Accessiblité et chemins dans un graphe. Applications.

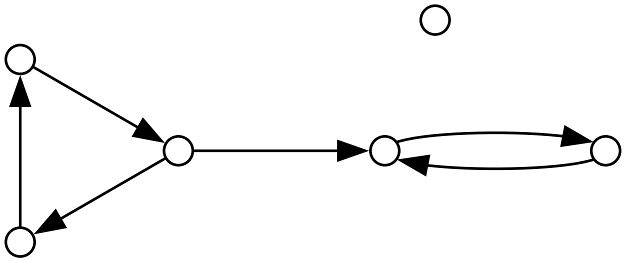
I. Définitions, généralités

A. Structure de Graphe [BAR 5.4.1 5.4.2] [TOR 8.1]

Définition 1 **Graphe** Un graphe G est un couple $G = (S, A)$ avec S un ensemble de sommets et A un ensemble d'arc/arêtes défini par une relation binaire entre les éléments de S.

Remarque 2 **Orienté vs Non-Orienté** Si G est orienté, les éléments de A sont des arcs. Si G est non orienté, les éléments de A sont des arêtes et A est défini par une relation symétrique.

Exemple 3 **Représentation Graphique**

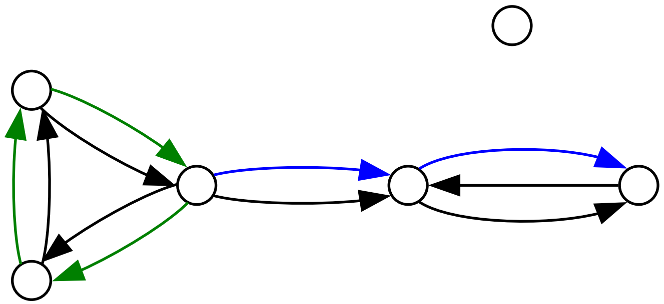


Définition 4 **Voisins d'un sommet** Les voisins d'un sommet $u \in S$ sont les sommets v tels que $(u, v) \in A$

Définition 5 **Chemin** Un chemin de $u \in S$ à $v \in S$ dans un graphe (S, A) est une séquence $((f_0, f_1), (f_1, f_2), ..., (f_{n-1}, f_n))$ où $f_0 = u$ et $f_n = v$.

Remarque 6 **Cycle** Un chemin de u à lui-même est appelé un cycle.

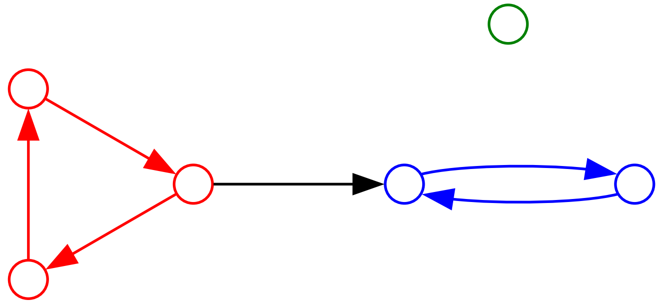
Exemple 7 **Chemin et cycle**



Définition 8 **Sommet accessible** Un sommet $u \in S$ est accessible depuis $v \in S$ si $u = v$ ou s'il existe un chemin de u à v.

Définition 9 **Composantes connexes** La composante fortement connexe de $u \in S$ est l'ensemble des $v \in S$ tels que u est accessible depuis v et v est accessible depuis u. Pour les graphes non orientés, on parle de composante connexe.

Exemples 10 **Composantes connexes**



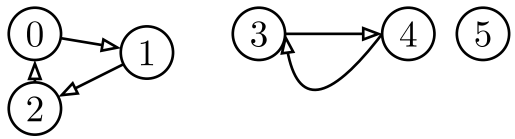
Application 11 **Applications des graphes** Les graphes sont une structure de donnée relationnelle qui permet de modéliser des problèmes dans des domaines variés : géographie, réseau, imagerie, ... Ils sont aussi la base d'outils formels tels que : les systèmes de transitions, automates finis et machine de Turing.

B. Représentation d'un graphe [BAR 5.4.4] [COR2 22.1] [TOR 8.2]

Définition 12 **Liste - Ensemble d'adjacence** On stocke pour chaque sommet l'ensemble de ses voisins.

Définition 13 **Matrice d'adjacence** On stocke une matrice de booléens $|S| \times |S|$ On a `true` en colonne i, ligne j ssi $(i, j) \in A$. Cette matrice est symétrique pour les graphes non orientés.

Exemple 14 **Représentation d'un graphe**



Remarque 15 **Représentation implicite** Certains graphes sont représentés de façon implicite par une fonction qui calcule les voisins d'un sommet (ex: graphes des configurations aux échecs.)

II. Parcours de Graphe

A. Parcours itératif [COR2 22.2] [BAR 9.1.1] [TOR 8.3.2]

Algorithme 16 **Parcours en largeur** L'algorithme suivant parcours un graphe G depuis un sommet u donné en entrée, en commençant par les sommets les plus proches :

```
Parcours_it(G, U)
  L = file qui contient juste u
  traité = [blanc \forall s in S]
  tant que L non vide :
    defiler s de L
    traite[s] = noir
    pour voisin v de s:
      si traité[v] = blanc:
        enfiler v dans L
        traité[v] = gris
```

À la fin de l'algorithme, tous les sommets traités sont en noir.

Proposition 17 Complexité en $O(|S| + |A|)$.

Application 18 Permet de calculer tous les sommets accessibles depuis u.

B. Parcours profondeurs récursifs [COR2 22.3 22.4 20.5] [BAR 9.1.2, 9.2] [TOR 9.3.1]

Algorithme 19

```
PARCOURS_REC(G)
  parent = [nil for s in S]
  pre = [-1 for s in S]
  post = [-1 for s in S]
  clock = 1
  tant qu'il existe u non traité
    EXPLORER(u)
```

```
EXPLORER(u)
  pre[u] = clock; ++clock
  pour v voisin de u tq pre[v]
= -1:
    parent[v] = u
    EXPLORER(v)
  post(u) = clock; clock++
```

Proposition 20 Complexité en $O(|S| + |A|)$.

Définition 21 **Forêt de parcours** La table `parent` définit des arborescences, qu'on nomme forêt de parcours.

Théorème 22 **Chemin Blanc** $u \in S$ est un descendant de $v \in S$ dans la forêt de parcours ssi quand on apelle `EXPLORER(v)` il existe un chemin de v à u qui ne passe que par des sommets non traités.

Définition 23 **Classification des arcs** Il existe une classification des arcs du graphe selon la forêt de parcours. Par exemple, un arc (`parent[u], u`) pour $u \in S$ est appelé arc de liaison. Un arc de u vers un de ses ancêtres est appelé arc arrière.

Application 24 **Détection de cycles** Un graphe contient un cycle si et seulement s'il a un arc arrière.

Application 25 **Tri topologique** Ordonner les sommets par `f` décroissant fournit un ordre topologique $\underset{t}{\leq}$ pour un graphe sans cycle.

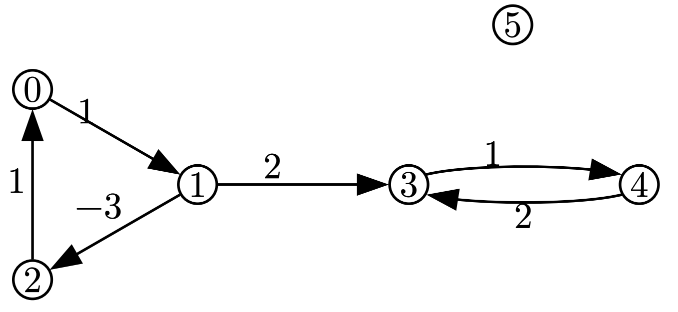
Application 26 **L'algorithme de Kosaraju** calcule les composantes fortement connexes d'un graphe.

III. Graphes pondérés

A. Définitions [BAR 5.4.1] [TOR 8.1.3]

Définition 27 **Graphe pondéré** Un graphe pondéré est un triplet $G_w = (S, A, w)$ où (S, A) est un graphe et $w : A \to \mathbb{R}$ une pondération.

Exemple 28 **Graphes pondérés**



Définition 29 **Poids d'un chemin** Le poids d'un chemin dans un graphe pondéré est la somme des poids des arêtes/arcs qui le composent.

Définition 30 Distance entre deux sommets La distance entre deux sommets u et v d'un graphe pondéré, est, si elle existe, le poids minimal des chemins qui relient u à v. On pose souvent $+\inf$ comme distance de u à v si elle n'est pas accessible depuis u.

Exemple 31 Calculs de distance

B. Algorithmes de calcul de distances [BAR 9.3] [COR2 24.3 25.2] [TOR 8.3.3]

Algorithme 32 Dijkstra Mono-source (d'un sommet vers tous les autres) dans un graphe sans poids négatifs. Paradigme glouton.

```
Dijkstra((S, A, w), s_0)
  d = [inf pour s in S]
  d[s_0] = 0
  parent = [nil pour s in S]
  F = S
  tant que F non vide:
    extraire p = argmin d[s] de F
    pour chaque voisin u de p:
      si d[u] > d[p] + w(p, u):
        d[u] = d[p] + w(p, u)
        parent[u] = p
```

Complexité avec la bonne file de priorité $O((|S| + |A|) \log |S|)$.

Remarque 33 En plus du calcul des distances, on peut retrouver le plus court chemin à l'aide du tableau `parent`.

Algorithme 34 A* On peut biaiser l'ordre de traitement des sommets à l'aide d'une heuristique. Si l'heuristique est une estimation de la distance restante, cet algorithme s'appelle alors A*. Si l'heuristique minore la distance réelle, alors A* est correct.

Algorithme 35 Floyd-Warshall Omni-source (de chaque sommet vers tous les autres). Le graphe ne peut pas posséder de cycle de poids négatifs. Paradigme de programmation dynamique. Calcul de matrices successives $O(|S|^3)$. $M^{n+1}_{i,j} = \min\left(M^n_{i,j}, M^n_{i,n+1} + M_{n+1,j}\right)$

Application 36 Routage réseau automatique Le protocole OSFP établit des tables de routages pour un réseau en deux étapes :

- les routeurs se mettent d'accord sur la matrice d'adjacence du réseau de façon distribuée
- chacun éxécute un algorithme de plus court chemin

C. Problème du voyageur de commerce [COR2 35.2]

Définition 37 Cycle Hamiltonien Un cycle dans un graphe est dit hamiltonien s'il passe exactement une fois par chaque sommet.

Définition 38 TSP Problème du Voyageur de Commerce

- Entrée : Un graphe (S, A, w) complet, pondéré par $w : A \to \mathbb{N}$, un entier $k \in \mathbb{N}$
- Sortie : oui ssi il existe un cycle hamiltonien de poids inférieur ou égal à k dans (S, A, w).

DEV

Propriété 39 Le problème du voyageur de commerce est NP-Complet.

Propriété 40 Il n'existe pas d'approximation à facteur constant en temps polynomial du problème du voyageur de commerce. Si l'on se limite aux graphes pour lesquels la pondération vérifie l'inégalité triangulaire, alors il existe une 2-approximation.

D. Arbre couvrant minimal [COR2 2.3] [BAR 9.4] [TOR 8.3.5]

Définition 41 Arbre couvrant Soit $G = (S, A)$ connexe, non orienté. Un arbre couvrant de G est un ensemble d'arête $T \subseteq A$ tq (S, T) connexe acyclique.

DEV

Algorithme 42 Kruskal

```
Kruskal(S, A, w):
  T = emptyset
  trier A par poids w croissants
  pour (u, v) in A:
    si u et v ne sont pas dans la même composante connexe dans
(S, T):
      T = T union {(u, v)}
  renvoyer T
```

Application 43 Égaliser des tensions en conception de circuits (par exemple relier toutes les portes logiques à la terre).

Accessiblité et chemins dans un graphe. Applications.

I. Définitions, généralités

A. Structure de Graphe [BAR 5.4.1 5.4.2] [TOR 8.1]

1 Def Graphe
2 Rem Orienté vs Non-Orienté
3 Ex Représentation Graphique
4 Def Voisins d'un sommet
5 Def Chemin
6 Rem Cycle
7 Ex Chemin et cycle
8 Def Sommet accessible
9 Def Composantes connexes
10 Exemples Composantes connexes
11 App Applications des graphes

B. Représentation d'un graphe [BAR 5.4.4] [COR2 22.1] [TOR 8.2]

12 Def Liste - Ensemble d'adjacence
13 Def Matrice d'adjacence
14 Ex Représentation d'un graphe
15 Rem Représentation implicite

II. Parcours de Graphe

A. Parcours itératif [COR2 22.2] [BAR 9.1.1] [TOR 8.3.2]

16 Algo Parcours en largeur
17 Prop
18 App

B. Parcours profondeurs récursifs [COR2 22.3 22.4 20.5] [BAR 9.1.2, 9.2] [TOR 9.3.1]

19 Algo
20 Prop
21 Def Forêt de parcours
22 Thm Chemin Blanc
23 Def Classification des arcs
24 App Détection de cycles
25 App Tri topologique
26 App L'algorithme de Kosaraju

III. Graphes pondérés

A. Définitions [BAR 5.4.1] [TOR 8.1.3]

27 Def Graphe pondéré
28 Ex Graphes pondérés
29 Def Poids d'un chemin
30 Def Distance entre deux sommets
31 Ex Calculs de distance

B. Algorithmes de calcul de distances [BAR 9.3] [COR2 24.3 25.2] [TOR 8.3.3]

32 Algo Dijkstra
33 Rem
34 Algo A*
35 Algo Floyd-Warshall
36 App Routage réseau automatique

C. Problème du voyageur de commerce [COR2 35.2]

37 Def Cycle Hamiltonien
38 Def TSP
39 Prop
40 Prop

D. Arbre couvrant minimal [COR2 2.3] [BAR 9.4] [TOR 8.3.5]

41 Def Arbre couvrant
42 Algo Kruskal
43 App

Commentaires

- La notion de cycle est mal définie, en effet, avec cette définition, tout graphe possédant un cycle en posséderait une infinité.
- La notion de couplage dans un graphe, au programme de MP et de MPI n'apparaît pas dans ce plan. Il est alors indispensable de justifier ce choix lors de la défense du plan.

A. Application : Couplage maximum dans un graphe biparti [TOR 8.3.6] [BAR 9.5]

Définition 44 **couplage, couplage maximum** Soit $G = (S, A)$ non orienté. Un couplage $M \subseteq A$ sur G est un ensemble d'arêtes non adjacentes. $(a, b) \neq (c, d) \in M, (a \neq c) \wedge (a \neq d) \wedge (b \neq c) \wedge (b \neq d)$ Un couplage sur G est maximum s'il a le plus grand nombre d'arêtes parmi les couplages sur G.

Exemple 45 **Exemples de couplages dont un maximum**

Définition 46 **Chemin améliorant** Soit M un couplage sur $G = (S, A)$. Un chemin améliorant est un chemin qui utilise de façon alternée une arête de $A \setminus M$ et une arête de M et qui commence et termine sur des sommets de $A \setminus M$.

Théorème 47 M est un couplage maximum ssi il n'admet pas de chemin améliorant.

Définition 48 **Graphe Biparti** Un graphe $G = (S, A)$ est biparti s'il existe S' et S'' tels que $S = S' \cup S''$, S' $S''\emptyset$ et $\forall u, v \in A, (u \in S' \wedge v \in S'') \vee (u \in S'' \wedge v \in S')$.

Algorithme 49 **Chemin améliorant dans un graphe biparti**

```
COUPLAGE_MAX(G biparti)
  M = emptyset
  tant qu'il existe C chemin améliorant M
    M = M Delta C
  renvoyer M
```

Bibliographie

[BAR] V. Barra, *Informatique MPI/MP2I.*

[TOR] T. Balabonski & S. Conchon & J. Filliâtre & K. Nguyen & L. Sartre, *MP2I MPI, Informatique Cours et exercices corrigés.*

[COR2] T. H. Cormen, *Introduction à l'algorithmique (2nd édition).*