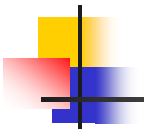


Dans cette partie...

3- Programmation

- Notion de programme
- Quelques instructions du langage Python

Qu'est-ce qu'un programme?

- C'est une suite d'instructions écrites dans un langage de programmation compréhensible par l'ordinateur.
- Cela permet à l'ordinateur d'appliquer un algorithme.
- Exemple : afficher les diviseurs de *n*

```
fonction affiche_diviseurs(n)
  si n > 0 alors
    pour tout entier i entre 1 et n faire
      si n est divisible par i alors
        afficher i
      finsi
    finpour
  finsi
```

```
def affiche_diviseurs(n):
    if n > 0:
        for i in range(1, n+1):
            if n % i == 0:
                print i
```

Remarque : `n % i` donne le reste de la division de *n* par *i*

L'affectation : ranger une valeur dans une variable

- Exemples :

```
i = 1
x = 2*i+1
i = i+1
x = x+i
```

- L'ordinateur effectue les instructions dans l'ordre. L'ordre des instructions est donc très important.
- Une variable désigne un emplacement dans lequel on peut mémoriser une valeur. Une variable a un nom.
- En python, le symbole = n'a pas la même signification qu'en mathématique. Il signifie calculer la valeur à droite du symbole =, et la ranger dans la variable dont le nom se trouve à gauche.

Instruction conditionnelle `if…: else :…`

Instruction conditionnelle si… alors… sinon…

```
i=10
x=6
if i > x :
  print "test VRAI"
  print i, "est supérieur à", x
else :
  print "test FAUX"
  print i, "n'est pas supérieur à", x
```

Attention à l'indentation !

Répétition : **while… :**

Répétition : tant que … faire…

```
i = 5
while i > 0 :
  print i
  i = i - 1
```

Ou :

```
i = 5
while i > 0 :
  i = i - 1
  print i
```

Attention à l'ordre des instructions !

Répétition : **for … in … range(….):**

Répétition : pour… parcourant la liste ….

```
# affiche les entiers de 0 à 9
for i in range(10) :
  print i
```

`range(a,b,c):` « donne la liste » des entiers appartenant à l'intervalle `[a,b[` par pas de `c` .

Exemple : `range(1,9,1) =>[1,2,3,4,5,6,7,8]`
`range(1, 9, 2) => [1,3,5,7]`

Remarque: `range(a,b)` ⇔ `range(a,b,1)`
`range(b)` ⇔ `range(0,b,1)`

Notion de fonction

En maths : soit la fonction $f : x \rightarrow 2x^2 + 1$

décrit la façon de calculer `f(x)` à partir de la donnée `x`

En Python :

```
def f(x) :
  return 2 * x * x + 1
# exemples d'appel de la fonction
y = 2 * f(2)
print f(5)
for k in range(10) :
  print f(k)
```

- Une fonction Python peut utiliser d'autres fonctions.
- Un programme est en général composé de plusieurs fonctions.

Quelques pièges à éviter !

Affichage ou sortie (`print` ou `return`)

Position du `return` (dans la boucle ou après la boucle)

```
def f(x):
  for i in range(x):
    print i

def g(x):
  for i in range(x):
    return i

def h(x):
  for i in range(x):
  return i
```

Quel sera le résultat de : `f, g` et `h` pour `x=3` ?

Il y a d'autres instructions dans le langage... **http://www.python.org/**

Dans cette partie...

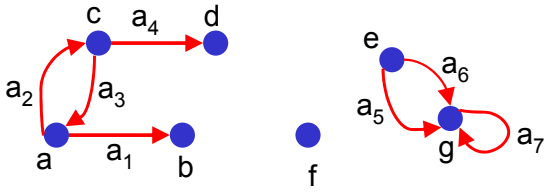
4- Introduction aux graphes

- 1ère sorte de graphes : les graphes orientés
- Quelques exemples de graphes orientés
- 2ème sorte de graphes : les graphes non orientés
- Quelques exemples de graphes non orientés
- Les graphes dans des problèmes courants

1ère sorte de graphes : les graphes orientés

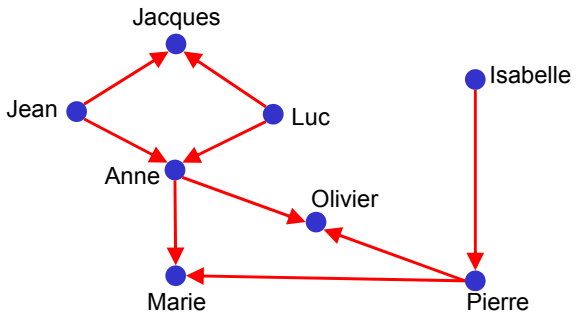
Un graphe orienté représente une relation non symétrique

- Il est défini par :
  - un ensemble fini de *sommets* représenté par un point ●
  - un ensemble fini d'*arcs* orientés
  - à chaque arc est associé est un *couple de sommets*.
  - un arc *a* associé à (s,t) est représenté par s ⟶ t
  - Exemple (graphe simple)
    - Sommets : {a,b,c,d,e,f,g}. Arcs : {a_1, a_2, a_3, a_4, a_5, a_6, a_7}.



Exemple de graphe orienté : parents

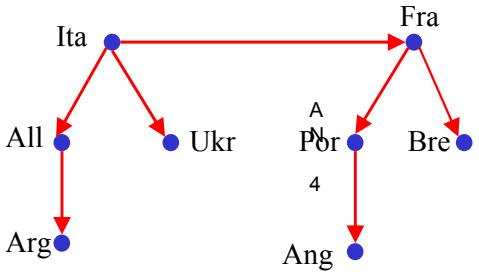
- Ensemble : toutes les personnes assistant à un repas de Noël.
- Relation : l'ensemble des couples de personnes (p1, p2) tels que p1 a pour parent p2.
- Représentation graphique (relation non symétrique, graphe orienté) :



- Qui sont les parents de Anne ?
- Même question pour Jacques.
- Ajouter une fille à Pierre

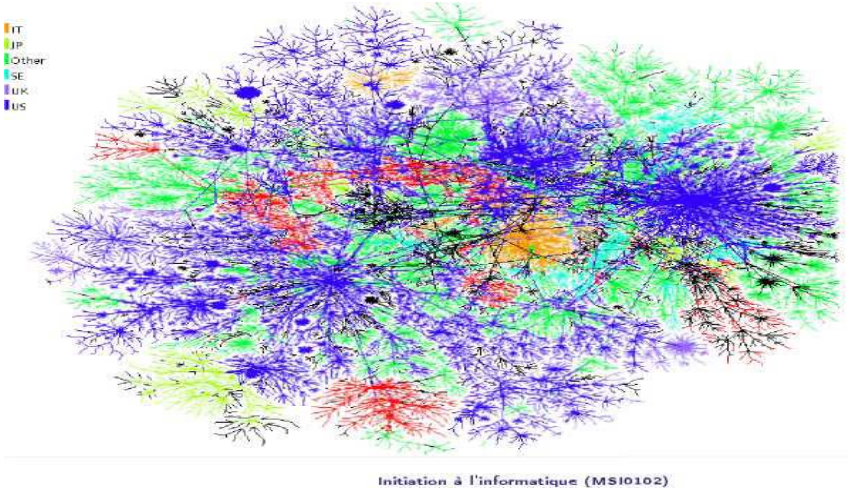
Exemple de graphe orienté : matchs de foot

- Ensemble : équipes de foot.
- Relation : l'ensemble des couples d'équipes *(eq1,eq2)* telles que *eq1* a battu *eq2* à partir des 1/4 finale de la coupe du monde 2006.
- Représentation graphique (pas de match nul, relation non symétrique, graphe orienté).



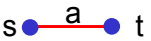
Changement d'échelle : Internet

- Dans la réalité les graphes sont de grande taille.
- On conserve une vue locale.



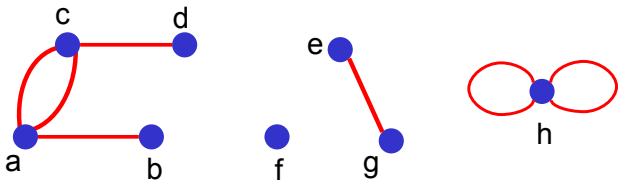
Initiation à l'informatique (MSI0102)

2ème sorte de graphes : graphes non orientés

- Un graphe non orienté représente une relation symétrique
- Dans un graphe non orienté, une arête est associée soit à une paire de sommets {s, t}, soit à un singleton {s}.
- On représente un graphe non orienté par un dessin, dans lequel les arêtes n'ont pas de flèche.
  - Un sommet est dessiné comme un point
  - Une arête a associée à {s, t} est représentée par 

  - Une arête a associée à {s} est représentée par 

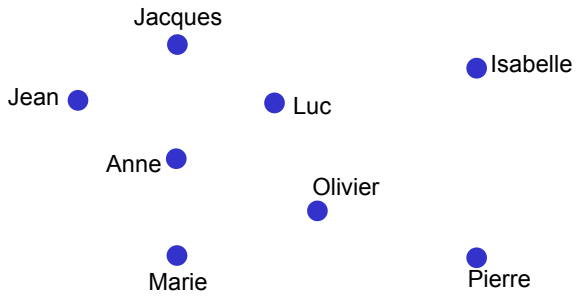
- Exemples



Exemple de graphe non orienté : cousins

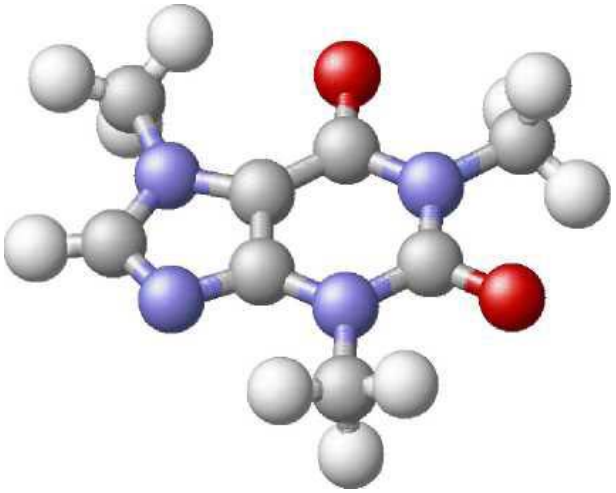
- Ensemble : toutes les personnes assistant à un repas de Noël.
- Relation : l'ensemble des couples de personnes (p1,p2) tels que p1 est un cousin de p2 (relation symétrique).
- Représentation graphique (relation symétrique, graphe non orienté)

En utilisant le graphe des parents compléter le graphe des cousins.



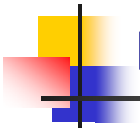
Exemple de graphe non orienté : molécule

- Ensemble : les atomes de la molécule de caféine.
- Relation : l'ensemble des couples d'atomes partageant au moins un électron (liaison covalente, relation symétrique)



Modélisation de problèmes par des graphes

- Les graphes sont des objets permettant de représenter plusieurs situations de la vie courante.
  - Des problèmes de la vie courante se traduisent en questions à résoudre sur les graphes.
  - Le développement d'algorithmes sur les graphes permet donc de répondre à des problèmes concrets et pratiques.
  - Cela permet aussi de reléguer la résolution de ces problèmes à des systèmes informatisés.



Exemple : Organisation d'examens

- Des étudiants passent des examens de Maths (M), Informatique (I), Chimie (C), Physique (P), Anglais (A) et Sport (S) de 2h.
- Des étudiants passent M, I, C, d'autres C, P, A et d'autres A, S, P

Question : Quel est le temps le plus court pour organiser les examens?

Exemple : Organisation d'examens

- On fait un graphe avec comme sommets M,I,C,P,S,A.
  - Si un étudiant doit passer deux des examens, les sommets correspondants sont incompatibles: les épreuves ne peuvent avoir lieu en même temps.
  - On forme le graphe en reliant 2 sommets s'ils sont incompatibles.

  1. Dessiner le graphe.

- On cherche à attribuer à chaque examen un créneau horaire.

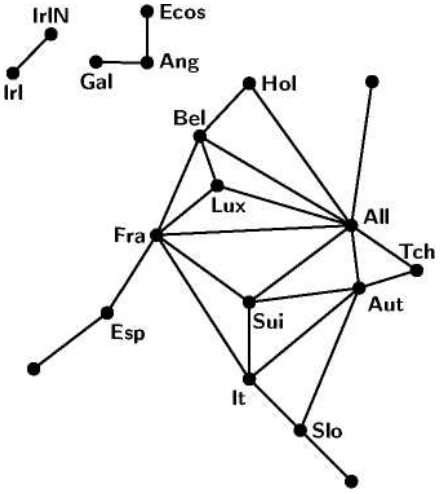
  - On colorie des sommets adjacents avec des couleurs différentes.
  - On souhaite utiliser un nombre minimum de couleurs (de créneaux).

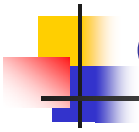
  2.1 Colorier le graphe solution
  2.2 Quel est le temps le plus court pour organiser les examens ?

Coloration de carte géographique

- 1852 Guthrie colorie la carte des cantons anglais avec 4 couleurs sans que 2 cantons adjacents n'aient la même couleur.
- 4 couleurs suffisent-elles pour colorier n'importe quelle carte?
- 1976 : Oui (Appel & Haken). Résultat obtenu en partie par ordinateur.
  - Carte géograhique ~> graphe.
    Les sommets représentent les pays, et 2 sommets sont reliés par une arête si les pays correspondants partagent une frontière.
  - But : montrer qu'on peut colorier le graphe avec 4 couleurs, sans que 2 sommets adjacents n'aient la même couleur.
- On peut extraire de la preuve un algorithme permettant de trouver la coloration.

Coloration de carte géographique



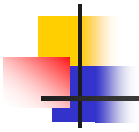


Coloration de graphes

- Partant d'un graphe quelconque (ne représentant pas une carte), on peut avoir besoin de plus de 4 couleurs.
- Tester si on peut colorier avec seulement 3 couleurs est difficile.
- Pour ce dernier problème, trouver un algorithme avec une complexité polynomiale, ou
- prouver qu'un tel algorithme n'existe pas est équivalent à l'un des problèmes ouverts les plus difficiles, sélectionné par le Clay Mathematical Institute (1 000 000$).

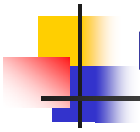
Dominos

- On dispose d'un jeu de dominos, le plus grand numéro étant n.
- Question : Peut-on placer tous les dominos (en suivant les règles)?
- Graphe. Sommets : {0, • • • , n}. Arêtes : toutes les paires $\{s, t\}$.
- L'arête $\{s, t\}$ représente le domino dont les numéros sont s et t.
  - Exemple : l'arête {1,2} représente 
  - Dessiner le graphe pour n =4
- On peut placer tous les dominos s'il existe un chemin dans le graphe passant une seule fois par chaque arête : chemin Eulérien.
- Un résultat vu plus loin permet de résoudre le problème



Problèmes de parcours

- Les graphes permettent de modéliser des voies de communication :
  - les sommets représentent des villes,
  - les arêtes représentent des routes.
- On peut associer à chaque arête d'un tel graphe (orienté s'il y a des sens uniques) un nombre représentant la distance entre les deux sommets reliés.



Problèmes de parcours

- Deux questions naturelles
  - Peut-on trouver un algorithme efficace pour calculer un trajet le plus court possible entre deux villes données?
  - Peut-on trouver un algorithme efficace pour calculer un trajet le plus court possible reliant toutes les villes?
- Réponses
  - Oui pour le 1er problème, (il y a des algorithmes efficaces).
  - ? ? pour le 2ème, sélectionné par le Clay Math Institute (1 M $).

Dans cette partie...

5- Graphes : définition

Une définition de graphe

Un *graphe* est un triplet (*S, A, inc*), où

- *S* est un ensemble d'objets appelés les *sommets* du graphe,
- *A* est un ensemble d'objets appelés les *arêtes* du graphe,
- *inc* est une fonction $inc : S \rightarrow P(A)$ telle que

$$\forall a \in A \ |\{s \in S \ , \ a \in inc(s)\}| \in \{1, 2\}$$

inc(s) = ensemble des arêtes incidentes au sommet *s*.
(incidentes = qui "touchent").

Interprétation de la définition

Exemple : pour décrire les vols d'une compagnie aérienne,

- on décrit chaque ville (= sommet) desservie,
- on décrit les lignes (= arêtes) vers ou depuis chaque ville.

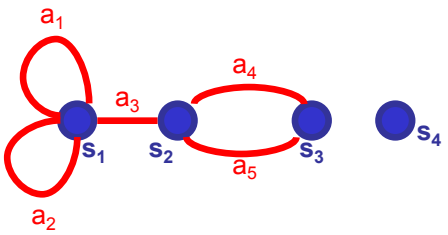
Par exemple : Ville Bordeaux, liaison AF6257 (vers Paris).

La fonction *inc* est appliquée à un sommet et renvoie l'ensemble des arêtes qui "touchent" le sommet.

La propriété de $inc : S \rightarrow P(A)$ pour forcer une arête à avoir une ou deux extrémités est :

$$\forall a \in A \; |\{s \in S \; , \; a \in inc(s)\}| \in \{1, 2\}$$

Exemple



$S = \{s_1, s_2, s_3, s_4\}$

$A = \{a_1, a_2, a_3, a_4, a_5\}$

$inc(s_1) = \{a_1, a_2, a_3\}$
$inc(s_2) = \{a_3, a_4, a_5\}$
$inc(s_3) = \{a_4, a_5\}$
$inc(s_4) = \varnothing$

$a_3 \in inc(s_1)$ et $a_3 \in inc(s_2)$ donc s_1 et s_2 sont les extrémités de l'arête a_3

Exercices

Dessiner le graphe défini par :

$S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$

$A = \{a_0, a_1, a_2, a_3, a_4, a_5, a_6\}$

$inc(s_0) = \{a_0, a_1, a_5\}$

$inc(s_1) = \{a_0, a_1, a_2\}$

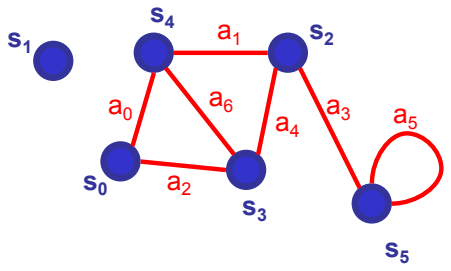
$inc(s_2) = \varnothing$

$inc(s_3) = \{a_3, a_6\}$

$inc(s_4) = \{a_2, a_3, a_4\}$

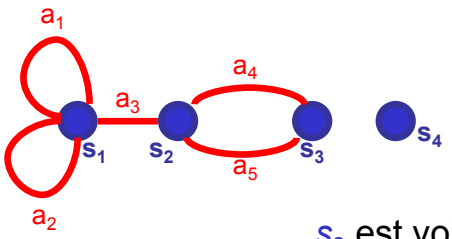
$inc(s_5) = \{a_4, a_5\}$

Déterminer les ensembles *A* et *S* ainsi que la fonction *inc* pour le graphe ci-dessous :



Voisins d'un sommet

Deux sommets sont voisins si ce sont les extrémités d'une même arête.



s_3 est voisin de s_2

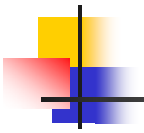
L'ensemble des voisins de s_1 est $\{s_1, s_2\}$

s_4 n'a pas de voisins (il est isolé)

Fonctions de manipulation de graphes

Plusieurs fonctions prédéfinies permettent de manipuler des graphes :

nom_graphe(G) : retourne le nom du graphe G.
nb_sommets(G) : retourne le nombre de sommets du graphe G.
liste_sommets(G) : retourne la liste des sommets du graphe G.
degre(s) : retourne le degré du sommet s.
liste_voisins(s) : retourne la liste des voisins du sommet s.
dessiner(G) : dessine le graphe G.

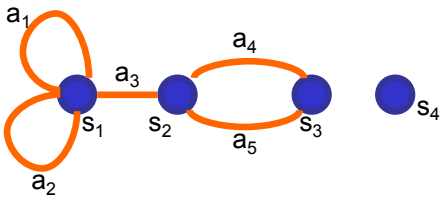


Dans cette partie...

6- Degré d'un sommet

Degré d'un sommet

- Le degré d'un sommet s, noté $d(s)$, est le nombre de brins d'arêtes ayant s comme extrémité.
- Une boucle compte deux fois.



Ici $d(s_1) = 5$, $d(s_2) = 3$, $d(s_3) = 2$, $d(s_4) = 0$.
Attention ! Le degré de s n'est pas le nombre de voisins de s.

Un théorème important

- Si G est un graphe (non orienté), on note
  - $S(G)$ l'ensemble de ses sommets.
  - $A(G)$ l'ensemble de ses arêtes.
- Théorème.
  Pour un graphe G ayant au moins un sommet, la somme des degrés des sommets est égale à deux fois le nombre d'arêtes. D'où la formule :
  $$\sum_{s \in S(G)} d(s) = 2\left|A(G)\right|$$
- Conséquence.
  Dans n'importe quel graphe, le nombre de sommets qui ont un degré impair est pair.

Preuves

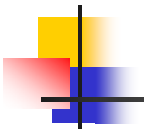
Preuve directe

Une idée de preuve : faire la somme des degrés compte chaque arête 2 fois, car chaque arête a 2 extrémités qui contribuent chacune à une unité de cette somme.

Preuve par induction (récurrence)

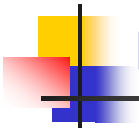
Vérifier que $\Sigma_{s \in S(G)} d(s) = 2|A(G)|$ pour un graphe sans arête, puis prouver que :

si $\Sigma_{s \in S(G)} d(s) = 2|A(G)|$ est vrai pour tout graphe ayant au plus k arêtes, alors c'est aussi vrai pour tout graphe ayant $k + 1$ arêtes.



Dans cette partie...

7- Chaînes dans un graphe



La notion de chaîne

- Il est souvent nécessaire de savoir si l'on peut aller d'un sommet à un autre en suivant des arêtes.
- Exemple d'utilité : Est-ce possible de prendre le train pour aller
  - De Bordeaux à Rome?
  - De Bordeaux à Oslo?
  - De Bordeaux à Reykjavik?
- La notion de chaîne exprime cette idée.

Définition de chaîne

- Une *chaîne* dans un graphe est une suite

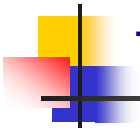
$$C = s_1, a_1, s_2, \ldots, s_k, a_k, s_{k+1}$$

de $k + 1$ sommets et k arêtes en alternance, telle que les extrémités de a_i sont s_i et s_{i+1}.

- On dit alors que C est une *chaîne entre* s_1 et s_{k+1}
- Remarques
  - Si $k = 0$, on obtient une chaîne sans arête $C = s_1$.
  - Pour définir une chaîne, se donner seulement la suite des arêtes, ou seulement la suite des sommets ne suffit pas (pourquoi ?).

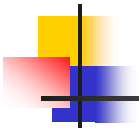
Chaîne simple

- Une chaîne $C = s_1, a_1, s_2, \ldots, s_k, a_k, s_{k+1}$ est *simple* si et seulement si:
  - Quels que soient i et j, alors ($i < j$ et $s_i = s_j$) *implique* ($i = 1$ et $j = k+1$) [où i et j varient entre 1 et $k+1$].
- Autrement dit "un sommet figure au plus une fois dans la chaîne" [sauf pour le sommet de début et de fin. S'ils sont les mêmes, il s'agit d'un *cycle*].



Théorème

- Dans un graphe *G,* s'il existe une chaîne entre deux sommets *s* et *t,* alors il existe une chaîne *simple* entre *s* et *t.*
- La preuve est *constructive.* On prend une chaîne non simple et on en supprime les cycles pour *construire* une chaîne simple.



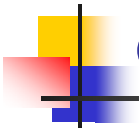
Existence d'une chaîne

- Vérifier s'il existe une chaîne entre un sommet s et un sommet *t* n'est pas forcément simple.
- Pour y arriver, nous allons utiliser une technique pour marquer et démarquer les sommets.
  - Les marques laissées sur les sommets agissent en quelque sorte comme une mémoire d'un parcours effectué dans le graphe.

Existence d'une chaîne entre *s* et *t*

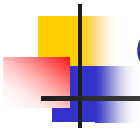
Algorithme :

- 1. démarquer tous les sommets
- 2. marquer s
- 3. tant que *t* n'est pas marqué,
  - 3.1 chercher une arête dont un sommet extrémité est marqué et l'autre ne l'est pas
  - 3.2 si une telle arête n'existe pas, renvoyer la valeur "**faux**"
  - 3.3 sinon marquer l'extrémité non encore marquée
- 4. renvoyer la valeur "vrai"



Quelques remarques sur l'algorithme

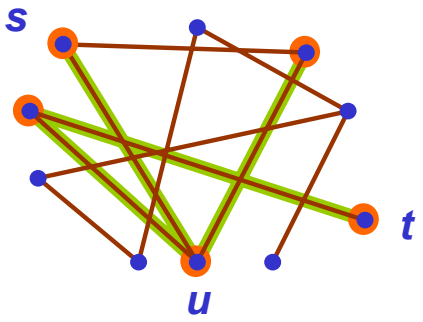
- La première phase consiste à se mettre dans un état de départ adéquat afin d'assurer le bon fonctionnement de l'algorithme.
- L'action « renvoyer » correspond à la réponse attendu et donné par l'algorithme, l'action « renvoyer » implique que l'algorithme est arrivé à sa conclusion et s'arête.
- On suppose ici que la recherche d'une arête qui satisfait la condition se fait en suivant une procédure connue.
- On suppose aussi que l'on a une façon facile de vérifier si un sommet est marqué et de le marquer ou de le démarquer le cas échéant.



Quelques remarques sur l'algorithme

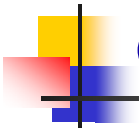
- Il reste encore à prouver que l'algorithme fait bien le travail que l'on attend de lui. C'est-à-dire:
  - Que l'on obtient éventuellement et à chaque fois une réponse de sa part,
  - Et que cette réponse est bien la bonne (!).

Existence d'une chaîne entre s et *t*



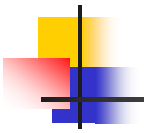
L'algorithme visite d'abord le voisin *u* de *s* qui se trouve au bas du graphe puis remonte vers le haut à droite. Il sélectionne ensuite voisin de *u* à gauche et en bas de *s* avant de « trouver » *t*.

.



Complexité de l'algorithme

- Étape 1 :
- Étape 2 :
- Étape 3.1 :
- Étapes 3.2 et 3.3 :

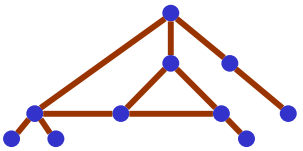


Dans cette partie...

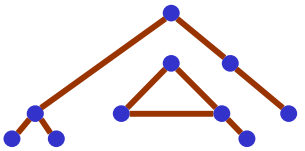
8 - Connexité

Connexité

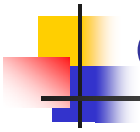
- La connexité exprime la possibilité d'aller de n'importe quel sommet du graphe à n'importe quel autre sommet du graphe.
- Informellement, un graphe est connexe s'il est en un seul morceau.



Connexe



Non connexe



Connexité

- La connexité exprime la possibilité d'aller de n'importe quel sommet du graphe à n'importe quel autre sommet du graphe.
  - Formellement, un graphe G est connexe si et seulement si

    $\forall s,t \in S(G)$, il existe une chaîne entre s et t.
- On met au point des algorithmes pour déterminer si un graphe est connexe.

Déterminer si un graphe est connexe

- Approche simple: la première idée est d'appliquer la définition.
  - Donc, pour déterminer si un graphe est connexe, on vérifie si pour chaque couple *(s, t)* de sommets, il existe une chaîne entre *s* et *t*.
  - Le nombre de couple *(s,t)* est $|S|^2$.
  - Au total, cette méthode a une complexité de l'ordre de

$$|S|^2 \cdot |S|(|A| + |S|) = |S|^3(|A| + |S|)$$

- Nous allons montrer qu'on peut résoudre le problème en remplaçant le facteur $|S|^3$ par $|S|^2$

Différence entre $O(n^3)$, $O(n^2)$ et $O(n)$

- Pour apprécier la différence entre $O(n^3)$ et $O(n)$, imaginons un ordinateur capable d'exécuter une instruction élémentaire en 10ns = 10^{-8} sec (ce qui est raisonnable).
  - Imaginons aussi un graphe de 1 000 000 sommets (un réseau routier par exemple).
- Exécuter n instructions élémentaires nécessite 10ms, soit 0,01sec.
- Exécuter n^2 instructions élémentaires nécessite 10^4 sec soit ~ 2h45.
- Exécuter n^3 instructions élémentaires nécessite 10^{10} sec soit ~ 321 ans.

Déterminer si un graphe est connexe

Theorème. Soit *G* un graphe. On considère les propriétés suivantes :

A. *G* est connexe.

B. Pour tout sommet *s* de *G,* il existe une chaîne entre *s* et chacun des sommets de *G*.

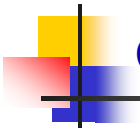
C. Il existe un sommet *s* de *G* tel qu'il existe une chaîne entre *s* et chacun des sommets de *G*.

Les propriétés *A, B* et C sont équivalentes, c'est-à-dire :

$$A \Leftrightarrow B \Leftrightarrow C$$

Preuve (parties *A => B et B => C)*

- $A \Rightarrow B$. On suppose que A est vraie, c'est-à-dire que G est connexe.
- $B \Rightarrow C$ (évident ?)
- Reste à montrer que $C \Rightarrow A$



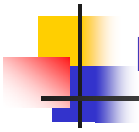
Conséquence: un algorithme plus efficace

- Pour déterminer si un graphe est connexe, on peut donc :
  - Choisir un sommet *s* arbitraire dans le graphe.
  - Vérifier si pour chaque sommet *t* du graphe, il existe une chaîne entre *s* et *t*
- Temps de calcul
  1. Il faut donc faire $|S|$ vérifications.
  2. Le coût de chaque vérification est $|S| \cdot (|A| + |S|)$
  3. Au total, la complexité est donc $|S|^2 \cdot (|A| + |S|)$

Encore une amélioration

- On observe que l'algorithme parcourt la même chaîne plusieurs fois. Dans le graphe suivant :

  - (On choisit s_0 comme sommet de départ)
  - Il va parcourir la chaîne entre s_0 et s_1,
  - puis entre s_0 et s_2, refaisant alors le parcours entre s_0 et s_1,
  - puis entre s_0 et s_3, refaisant alors le parcours entre s_0 et s_2, etc.



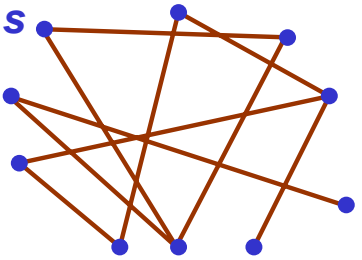
Méthode finale pour tester la connexité

- Voici un algorithme encore plus efficace.
  1. démarquer tous les sommets
  2. marquer un sommet arbitraire s
  3. tant qu'il existe une arête « bicolore » :
     3.1 marquer l'extrémité non encore marquée
  4. si tous les sommets sont marqués, renvoyer "vrai"
  5. sinon renvoyer "faux"

- Complexité $|S| \cdot (|S| + |A|)$ car on applique une seule fois l'algorithme de « parcours » du graphe depuis s.

Méthode finale pour tester la connexité

- L'algorithme ressemble à celui de recherche d'une chaîne
  - Recherche d'arêtes « bicolores »
  - Marquage des sommets



Utiliser l'algorithme précédent pour vérifier si le graphe ci-contre est connexe ?