Des villes, des réseaux et une souris

1 Connexité

Exercice 1.1 (Algernon et le labyrinthe)

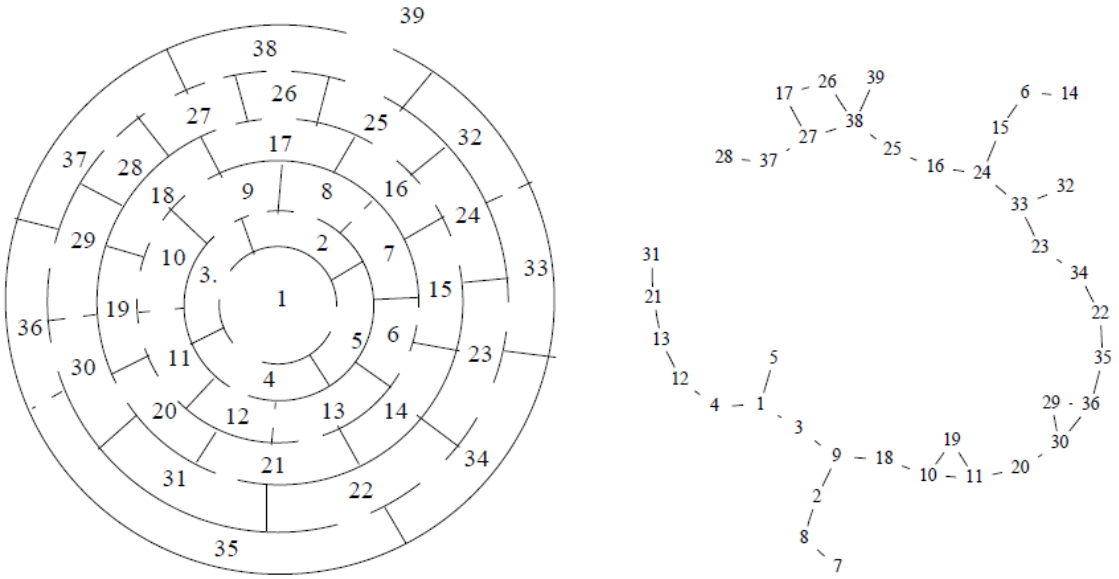

Figure 1 – Le labyrinthe d'Algernon

Algernon, petite souris blanche de laboratoire, apprend à sortir de labyrinthes.

1. Algernon est placée au centre du labyrinthe. Est-ce qu'elle pourra en sortir ? Est-ce qu'elle a plusieurs solutions ?
2. Pour les essais suivants, Algernon est placée à un nouvel endroit du labyrinthe à chaque fois. Peut-elle sortir du labyrinthe à tous les coups ?
3. Algernon est maline, elle a réussi à trouver un chemin depuis le centre et s'en souvient. Les chercheurs ferment donc toutes les portes par lesquelles elle est passée. Peut-elle encore sortir du labyrinthe depuis le centre ?

Exercice 1.2 (Réseau de routeurs)

Le fournisseur de service Internet (Internet Service Provider : ISP) *MarWaNet*[1] doit mettre en place son réseau de routeurs. Chaque routeur se trouve dans un *point d'interconnexion* pour avoir accès aux routeurs des autres ISP. Désirant établir son propre réseau (faire passer du trafic au travers du réseau d'un autre ISP étant facturé), il fait appel à différents *cablo-opérateurs* présents sur chaque site. Les liaisons proposées sont toutes des liaisons *point à point* permettant de relier un routeur avec un autre. À partir des offres proposées, la direction a obtenu la solution présentée en figure 2.

1. L'architecte réseau doit vérifier que les connexions proposées permettent réellement de relier tous les routeurs entre eux.
   (a) Quelle propriété de la théorie des graphes ce réseau doit-il respecter ?
   (b) Rappeler toutes les définitions en relation avec cette propriété (chaîne, chaîne élémentaire. . .).

1. Qui fêtera ses ? ? années d'existence ce 6 février. . .

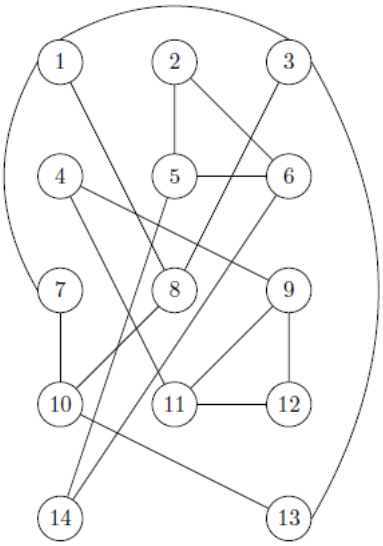

FIGURE 2 – Réseau de routeurs

2. Graphe connexe (*connected graph*) :
   (a) Donner le principe de vérification de la propriété de la question 1.
   (b) Appliquer ce principe au réseau de *MarWaNet*. Ce réseau permet-il de connecter tous les routeurs ?
3. Composantes connexes (*connected component*) :
   (a) Rappeler la définition de *composante connexe*.
   (b) Adapter le principe de l'algorithme précédent pour qu'il calcule le nombre de composantes connexes d'un graphe.

Exercice 1.3 (Minimiser les liaisons)

Louer une liaison *point à point* coûte cher à *MarWaNet*, le comptable demande donc à l'architecte réseau de trouver une solution garantissant que tous les routeurs sont reliés, mais en utilisant le minimum possible de liaisons *point à point*.

1. Comment peut-on satisfaire le comptable : comment obtenir un réseau où tous les routeurs sont reliés mais en utilisant un minimum de liaisons ?
2. Un tel graphe est un *arbre*.
   (a) Que se passe-t'il si on ajoute une arête quelconque à un *arbre* ?
   (b) Et si on lui enlève une arête quelconque ?
   (c) Quelles sont les trois propriétés d'un graphe qui est un *arbre* ?
3. Algorithme :
   (a) Est-il nécessaire de vérifier les trois propriétés de la question précédente pour déterminer si un graphe est un *arbre* ?
   (b) En déduire les différents méthodes pour vérifier qu'un réseau est bien interconnecté et est constitué d'un nombre minimum de liaisons.
   (c) Écrire un algorithme qui utilise obligatoirement un parcours profondeur pour vérifier si un graphe en représentation dynamique est un arbre (**contrôle n° 2 - 2013**).

Exercice 1.4 (I want to be a tree – *Contrôle 2012*)

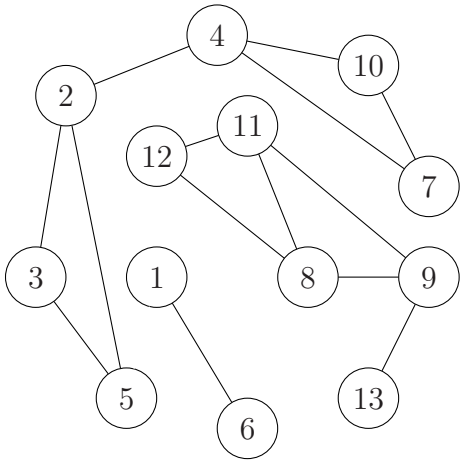

FIGURE 3 – Not a tree yet

Le but de cet exercice est de transformer un graphe en arbre en le modifiant le moins possible, à l'aide d'un parcours profondeur.

1. On effectue un parcours profondeur du graphe :
   (a) Quelles sont les arêtes qui peuvent être enlevées du graphe sans augmenter le nombre de composantes connexes ?
   (b) Comment repérer ces arêtes lors du parcours profondeur ?
   (c) Appliquer au graphe de la figure 3 : donner la liste des arêtes que l'on supprimera lors du parcours profondeur en choisissant les sommets dans l'ordre croissant (y compris pour les successeurs).

2. Pendant le parcours profondeur, on attribue à chaque sommet un numéro de composante connexe (de 1 à k, s'il y a k composantes) :
   (a) Combien d'arêtes faut-il ajouter pour rendre le graphe connexe ?
   (b) Comment, lors du parcours, savoir quelles arêtes ajouter pour rendre le graphe connexe ?
   (c) Donner le tableau des composantes connexes du graphe de la figure 3 obtenu lors du parcours profondeur toujours en choisissant les sommets dans l'ordre croissant.

3. **L'algorithme :**
   L'algorithme demandé ici sera un parcours profondeur d'un graphe non orienté en représentation statique.
   L'algorithme devra :
   — Construire le vecteur des composantes connexes du graphe initial, dans le vecteur *cc*.
   — Ajouter au graphe les arêtes qui permettent de le rendre connexe.
   — Supprimer du graphe les arêtes "inutiles" : sans augmenter le nombre de composantes.

4. **En plus :** *trouver - réunir*.
   (a) Adapter les algorithmes *trouver et réunir* (versions optimisées) du cours pour la représentation statique des graphes.
   (b) En déduire un algorithme qui rend un graphe connexe (en représentation statique), en y ajoutant les éventuelles arêtes manquantes.
   (c) Que faut-il modifier pour éliminer les arêtes inutiles ?

2 Forte connexité

Exercice 2.1 (Γ^+, Γ^-)

On considère l'algorithme suivant :

```
Marquer ⊕ et ⊖ un sommet arbitraire x du graphe
Tant que c'est possible faire
  Marquer ⊕ tout successeur d'un sommet marqué ⊕
  Marquer ⊖ tout prédécesseur d'un sommet marqué ⊖
Fin tant que
```

1. Quelle est la complexité de cet algorithme ?
2. Que représente l'ensemble des sommets qui sont marqués à la fois ⊕ et ⊖ ?
3. Comment utiliser cet algorithme pour déterminer si un graphe est fortement connexe ? Ou, s'il ne l'est pas, pour déterminer ses différentes composantes fortement connexes ?

Exercice 2.2 (Circulation à sens unique)

Monsieur Voie (de la mairie de Paris) est chargé d'installer des sens uniques dans tout un quartier afin que la circulation soit plus fluide (les rues étant trop étroites le stationnement et la circulation posent un problème). Voici le plan avec le sens de circulation de chaque rue. M. Voie se demande si on peut vraiment se déplacer partout dans le quartier sans en sortir.

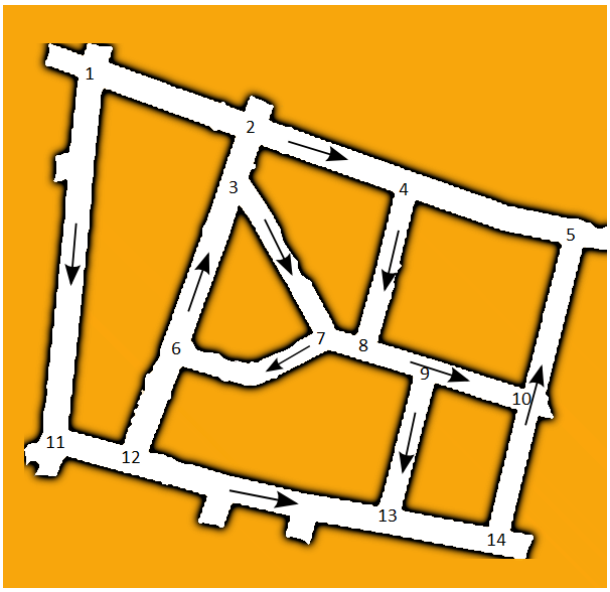

FIGURE 4 – Plan du quartier

1. Graphe fortement connexe (*strongly connected graph*).
   (a) Quelle propriété le plan de M. Voie doit-il respecter ?
   (b) Rappeler toutes les définitions en relation avec cette propriété (chemin, chemin élémentaire...).

2. **Méthode 1 :** deux parcours

   (a) Qu'observe-t-on lorsque l'on considère le graphe transposé *(inverse)* et la propriété que le graphe initial devrait respecter ?

   (b) Donner le principe de vérification de la propriété de la question 1a utilisant un double parcours.

   (c) Écrire l'algorithme correspondant pour une représentation dynamique d'un graphe.

   (d) Que faut-il modifier à cet algorithme pour construire le vecteur des composantes fortement connexes ?

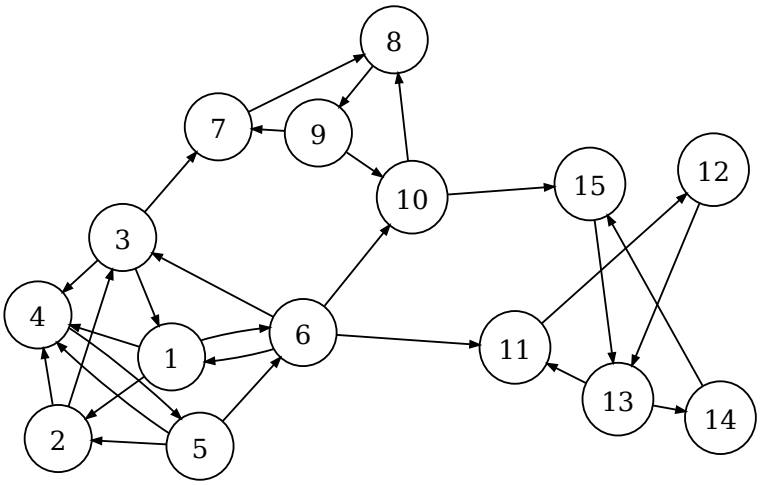

FIGURE 5 – Un autre plan

3. **Méthode 2 :** Tarjan, le vrai...

   (a) Quel célèbre principe peut-on aussi utiliser ?

   (b) Appliquer cette nouvelle méthode pour vérifier cette propriété sur le graphe de la figure 5.

   (c) Écrire l'algorithme qui détermine les composantes fortement connexes d'un graphe en représentation dynamique.

   (d) Et si on veut uniquement tester la forte connexité du graphe ?

Exercice 2.3 (Forte connexité : élimination des inutiles – *Contrôle 2014*)

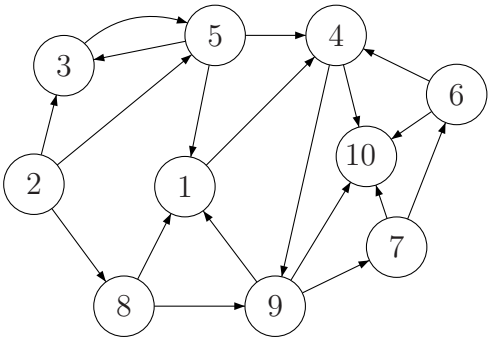

FIGURE 6 – Graphe G_2

1. Lors du parcours profondeur d'un graphe orienté, quels sont les arcs qui peuvent être enlevés *simplement* du graphe sans risquer d'augmenter le nombre de composantes fortement connexes ?

2. On numérote les sommets en ordre préfixe de rencontre, dans le vecteur $pref$. Comment repérer les arcs de la question 1 lors du parcours ?

3. En considérant un parcours profondeur du graphe de la figure 2 qui choisit les sommets en ordre croissant, quels sont les arcs qui pourront être retirés ?

4. Compléter les deux algorithmes qui permettent de supprimer d'un graphe orienté (en représentation **statique**) les arcs de la question 1. Le premier est l'algorithme d'appel, le second l'algorithme récursif qui effectue le parcours profondeur.

5. **Bonus :** Comment, lors du parcours profondeur, ajouter *simplement* des arcs (le moins possible) afin de rendre le graphe fortement connexe.

Exercice 2.4 (Graphe réduit)

Soit G un graphe orienté admettant p composantes fortement connexes : C_1, C_2, ..., C_p. On définit le *graphe réduit* de G (noté G_R) par $G_R = < S_R, A_R >$ avec :
- $S_R = \{C_1, C_2, \cdots, C_p\}$
- $C_i \to C_j \in A_R \Leftrightarrow$ Il existe au moins un arc dans G ayant son extrémité initiale dans la composante fortement connexe C_i et son extrémité terminale dans la composante fortement connexe C_j.

1. Montrer qu'un graphe réduit est toujours sans circuit.

2. Déterminer le graphe réduit du graphe de la figure 5.

3. Expliquer en quoi le graphe réduit peut permettre de simplifier la recherche d'un chemin (itinéraire) entre deux sommets donnés.

4. Écrire un algorithme qui construit le graphe réduit G_R d'un graphe G (les deux graphes sont en représentation statique). L'algorithme prendra en entrées le vecteur des composantes fortement connexes cfc, qui contient pour chaque sommet le numéro de la composante fortement connexe à laquelle il appartient, ainsi que le nombre de composantes fortement connexes du graphe G.

3 Pour finir

Exercice 3.1 (Résister aux pannes)

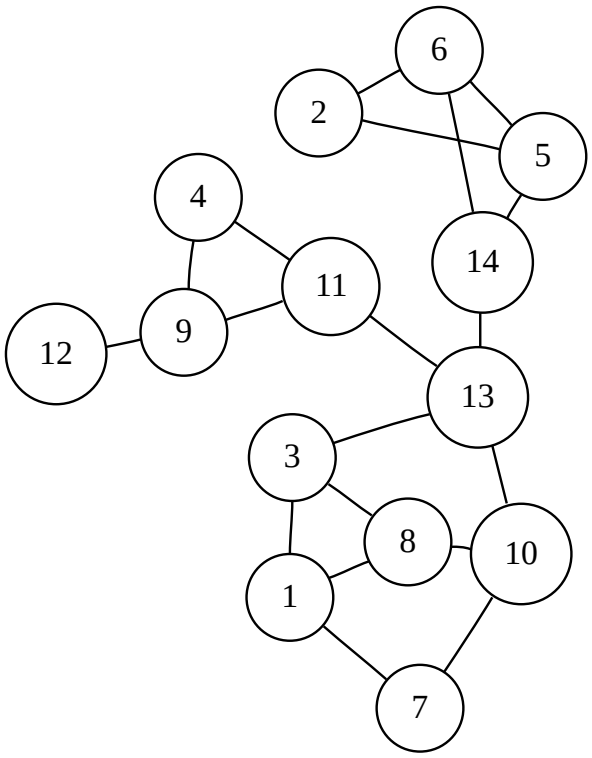

FIGURE 7 – Amélioration du réseau

Bien que minimiser le nombre de liaisons soit important, il convient à notre ISP d'avoir un réseau tolérant aux fautes. En effet, si un routeur se trouve coupé d'un autre, il devra emprunter une route alternative via les connexions externes des routeurs, or le trafic sur ces connexions (trafic de transit) a un coût relativement élevé. On s'intéresse donc à la résistance aux pannes, celles-ci peuvent être de deux sortes : panne d'un routeur et panne d'une liaison.

1. **Panne d'un routeur :**

   (a) Quel problème pose le réseau à nombre minimum de liaisons, en cas de panne de certains routeurs ? Quels sont ces routeurs ?

   (b) En considérant le réseau comme un graphe, comment appelle-t-on un sommet qui pose ce type de problème ?

   (c) Montrer qu'un tel sommet a au moins 2 fils s'il est racine d'une arborescence de parcours.

   (d) Montrer qu'un tel sommet nommé v, s'il n'est pas racine de l'arborescence, possède au moins un fils s tel qu'il n'existe aucun arc arrière partant de s ou d'un descendant de s et arrivant à un ancêtre de v.

   (e) Quelle information doit-on associer à chaque sommet pour représenter la propriété de la question précédente ? Comment calcule-t-on cette information ?

   (f) Donner le principe de détermination d'un tel sommet, en utilisant les réponses aux trois dernières questions.

   (g) Appliquer au graphe de la figure 7.

   (h) Écrire l'algorithme correspondant à ce principe pour un graphe en représentation dynamique.

2. **Panne d'une liaison :**

   (a) Quel problème pose le réseau à nombre minimum de liaisons en cas de liaison défectueuse ? En théorie des graphes, comment appelle-t-on une liaison qui pose ce type de problème.

   (b) Montrer qu'une telle liaison n'appartient à aucun cycle élémentaire du graphe.

   (c) Comment identifier toutes les liaisons ayant la propriété de la question 2a ?

   (d) De telles liaisons existent-elle dans le graphe de la figure 7 ?

   (e) Modifier le principe de la question 1f (et l'algorithme correspondant) pour qu'il détermine aussi ces liaisons.

3. **Sous-réseau sûr :**

   (a) Comment appelle-t-on les sous-graphes qui ne contiennent pas de sommets du type de la question 1b ?

   (b) Comment peut-on identifier ces sous-graphes en utilisant les résultats déjà obtenus dans les questions précédentes ?

   (c) Déterminer les sous-graphes ayant la propriété de la question 3a, dans le graphe de la figure 7 ?

   (d) Modifier le principe de la question 2e, pour qu'il détermine aussi ces sous-graphes.

Exercice 3.2 (Algernon et le labyrinthe (suite))

Le nouveau labyrinthe d'Algernon est maintenant muni de portes ne pouvant s'ouvrir que dans un sens.

1. Le labyrinthe est très grand et contient des circuits. Comment savoir quels sont les différents chemins impossibles, sans parcourir tout le graphe à chaque fois ? Comment repérer les impasses dont elle ne pourra plus repartir ?
2. Comment, à partir du labyrinthe initial, savoir quelles sont les portes qui doivent s'ouvrir dans les deux sens pour assurer qu'Algernon pourra aller partout sans pour autant être bloquée.