SAÉ S2.02 : Exploration algorithmique d'un problème
Propagation dans un réseau de contacts téléphoniques

Thématique générale : Modéliser un réseau de contacts téléphoniques de différentes façons, résoudre des problèmes de propagation de l'information, comparer différentes implémentations.

Sujet : Pour un groupe de personnes donné, on considère les liens qui existent entre ces personnes par le biais de leur répertoire de contacts (répertoire téléphonique). Il se peut que parmi ces personnes, certaines personnes ne conservent aucun contact, ou alors quelques contacts, ou énormément de contacts. Et ce n'est pas parce qu'une personne A conserve une personne B dans ses contacts, que la personne B conserve A dans ses contacts également. Dans l'exemple de la Figure 1, Bob a 3 contacts : Domi, Elie et Cali; alors que Domi n'a pas de contact; et personne n'a Anne dans ses contacts.

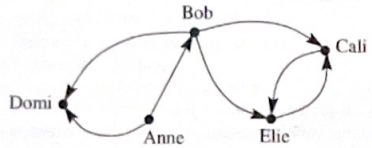

Figure 1: Exemple de réseau de contacts

On souhaite disposer d'un programme permettant de répondre à la question suivante :

Si une personne A détient une information importante, quelles sont toutes les personnes qui pourront être mises au courant de cette information grâce à ce réseau de contacts (par un SMS direct, ou indirectement par propagation de SMS)?

De plus, on souhaite modéliser le réseau de contacts de différentes façons afin de comparer l'efficacité de ces implémentations. Pour ce faire, on peut représenter le réseau de plusieurs façons, voici quelques exemples :

1. Une liste de personnes. Chaque personne contient la liste des personnes/contacts à qui il peut envoyer directement un SMS, puisqu'elles sont dans son carnet de contacts.
   Avec l'exemple de la Figure 1:
   Personnes : {A,B,C,D,E}
   Contacts de :
   A : {B,D}
   B : {C,D,E}
   C : {E}
   D : {}
   E : {C}

2. Une liste de personnes, et une liste séparée contenant les liens directs entre ses personnes. Un lien direct est un couple (A,B) et signifie que A a B dans ses contacts.

   Avec l'exemple de la Figure 1:
   Personnes : {A,B,C,D,E}
   Liens directs : {(A,B), (A,D), (B,C), (B,D), (B,E), (C,E), (E,C)}

3. Une liste de personnes. Un lien direct entre une personne et l'un de ses contacts est représenté par un pointeur vers la personne contact. Chaque personne pointe vers les liens directs dont elle est le "point de départ".

   Avec l'exemple de la Figure 1:
   Personnes : {A,B,C,D,E}
   Liens et le contact : {L1:B, L2:C, L3:D, L4:D, L5:E, L6:C, L7:E}
   Liens partant de :
   A : {L1, L3}
   B : {L2, L4, L5}
   C : {L7}
   D : {}
   E : {L6}

4. Les n personnes du groupe sont numérotées de 0 à n-1, et les liens directs entre elles sont stockés dans une matrice (tableau à deux dimensions) où la case (i,j) contient 1 si la personne n°i a la personne n°j dans ses contact (il existe un lien de la personne i vers la personne j), et 0 sinon.

   Avec l'exemple de la Figure 1:
   Personnes : {A=0, B=1, C=2, D=3, E=4}

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Travail demandé :

- ***Modélisation des réseaux.***
  Modéliser le réseau de contacts en C++ et en utilisant les paradigmes objet. Il faudra le faire d'au moins deux façons différentes parmi celles proposées ci-dessus (ou d'autres vous semblant pertinentes) afin de pouvoir les comparer par la suite. Votre code sera modulaire, pour que votre algorithme fonctionne indifféremment avec les

deux implémentations de réseau choisies. Pour cela, vous créerez une classe représentant un réseau. Chaque implémentation de réseau disposera d'une fonction permettant d'obtenir la liste des personnes pouvant recevoir un SMS directement envoyé par une personne A donnée (sans propagation de la part de personnes intermédiaires).
Pour chaque implémentation de vos réseaux, vous devrez concevoir les structures de données adaptées. Vous pourrez utiliser des listes, des tableaux... : à vous de choisir.

- ***Algorithme de propagation.***
  Développer l'algorithme de propagation demandé. Attention, comme le réseau peut avoir des cycles, pour éviter de boucler indéfiniment, il faudra trouver un moyen pour que l'algorithme sache quelle partie du réseau il a déjà explorée.
- ***Tests.***
  Tester votre programme sur des exemples de réseau suffisamment complexes (entre 10 et 20 personnes par exemple).
- ***Rapport.***
  Rédiger un rapport décrivant les choix d'implémentation et répondant aux questions détaillées plus bas (voir la section « Livrables »).

Organisation : Vous travaillerez par groupes de quatre étudiants. Les groupes ne sont pas libres mais formés par l'équipe enseignante.

Livrables : À l'issue de cette SAÉ, les différents livrables que vous devez fournir sont :

- Le code de l'application avec sa documentation. Vous détaillerez le travail effectué par chaque membre du groupe. Vous écrirez aussi dans les commentaires du code, qui a implémenté telle ou telle classe/méthode.
- Un rapport au format PDF contenant :
  1. Une description des implémentations du réseau que vous avez choisies, avec le détail des structures de données utilisées.
  2. Une explication des algorithmes développés pour résoudre le problème d'accessibilité. Cette explication doit être compréhensible par un non-informaticien. Vous pouvez illustrer cette explication par un schéma représentant un exemple de petit réseau, et dérouler les étapes de l'algorithme sur cet exemple.
  3. Une comparaison de la complexité de l'algorithme, en fonction des deux méthodes de représentation du réseau que vous avez choisies. Par complexité on entend le nombre d'opérations effectuées, au pire des cas, en fonction de la taille du réseau (la taille est le nombre personnes et de liens). À votre avis, laquelle des deux méthodes choisies est plus pertinente ? Pourquoi ?
  4. À votre avis, l'algorithme que vous avez fourni peut-il résoudre un ou plusieurs autres problème(s) similaire(s) dans un autre contexte ? Si oui lequel ?

Attention, il est important de justifier et d'expliquer toutes vos réponses.

Évaluation :
Qualité et complexité du code et de la documentation : 6 points
Compte-rendu, partie 1 : 4 points
Compte-rendu, partie 2 : 5 points
Compte-rendu, partie 3 : 4 points
Compte-rendu, partie 4 : 1 point

Modalités de rendu :
Vous rendrez une archive intitulée NOM1-NOM2-NOM3-NOM4.zip contenant le code et le rapport au format PDF, nommé NOM1-NOM2-NOM3-NOM4.pdf, avant le vendredi 4 mars 2022 à minuit :

- sur Karuta (chaque étudiant dépose l'archive dans son portfolio),
- ainsi que par e-mail à votre enseignant du cours R201 - Développement orienté objets (un seul envoi par groupe). Précisez vos noms dans l'objet de l'e-mail.

Compétence et apprentissages critiques concernés :
Compétence 2 (Optimiser des applications informatiques / Appréhender et construire des algorithmes)

- Apprentissage Critique (AC1) : Analyser un problème avec méthode (découpage en éléments algorithmiques simples, structures de données...)
- Apprentissage Critique (AC2) : Comparer des algorithmes pour des problèmes classiques (tris simples, recherche...)
- Apprentissage Critique (AC3) : Expérimenter la notion de compilation et les représentations bas niveau des données
- Apprentissage Critique (AC4) : Formaliser et mettre en œuvre des outils mathématiques pour l'informatique