Théorie Des Graphe

Mendy Fatnassi

10 décembre 2020

Table des matières

1 Les graphes **3**
- 1.1 Graphe Orienté/Non-Orienté . . . 4
- 1.2 Connexité/Forte Connexité . . . 5
  - 1.2.1 Composante Connexe/Fortement Connexe . . . 5

2 Parcours de graphe **7**
- 2.1 Parcours Eulerien . . . 7
- 2.2 Parcours Hamiltonien . . . 7

3 Graphe Planaire **8**

4 Bipartisme **9**
- 4.1 Affectation de registre . . . 9
  - 4.1.1 Algorithme Heuristique . . . 9
  - 4.1.2 Graphe Bipartie . . . 10

5 Structure et algos de base **11**
- 5.1 Structure . . . 11
- 5.2 Matrice d'incidence . . . 12
- 5.3 Graphe Inverse . . . 12
  - 5.3.1 Algos Graphe Inverse . . . 12
- 5.4 Tri Topologique . . . 13
  - 5.4.1 Algos Tri Topologique . . . 13
- 5.5 Exploration de graphe . . . 14
  - 5.5.1 Algos . . . 14

6 Parcours en Profondeur/Largeur **16**
- 6.1 Parcours en Profondeur . . . 16
- 6.2 Parcours en Largeur . . . 16

7 Distancier **17**
- 7.1 Algos Distancier . . . 17
- 7.2 Algos Potentiel tache . . . 17

8 Dijsktra **18**
  - 8.0.1 Plus long chemin . . . 18

Chapitre 1 : Les graphes

Les graphes sont utilisé en mathematique et en informatique.
Un graphe G est representer par ces *sommet* **S** et A des relations entre ces sommet qu'on appele *arcs* , on le note donc $G =< S, A >$.

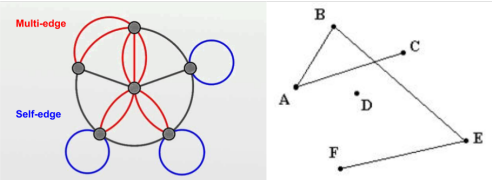
Vocabulaire :

Densité d'un graphe : Représente un nombre compris entre [0,1.] et se note : **Orienté** : $D = \frac{A}{S(S-1)}$ *ou* **Non-Orienté** : $D = \frac{A}{\frac{S(S-1)}{2}}$ (!Ne marche pas avec un graphe a 1 sommet).

Graphe Simple : Graphe dans lequel chaque paire de sommets est reliée par au plus une arête et aucun sommet ne possède de boucle.

Multigraphe : A l'inverse du graphe simple (simple graph), le multigraphe (multigraph) est un graphe qui autorise des liens multiples (aussi appelés liens parallèles). En d'autres termes, dans un multigraphe, deux sommets peuvent être connectés par plus d'un lien.

Exemple graphe Simple/Multigraphe :



Graphe Complet : C'est un graphe simple qui a toutes ses arretes (ou arc) relier. Les graphes K3,3 ;K4 et K5 sont es graphes complet (*cf.Graphe Planaire*).

1.1 Graphe Orienté/Non-Orienté

Un graphe peux etre soit :

-**Non-Orienté** cad que les sommets seront liée simplement par un trait il n'y a pas de sens pour aller d'un sommet quelconque vers un autres.

-Une **aretes** : ensemble fini de paires de sommets.

-Une **chaine** : relie un sommet S0 a un sommet Sn correspondant a une suite de sommets ou deux element successif sont relier par une arete.

-Un **cycle** : est une chaine de longueur non nulle reliant un sommet a lui meme.

-**Orienté** c'est a dire qu'il y a un sens de parcours du graphe , les trait seront representer par des flèches .

-Un **arcs** : ensemble fini de couples de sommets.

-Un **chemins** : relie un sommet S0 a un sommet Sn correspondant a une suite de sommets ou deux element successif sont relier par un arc .

- Un **circuit** : est un chemin de longueur non nulle reliant un sommet a lui meme .

Exemple :

Longueur d'un chemin (chaine) : Nombre d'arcs(aretes) empruntés .
Chemin elementaire : Chemin dont tous les sommets sont distincts.

1.2 Connexité/Forte Connexité

Vocabulaire :

- Un sous-graphe correspond au graphe de départ moins certain sommet. On le notera S'(prime).
- Un graphe partiel correspond au graphe initiale moins certain liens.
- Un sous-graphe partiel : S' $<$ S moins certain liens.

Connexe :
Un graphe non-orienté est connexe si toute paires de sommets distincts est reliée par une chaine.

Fortement Connexe :
Un graphe orienté est fortement connexe si tout couple de sommets distincts est reliée par un chemin.
Il existe un chemin de tout sommet i vers j et inversement.

1.2.1 Composante Connexe/Fortement Connexe

Composante Connexe (CC) :
Un sous-graphe d'un graphe non-orienté est une composante connexe , si il est connexe et si il est maximal. Un sous-graphe connexe est maximal si on ajoute un sommet dans ce sous-graphe et qu'il n'est plus connexe .

Une composante connexe est donc un sous-graphe connexe maximal.

Composante Fortement Connexe CFC) :
Un sous-graphe d'un graphe orienté est une composante fortement connexe , si il est fortement connexe et si il est maximal. Un sous-graphe connexe est maximal si on ajoute un sommet dans ce sous-graphe et qu'il n'est plus connexe .
On peux trouver une chemin de i vers j et inversement .

Tableau recapitulatif des terminologie :

Orienté	Non Orienté
Arc	Arête
Chemin	Chaîne
Circuit	Cycle
Fortement Connexe	Connexe

Chapitre 2 : Parcours de graphe

2.1 Parcours Eulerien

Un parcours eulerien passe par chaque arrete du graphe (pas forcément une 1 seul fois).

Théoreme d'euler :
Si le nombre de sommet de degré impair est egale a :
- 0 : Il existe un cycle solutions
- 2 : Il existe toujoure une chaine qui vas d'un sommet impair vers un autre sommet impair

Sinon il n'y a pas de solution.

2.2 Parcours Hamiltonien

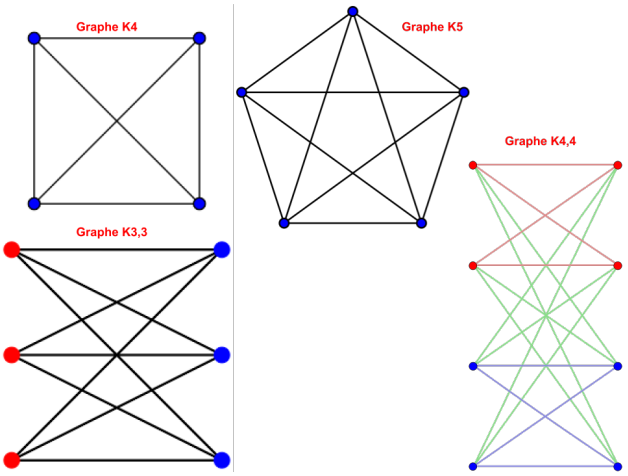
Un parcours hamiltonien passe par chaque arréte 1 seul fois.

Chapitre 3 : Graphe Planaire

Un graphe est dit planaire si on peux le dessiner sur un plans et n'ayant aucun croisement.

Théoreme de Kuratowski :
Un graphe est planaire si et seulement si il ne contient aucun sous-graphe reductible a K3,3 ou K5.

Voici un exemple de graphe K3,3 , K4 et K5 qui d'ailleur sont des graphes complet :

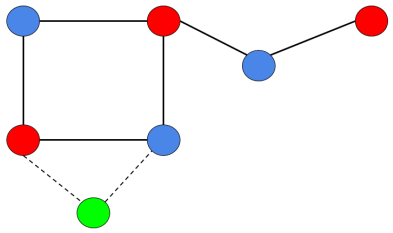


Chapitre 4 : Bipartisme

4.1 Affectation de registre

L'affectation de registre dans un graphe revient a traité un probléme de coloriage de graphe.
Pour savoir quelle info doit etre traité avant (Si c'est plus important ou non).

Pour savoir combien de registre on vas affecter au graphe nous allons essayé de le colorier avec le moins de couleur(registre) possible.
On a une régle a respecter : On ne peux pas avoir cote-a-cote une meme couleur.



4.1.1 Algorithme Heuristique

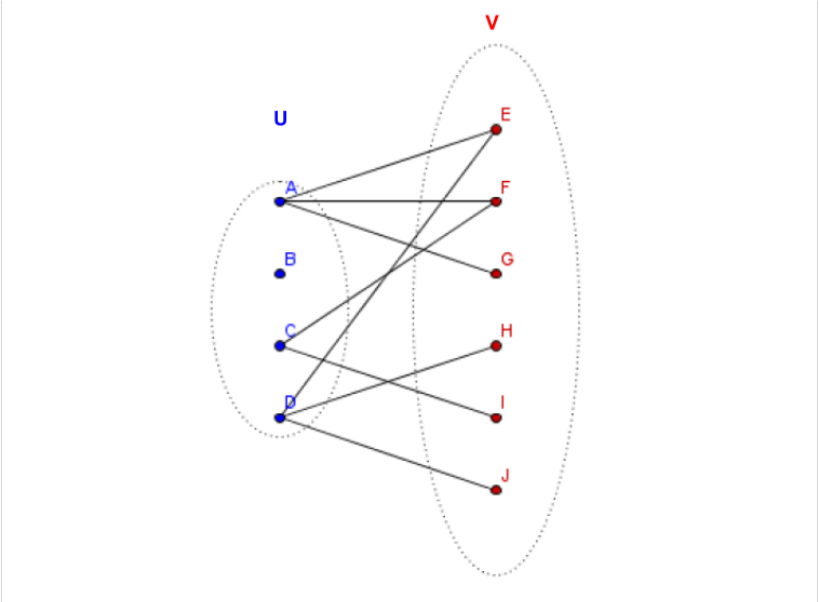
```
rep
  x <- sommet+haut degré
  empiler                                    // Detruie G
  G <- G\{x}
frep

rep
  x <- depiler()
  colorier                                   // Construire G
  G <- G U {x}
frep
```

4.1.2 Graphe Bipartie

Un graphe est dit biparti si on peut partager son ensemble de sommets en deux parties A et B tels qu'il n'y ait aucune arête entre éléments de A et aucune arête entre éléments de B.

Autrement dit, les graphes bipartis sont ceux que l'on peut colorer en utilisant au plus deux couleurs. Le théorème suivant, dû à König en 1916, caractérise les graphes bipartis :

Théorème de konig : Un graphe est biparti si et seulement s'il ne contient pas de cycles de longueur impaire.



Rappelons que la longueur d'un cycle est égale au nombre d'arêtes qu'il contient. En particulier, d'après le théorème précédent, les arbres sont des graphes bipartis.

Chapitre 5 : Structure et algos de base

Probléme : On veux enumérer les graphes simple a 8 sommets et 3 couleur par arrete ?

Le nombre de graphe seras : 4x4x4....x4=$4^28 = 2^56 = 2^{60-4} = \frac{2^{30}x2^{30}}{2^4} \approx 7.10^{16}$

Hypothése : Le super ordinateur énumere $2^{30}(10^9)$ graphes par secondes.

Temps totale : $\frac{2^{30}}{16} = 67108864sec$ donc en 1 année = $\pi x10^7$ sec avec une erreur $<$ a 1% .
$t = \frac{\pi x10^7 x10^2}{\pi x16} = \frac{100}{16\pi} anne 2ans$

Il faut donc 2ans pour tout enumérer. L'enumeration n'est donc pas une bonne solution, surtout quand le nombre d'echantillon a traité est grand, on préferera utilisé :
-Des Structure de Donnée
-Des Algorithme sur les graphes

Certain probléme de graphe nécessite beaucoup de temps de calcule, or on veux un resultat cas immédiat (1 semaine au plus) pas dans 2 ou 5ans.

5.1 Structure

-Pile ou File -> Dequeu (double queu)
-Une dequeu fonctionne soit comme une pile soit comme une file (ne pas melanger cela depend du contexte).

Primitive :
-empiler(), depiler(), enfiler(), defiler() => O(1)
-cardinal, vide, present => O(1)

-tas monceau (heap) : Maximier, Minimier

5.2 Matrice d'incidence

Avantage :

-plus agréable pour l'homme.
-ajout/suppression des arcs ne change pas la dimension du tableau.

Inconvénient :

-Traitement en $O(S)^2$
-Met plus de temps a trouver l'arc suivant.

5.3 Graphe Inverse

Principe :
-Determiner la taille de chaque plexe dans H, graphe inverse de G.
-Remplir H.Suk et H.Head

5.3.1 Algos Graphe Inverse

```
//deg[] degrés sortant dans H

Razer (deg)

rep pour k de 1 a G.A   //A:Arrete
  deg[G.Suk[k]]++
frep
//H.Head <-- au dela de chaque bloc

H.Head[0] <- 1

rep x de 1 a G.S      //S:Sommet
  H.Head[x+1] <- H.Head[x]+deg[x]
frep
//H.Suk <-- remplie en marche arriere

rep x de 1 a G.S
```

```
  rep k de G.Head[x] a H.Head[x+1]-1
    y <- G.Suk[k]
    H.Head[x]--
    H.Suk[H.Head[y]] <- x
  frep
frep

//terminer la structure

H.S=G.S
H.A=G.A
H.Head[H.S+1]=H.A+1
H.Head[0]=0
```

temps totale : O(S+A) , on considére A S dans les graphes connexe.

5.4 Tri Topologique

Les champs sont inversé ; Suk devient Kus et Hed devient Deh.
Possible si le graphe ne contient pas de circuit.

Principe :
On commence par noté le premier sommet par 1 (ordre croissant), ensuite on marque chaque sommet S si le precedent de S est deja marqué.

5.4.1 Algos Tri Topologique

```
\\deg[] degré entrant
Razer(nivo)
Razer(deg)

Rep pour k de 1 a G.A
  deg[Suk[k]]++
Frep

//Initialiser la deque
Rep x de 1 a S
  Si deg[x]==0
```

```
    enfiler(x)
  Fsi
  N <- 0
Frep

Rep jusqu'a dequeVide()
  N++

  Rep de 1 a dequeCardinal()
    x <- defiler
    nivo[x] <- N

    Rep pour tout succ de H y de x
      deg[y]--
      Si deg[y]==0
        enfiler(x)
      Fsi
    Frep
  Frep
Frep
```

5.5 Exploration de graphe

Marquae des sommets.

5.5.1 Algos

```
Next <- Head
Mark <- 0 partout
x <- 1
Mark[x] <- vrai
E <- {x}

Rep jusqu'a E vide
  x <- 1 sommet qlq de E (utile plus tard)

  Si Next[x] < Hed[x+1]
    y <- Next[x]
```

```
    Si y non-marqué
      E <- E U {y}
    Fsi
    Next[x]++

    Sinon
      E <- E\{x}
  Fsi
Frep
```

Chapitre 6 : Parcours en Profondeur/Largeur

6.1 Parcours en Profondeur

On prend les sommets par ordre croissant, marqué le 1er sommet ensuite marqué ces successeur.
Si pas de successeur, revenir au sommet precedent (depiler) pour trouver un nouveau successeur tant qu'on est pas arrivée jusqu'a trouvé un nouveau successeur.

Resultat G1 : 1,2,3,4
Resultat G2 : 1,2,3,4,6 (5,7 et 8 ne font pas partie du parcours)
Resultat G3 : 1,2,3,4,7,6,10,5,9,8

6.2 Parcours en Largeur

Resultat G1 : 1|2,3|4|6
Resultat G2 : 1|2,3|4|6
Resultat G3 : 1|2,5,8|3,6,9|4,7,10

Chapitre 7 : Distancier

Un graphe G(S,A), on veut calculerla distance la plus courte entre tous les sommets (X,Y).
Changement de représentation des données.

7.1 Algos Distancier

```
Rep k de 1 a S
  Rep i de 1 a S
    Rep j de 1 a S
      Si D[i,j] > D[i,k]+D[k,j]
        D[i,j] <- D[i,k]+D[k,j]
        P[i,j] <- P[i,k]
      Fsi
    Frep
  Frep
Frep

D[][]

Rep pour i de 1 a S
  D[i][i] <- 0
  Rep j succ h de i dans G
    D[i][j] <- [i][j]
  Frep
Frep
```

7.2 Algos Potentiel tache

Donnée : Un graphe sans circuit , description de tache donnée par G.
Resultat : Date au plus tot/tard, Marges, Chemin critique.

Chapitre 8 : Dijsktra

8.0.1 Plus long chemin

Il y a une boucle `{2,3,2}`, ducoup pour trouvé le chemin le plus "long" on pourrai basculé a l'infinie or on veux un parcours elemenetaire (1 fois), ducoup pas possiblepour le graphe ci-dessus.