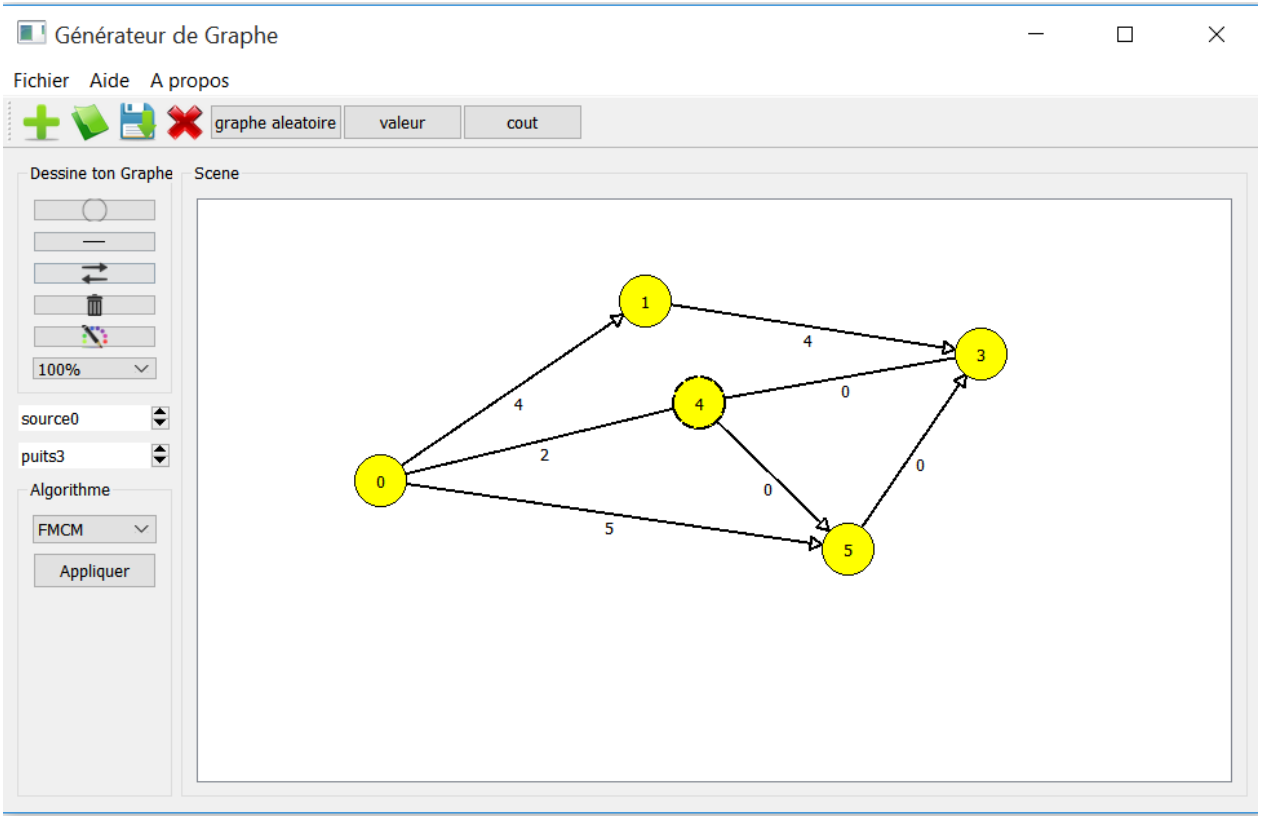
Rapport final :



Les intervenants

MBARKI Anis

FILAH Anas

KADIM Mouad

IBNOUKHALKANE Tarik

OUARHOU Lhoussaine

Les encadrants

Mme ZRIKEM Maria

Mr EL MARZOUKI Nabil

Mme BENCHIKHI Loubna

SOMMAIRE

1-CONTEXTE GLOBALE DU PROJET

1-1. INTRODUCTION

Dans le cadre du projet du second semestre de la 1ere année, nous devons réaliser un Générateur de graphe avec le problème de flot comme sujet. Le but de ce projet et de tester les connaissances de l'étudiant ainsi que de mobiliser ses compétences acquises jusqu'à présent.

Ce rapport est le résultat d'un travail collectif.

On commencera par présenter l'interface, le générateur et les algos.

1-2. CAHIER DE CHARGE

- Générateur de graphe : de toutes tailles, tenir compte de la densité ……
- Un outil graphique de traçage et visualisation de graphe
- Opérations de base sur les graphes : ajout et suppression de sommets, d'arêtes ….
- Le graphe peut être :
    - saisi directement par l'outil graphique ensuite stocké dans la structure de données,
    - produit par le générateur
    - lu à partir d'un fichier et visualiser par l'outil graphique
- Flot max et flot max à cout minimum et floit max sous contraintes de capacité.

1-3. PLANNING

Etape 1 : Réalisation de l'interface sans utiliser Qt designer.

Etape 2 : Réaliser le générateur manuel et automatique, avec la possibilité de supprimer les sommets, arcs ou arêtes, et d'enregistrer le projet.

Etape 3 : Implémenter les Algorithmes de flot.

Planning de la realisation du projet sur les graphes

- **Le mercredi 18 Avril** :

1) présenter à quoi va ressembler l'interface.
2) Collecter des infos sur les options que présente l'application

- **Le mercredi 25 Avril :**

Présentation du planning

- **le mercredi 2 mai :**

1) Présenter une application qui permet de dessiner des arcs, des arêtes Et des sommets, ainsi que de créer des nouvelles pages vides.
2) Proposer une manière de sauvegarde, sur papier.

- **Le mercredi 9 mai :**

1) Présenter un générateur qui a toute les fonctionnalités Fondamentales (créer des nouvelles pages, y dessiner des graphes Quelque soit la figure voulu, et pouvoir les sauvegarder pour une Utilisation future).

- **<u>Le mercredi 16 mai :</u>**

1) Ajouter des options à l'application et finaliser le générateur,

la possibilité d'affecter des valeurs au arcs et au arêtes.

2) Présenter des études et des recherche sur le flot, afin d'établir

L'adéquat algorithme pour gérer les flots.

- **<u>Le mercredi 30 mai :</u>**

1) Traduire l'algorithme au langage C++(Qt).

2) Avoir une application apte de resoudre le problème des flots

- **<u>Le mercredi 06 juin :</u>**

1) Présenter une application qui peut générer les graphe et traiter les

Flots.

2-ÉTUDE TECHNIQUE :

Lors de notre projet, nous allons utiliser :

- Les bibliothèques de C++.
- Le développement va être effectué avec QtCreator.

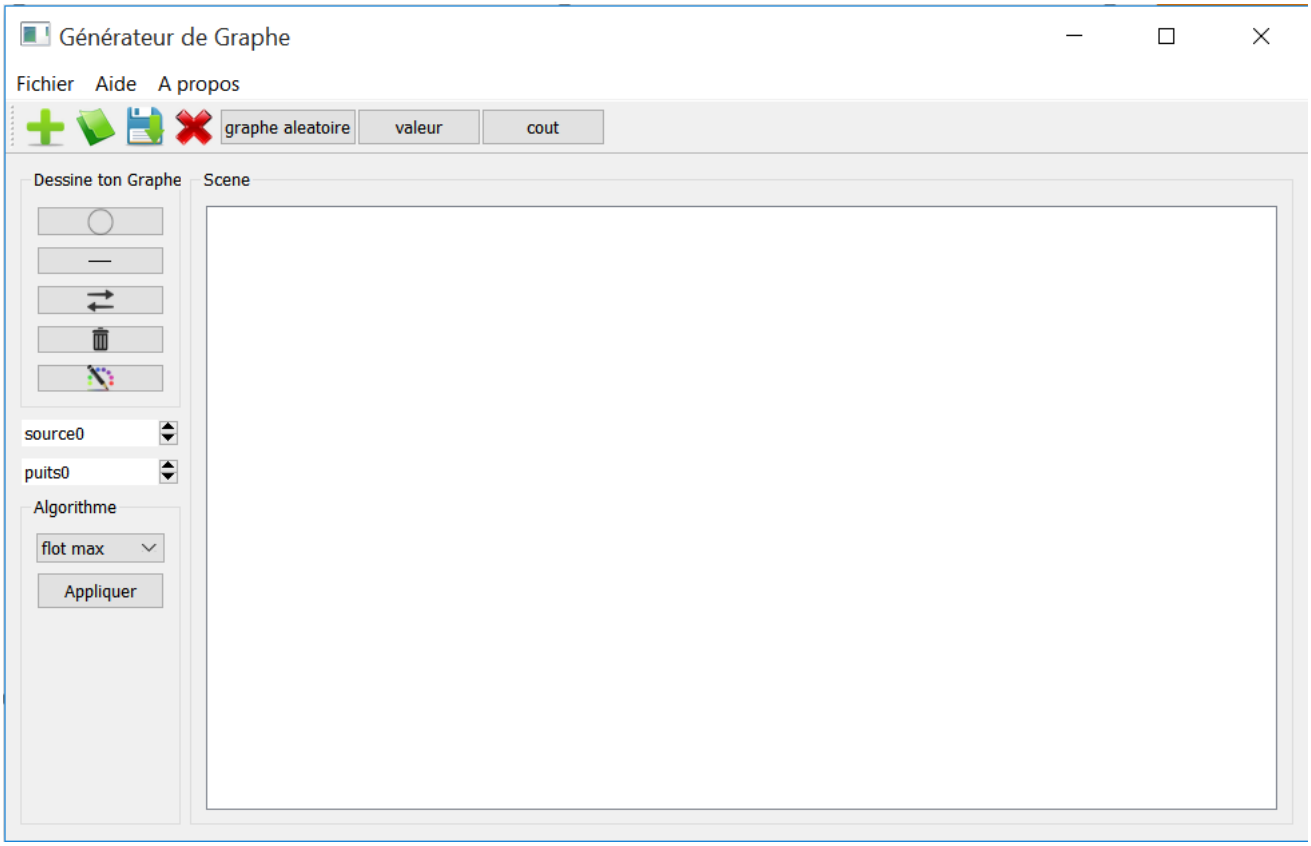
Qt est écrite en C++ et elle est, à la base, conçue pour être utilisée en C++. Toutefois, il est aujourd'hui possible de l'utiliser avec d'autres langages comme Java, Python, etc.

Plus fort qu'une bibliothèque : un Framework

Qt (voir logo en figure suivante) est en fait… bien plus qu'une bibliothèque. C'est un ENSEMBLE DE BIBLIOTHEQUES. Le tout est tellement énorme qu'on parle d'ailleurs plutôt de **framework** : cela signifie que vous avez à votre disposition un ensemble d'outils pour développer vos programmes plus efficacement

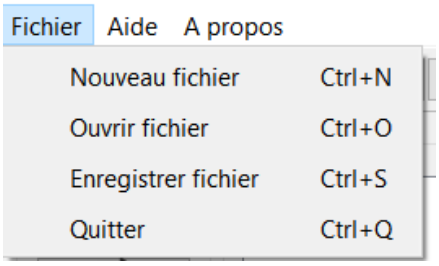
C++ est un langage de programmation compilé permettant la programmation sous de multiples paradigmes (comme la programmation procédurale, orientée objet ou générique). Ses bonnes performances, et sa compatibilité avec le C en font un des langages de programmation les plus utilisés dans les applications où la performance est critique.

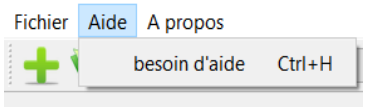
3-INTERFACE :

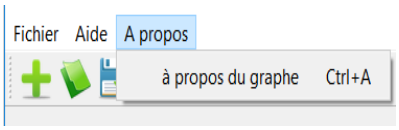

Générateur de Graphe
Fichier
Aide
A propos
graphe aleatoire
valeur
cout
Dessine ton Graphe
Scene
source0
puits0
Algorithme
flot max
Appliquer

LES ELEMENTS DE L'INTERFACE:

MENUBAR





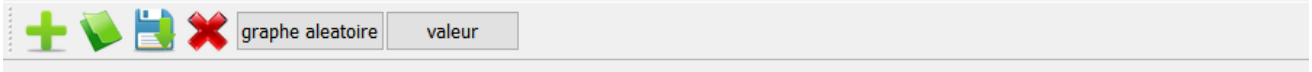


Menu fichier: Permet d'afficher des fonctionnalités qui permettent de créer un fichier, d'ouvrir un fichier, d'enregistrer un fichier et de quitter le programme.

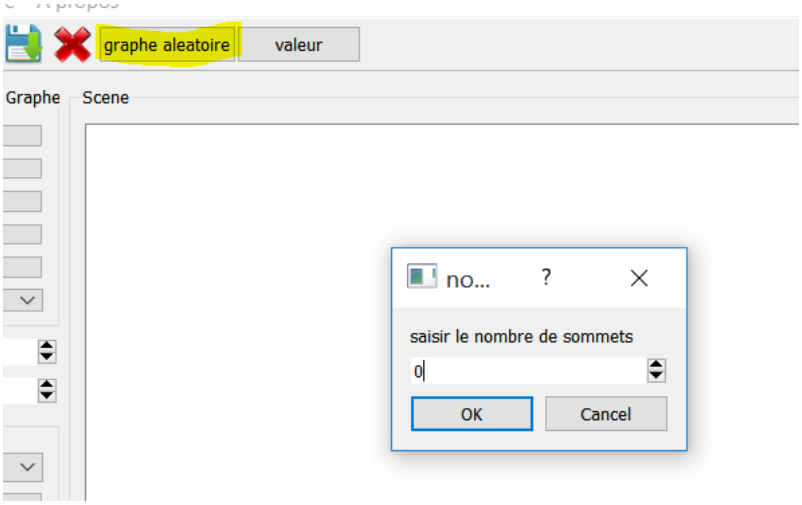
Menu Aide: Permet de donner quelques informations afin d'utiliser les fonctionnalités du programme.

Menu A propos : Permet d'afficher quelques informations sur le projet.

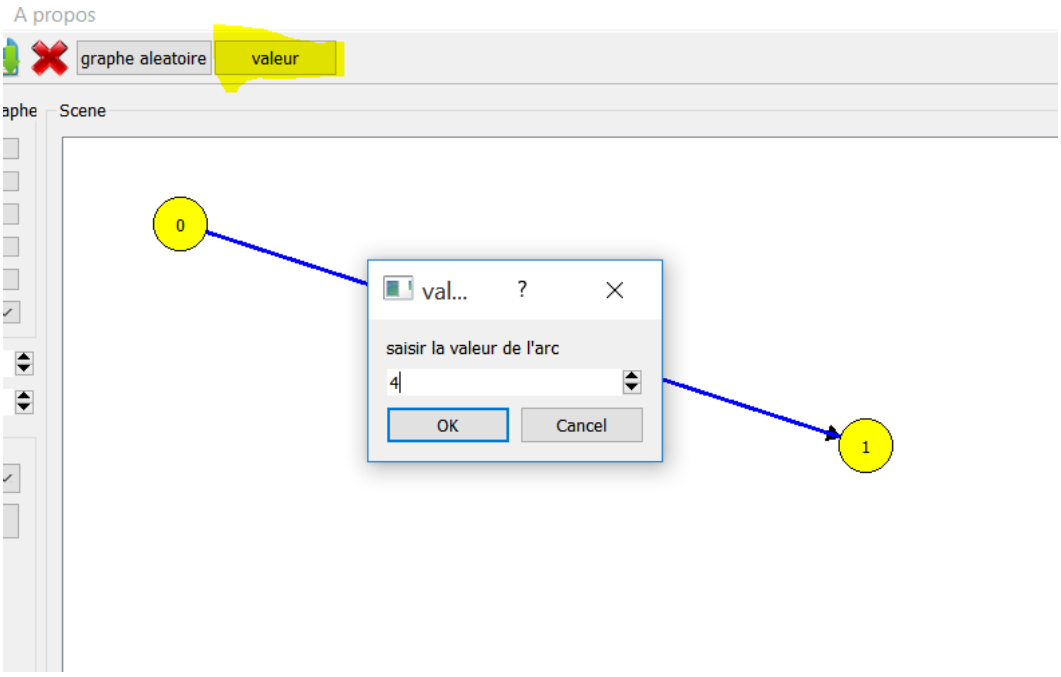
TOOLBAR



Le Toolbar contient 4 boutons qui permettent d'ouvrir un nouveau fichier, d'ouvrir un fichier, de sauvegarder un fichier ou de quitter le programme.

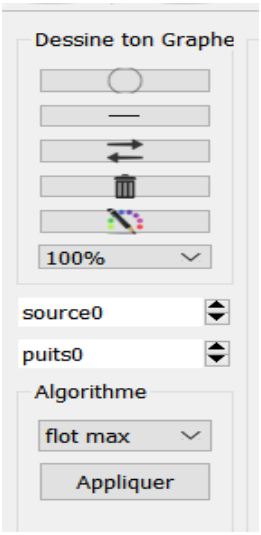


Le Toolbar contient aussi un widget « graphe aleatoire »qui permet d'afficher 3 QInputDialog qui permettent d'entrer le nombre de sommets, d'arêtes et d'arcs. Lorsqu'on appuie sur le bouton OK, le graphe sera générer d'une façon aléatoire.



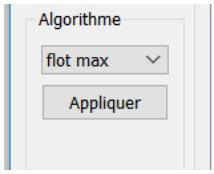
Le Toolbar contient aussi un widget « valeur » qui permet d'afficher un QInputDialog afin de donner la capacité d'un arc ou d'un arête.

GROUPBOX



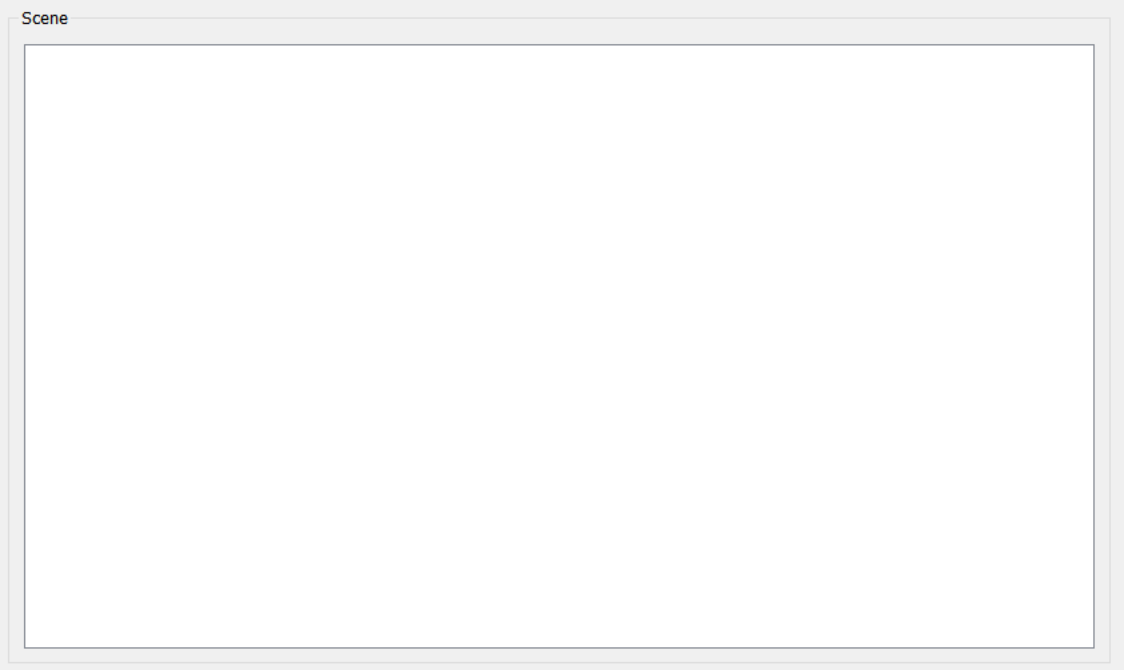
Le groupbox contient des boutons qui permettent de dessiner le sommet, l'arête et l'arc, de supprimer des éléments de la scène ou de les sélectionner.

Les deux Qspinbox permettent de designer le sommet qui sera considérer comme source, et celui qui sera considérer comme puits.



Le flot box permet de choisir l'algorithme qu'on veut utiliser, et on appuie sur Appliquer pour l'appliquer sur le graphe déjà réaliser.

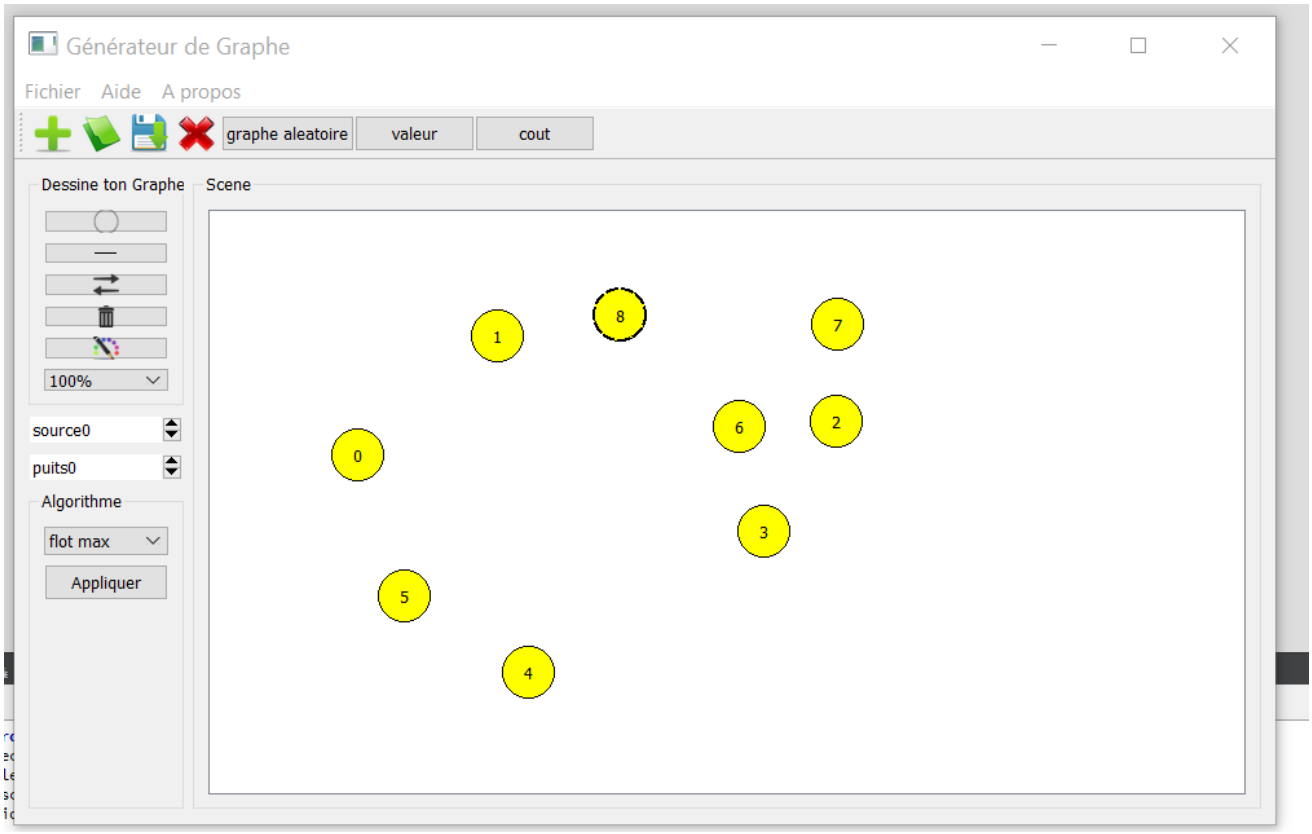
SCENE



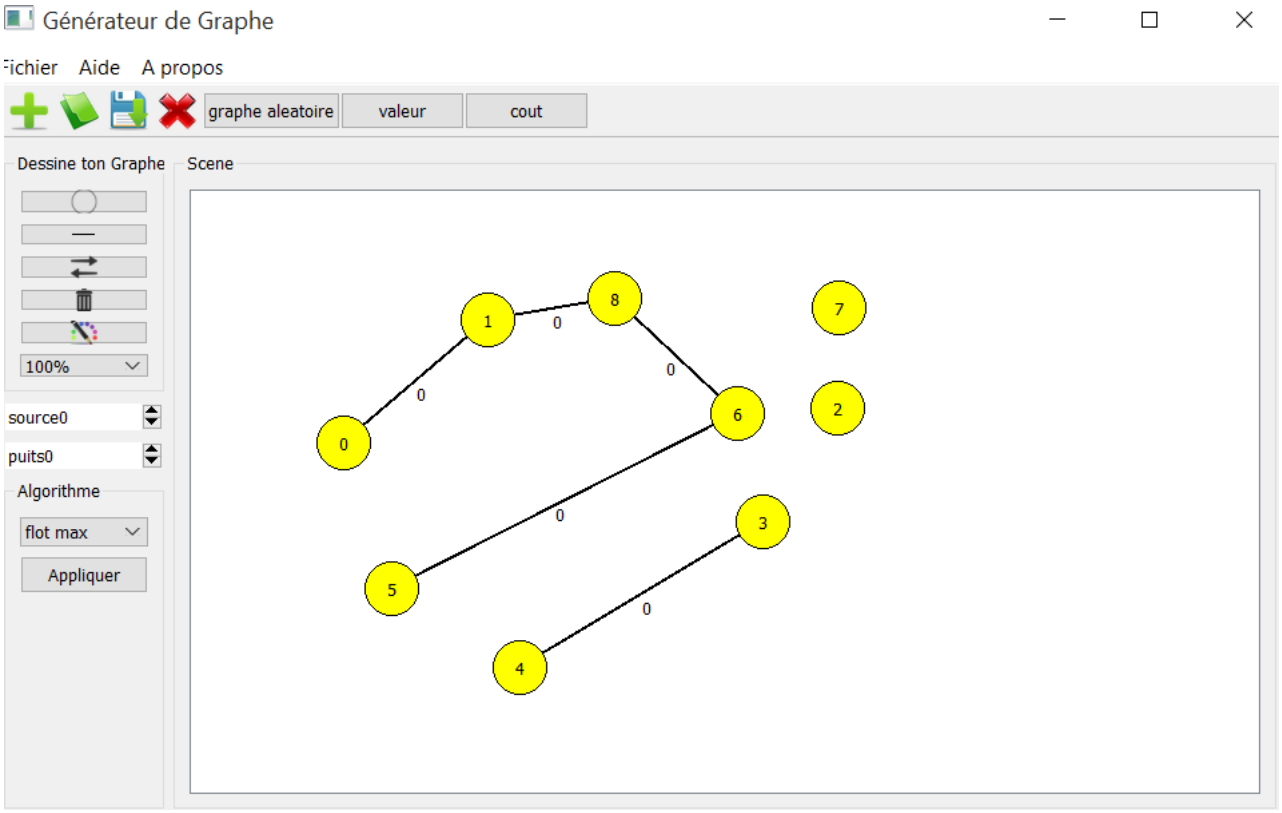
La scène est l’emplacement où l’utilisateur peut dessiner son graphe

LE FONCTIONEMENT DU PROGRAMME

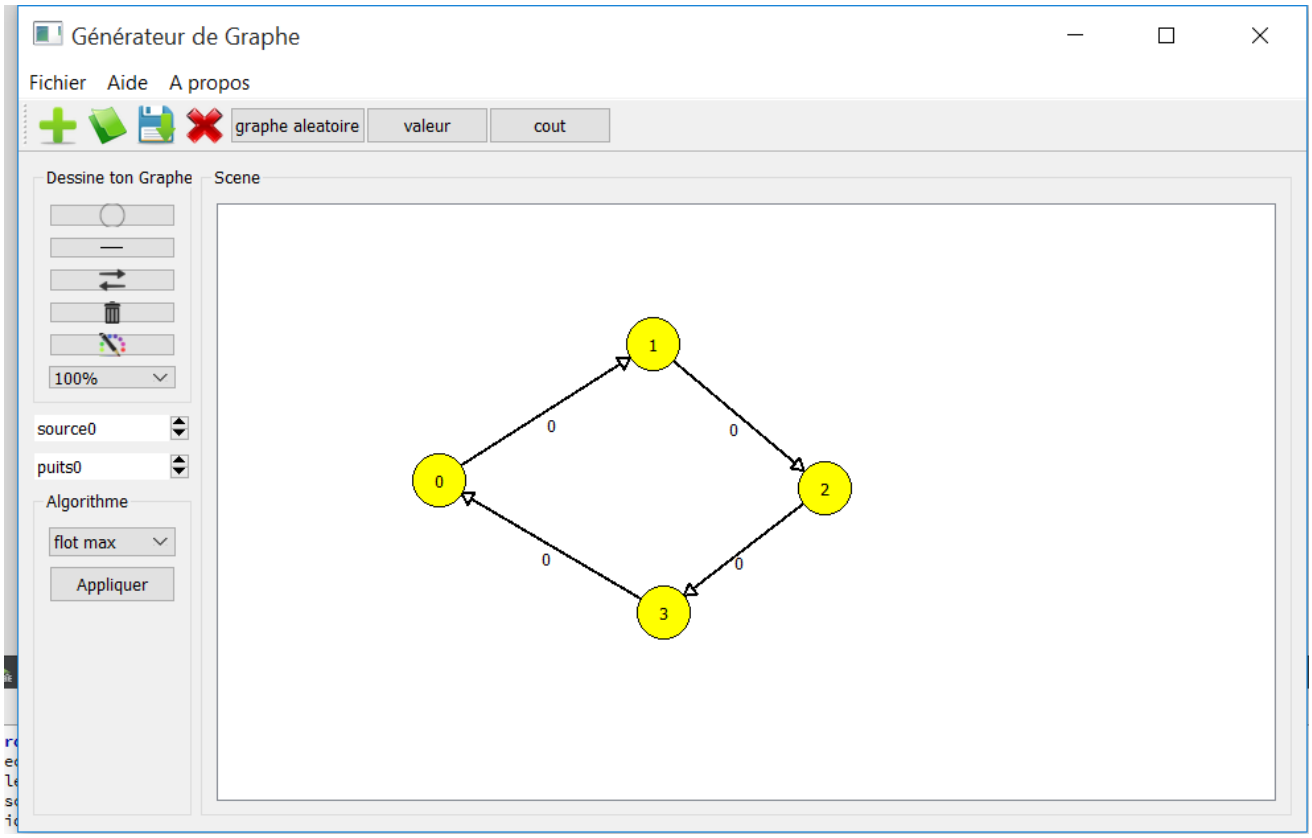
AJOUT D'UN SOMMET



AJOUT D'UNE ARETE



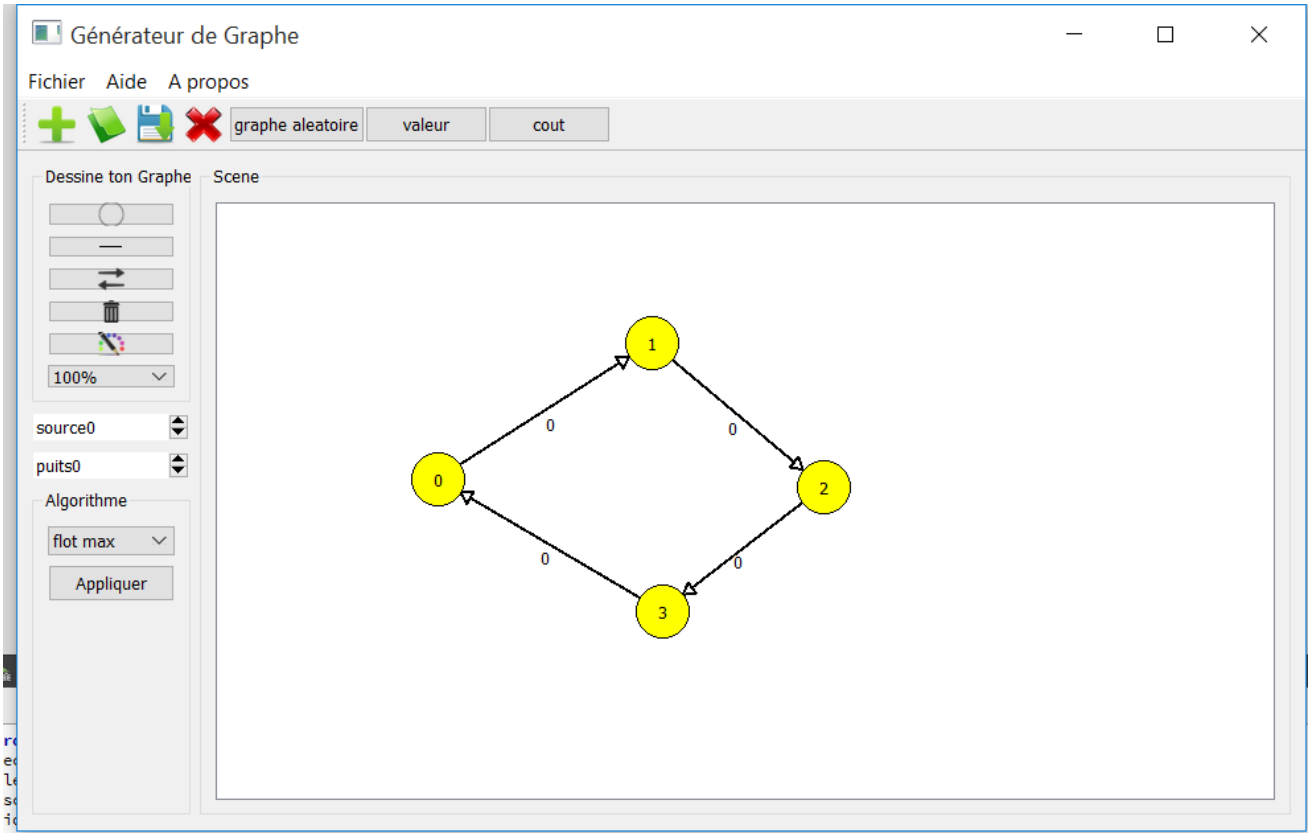
AJOUT D'UN ARC

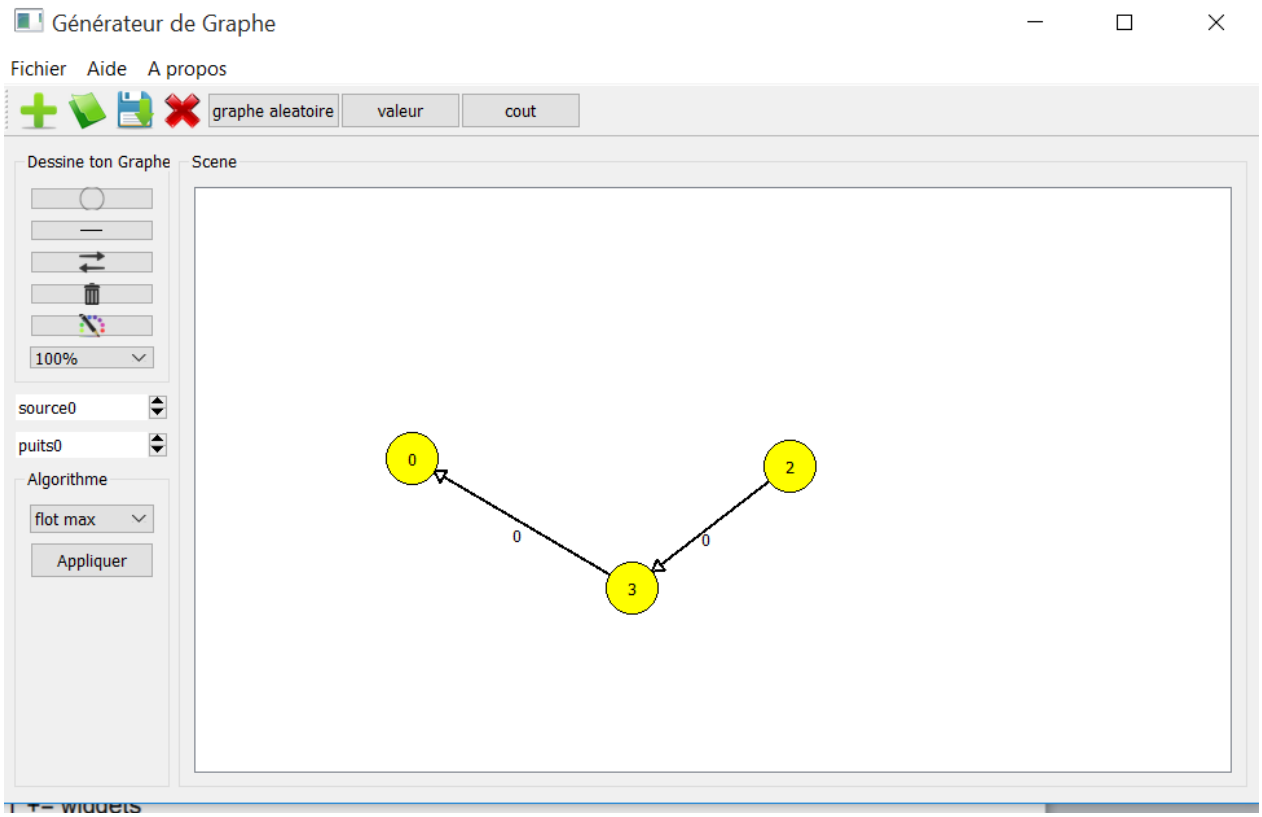


SUPPRIMER

Pour supprimer un élément, il suffit d'appuyer sur le bouton sélectionner, puis appuyer sur l'élément et ensuite sur le bouton delete.

Par exemple pour supprimer le sommet numéro 1 :

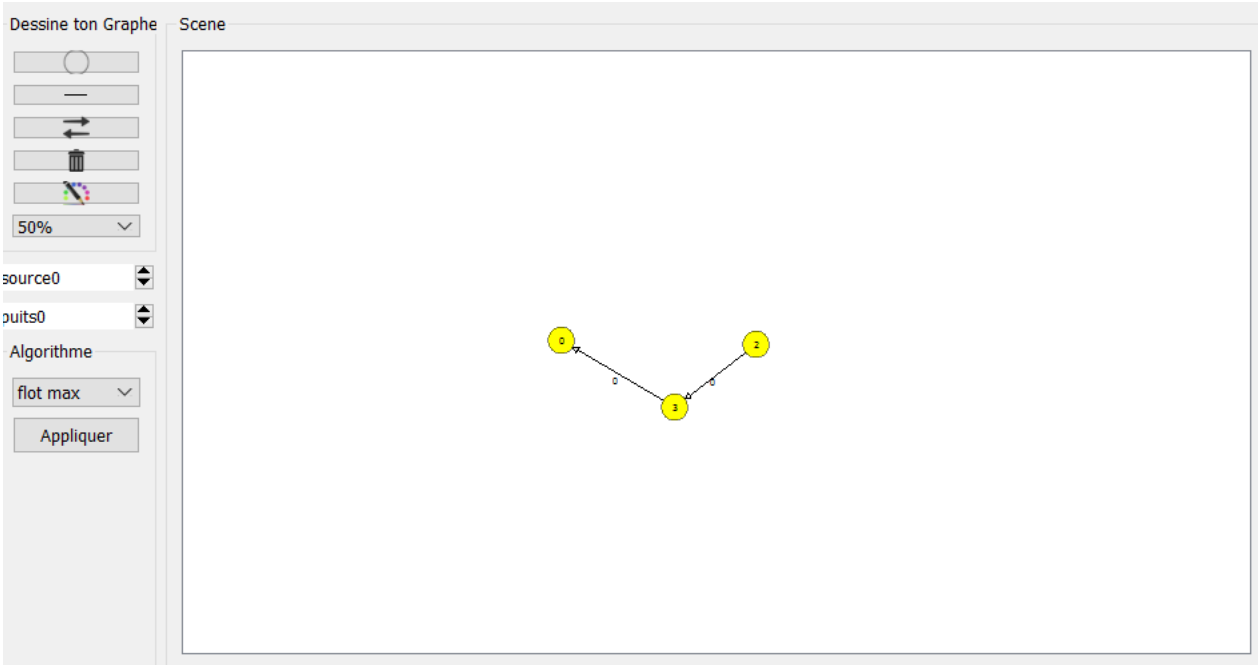




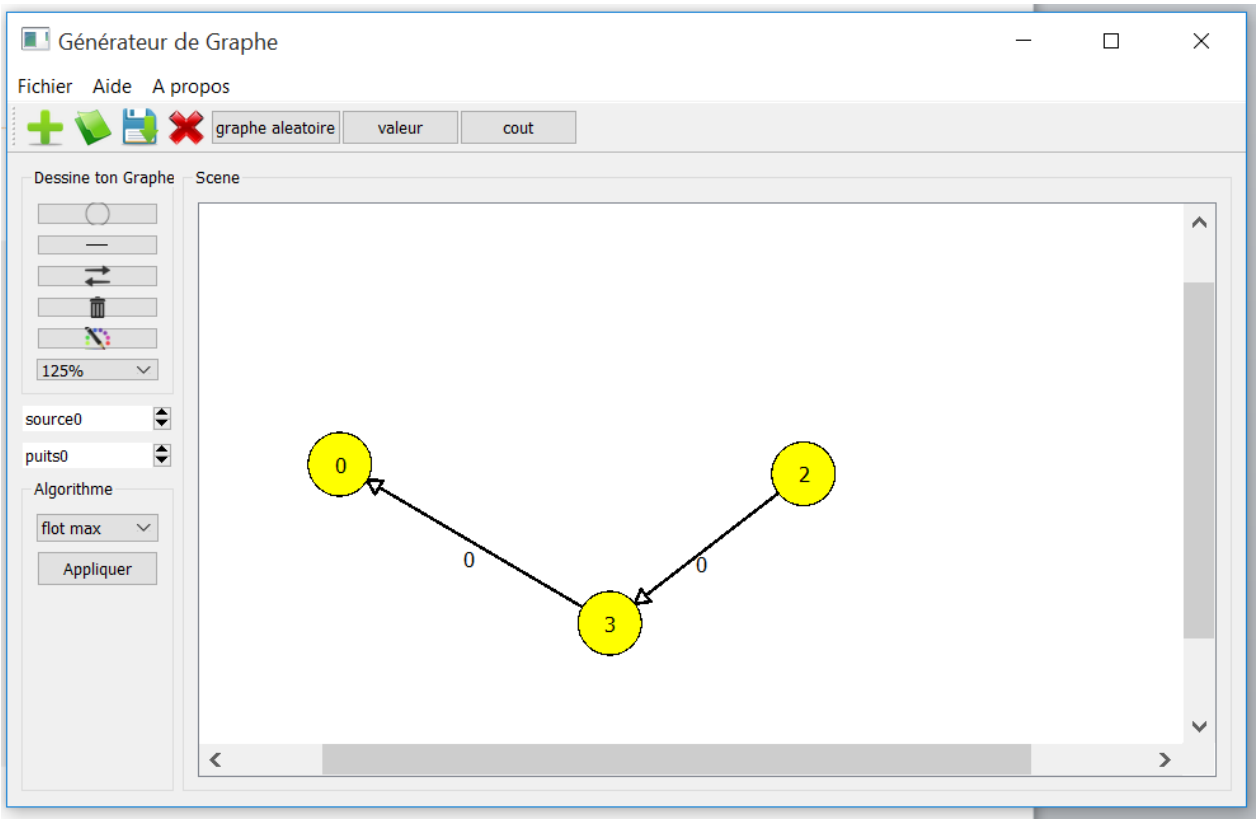
Remarque : Tous les arcs et les arêtes qui sont liées à notre sommet vont être supprimé.

ZOOM

50%

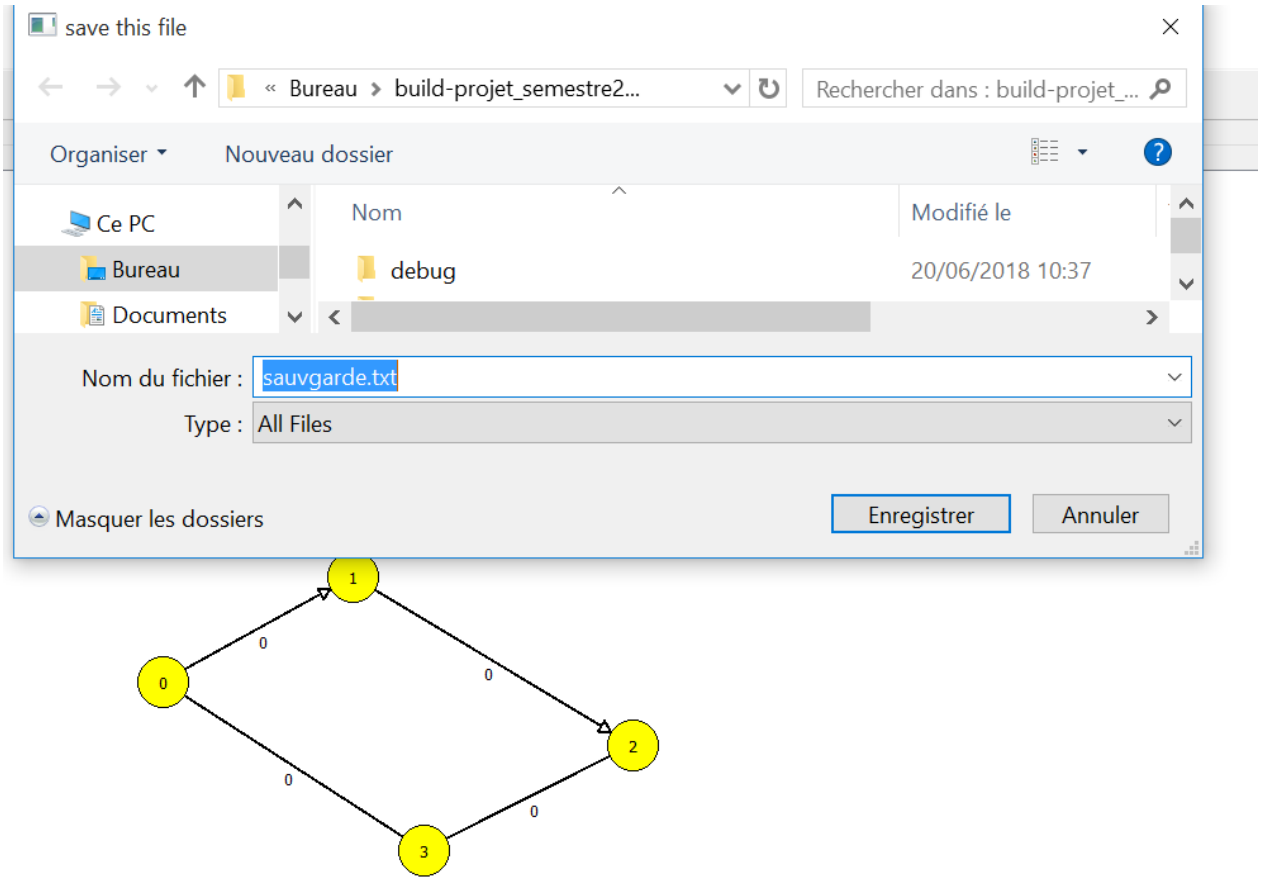

Dessine ton Graphe
Scene
50%
source0
puits0
Algorithme
flot max
Appliquer
0
2
3

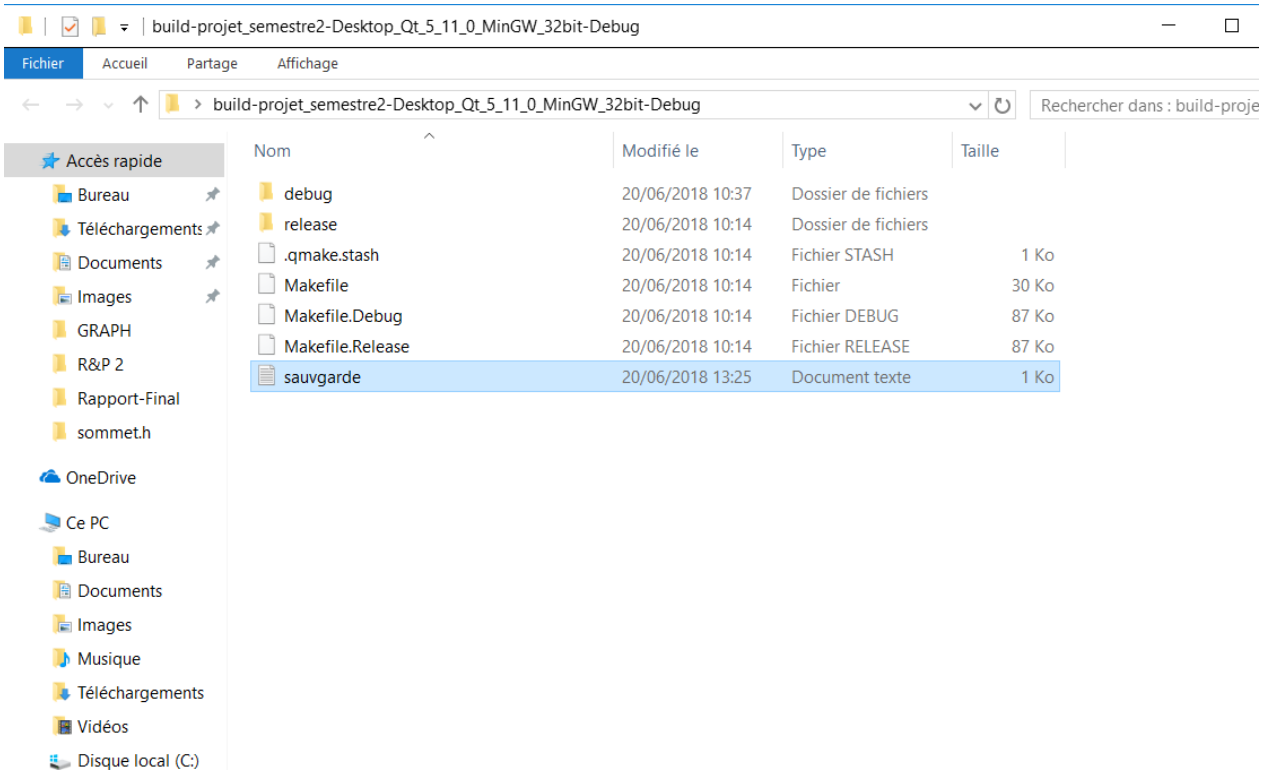
125%


Générateur de Graphe
Fichier
Aide
A propos
graphe aleatoire
valeur
cout
Dessine ton Graphe
Scene
125%
source0
puits0
Algorithme
flot max
Appliquer
0
2
3
0
0

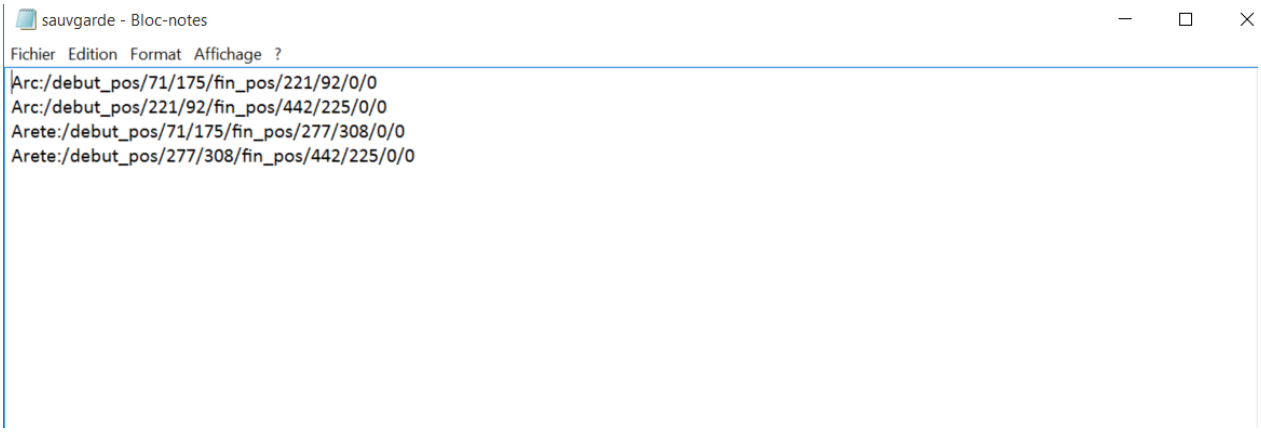
POUR SAUVGARDER ET OUVRIRE

Apres avoir créer un graphe il suffit de cliquer sur le bouton save pour le sauvegarder.

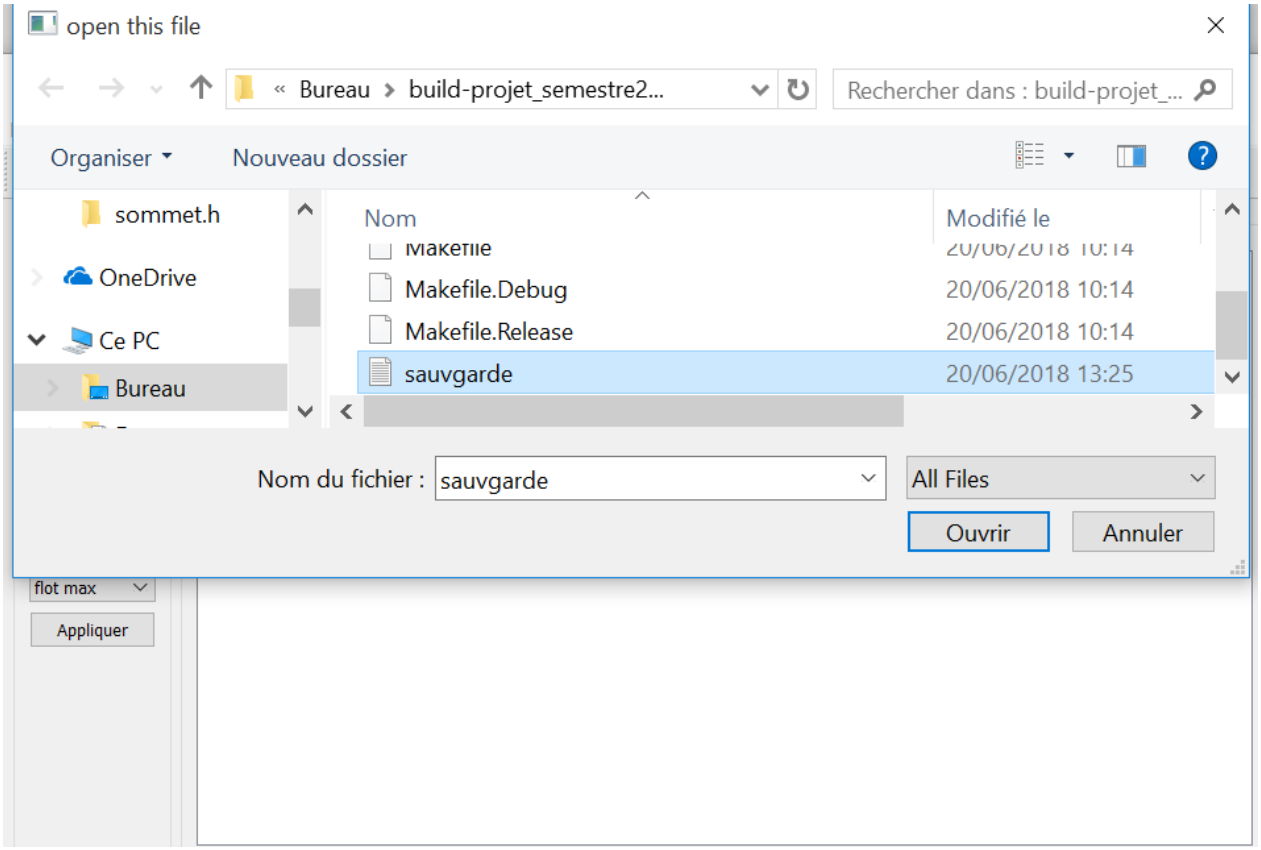




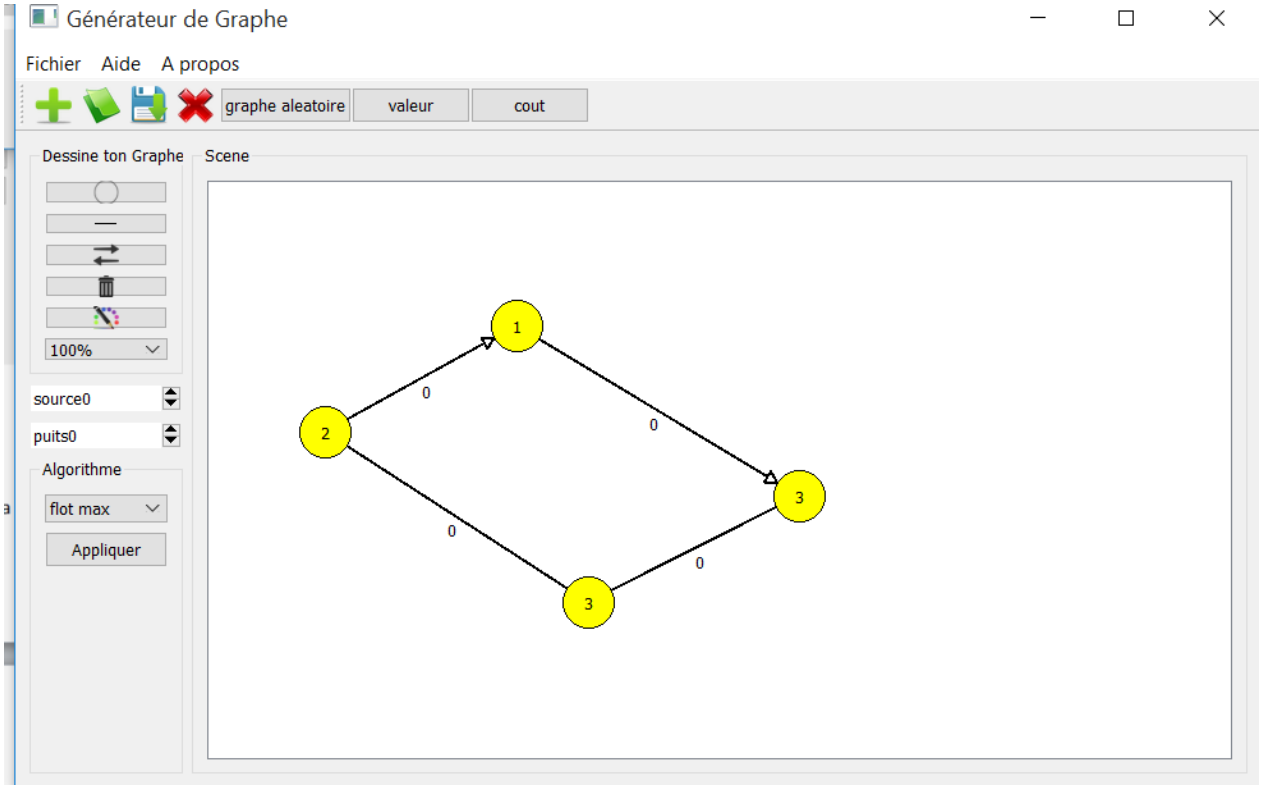
La sauvegarde est sous forme d'un fichier texte qui contient la position du sommet de début et de fin de chaque arc/arête.



OUCRIR UNE SAUVEGARDE



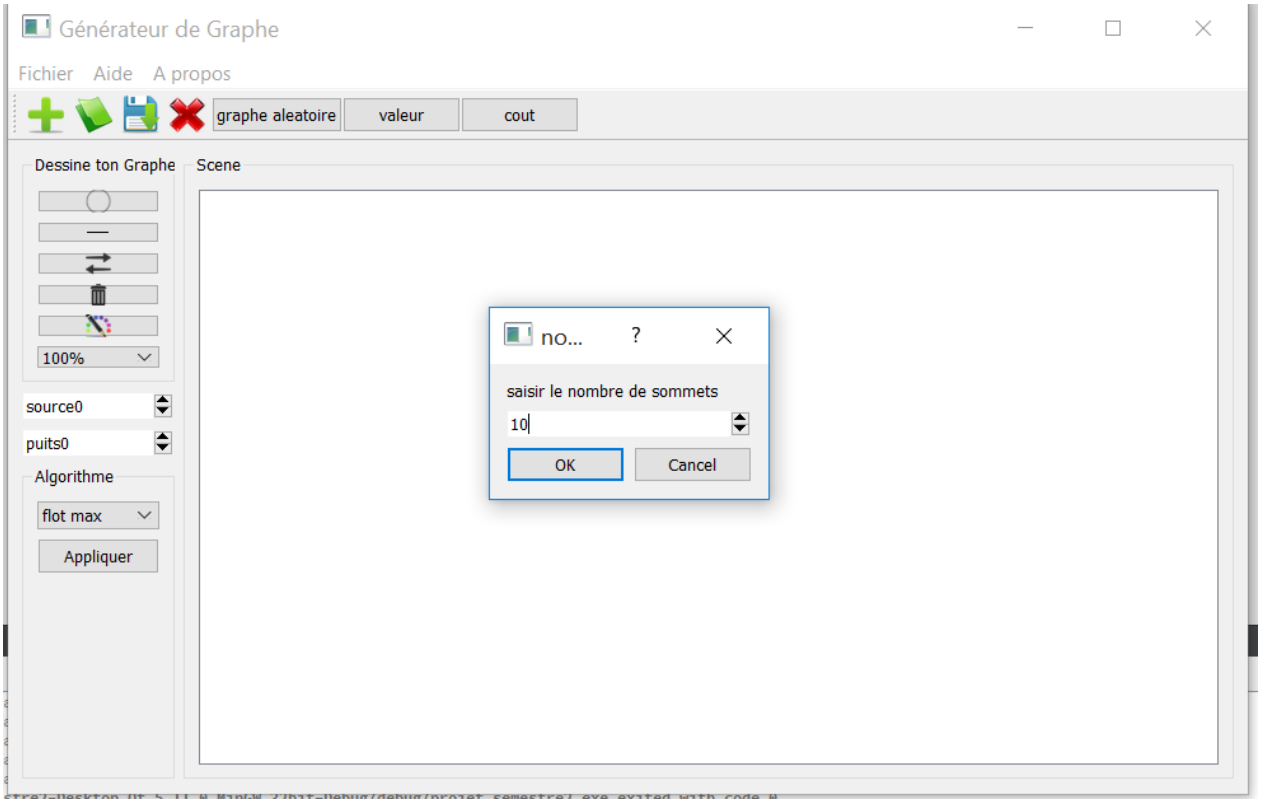
On appuie sur le bouton open file, l'algorithme traduit le fichier texte en un graphe ,en récupérant les positions des sommets qui relient les arcs/arêtes.



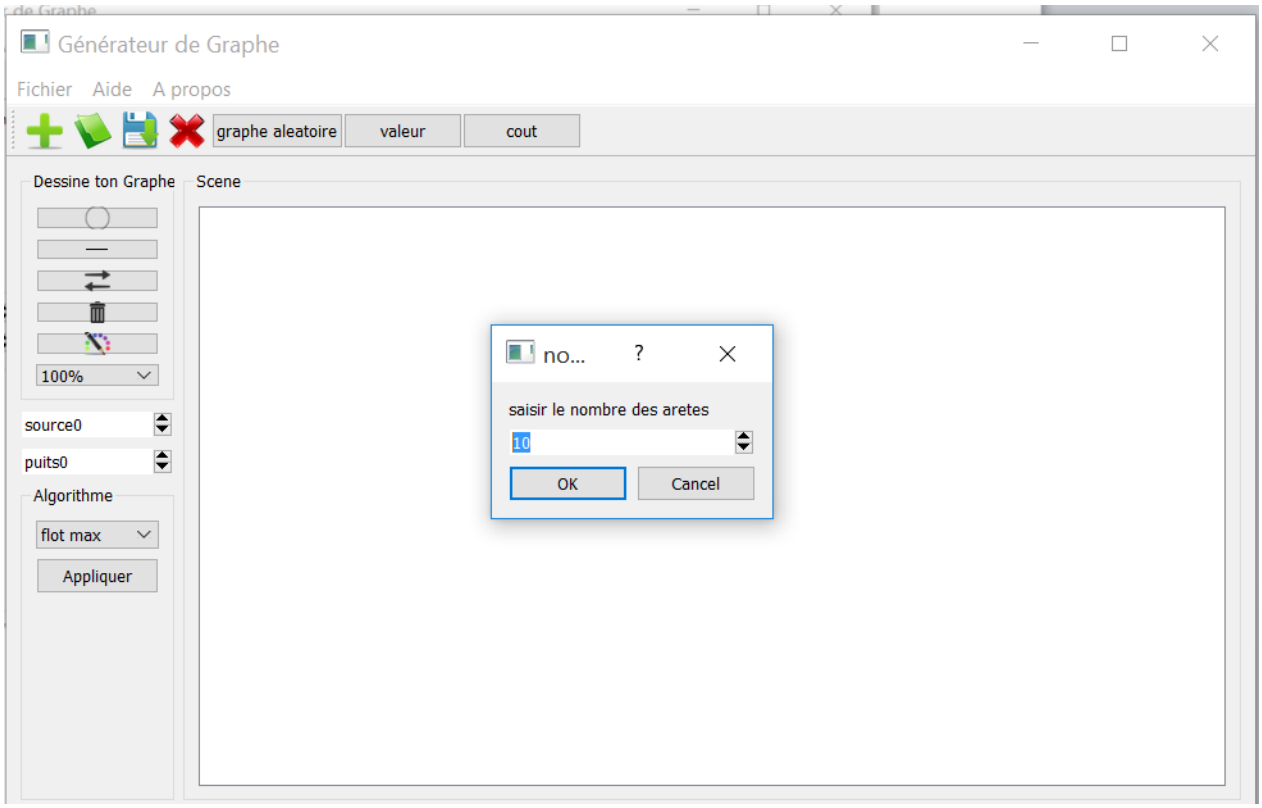
L’algorithme réussit a ouvrir le fichier texte.

On appuie sur le bouton graphe aleatoire ce qui génère l'apparition d'une fenêtre(QInputDialog), on saisit le nombre de sommets, d'arêtes et d'arcs.

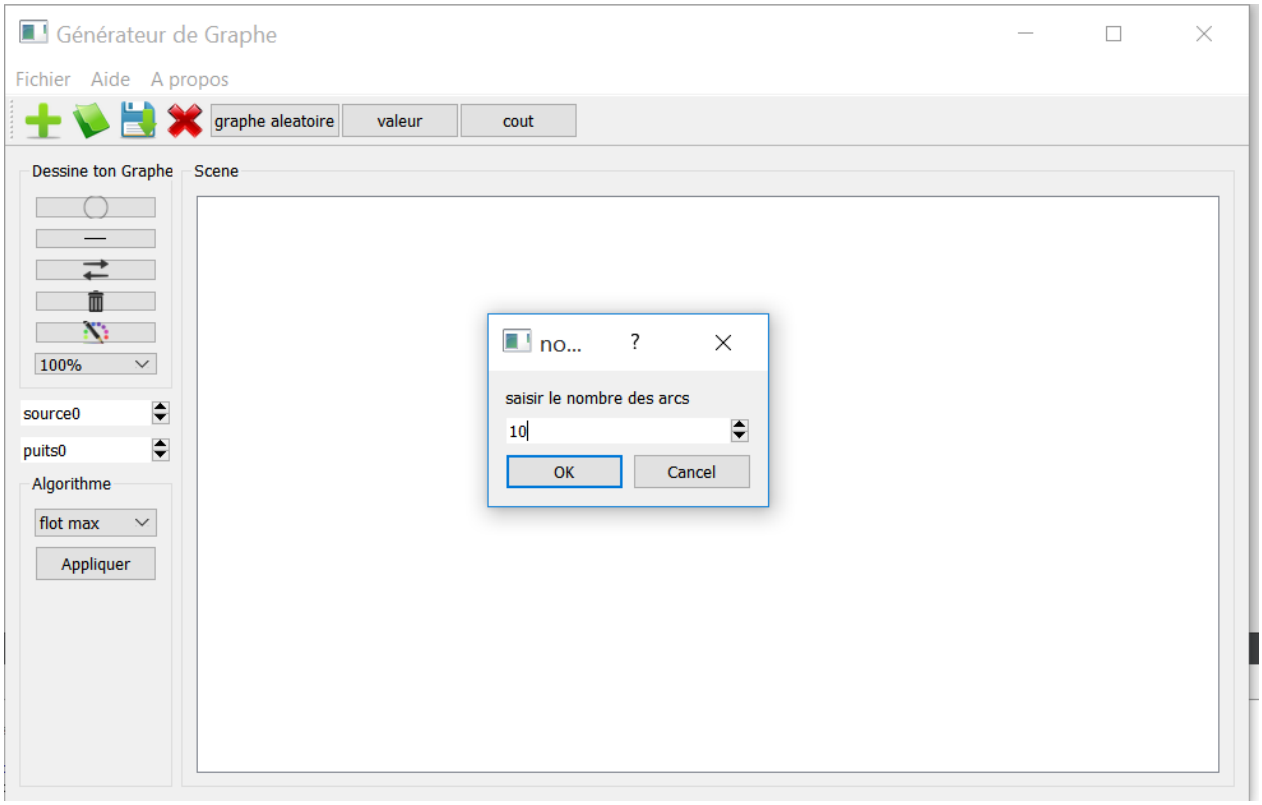
- <u>Nombre de sommets :</u>



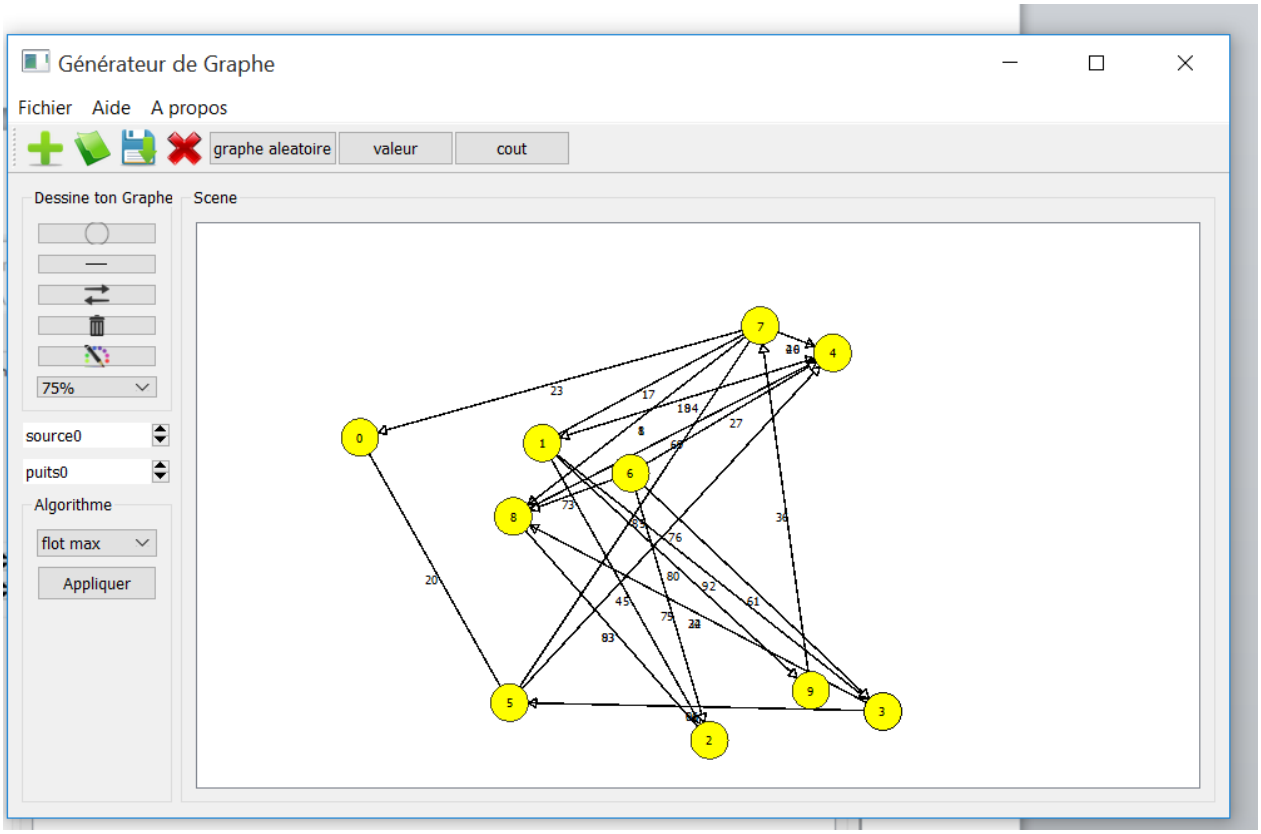
- Nombre d'arêtes :



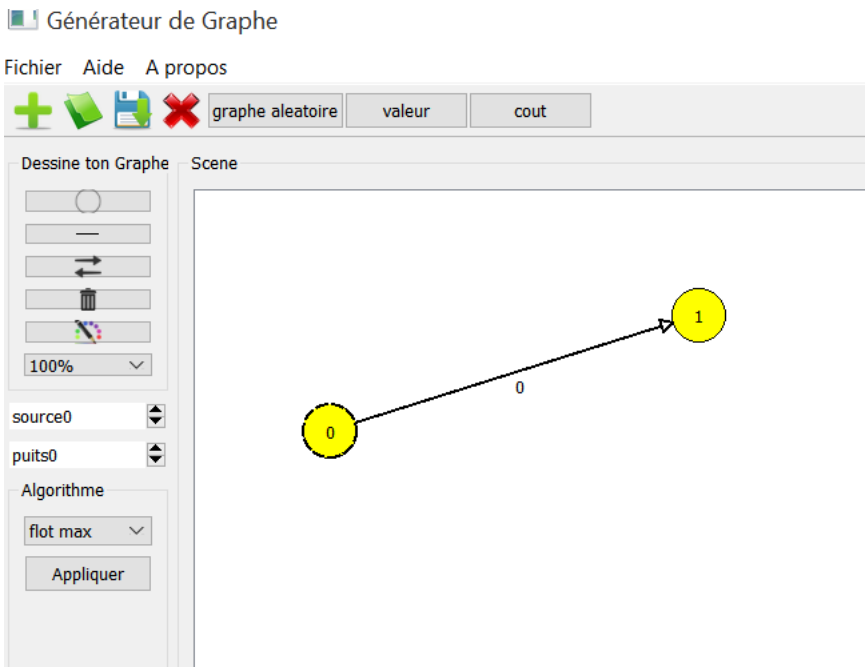
- Nombre d'arcs :



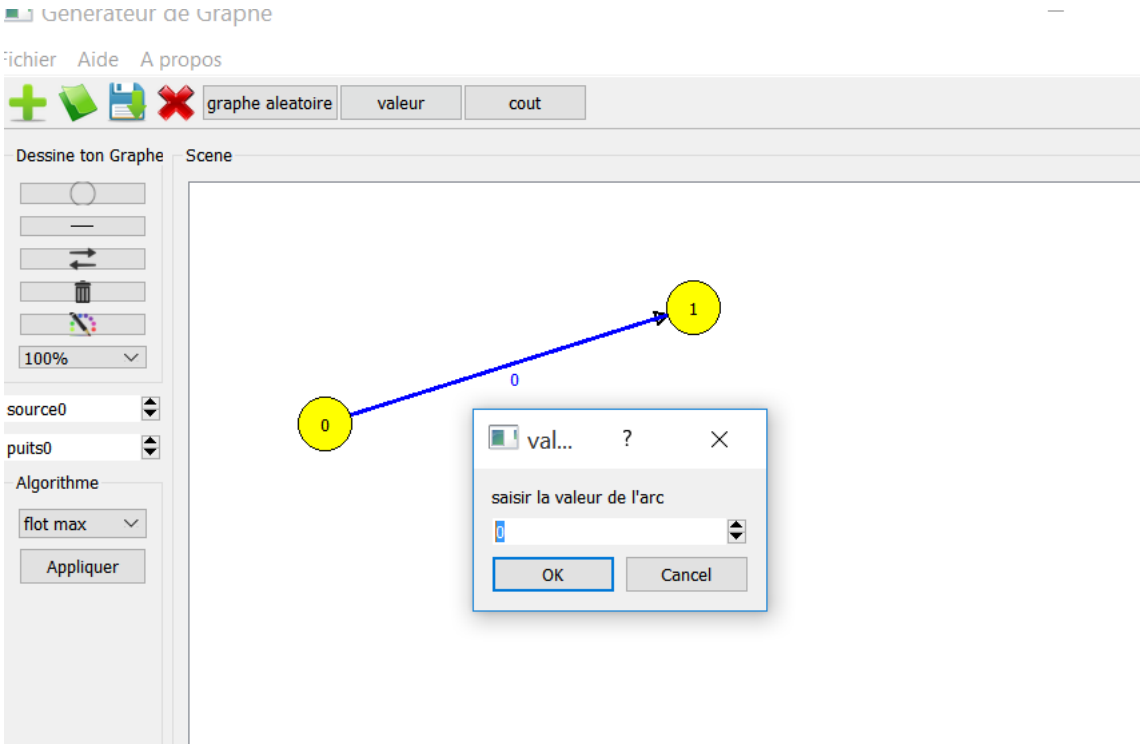
L’algorithmes a reussit a dessiner le graphe :

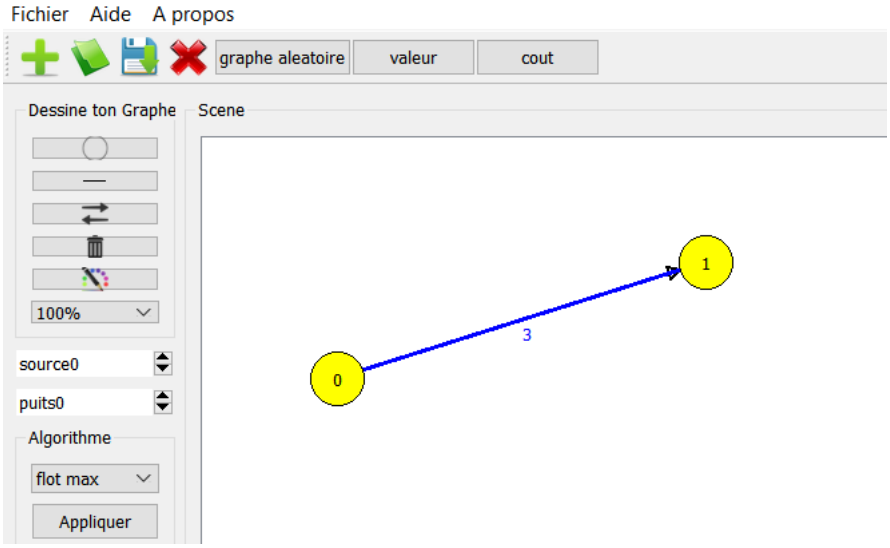


Par default la valeur c’est 0

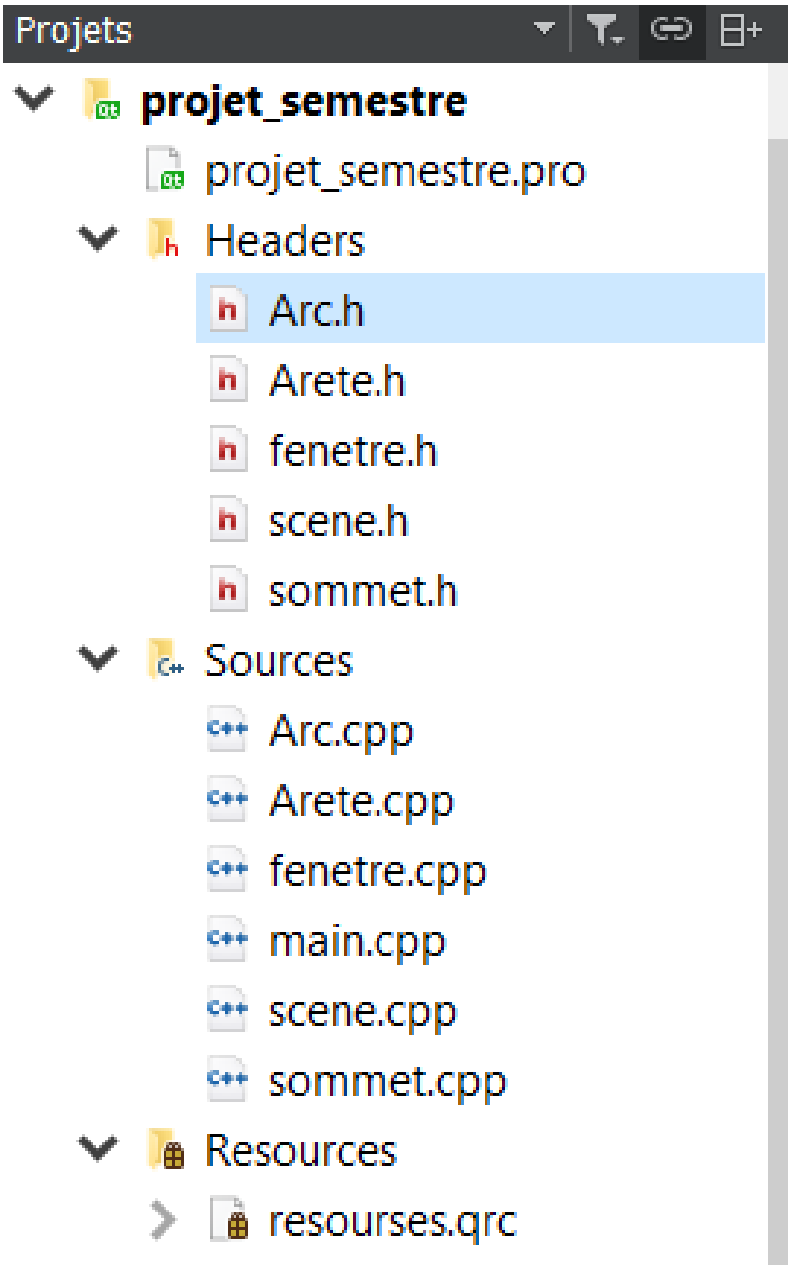


On appuie sur le bouton selectionner puis sur l’arcs et on lui affecte une valeurs





4-STRUCTURE DE PROJET



5-LES METHODES UTILISEES DANS CHAQUE CLASSE :

DANS LA CLASSE SOMMET :

```
 int type() const { return Type;}

 //*******
sommet();
//*******
int nombre();
bool getmarquer();
void setmarquer(bool);
void addArc(Arc *arc);
void addArete(Arete *art);
QRectF boundingRect()const;
void paint(QPainter * painter, const QStyleOptionGraphicsItem * option, QWidget * widget = 0);
QVariant itemChange(GraphicsItemChange change, const QVariant &value);
void removeArc(Arc *arc);//delete arc
void removeArcs();//delete arcs

QList<Arc *> getarcs(){return arcs;}
QList<Arete *> getaretes(){return aretes;}
void removeArete(Arete *art);//delete arc
void removeAretes();
 int sommetId();
```

DANS LA CLASSE SCENE

```
void mousePressEvent(QGraphicsSceneMouseEvent *event);
void mouseMoveEvent(QGraphicsSceneMouseEvent *event) ;
void mouseReleaseEvent(QGraphicsSceneMouseEvent *event) ;
```

DANS LA CLASSE FENETRE

```
Q_OBJECT
public :
    fenetre();
    //l'interface *****************
    void menu();
    void toolbar();
    void graphboutton();
    void algolayout();
    void scene();
    //*****************************
    void newfile();
    int nb(int n);
    friend void generergraphe();
    bool z2=false;
    bool y2=false;
    //les fonctions:
    friend void initialisertableaux(int);
    void raffraichirlestableaux();
   //flot max:
   void chaineameliorante();
   int flotmax();
   void initialiser_Lij();
   int flotmax_coutMin(sommet*,sommet*);
   bool existancechaineameliorante(sommet *s,sommet *t);
   int nbrefils(sommet*);
   //flot max a cout minimum:
     sommet*getsource();
     sommet*getpuits();
```

```
//*******algo*****************************************************************************
int n=0;
void fenetre::raffraichirlestableaux(){ ...}
void fenetre::initialiser_Lij() {...}

//*
void fenetre:: chaineameliorante(){ ...}


//*************************************************************************
int fenetre:: flotmax(){ ...}

QQueue<sommet*>fcm;
int fenetre::flotmax_coutMin(sommet *s,sommet *t) {...}
sommet *fenetre::getsource() {...}

sommet *fenetre::getpuits() {...}

int fenetre::nbrefils(sommet *s) {...}
void fenetre::recupererindicealgo(){ ...}
void fenetre::appliqueralgo() {...}
//*****************************

//fin algo1*****************************************************************************
```

DANS LA CLASSE ARETE

```
int type() const override { return Type; }
QRectF boundingRect() const override;
sommet *sommet_deb() const {  return sommet_debut; }
sommet *sommet_f() const {  return sommet_fin; }
 void updatePosition();
 QString creerstring(Arete *item);
 //********
int getflot1();
void setflot1(int t);
 int getflot2();
 void setflot2(int t);
 void setcout(int c);
int getcout();
 int getcapacite();
 void setcapacite(int);
 void setarete(int);
 int getarete(){return arete;}
ected:
void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget = 0) override;
```

DANS LA CLASSE ARC

```
   int type() const override{ return Type; }//[virtual] int QGraphicsItem::type() const
   sommet *sommet_deb() const {  return sommet_debut; }
   sommet *sommet_f() const {  return sommet_fin; }
   void updatePosition();
   QString creerstring(Arc *item);
   QRectF boundingRect() const override;
/**********
   int getflot();
   void setflot(int t);
   void setcout(int c);
   int getcout();
   int getcapacite();
   void setcapacite(int t);
   int getarc();
   void setarc(int);
   //*********
otected:
   void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget = 0) override;
ivate:
```

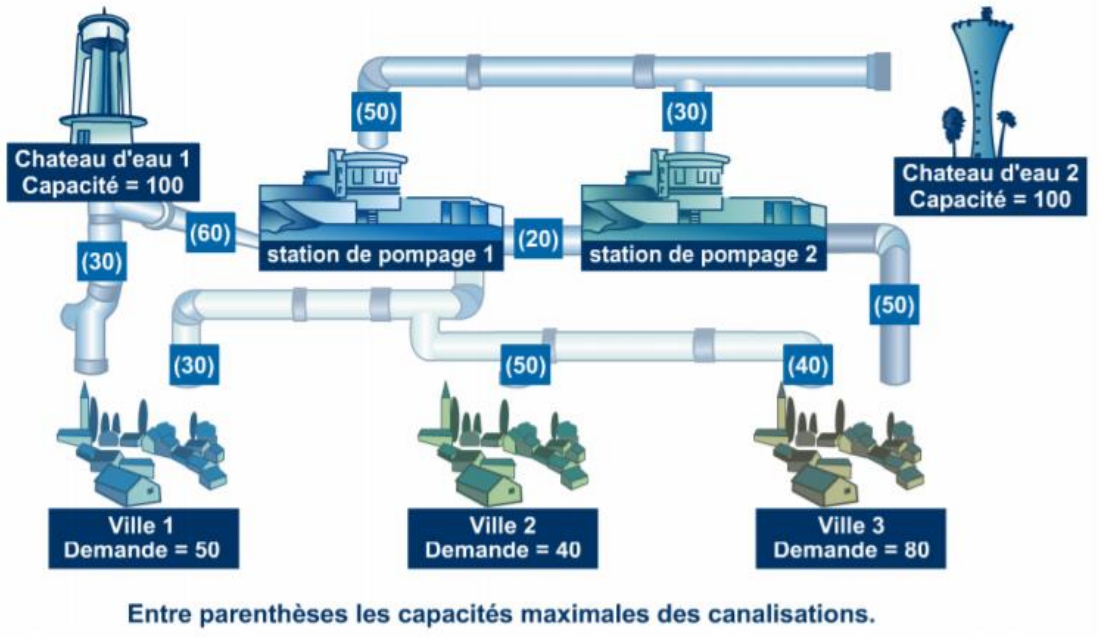
6-GENERALITE ET ETUDE FONCTIONNELLE

DESCRIPTION DU PROGRAMME:

- Lorsqu'on lance le programme, une fenêtre apparait illustrant un menu qui donne à l'utilisateur la possibilité de dessiner une graphe et d'appliquer un algorithme sur ce dernier.
- Pour générer le programme, soit on le clique manuellement en appuyant sur les boutons des sommets, arcs et arêtes, soit on cliquer le bouton graphe aleatoire et de taper les nombres des éléments nécessaire afin de générer un graphe aleatoire.
- On appuie sur le bouton sélectionner et on sélectionne l'arête qui nous intéresse, puis on appuie sur le bouton 'valeur' afin de lui affecter une capacité.
- On désigne le numéro de la source et du puits.
- On choisit l'algorithme qu' on veut appliquer sur le graphe déjà dessiner.

7-INITIALISATION DU PROBLEME :

Deux châteaux d'eau alimentent 3 villes à travers un réseau de canalisations au sein duquel se trouvent également des stations de pompage. Les châteaux d'eau ont une capacité limitée qui s'élève pour chacun d'eux à 100 000 m3 .



Entre parenthèses les capacités maximales des canalisations.

Les villes ont exprimé une demande qui est au minimum de 50 000 pour la ville 1, 40 000 pour la 2 et 80 000 pour la ville 3 en m3 . Les canalisations entre les châteaux d'eau et les villes ont des débits limités. Par exemple, pour la canalisation reliant le château 1 à la ville 1, le débit maximum est de 30 alors que celui de la canalisation reliant la station de pompage 1 à la ville 2 est de 50 en milliers de m3 . Ces valeurs figurent sur le graphique entre parenthèses le long des canalisations. Un premier problème est de déterminer s'il est possible de satisfaire à travers ce réseau la demande des 3 villes et comment ? Pour résoudre ce problème il faut dans un premier temps le modéliser. Pour cela, nous introduisons un nouveau problème standard qui est celui du flot maximal sur un réseau.

8- PROBLEME DE FLOT :

Pour un graphe orienté G(V,E),à chaque arc partant d'un sommet u et arrivant sur un sommet v on associe une valeur dite capacité c(u,v),et on peut définir une fonction dite flot avec les contraintes suivantes f(u,v)<=c(u,v) et f(u,v)=-f(v,u). autre definition : capacité residuelle,reseau residuel,chemin croissant.

THEOREME DE FORD-FULKERSON

Théorème de Ford-Fulkerson

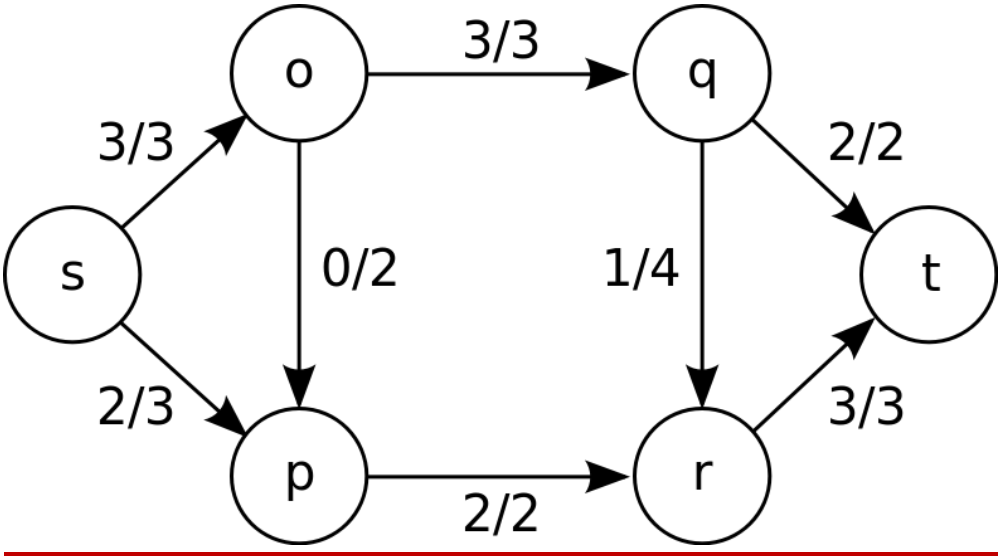
Soit $G = (E, \Gamma, c)$ un graphe valué. Pour tout flot réalisable f et toute coupe $(X, \overline{X})$, on a

$$v(f) \leq c(X, \overline{X})$$

où $v(f)$ est la valeur du flot f.

FLOT MAX

Le problème de flot maximum consiste à trouver, dans un réseau de flot, un flot réalisable depuis une source unique et vers un puits unique qui soit maximum. Ex : On veut par ex. trouver le trafic maximal entre deux villes d'un réseau routier dont on connait la capacité (nbr de voiture par heure sur chaque tronçon).



Algorithme de Ford-Fulkerson :

Algo principal :

Algorithme 15: Algorithme de Ford-Fulkerson

Données : Un réseau $G = (X, A, C)$
Résultat : Un flot Φ

```
1 Initialiser Φ à 0 sur tous les arcs
2 répéter
3     Chercher une chaîne améliorante μ de s à t
4     si μ existe alors
5         Calculer δ
6         Augmenter Φ sur les arcs de μ+ de δ
7         Diminuer Φ sur les arcs de μ− de δ
8 jusqu'à Il n'existe plus de chaîne améliorante
9 retourner Φ
```

Principe

Tant qu'il existe un chemin augmentant dans le graphe, on ajoute un flot le long de ce chemin.

Chemin améliorant ou augmentant

Un chemin améliorant ou chemin augmentant, est un chemin de $G_{\text{résiduel}}$, allant de s à p et sans circuit

Principe

Créer un graphe résiduel $G_r = G$
Initialement tous les arcs sont valués par leur capacité
Déterminer un chemin augmentant
TantQue il existe un chemin augmentant de s vers p **Faire**
    mettre à jour le graphe résiduel
    chercher un nouveau chemin augmentant
finTantQue

On construit un graphe résiduel pour pouvoir utiliser l´algo de Ford Fulkerson, qui consiste a :

1/ trouver une chaine améliorante de la source au puits

2/ajouter la valeur résiduelle la plus faible sur la chaine améliorante jusqu' a ce que un arc soit sature

Et en même temps augmenter le flot max

3/on s'arrête dès qu'on trouve plus de chaine améliorante et on retourne le flot max.

FLOT MAX A COUT MINIMUM

Definition :

Définition du problème

Soit $G = (N, A)$ un réseau de transport, c'est-à-dire un graphe orienté sur lequel sont définis :

- $d : N \to \mathbb{R}$ une fonction prenant des valeurs positives pour les nœuds sources (*i.e.* produisant des ressources), négatives pour les nœuds puits (*i.e.* utilisant des ressources) et nulles pour les nœuds dits de transit ;
- $f_{max} : A \to \mathbb{R}$ une fonction associant à chaque arête le flot maximum qu'elle peut supporter ;
- $c : A \to \mathbb{R}$ une fonction mesurant le coût du transport par unité de flot pour une arête donnée.

En supposant qu'il existe un flot réalisable, le problème du flot de coût minimal consiste, à trouver un flot $f : A \to \mathbb{R}$ minimisant le coût total : $\sum_{a \in A} c(a) f(a)$

sous les contraintes :

- conservation du flot : $\forall n \in N, \sum_{a=(n,s) \in A} f(a) - \sum_{a=(s,n) \in A} f(a) = d(n)$
- contrainte de capacité : $\forall a \in A, 0 \leq f(a) \leq f_{max}(a)$

L'algo :

TROUVER UN FLOT RÉALISABLE

- Vérifier que l' offre totale égale la demande totale:

$$\sum_{i \in N} b_i = 0$$

- Définir une source s et un puits t artificiels.
- Pour tout noeud source, i avec $b_i > 0$, créer un arc (s, i) de capacité $u_{s\,i} = b_i$.
- Pour tout noeud puits, i avec $b_i < 0$, créer un arc (i, t) de capacité $u_{i\,t} = -b_i$.
- On trouvera une solution réalisable pour le problème initial en résolvant le problème de flot maximum dans le réseau modifié.

Theorem

$\exists x$ tel que $\sum_{j \in N} x_{i\,j} - \sum_{j \in N} x_{j\,i} = b_i \quad \forall i$ and $x_{i\,j} \leq u_{i\,j}\ \forall (i,j)$ ssi le flot maximum de s à t dans le réseau modifié est $v = \sum_{i:\, b_i > 0} b_i = -\sum_{i:\, b_i < 0} b_i$

UTILISATION DE L'ALGORITHME DE FORD

On utilise l'algorithme de Ford afin de trouver le chemin qui a le plus faible cout.

Algorithme de FORD

Hypothèse particulière : aucune

Algorithme

- On pose $\lambda_1 = 0$ et $\lambda_i = +\infty \quad \forall \ i \neq 1$
- On cherche un arc (x_i, x_j) tel que $\lambda_j - \lambda_i > l_{ij}$ et on remplace λ_j par $\lambda'_j = \lambda_i + l_{ij}$
- On continue ainsi jusqu'à ce qu'aucun arc ne permette de diminuer les λ_j
- Il existe nécessairement un sommet x_{p_1} tel que $\lambda_n - \lambda_{p1} = l_{p_1 n}$

  un sommet x_{p_2} tel que $\lambda_{p_1} - \lambda_{p_2} = l_{p_2 p_1}$

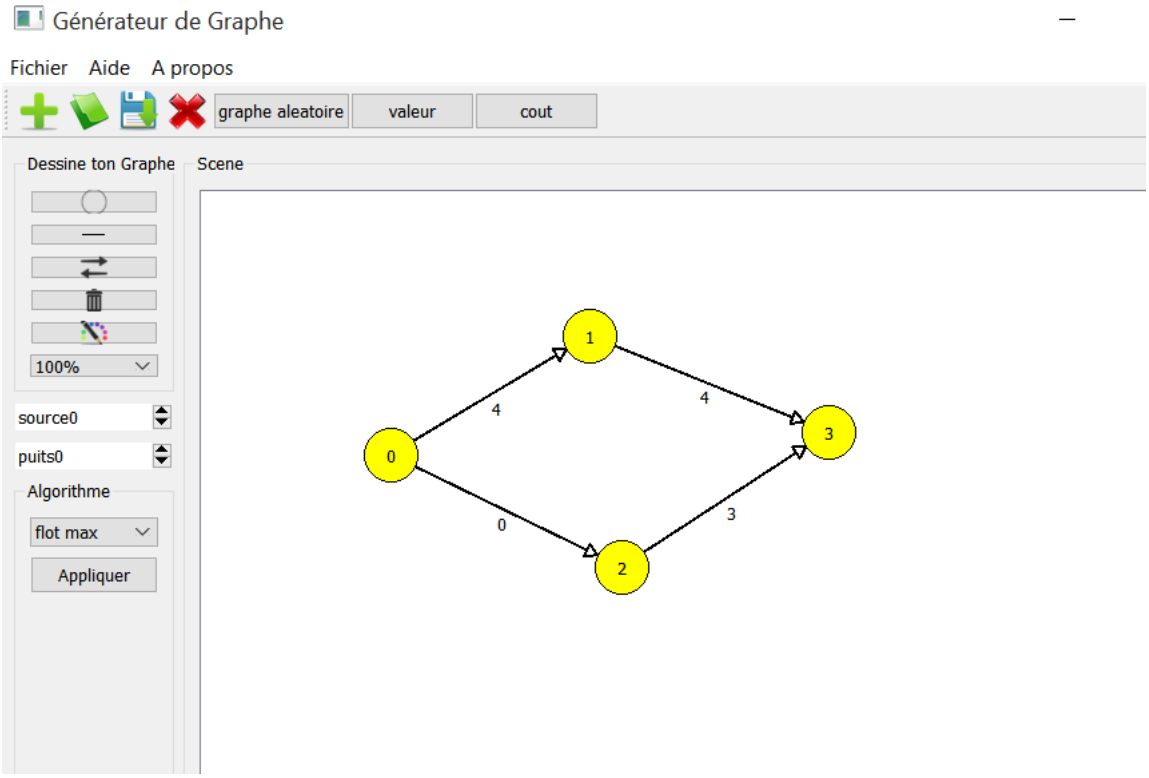
  et ainsi de suite jusqu'à ce que $x_{p_k} = x_1$

Le chemin $(x_1, x_{p_{k-1}}, x_{p_{k-2}}, \ldots, x_{p_1}, x_n)$ est optimal

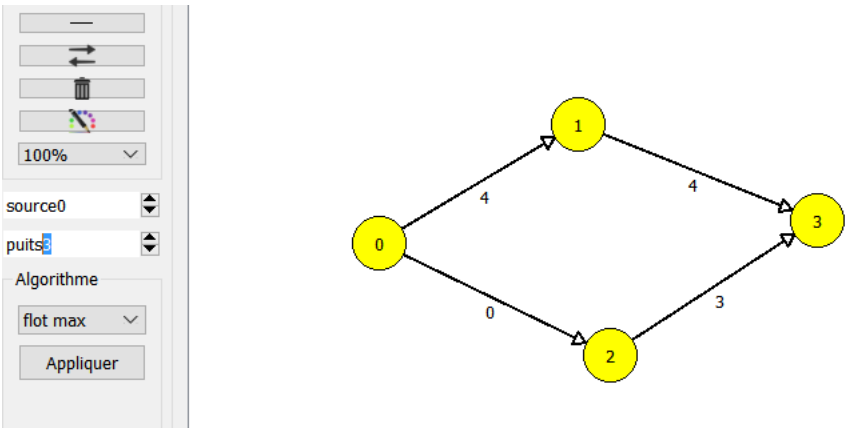
9-APPLICATION DES ALGORITHMES :

Pour l'algorithme de flot max :

- On construit notre graphe, en lui affecte les valeurs (capacités).

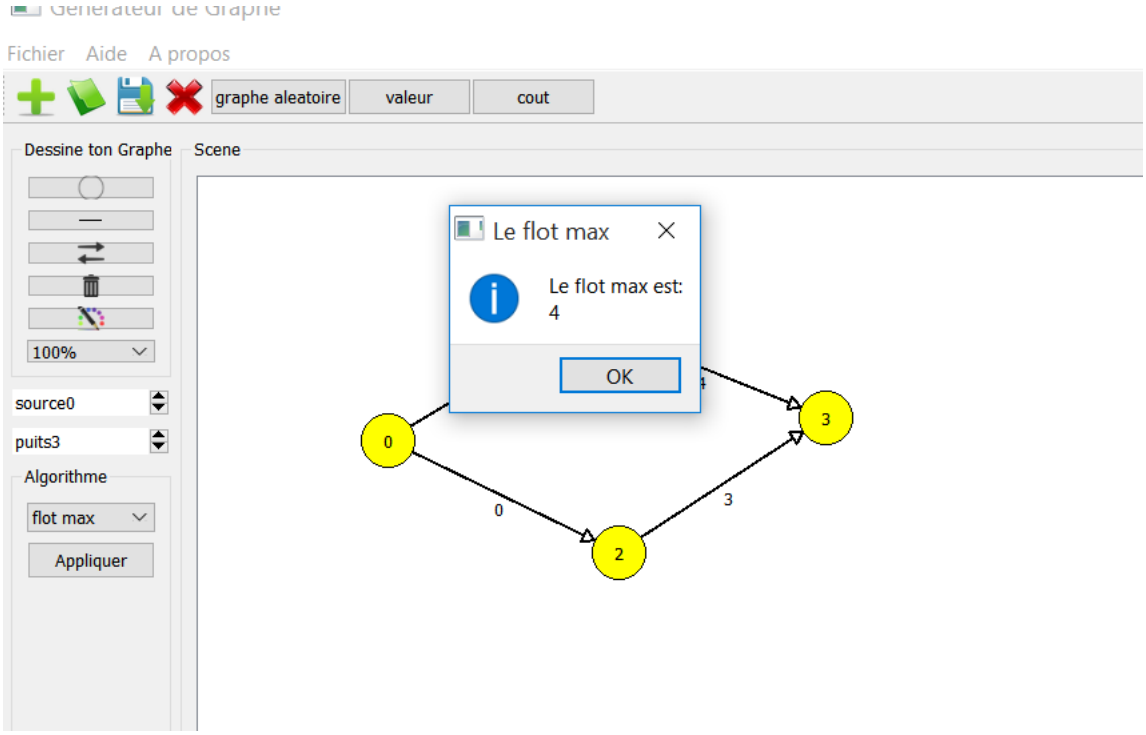


- On choisit la source et le puit.



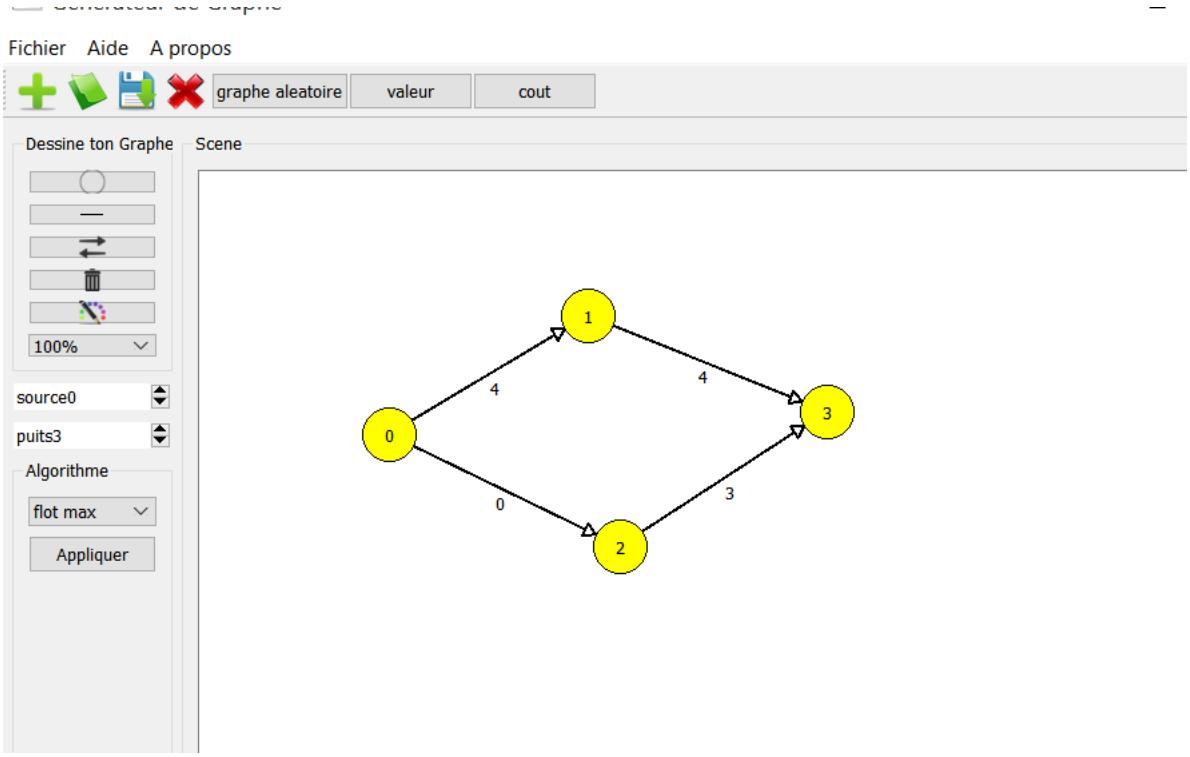
- On choisit l’algorithme du flot max.

- L’algorithme affiche le résultat.

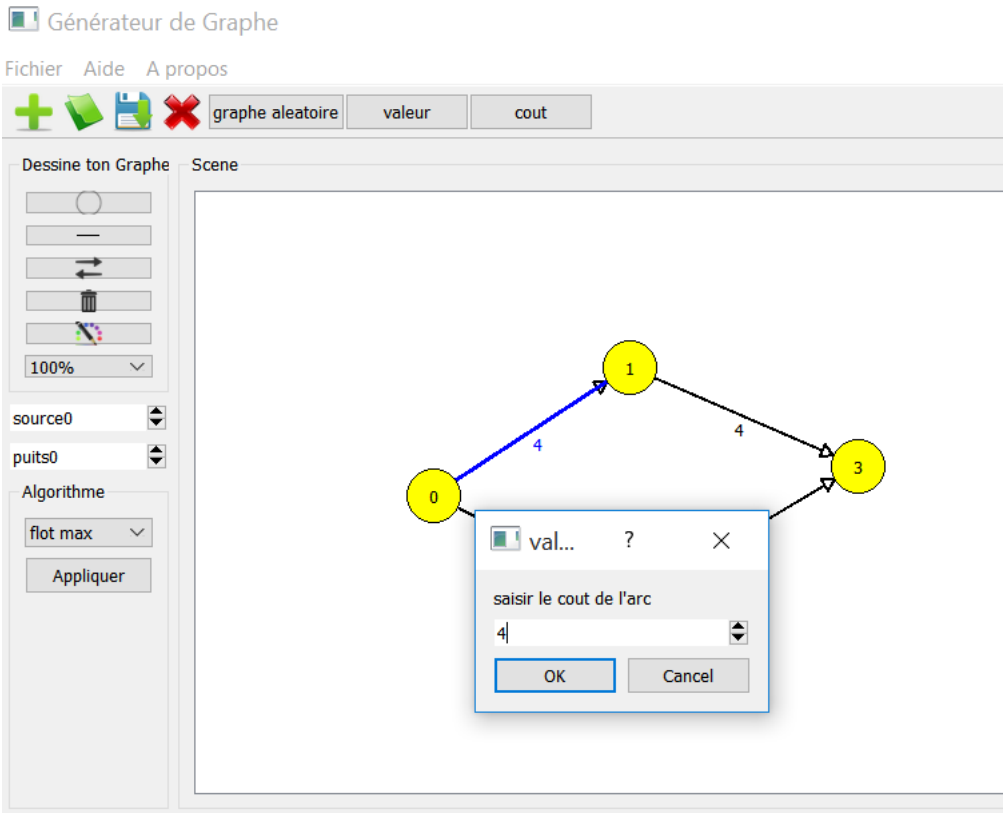


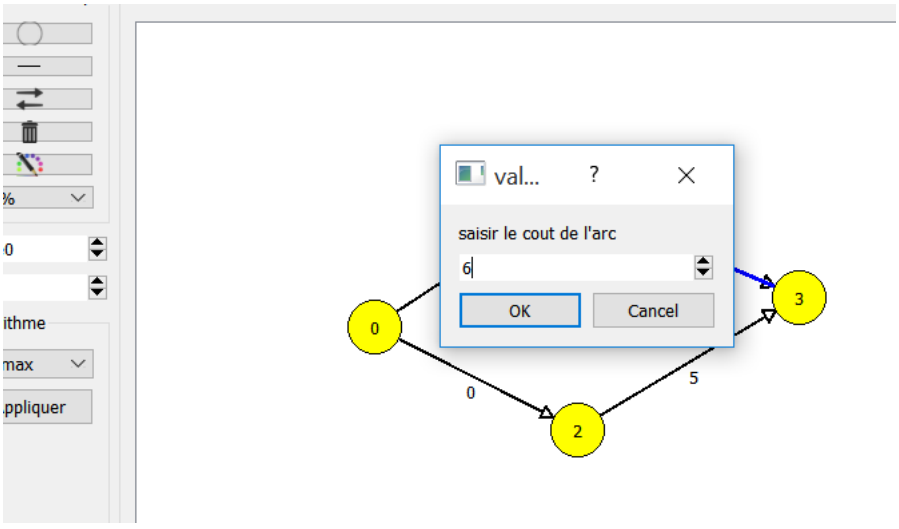
Pour l’algorithme de flot a cout minimum :

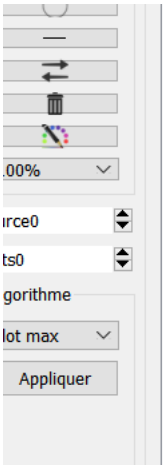
En prend le même exemple :

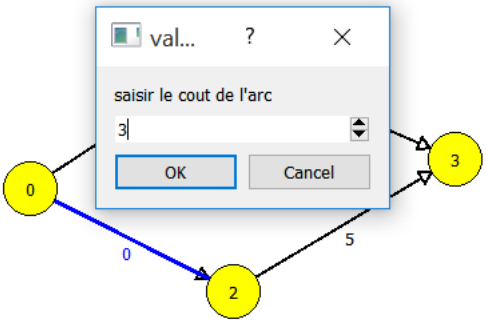


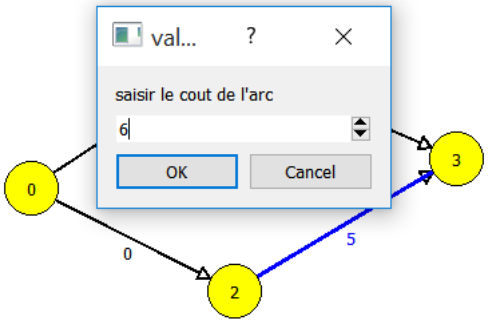
- Il faut ajouter les couts a chaque arc :



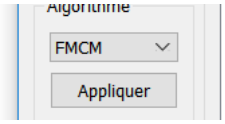




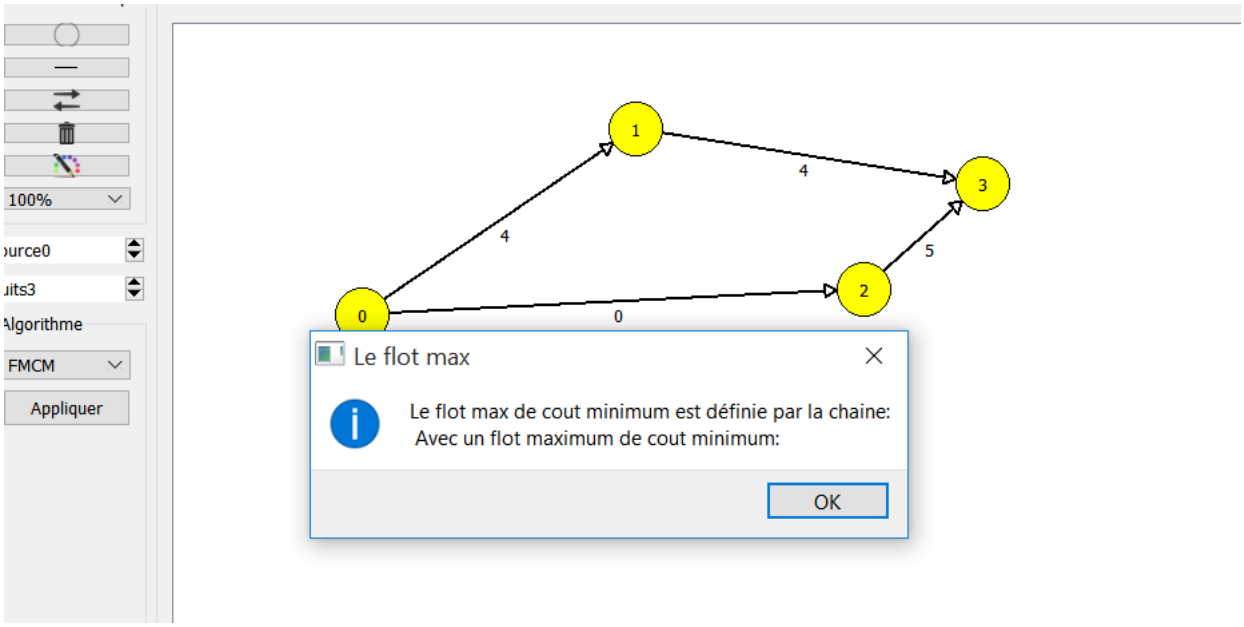




- Apres on choisit le deuxième algorithme.



Problème : l’algorithme n’affiche pas le bon résultat.



Conclusion

Le problème de flot maximal est un problème intéressant qui permet de trouver une répartition efficace des différentes quantités repartis sur un réseau de capacité.

D'autre problèmes ont apparu et ont empreinte le principe de flot maximal à cout minimum.

Depuis l'algorithme de FORD-FULKERSON jusqu'à nos jours, différents algorithmes ont été proposés pour le calcul de flot maximum, ces algorithmes visent, principalement à diminuer la complexité.