Algorithme 1 : ***Parcours en largeur d'abord (Breadth First Search (BFS))***

Données : G : un graphe connexe orienté. i_0 : sommet de G
Variables : La liste OUVERT : sommets en attente d'être traités
La liste FERMÉ : sommets déjà traités
i : sommet courant

```
OUVERT ← (i0) ; FERMÉ ← ()
tant que OUVERT n'est pas vide faire
    soit i le premier élément d'OUVERT
    si i n'est pas dans FERMÉ alors
        mettre les successeurs de i qui ∉ FERMÉ en fin d'OUVERT (en mémorisant
          que i est leur père et en supprimant les répétitions)
        effectuer le traitement pour i
        mettre i dans FERMÉ
    supprimer i d'OUVERT
```

Algorithme 2 : ***Algorithme glouton de coloration***

```
début
    Introduire une première couleur et colorier le premier sommet
    pour chaque sommet s non encore colorié faire
        pour chaque couleur c déjà crée (envisagée dans l'ordre de création) faire
            si s n'est relié à aucun sommet colorié par c alors
                s est colorié en c
            sinon
                on passe à la couleur suivante
        si s n'est pas colorié alors
            on introduit une nouvelle couleur c
            on colorie s avec c
```

/*NB : l'ordre dans lequel on considère les sommets est important, l' algorithme ne fournit qu'une borne supérieure du nombre chromatique. */

Algorithme 3 : ***Bellman-Kalaba***

Données : G : un graphe connexe orienté pondéré par l (l_{ij} : poids de l'arc (i,j))
i_0 : sommet de G, *on ajoute un arc fictif* (i_0, i_0) *de poids 0.*
Résultat : Si G a un circuit de longueur négative pas de solution sinon pour chaque sommet $i \neq i_0$, distance du plus court[long] chemin de i_0 à i ou ∞
Variables : j : sommet courant, k : étape courante

$\lambda_0(i_0) \leftarrow 0$
pour *tout sommet* $i \neq i_0$ **faire** $\lambda_0(i) = \infty\ [-\infty]$

$k \leftarrow 1$
tant que $k \leq n$ **faire**
- **pour** *tout sommet* j **faire** $\lambda_k(j) = \min[\max]_{i \in \Gamma^-(j)}(\lambda_{k-1}(i) + l_{ij})$
- **si** $\lambda_k = \lambda_{k-1}$ **alors** STOP
- **sinon** $k \leftarrow k+1$

si $k = n+1$ **alors** G admet un circuit < 0 $[> 0]$
sinon $\forall j \in X \setminus \{i\}, \lambda_k(j)$ est la longueur d'un chemin i_0j-minimal [maximal]

Algorithme 4 : ***Bellman pour les graphes sans circuits***

Données : G : graphe orienté pondéré sans circuit (poids de signe quelconque),
les sommets *étant numérotés de 1 à n en accord avec les niveaux de G*

Résultat : pour chaque sommet $i \neq 1$, $\lambda(i)$ distance du plus court[long] chemin du sommet 1 autres sommets.

$\lambda(1) \leftarrow 0$

pour *tout sommet* $j \in [\![2, n]\!]$ **faire** $\lambda(j) = \min[\max]_{i \in \Gamma^-(j)}(\lambda(i) + l_{ij})$

Algorithme 5 : ***Bellman-Ford***

Données : G : un graphe connexe orienté pondéré par l (l_{ij} : poids de l'arc (i, j))
i_0 : sommet de G.

Résultat : Si G a un circuit de longueur négative pas de solution sinon pour chaque sommet $i \neq i_0$, $\lambda(i)$ distance du plus court chemin de i_0 à i ou ∞

Variables : j : sommet courant, k : étape courante

$\lambda(i_0) \leftarrow 0$

pour *tout sommet* $i \neq i_0$ **faire** $\lambda(i) = \infty$

pour $k = 1$ *à* $n - 1$ **faire**
  pour *tout arc* (i, j) **faire**
    si $\lambda(j) > \lambda(i) + l_{ij}$ **alors** $\lambda(j) \leftarrow \lambda(i) + l_{ij}$; $pere(j) \leftarrow i$

pour *tout arc* (i, j) **faire si** $\lambda(j) > \lambda(i) + l_{ij}$ **alors** G admet un circuit < 0

Algorithme 6 : ***Roy-Floyd-Warshall***

Données : un graphe orienté pondéré poids de signe quelconque

Résultat : S'il $\exists$ un circuit de longueur < 0 $[> 0]$ alors pour tout i du circuit, $\mathcal{M}_{ii} < 0$ $[> 0]$ sinon $\forall i, j$, $\mathcal{M}_{ij}$ = longueur chemin ij-minimal[maximal]

Initialisation : $\forall i, j \quad M_{ij} = l_{ij}$ si $(i, j) \in U$, $\infty[-\infty]$ sinon.

pour $k \leftarrow 1$ *à* n **faire**
  pour $i \leftarrow 1$ *à* n **faire**
    pour $j \leftarrow$ *1 à* n **faire**
      $\mathcal{M}_{ij} \leftarrow \min[\max](\mathcal{M}_{ij}, \mathcal{M}_{ik} + \mathcal{M}_{kj})$

Algorithme 7 : ***Dijkstra (appelé aussi Moore-Dijkstra)***

Données : un graphe connexe orienté pondéré **positivement** de racine i_0

Résultat : pour chaque sommet $i \neq i_0$, $\lambda(i)$ distance du plus court chemin de i_0 à i et $pere(i)$ sommet précédant i sur ce chemin

Variables : S : sommets explorés ;
i : sommet courant ; j : successeur de i non exploré

pour *tout sommet* i **faire** $\lambda(i) \leftarrow \infty$

$\lambda(i_0) \leftarrow 0$

$S \leftarrow \emptyset$

tant que $S \neq X$ **faire**
  sélectionner i dans $X \setminus S$ tel que $\lambda(i) = \min_{j \in X \setminus S} \lambda(j)$
  $S \leftarrow S \cup \{i\}$
  pour *tout* j *de* $\Gamma^+(i) \cap (X \setminus S)$ **faire**
    si $\lambda(i) + l_{ij} < \lambda(j)$ **alors**
      /* ajustement de j à partir de i (distance et père) */
      $\lambda(j) \leftarrow \lambda(i) + l_{ij}$; $pere(j) \leftarrow i$