. The adjacency matrix is a (0,1)-matrix with zeros on its diagonal in the particular situation of a finite simple graph.

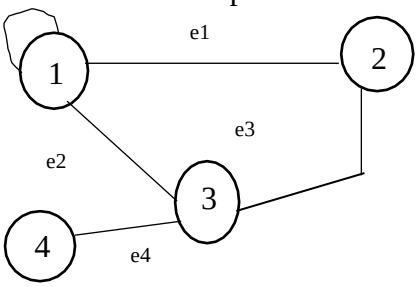
<table>
<tr><th>Matrix</th><th colspan="2">Adjacency matrix</th></tr>
<tr><td>Cases</td><td>In indirecte graph</td><td>In directe graph</td></tr>
<tr><td>Values</td><td>Gij=1 : If the vertex vj and vertex vi are adjacency .
Gij=0 : If the vertex vj and vertex vi are not adjacency .
Gij=2 : If there is a loop in vertex vi.</td><td></td></tr>
<tr><td>Exemples</td><td>Graph G1
 

Fig 1.32.: Graph G1.

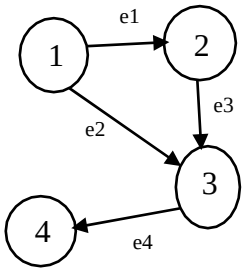
Adjacency matrix
<table><tr><td></td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>1</td><td>2</td><td>1</td><td>1</td><td>0</td></tr><tr><td>2</td><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>3</td><td>1</td><td>1</td><td>0</td><td>1</td></tr><tr><td>4</td><td>0</td><td>0</td><td>1</td><td>0</td></tr></table>Tab 1.2.: Adjacency matrix.</td><td>Graph G2
 

Fig 1.33.: Graph G2.

Adjacency matrix

If the graph G2 are direct the adjacency matrix is not necessarily.</td></tr>
</table>

Tab 1.3.: Adjacency matrix.

2) Incidence matrix (Vertices - edges):

In graph theory, an incidence matrix is a popular graph representation, represent the connection between vertex and edge pairs.

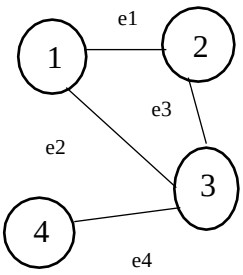
<table>
<tr><td>Matrix</td><td colspan="3">Incidence matrix</td></tr>
<tr><td>Cases</td><td>In indirecte graph</td><td>In directe graph</td><td>In wrightie graph</td></tr>
<tr><td>Values</td><td>Gij=1 : If the edge ej and vertex vi are incident .
Gij=0 : If the edge ej and vertex vi are not incident .</td><td>Gij=1 : If the edge ej enters vertex vi .
Gij=0 : If the edge ej and vertex vi are not incident .
Gij=-1 : If the edge ej leaves vertex vi .</td><td>Gij=Wej : If the edge ej and vertex vi are incident .
Gij=0 : If the edge ej and vertex vi are not incident .</td></tr>
<tr><td>Exemples</td><td>Graph G1
 

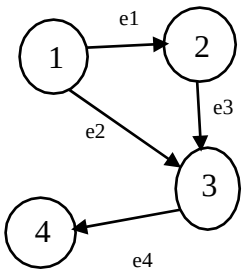
Fig 1.34.: Graph G1.
Incidence matrix M (N,Nedge)
<table><tr><td></td><td>e1</td><td>e2</td><td>e3</td><td>e4</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td>2</td><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>3</td><td>0</td><td>1</td><td>1</td><td>1</td></tr><tr><td>4</td><td>0</td><td>0</td><td>0</td><td>1</td></tr></table>Tab 1.4.: Adjacency matrix of G1.</td><td>Graph G2
 

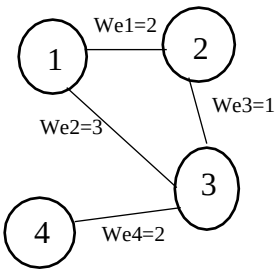
Fig 1.35.: Graph G2.
Incidence matrix M (N,Nedge)
<table><tr><td></td><td>e1</td><td>e2</td><td>e3</td><td>e4</td></tr><tr><td>1</td><td>-1</td><td>-1</td><td>0</td><td>0</td></tr><tr><td>2</td><td>1</td><td>0</td><td>-1</td><td>0</td></tr><tr><td>3</td><td>0</td><td>1</td><td>1</td><td>-1</td></tr><tr><td>4</td><td>0</td><td>0</td><td>0</td><td>1</td></tr></table>Tab 1.5.: Adjacency Matrix of G2.</td><td>Graph G3
 

Fig 1.36.: Graph G3.
Incidence matrix M (N,Nedge)
<table><tr><td></td><td>e1</td><td>e2</td><td>e3</td><td>e4</td></tr><tr><td>1</td><td>2</td><td>3</td><td>0</td><td>0</td></tr><tr><td>2</td><td>2</td><td>0</td><td>1</td><td>0</td></tr><tr><td>3</td><td>0</td><td>3</td><td>1</td><td>2</td></tr><tr><td>4</td><td>0</td><td>0</td><td>0</td><td>2</td></tr></table>Tab 1.6.: Adjacency Matrix of G3.</td></tr>
</table>

Tab 1.7.: Incidence matrix

3) List of successors:

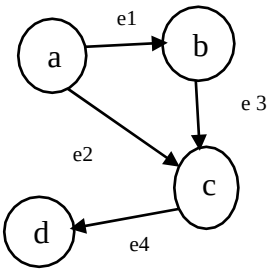
<table>
<tr><th colspan="2">Example of list of successors</th></tr>
<tr><td>Graph G</td><td> 

Fig 1.37.: Graph G.</td></tr>
<tr><td>List of successors</td><td>M (N, dmax +)
<table>
<tr><td>1</td><td>a</td><td>b</td><td>c</td></tr>
<tr><td>2</td><td>b</td><td>c</td><td></td></tr>
<tr><td>3</td><td>c</td><td>d</td><td></td></tr>
<tr><td>4</td><td>d</td><td></td><td></td></tr>
</table>
Tab 1.8.: List of successors.</td></tr>
</table>

Tab 1.9.: list of successors

4) List of predecessors:

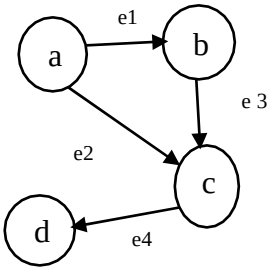
<table>
<tr><th colspan="2">Example of list of predecessors</th></tr>
<tr><td>Graph G</td><td> 

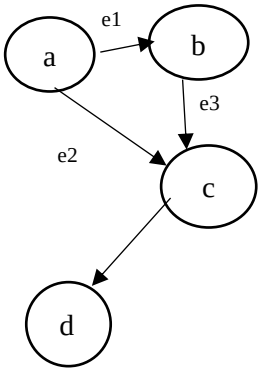
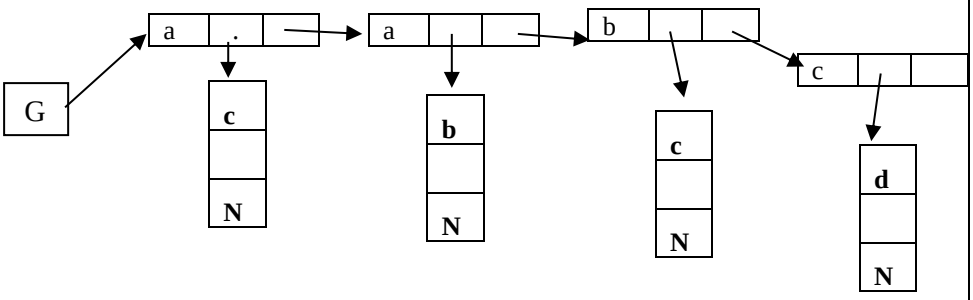
Fig 1.38.: Graph G.</td></tr>
<tr><td>List of predecessors</td><td>M (N, dmax -)
<table>
<tr><td>1</td><td>a</td><td></td><td></td></tr>
<tr><td>2</td><td>b</td><td>a</td><td></td></tr>
<tr><td>3</td><td>c</td><td>a</td><td>b</td></tr>
<tr><td>4</td><td>d</td><td>c</td><td></td></tr>
</table>
Tab 1.10.: List of predecessors.</td></tr>
</table>

Tab 1.11.: list of predecessors.

1.2.13.2. Dynamic methods

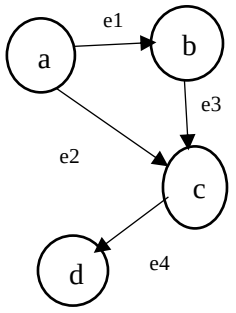
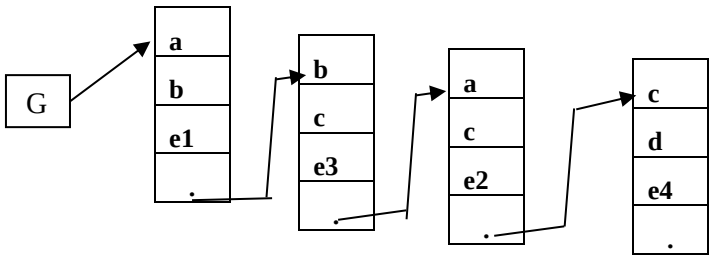
these methods represent graphs using linked lists:

1) Dynamic list of vertices

Example of Dynamic list of vertices	
Graph G	 **Fig 1.39.: Graph G.**
Dynamic list of vertices	 **Fig 1.40.: Dynamic list of vertices of graph G.**

Tab 1.12.: Dynamic list of vertices.

2) Dynamic list of arcs

Example of Dynamic list of arcs	
Graph G	 **Fig 1.41.: Graph G.**
Dynamic list of arcs	 **Fig 1.42.: Dynamic list of arcs of graph G.**

Tab 1.13.: Dynamic list of arcs.

1.2.14. Some famous algorithms in graph theory

1.2.14.1. Search algorithm for a simply connected component CSC of a vertex S

a component of an graph is an induced subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices in the rest of the graph, And by this algorithm we can find simply connected component CSC of a vertex S:

S is a vertex in the graph G (X, U) :

1) Mark the vertex S (by *).
2) Mark any adjacent vertex (next / previous) to an already marked vertex.
3) Repeat (2) until no more peaks can be marked.
4) The vertices marked by (*) form the simply connected component of S.

Algo 1.1.: CSC Algo.

1.2.14.2. Algorithm for finding a strongly connected component CFC of a vertex S

a graph is said to be strongly connected if every vertex is reachable from every other vertex. The strongly connected components of an arbitrary directed graph form a partition into

subgraphs that are themselves strongly connected.

And by this algorithm we can find a strongly connected component CFC of a vertex S:

S is a vertex in the graph G (X, U):

1) Mark the vertex S with (+ and -)

2) Mark with (+) any following (not yet marked +) from a vertex already marked(+)

3) Mark with (-) any preceding (not yet marked -) of a vertex already marked (-)

4) The vertices marked with both + and - form a CFC containing S

Algo 1.2.: CFC Algo.

Chapter 2: Shortest Path Problem (SPP)

2.1. Shortest path problem

2.1.1. Definition of SPP

This is the graph G = (X, U, L) with X = {1, 2,…, n}; U = {u1, u2,…, um}
L is the set of values on the arcs called lengths, The problem is to find a shorter path SP for one of the following cases:

- From a given vertex x to another given vertex y.
- From a given vertex x to all other vertices.
- From all vertex x to all vertex y.

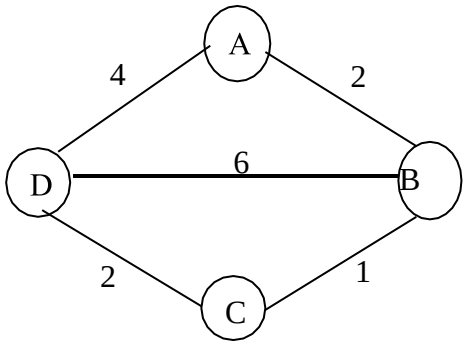


Fig 2.1.: Description of the SPP.

2.2. Negative circuit problem

A necessary condition for the existence of a SPP from 1 to i is that there is no path from 1 to I with a circuit of negative length generating a path of shorter length if a random number of times.[1]

2.3. Some shorter path properties to revise

- Everything sub path of a minimum length is itself of a minimum length.
- We can also apply this property with the paths of maximum length and we say in general: everything under the path of an optimum path is itself optimum.
- The shortest path SP = (x0, x1, x2,…, xn) between x0 and xn form a tree with root x0 and containing all vertices xi (i = 1, 2,…, n) .[1]

2.4. SPP types

2.4.1. Finding SP from a given vertex x to another given vertex y:

example of looking for a SPP between A and F:

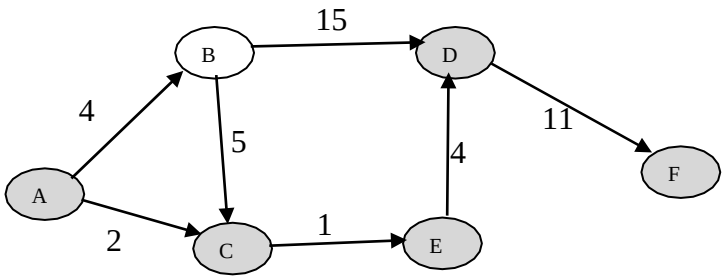


Fig 2.2.: looking for a SP.

P1(A, F) = {A, B, D, F} ==> w(P1) = 31

P2(A, F) = {A, B, C, E, F} ==> w(P2) = 25

P3(A, F) = {A, C, E, D, F} ==> w(P3) = 16

As we see P3<P1<P2 so {A, C, E, D, F} is the shortest path SP.

2.4.2. Finding SP from a given vertex x to all other vertices

Some famous algorithms are:

2.4.2.1. Graph with a negative circuit

we use:

- Floyd_warshell's algorithm.
- Dantzig's algorithm.

2.4.2.2. Graph with no negative circuit

we use:

- Dijkstra's algorithm.
- Bellman's algorithm (with no negative weights).
- Floyd_warshell's algorithm.
- Dantzig's algorithm.
- Breadth Search First (BFS).
- Linear Program (LP).

2.4.2.3. Finding SP by a heuristic method[10][11]

- Greedy algorithm (GA). [12]
- A* (best-first search).

2.4.3. Finding SP from all vertices to all other vertices

Using:

- Dantzig's algorithm.
- Floyd_warshell's algorithm.

2.4.3.1. Finding SP by an Approximate methods

Analytical approaches for producing solutions in the form of functions that are near to each other are known to as approximation methods, we using approximation method to f Finding SP from all vertices to all other vertices because that problem is NP-Hard problem(A problem that can be divided into sub-problems and solved separately). [13][14][8]

❖ With single solution based:

- Local Search (LS).
- Simulated Annealing (SA).
- Hill Climbing (HC).[15]
- Tabu Search (TS).
- Scratcher Search (SS).

❖ With population solution based:

- Genetic Algorithm (GA). [12]
- Ant colony optimization ACO.
- Bee colony optimization BCO.
- particle swarm optimization (PSO).

2.5. Shortest path search algorithms

The most famous algorithms are:

- ✓ Dijkstra's algorithm.
- ✓ Bellman's algorithm.
- ✓ Floyd warshell's algorithm.

2.5.1 Dijkstra's algorithm:

This algorithm gives all the shortest paths from a given vertex to all the other verticesof the graph, and it is a Greedy algorithm and time complexity is O(nLog(n)) (with n is the number of vertices).

start

i) S = {s}; □ = s; □(s) = □(□) = 0 ; □(x) = +□ □x□(X – S) ; A(x) = □ □x□(X –S)

ii) We consider all the arcs u such that: I(u) = □ et T(u) = x□(X – S) Si □(□)+d(u) < □(x)then ask □(x) = □(□)+d(u); A(x) = {u} Otherwise continue

iii) Choose a vertex y□(X – S); □(y) = Min[□(x)] □x□(X – S) Si □(y) = +□ then ending Sis not root of G Otherwise ask □=y S = S□{□} Si S = X then terminate {the problem is solved} Otherwise go to (i)

Algo 2.1.: Dijkstra's algorithm. [16][17]

2.5.2 Bellman's algorithm:

The graph must be without circuit, and l(u) any, and time complexity of Bellman is O(n!) (with n is the number of vertices).

Start

i) S = {s} ; $\pi(s) = 0$; A = ϕ

ii) Find a vertex $x \in (X – S)$ as all its predecessors are in S (Γ - (x)$\in$S) Si (x does not exist) then finish with: {*S* is not a root or ∃ circuit in G S = X else ($\exists x \in (X – S)$ then continue

to pose $\pi(x) = Min[\pi(I(u))+d(u)]$ with $I(u) \in S$, T(u)=x $S \cup \{x\}$; $A = A \cup \{u\}$; go to (ii

Algo 2.2.: Bellman's algorithm. [18][19]

2.5.3. Floyd_warshell's algorithm:

It is a dynamic programming algorithm with $O(|n|^3)$ time complexity and $O(|n|^2)$ space complexity (with n is the number of vertices).

```
Start
    V: Array storing the values of SPP for each vertex of G
    // Initialize array V
    For  i=0, n do
        V[xi] ←0;
    End for
    S ← {x0}                    // S contains the starting vertex x0
    V[x0]← 0                    // the SP value of the starting vertex x0 equal to 0
    // Initialize the values of SP for the other vertices of G apart from s at +∞
For i = 1, n do                 // excluding the vertex x0
        V[xi] ←+∞               // Initialize the value of SP of each vertex at +∞
 End For
// Examine the arcs
For k = 1, n do
     For i = 0, n do            // for each vertex xi
          Find the arcs u such that : I(u) = xi  and T(u)=xj      // the set of arcs from xi
         if V[xi]+d(u)] < V[xj] so
```

$V[x_j] \leftarrow V[x_i]+d(u)$

End if

End for

End for

End

Algo 2.3.: Floyd_warshell's algorithm.

Chapter 3: Programming Environment

3.1.C

3.1.1. Definition

C # (pronounced "See Sharp") has its roots in the C language family and easy-to-learn familiar to C, C ++, Java, and JavaScript programmers. It is a modern, object-oriented and secure programming language. C # allows developers to create many types of powerful applications that It works in the .NET system.[18]

3.1.2. C# History:

C# is now the most widely used programming language on the planet. Microsoft created C# with the development of the .NET framework, and later ECMA certification as a standard (ECMA-334) The C# programming language is a general-purpose programming language based on OOPS. In 2002, Anders Hejlsberg led the C# development team, the C# programming language is one of the languages created for the Common Language Initiative (CLI).

In 2002 : C# 1.0 .NET framework 1.0,1.1 , CLR version 1.0, Visual Studio 2002.

In 2005 : C# 2.0 .NET framework 2.0 , CLR version 2.0, Visual Studio 2005.

In 2007 : C# 3.0 .NET framework 3.0,3.5 , CLR version 2.0, Visual Studio 2008.

In 2010 : C# 4.0 .NET framework 4.0 , CLR version 4.0, Visual Studio 2010.

In 2013 : C# 5.0 .NET framework 4.5 , CLR version 4.0, Visual Studio 2012,2013.

In 2015 : C# 6.0 .NET framework 4.6 , CLR version 4.0 ,Visual Studio 2013,2015.

In 2017 : C# 7.0 .NET framework 4.6,4.6.1,4.6.2 , CLR version 4.0, Visual Studio 2015, 2017 RC.

In 2019 : C# 8.0 .NET framework 4.6,4.6.1,4.6.2 , 5 ,Visual Studio 2019 .

3.1.3. Microsoft Visual Studio 2019

3.1.3.1. Definition

Microsoft Visual Studio is the company's integrated development environment (IDE). It's used to make websites, web apps, online services, and mobile apps, among other things. Windows API, Windows Forms, Windows Presentation Foundation, Windows Store, and Microsoft Silverlight are some of the Microsoft software development platforms used by Visual Studio. It can write native code as well as manage it.

You can download it from the Microsoft website:

The URL: https://visualstudio.microsoft.com/fr/downloads/, After entering the site, we choose the free version (**Visual Studio Community**) intended for students, beginners, and developers for uncomplicated applications

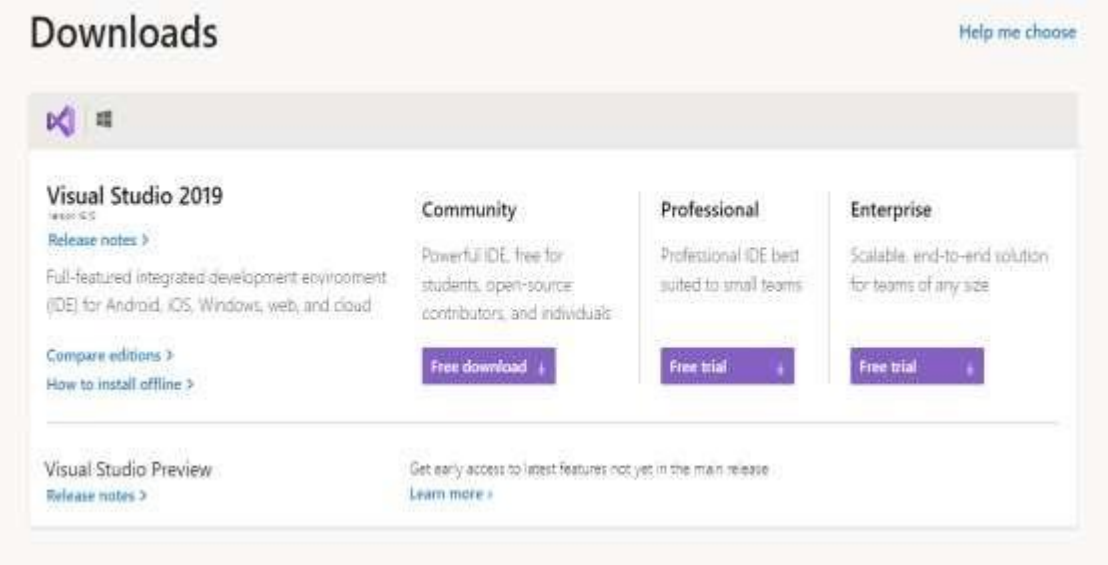


Fig 3.1.: Visual Studio 2019 Versions.

After downloading, and during installation, we choose what we need from aprogramming language (I chose c#), framework (I chose.NET framework 5.0), and features and the app type (I chose desktop app).

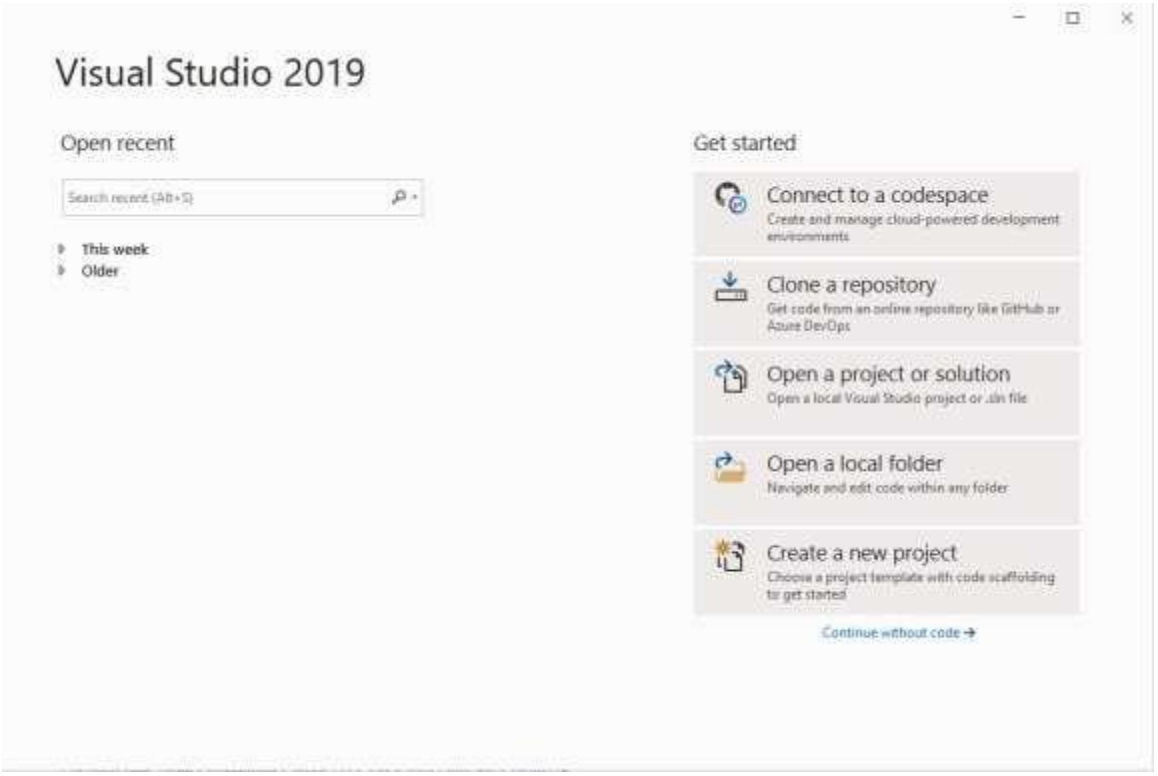


Fig 3.2.: visual studio 2019.

3.1.3.2. Features

- Code editor.
- Debugger.
- Designer.
    - Windows Forms Designer.
    - WPF Designer :This is what we use.
    - Web designer/development.
    - Class designer.
    - Data.
    - Mapping designer.

3.2. WPF

3.2.1. Definition

WPF (Windows Presentation Foundation) is a free and open-source graphical subsystem (similar to WinForms) that was created by Microsoft to render user interfaces in Windows-based applications.

WPF, formerly known as "Avalon," was first introduced in 2006 as part of the.NET Framework 3.0. WPF is based on DirectX and attempts to provide a uniform programming model for application development.[19]

3.2.2. Syncfusion Essential Studio for WPF

Syncfusion Essential Studio for WPF is a complete set of over 90+ essential WPF controls for developing strong line-of-business Windows applications quicker, including DataGrid, Chart, Diagram, and PDF Viewer. Unparalleled performance, gorgeous built-in themes, touch-friendly UI, internationalization, and easy interaction with Visual Studio are all features of Syncfusion WPF controls.

3.2.3. Features

- Direct3D.
- Data binding.
- Media services.
- Templates:
  - Control templates.
  - Data templates.
- Animations.
- Imaging.
- Effects.
- Documents:
  - per-page or scrollable.
  - FlowDocumentReader.
  - DocumentViewer.

- reflows text.

- Text.
- supporting OpenType.
- Alternative input.
- Accessibility.

3.3.SQL

3.3.1. Definition

Structured Query Language (SQL) is a query language that is used to manage data in a relational database management system (RDBMS) or for stream processing in a relational data stream management system (RDSMS). It's especially beneficial for dealing with structured data, or data that has relationships between entities and variables.[20]

3.3.2. SQL server 2019

SQL Server 2019 expands on SQL Server 2017's industry-leading2 capabilities, setting new benchmarks in areas like:

- Performance.
- Security.

SQL Server 2019 is the latest version of SQL Server's growth, delivering new capabilities to the modern data ecosystem to improve data management and data-driven applications.

The updates in SQL Server 2019 are divided into five categories:

- Reason over data anywhere.
- Choice of language and platform.
- Industry leading performance and security.
- The only commercial database with AI built in.
- Enhancing SQL Server on Linux.

You can download it from the Microsoft website:

The URL: **https://www.microsoft.com/en-au/sql-server/sql-server-downloads**

- On-premises or cloud-based server
    - SQL server in azur.
    - SQL server at the edge.
    - SQL server on-premises.

- The free specialized Edition (that's the version I chose)
    - SQL Server 2019 Developer:

        It is a full-featured free edition licensed for use in a non-production environment as a development and test database.

    - SQL Server 2019 Express: (that's the version I chose)

        It is a free edition of SQL Server that is intended for desktop, online, and small server application development and deployment.

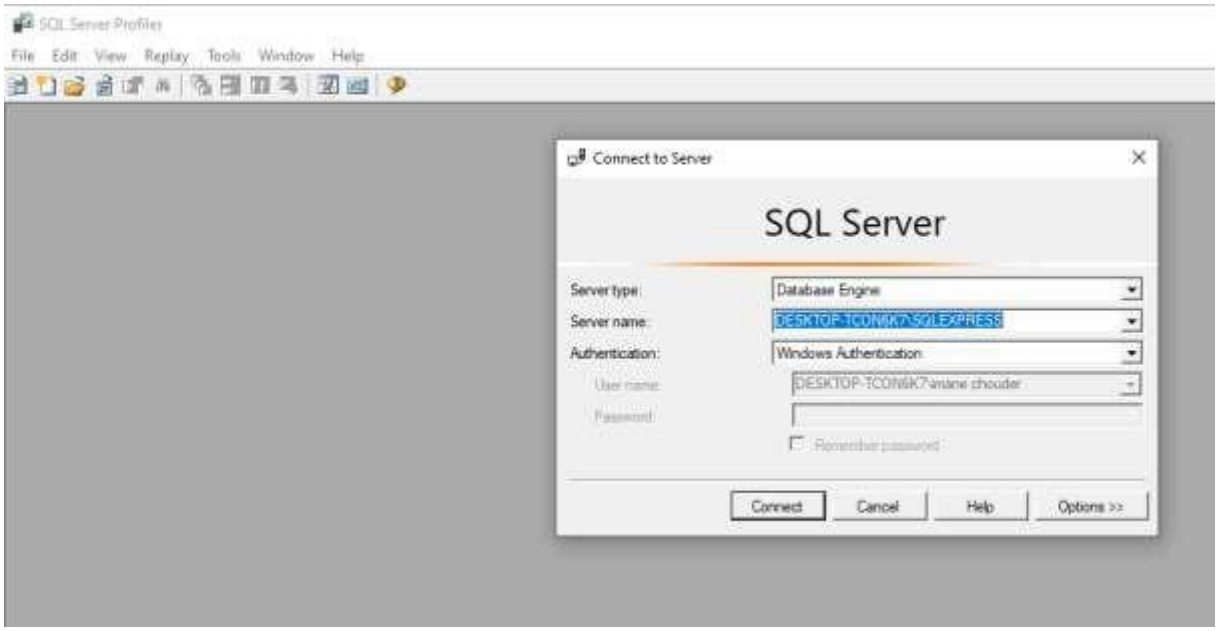


Fig 3.3.: SQL Server Profiler.

3.3.3. SQL Server Management Studio (SSMS)

SQL Server Management Studio (SSMS) provides a unified environment for managing SQL infrastructure, from SQL Server to Azure SQL Database, SSMS is a set of tools for configuring, monitoring, and managing SQL Server and database instances, and SSMS can be used to deploy, monitor, and upgrade data components that Your applications depend on it, as well as for creating queries and scripts, SSMS allows you to query, construct, and administer databases and data warehouses on your local computer or in the cloud.

Fig 3.4.: SQL Server Management Studio.

3.4.StarUml

3.4.1. Definition

StarUML is a UML modelling program wich was ceded open source by its publisher, Currently, only a proprietary license is available for StarUML V3.

The majority of the diagrams in the UML 2.0 standard are supported by StarUML. Delphi1 is used to create StarUML.

3.4.2. Features

- Multi-platform support (MacOS, Windows and Linux)
- UML 2.x standard compliant
- SysML support
- Entity-Relationship diagram (ERD)

- Data-flow diagram (DFD)
- Flowchart diagram
- Multiple windows
- Modern UX
- Dark and light themes
- Retina (High-DPI) display support
- MacPro Pro's Touch Bar support
- Model-driven development
- Open APIs
- Various third-party extensions
- Asynchronous model validation
- Export to HTML docs
- Automatic updates.

3.4.3. Users of StarUML

- Agile and small development teams.
- Professional persons.
- Educational institutes.

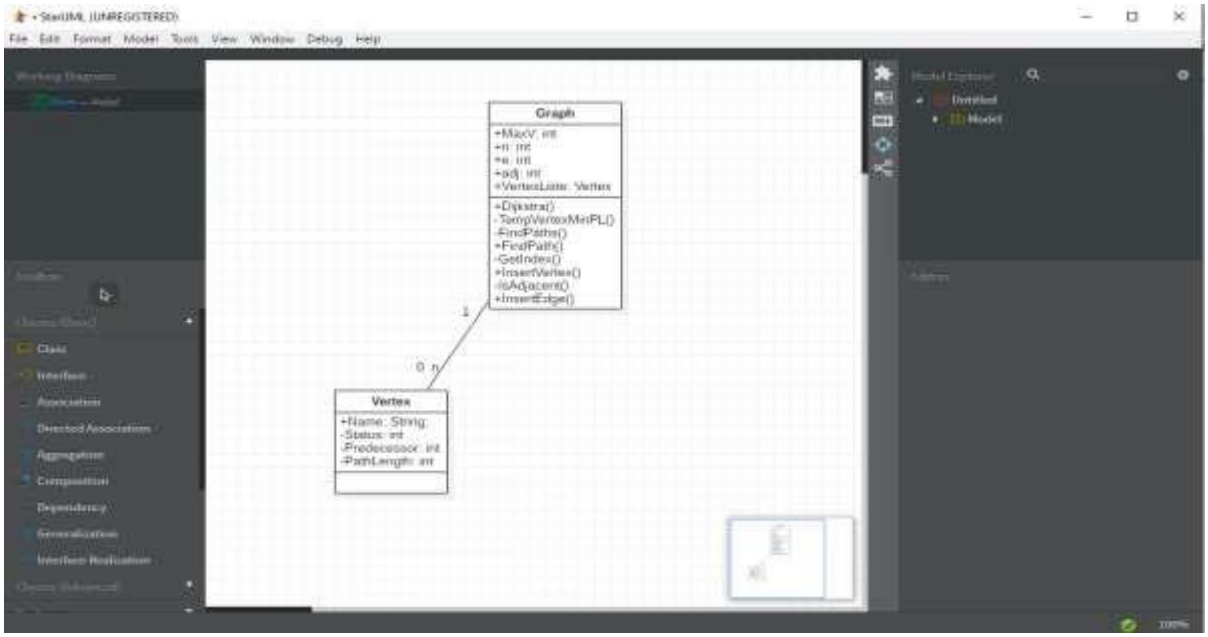


Fig 3.4.: SQL Server Management Studio.

Chapter 4: Realized Work

4.1. Class diagram

A class diagram in the unified Modelling Language (UML) is a sort of static strut a class diagram in software engineering that depicts the structure of a system by displaying system's classes, properties, operations (or method), and connections among objects.[21]

The class diagram is a fundamental component of object-oriented modelling and thisis our Class Diagram:

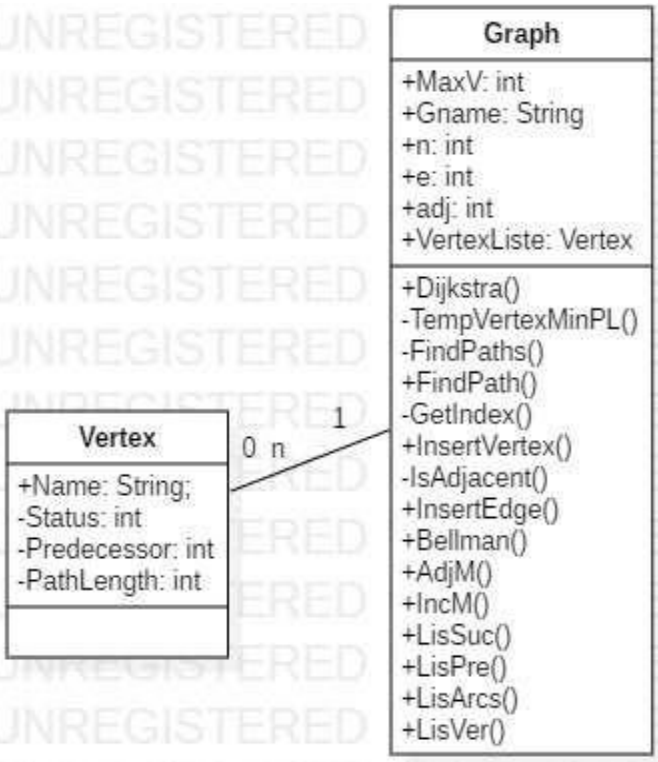


Fig 4.1.: Class diagram.

4.1.1. Explanation

Our class diagram shows the two classes used in our SPP App, the Attribute group,and the function used in every class.

In the following table, we will explain the function of the most important Operationfound in our class diagram.

Method	Function
Dijkstra()	Finding SP from all vertices to all other vertices using Dijkstra's algorithm.
Bellman()	Finding SP from all vertices to all other vertices using bellman's algorithm.
AdjM()	Finding Adjacence matrix.
IncM()	Finding Incidence matrix.
LisVer()	Finding List of vertices.
LisArcs()	Finding List of Arcs.

Tab 4.1.: Our operations.

4.2. Difference between Bellman's and Dijkstra's algorithms to find SP

Bellman's algorithm and Dijkstra's algorithm are single-source shortest path algorithms.

Bellman-Ford algorithm allows for negative edge weight and can detect negativecycles in a graph, but in Dijkstra algorithm the weight of all the edges must be non-negative.

4.3.Application Interface

When running the software, we will first see an image of the main window.

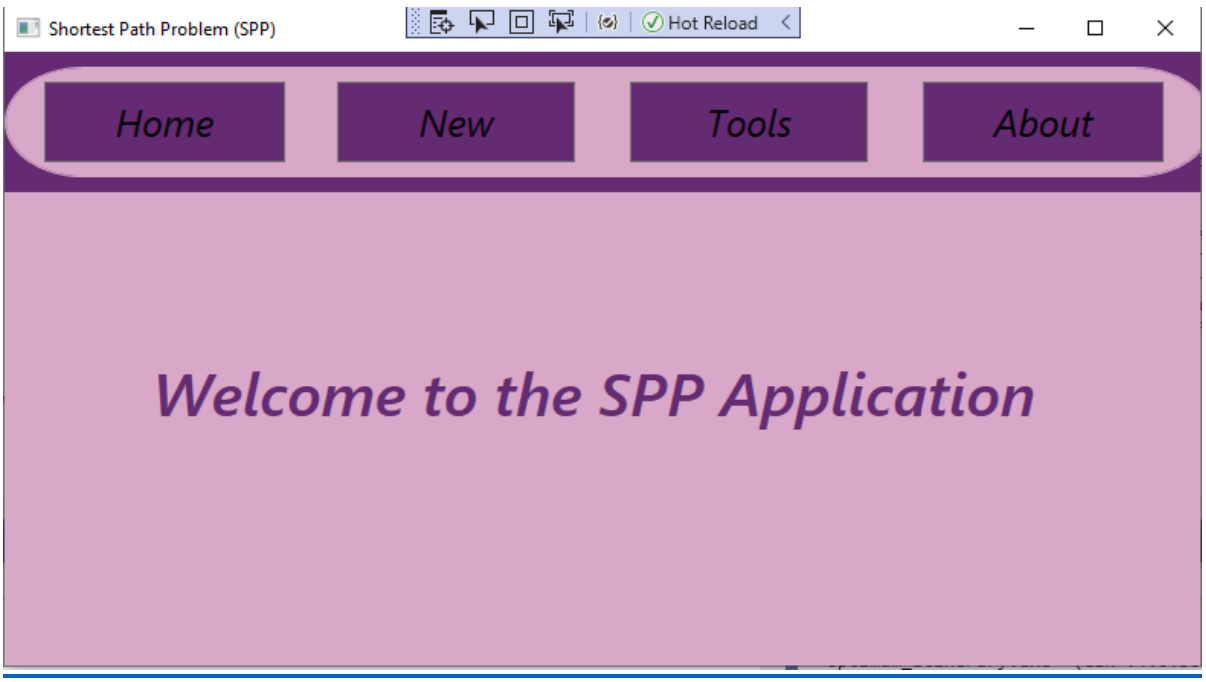


Fig 4.2.: The main of SPP Application.

Firstly, we have the home button. When we click on this button, we back to main ofSPP Application.

Secondly, we have the tools button. When we click on this button the page in Fig4.3. willshow in the frame of SPP Application.

Fig 4.3.: tools page.

Third**,** we have About button. When we click on this button the page in Fig4.3. will showin the frame of SPP Application.

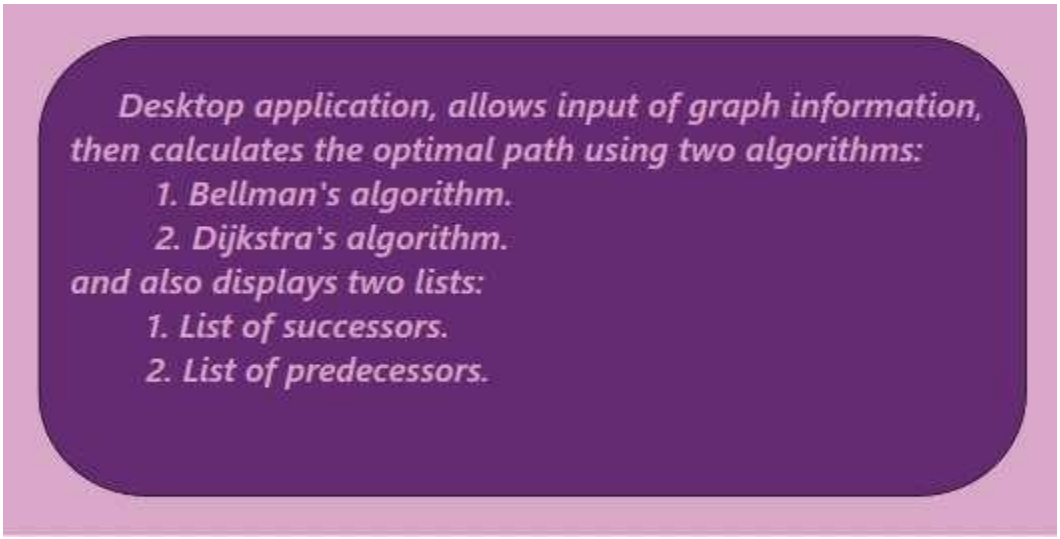


Fig4.4.: about page.

4.4. Results

Negative weight edges are possible in the graph, For this problem, we evaluated only at Dijkstra's algorithm. Dijkstra's algorithm is a greedy algorithm with an O(nLogn) time complexity Dijkstra does not operate for graphs with negative weight edges; however, Bellman-Ford does. Bellman-Ford is also less complicated than Dijkstra and is well suited to distributed systems. But time complexity of Bellman-Ford is O(VE), which is more than Dijkstra.

Conclusion

Conclusion

The field of graph theory defines a constant activity of researchers with different specializations: sports, computer science, electronics, networks, and many others.

the shortest path problem is the algorithmic problem of finding a path from one vertex to another so that the sum of the weights of the edges of this path is minimal and the SPP is considered one of the most discussed topics in this field because of its importance because we can apply it to many areas such as transportation, water transport, the completion of wired and wireless networks, road construction, delivery, and many more.

we got acquainted with the types of SPP and talked about the algorithms that can be applied to each type to obtain the SP as follows:

1. The first type, which is the simplest type, is about Finding SP from a given vertex x to another given vertex y, and we have given an illustrative example of the problem and how to find the solution.

2. The second type is about Finding SP from a given vertex x to all other vertices and we divided the algorithms as follows

   - Graph with a negative circuit :

(Dantzig’s algorithm, Floyd_warshell’s algorithm)

   - Graph with no negative circuit

(Floyd_warshell’s algorithm, Dantzig’s algorithm, BFS, LP, Bellman's algorithm, Dijkstra's algorithm)

   - Finding SP by a heuristic method: (best-first search, Greedy algorithm)

This was the type that focused our study on and we applied two algorithms of it in the desktop application that we created (Dijkstra's algorithm, Bellman's algorithm).

3. The third type is about Finding SP from all vertices to all other vertices

We mentioned the following algorithms:

Floyd_warshell's algorithm, Dantzig's algorithm

Talk about Finding SP by an Approximate method We divided it into two parts:

- With population solution-based (Local Search, Simulated Annealing, Hill Climbing, Tabu Search, Scratcher Search).

- With population solution-based (Genetic Algorithm, Ant colony optimization, Bee colony optimization, particle swarm optimization).

In our work, we focused on the second type of SPP and included the algorithms that are used in it.

We hope that we have covered a significant part of the field of graph theory in general, the topic of shortest path problem in particular, despite the short time, and we hope to be able to contribute to the development of similar research in the same field or beyond.

References

[1] D. G. Said, “Gadri said,” *Les Fondements de la Théorie des Graphes*, 2018. .

[2] L. Euler, “The seven bridges of Königsberg,” *world Math.*, vol. 1, pp. 573–580, 1956.

[3] L. Euler, “No Title,” *Solutio problematis ad geometriam situs pertinentis*, 1735. .

[4] R. J. Trudeau, *Introduction to graph theory*. Courier Corporation, 2013.

[5] I. Beg and A. R. Butt, “Fixed point of set-valued graph contractive mappings,” *J. Inequalities Appl.*, vol. 2013, no. 1, pp. 1–7, 2013.

[6] M. Bianchi, D. Chillag, M. Lewis, and E. Pacifici, “Character degree graphs that are complete graphs,” *Proc. Am. Math. Soc.*, vol. 135, no. 3, pp. 671–676, 2007.

[7] L. Zedam, N. Jan, E. Rak, T. Mahmood, and K. Ullah, “An approach towards decision-making and shortest path problems based on T-spherical fuzzy information,” *Int. J. Fuzzy Syst.*, vol. 22, pp. 1521–1534, 2020.

[8] H. Allaoua, “Combination of genetic algorithm with dynamic programming for solving TSP,” *Int. J. Adv. Soft Compu. Appl*, vol. 9, no. 2, 2017.

[9] M. Benazi and C. Lamiche, “An efficient Genetic Algorithm with Modified Crossover Operator for Community Detection in Social Networks,” in *2020 4th International Symposium on Informatics and its Applications (ISIA)*, 2020, pp. 1–7.

[10] R. LEBZA and M. AYMEN, “Meta-heuristic based approach for Minimum Vertex Cover Problem.” UNIVERSITE MOHAMED BOUDIAF-M’SILA FACULTE DES MATHEMATIQUES ET DE L …, 2019.

[11] S. BOUCHAREB, “Sieve Optimization Method A Survey and Applications.” Faculté des Mathématiques et de l’Informatique-Université Mohamed BOUDIAF-M’sila,

[12] 2017.

[13] K. Boudjelaba, D. Chikouche, and F. Ros, “Evolutionary techniques for the synthesis of 2-D FIR filters,” in *2011 IEEE Statistical Signal Processing Workshop (SSP)*, 2011, pp. 601–604.

[14] Y. Deng, Y. Chen, Y. Zhang, and S. Mahadevan, “Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment,” *Appl. Soft Comput.*, vol. 12, no. 3, pp. 1231–1237, 2012.

[15] H. Wang, Y. Yu, and Q. Yuan, “Application of Dijkstra algorithm in robot path-planning,” in *2011 second international conference on mechanic automation andcontrol engineering*, 2011, pp. 1067–1069.

[16] A. Goldberg and T. Radzik, “A heuristic improvement of the Bellman-Ford algorithm,” STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1993.

[17] H. Allaoua and B. Brahim, “Hybrid algorithm for optimization problems applied to single machine scheduling,” *Int. J. Comput. Appl.*, vol. 66, no. 24, 2013.

V. Sakharov, S. Chernyi, S. Saburov, and A. Chertkov, “Automatization Search for the Shortest Routes in the Transport Network Using the Floyd-warshell Algorithm,” *Transp. Res. Procedia*, vol. 54, pp. 1–11, 2021.

[18] Wikibooks.org, *C Sharp Programming*. .

[19] V. R. Mahesh Chand, Rikam Palkar, “WPF,” in *CHAPTER I: INTRODUCTION TO WPF*, p. 34.

[20] M. Stonebraker, “SQL databases v. NoSQL databases,” *Commun. ACM*, vol. 53, no. 4, pp. 10–11, 2010.

[21] S.-K. Kim and C. David, “Formalizing the UML class diagram using Object-Z,” in *International Conference on the Unified Modeling Language*, 1999, pp. 83–98.

Abstract

The work presented in this note revolves around identifying the field of graph theory, graph its properties, types, and ways of presenting it, and mentioning some of the problems specific to this field, especially the focus on SPP and its three types, especially the second type, which is related to Finding SP from a given vertex x to all other vertices , There is also an explanation of a desktop application for finding SP in Weighted directed graph by Dijkstra's algorithm && Bellman's algorithm .

Keywords:

graph theory, graph, arcs, vertex, Shortest path problem, SSP, Shortest path, SP, Graph's Type, valued graph, Complete graph, Simple graph , Empty graph, Trivial graph, Refletive graph , Transitive graph, inverse of the graph, Complementary graph, Multigraph graph, Bipartite graph, Planar graph, Isomorphic Graph, Subgraph, partial graph, partial subgraph.

ملخص

يتمحور العمل المقدم في هذه المذكرة حول نظرية المخططات أو نظرية البيان, و خصائصه وأنواع وطرق عرض الرسم المخطط او البيان، مع ذكر بعض المشاكل الخاصة بهذا المجال ، لا سيما التركيز على مشكلة إيجاد المسار الأمثل وأنواعه الثلاثة ، خاصة النوع الثاني ، المرتبط بإيجاد المسار الأمثل من رأس إلى جميع الرؤوس الأخرى للبيان ، وهناك أيضًا شرح لتطبيق سطح المكتب خاص بإيجاد المسار الأمثل من رأس إلى جميع الرؤوس الأخرى للبيان الموجه الذي تكون اضلاعه ذات قيم عددية, وأيضا يعرض هذا التطبيق عدة طرق لشرح بيان .

كلمات مفتاحية:

نظرية المخططات، نظرية البيان، البيان ، المخطط ، الأضلاع ، الرأس ، أقصر مسار، المسار الأمثل ، مشكلة المسار الأمثل، أنواع المخطط ، أنواع البيان ، بيان ذو قيم ، بيان كامل ، بيان بسيط ، بيان فارغ ، بيان تافه ، بيان انعكاسي ، بيان انتقالي ، معكوس البيان ، بيان تكميلي ، بيان متعدد ، بيان ثنائي الأجزاء ، بيان مستوي ، بيان متماثل ، بيان فرعي ، بيان جزئي ، بيان فرعي جزئي..

Résumé

Le travail présenté dans cette note s'articule autour d'identifier le domaine de la théorie des graphes, de représenter graphiquement ses propriétés, ses types et les manières de le présenter, et de mentionner quelques-uns des problèmes spécifiques à ce domaine, notamment deuxième type, qui est lié à la recherche de SP d'un sommet x donné à tous les autres sommets,

Il existe également une explication d'une application de bureau pour trouver SP dans un graphe

orienté pondéré par l'algorithme de Dijkstra et l'algorithme de Bellman.

Mots clés:

théorie des graphes, graphe, arcs, sommet, Problème du plus court chemin, PCC, Chemin le plus court, CP, Type de graphe, graphe valué, graphe complet, graphe simple, graphe vide, graphe trivial, graphe réflexif, graphe transitif, inverse du graphe, graphe complémentaire, graphe multigraphe, graphe bipartite, graphe planaire, graphe isomorphe, sous-graphe, graphe partiel, sous-graphe partiel.