# Arbre et Arborescence

Sy, Ibrahima

Institut Supérieur Informatique (**ISI**)
Licence 2 RI

April 21, 2021

# Overview

# Définition et concepts de base

## Définition

1. Un graphe non orienté $G = (S, A)$ est dit connexe si quels que soient $u$ et $v$ de $S$ les sommets , il existe une chaîne reliant $u$ à $v$ .
2. Un sous-graphe connexe maximal d'un graphe non orienté quelconque est une composante connexe de ce graphe.
3. Pour un graphe orienté, on parle de connexité si en oubliant l'orientation des arêtes, le graphe est connexe. On parle de forte connexité s'il existe un chemin orienté depuis tout nœud $u$ vers tout nœud $v$.

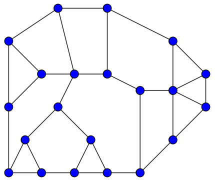


Figure: graphe connexe

# Définition et concepts de base

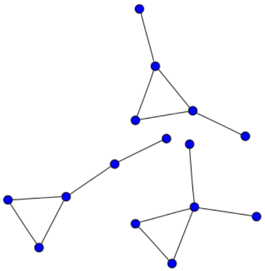


Figure: graphe non connexe

# Définition et concepts de base

## Définition

Un arbre est un graphe connexe et sans cycle. Il est noté $= (S, A)$ ou $S$ est l'ensemble des sommets et $A$ est celui des arcs ou arêtes

## Définition

Un sommet $v$ d'un arbre est $T$ est pendants sil est de degré égal a 1 i.e $d(v) = 1$

## Remarque

Le graphe $T = (\{u, v\}, (u, v))$ est définie par un seul arc. Les deux extrémité sont donc pendantes

# Définition et concepts de base

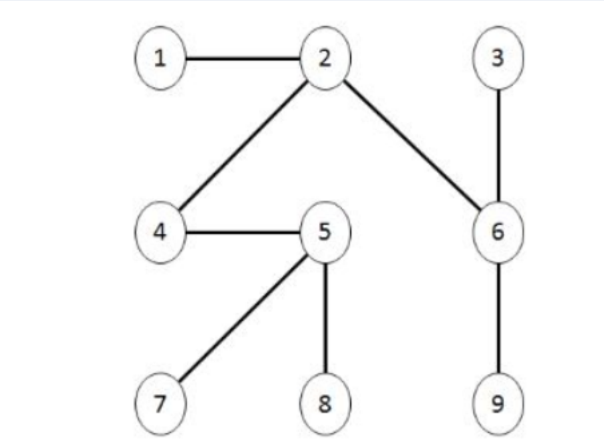



Figure: exemple arbre

# Définition et concepts de base

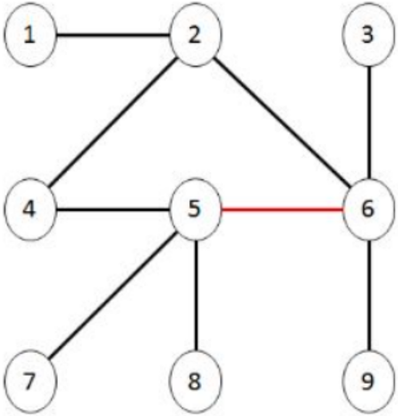



Figure: c'est pas un arbre

# Définition et concepts de base

## Arbre orienté

Un arbre $T = (S, A)$ est oriente si les éléments de $A$ sont orientés. L'exemple ci dessous présente un arbre oriente

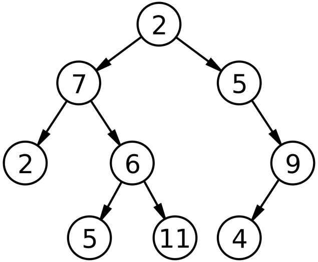



Figure: Arbre orienté

# Définition et concepts de base

Les arbres sont très souvent utilisés a tous les niveaux de la société. C'est le cas des arbres définies par hiérarchie qui permettent de mieux structurer des regroupements associatifs ou des bases de données

**Proposition**

Soit $T$ un arbre avec $n$ sommets $n \geq 2$ alors l'arbre $T$ possède au moins deux extrémités pendantes

**Théorème**

Soit $T = (S, A)$ un arbre avec $n$ sommets $n \geq 1$ alors l'arbre $T$ possède $n - 1$ arcs

# Définition et concepts de base

Le graphe $T = (S, A)$ ci dessous est arbre avec 9 sommets et 8 arcs

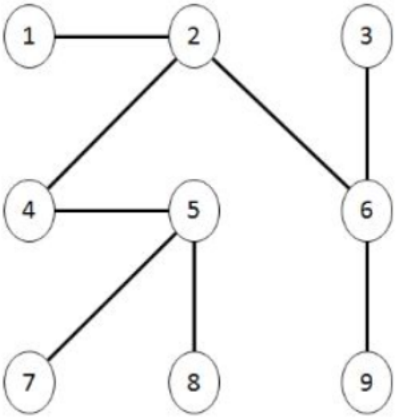



Figure: Caption

# Définition et concepts de base

## Définition

Une Forêt est un graphe dont les composantes connexes sont des arbres. C'est donc un graphe non connexe mais sans circuit. Elle est constitué d'arbre et est noté $F = (S, A)$

## Corollaire

Si $F$ est une forêt à $r$ arbres, alors $F$ possède $n - r$ arcs

# Définition et concepts de base

**Théorème**

Soit $T = (S, A)$ un arbre à $n$ sommets, alors les assertions suivantes sont équivalentes

1. $T$ est un arbre
2. $T$ est sans cycle avec $n - 1$ arcs
3. $T$ est connexe et possède $n - 1$ arcs
4. $T$ est connexe et tout arcs est un pont
5. Deux sommets sont liées par un seul chemin
6. $T$ est sans cycle mais l'addition d'un arcs quelconque crée un cycle

# Arbre de recouvrement

Parmi ces arbres, une catégorie occupe une position plus ou moins honorable. Ce sont les arbres de recouvrements de graphe. Ils permettent dans un réseau de graphe de connecter tous les sommets. En conséquence, ils sont beaucoup utilises dans des problèmes pratique tels que l'installation de câbles de télévision ou de lignes électriques dans un réseau

**Définitions**

Soit le graphe $G = (S, A)$ le sous graphe $T$ est un arbre de recouvrement si $T$ est arbre contenant tous les sommets $S$ de $G$

**Théorème**

Tout graphe $G$ connexe à $n$ sommets possède un arbre de recouvrement

# Arbre de recouvrement

Dans le cas ou le graphe est value, à chaque arc est associé une valeur appelée coût de l'arc. Les arbres de recouvrements les plus intéressant , dans ce cas, sont ceux a coût minimale. Ils sont l'objet de la prochaine section.

# Arbre de recouvrement minimal

Dans cette sous section, nous présentons les différentes étapes d'algorithmes de déterminations de l'arbre de recouvrement minimale d'un graphe $G = (S, A)$. Il s'agit de deux principaux algorithmes d'abord celui de **Kruskal** ensuite celui de **Prim**

# Arbre de recouvrement minimal

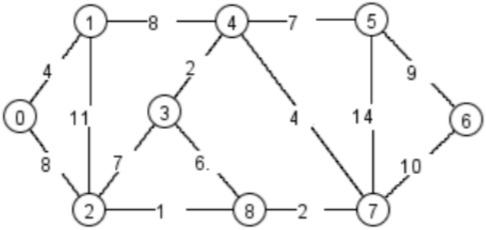
L'algorithme de **Kruskal** détermine à partir d'un graphe $G = (S, A)$ un arbre de recouvrement minimal noté $T$. La méthode détermine initialement un arc à coût minimale. Ensuite, la démarche est d'y attacher successivement les arcs de coût minimale jusqu'à obtenir un arbre de recouvrement.

# Arbre de recouvrement minimal

```
Kruskal(G) :
    A := ∅
    pour chaque sommet v de G :
       créerEnsemble(v)
    trier les arêtes de G par poids croissant
    pour chaque arête (u, v) de G prise par poids croissant :
       si find(u) ≠ find(v) :
          ajouter l'arête (u, v) à l'ensemble A
          union(u, v)
    renvoyer A
```

Les fonctions **makeset**, **find** et **union** sont les trois opérations d'une structure de données Union-Find – qui, respectivement, ajoute une classe singleton à la structure, renvoie un représentant de la classe d'un élément et fusionne deux classes d'équivalence.

# Exemple



# Arbre de recouvrement minimal

L'algorithme consiste à faire croître un arbre depuis un sommet. On commence avec un seul sommet puis à chaque étape, on ajoute une arête de poids minimum ayant exactement une extrémité dans l'arbre en cours de construction. En effet, si ses deux extrémités appartenaient déjà à l'arbre, l'ajout de cette arête créerait un deuxième chemin entre les deux sommets dans l'arbre en cours de construction et le résultat contiendrait un cycle.

# Arbre de recouvrement minimal

```
fonction prim(G, s)
      pour tout sommet t
             cout[t] := +∞
             pred[t] := null
      cout[s] := 0
      F := file de priorité contenant les sommets de G avec cout[.] comme priorité
      tant que F ≠ vide
           t := F.defiler
           pour toute arête t--u avec u appartenant à F
               si cout[u] >= poids de l'arête entre les sommets t et u
                     pred[u] := t
                     cout[u] := poids de l'arête entre les sommets t et u
                     F.notifierDiminution(u)

      retourner pred
```

# Arborescences

**Définition**

une arborescence est un graphe $G = (S, A)$ avec un sommet $v_0$ appelé racine

1. $G$ est arbre
2. $\forall u \in S$ il existe un chemin $v_0$ a $u$

## Définition

Un arbre a racine est dit binaire si les degrés des sommets de $T$ sont de degré au plus égal a 3