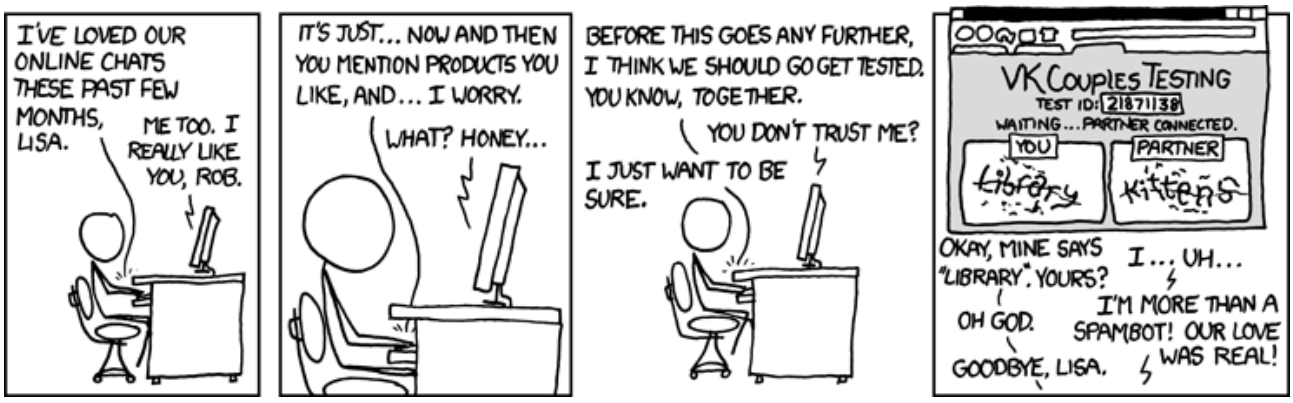
Prologin 2013

Concours national d'informatique

Épreuve écrite d'algorithmique
Paris II

Samedi 16 mars 2013

The Imitation Game



XKCD – Suspicion, par Randall Munroe.
Attention : cette image n'a absolument aucun rapport avec le reste du sujet.

1 Préambule

Bienvenue à **Prologin**. Ce sujet est l'épreuve écrite d'algorithmique et constitue la première des trois parties de votre épreuve régionale. Sa durée est de 3 heures. Par la suite, vous passerez un entretien (20 minutes) et une épreuve de programmation sur machine (3 heures 30).

Conseils

- Lisez bien tout le sujet avant de commencer.
- **Soignez la présentation** de votre copie.
- N'hésitez pas à poser des questions.
- Si vous avez fini en avance, relisez bien, ou préparez votre présentation pour l'entretien.
- N'oubliez pas de passer une bonne journée.

Remarques

- Le barème est donné à titre indicatif uniquement.
- Indiquez lisiblement vos nom et prénom, la ville où vous passez l'épreuve et la date en haut de votre copie.
- Tous les langages sont autorisés, veuillez néanmoins préciser celui que vous utilisez.
- Ce sont des humains qui lisent vos copies : laissez une marge, aérez votre code, ajoutez des commentaires (**seulement** lorsqu'ils sont nécessaires) et évitez au maximum les fautes d'orthographe, sinon ça va barder.
- Le barème récompense les algorithmes les plus efficaces : écrivez des fonctions qui trouvent la solution le plus rapidement possible.
- Si vous trouvez le sujet trop simple, relisez-le, réfléchissez bien, puis dites-le-nous, nous pouvons ajouter des questions plus difficiles.

2 Sujet

Introduction

Ça y est, l'épreuve écrite de Prologin 2013 a commencé. Il est donc temps de vous concentrer sur votre principale activité des trois prochaines heures, à savoir : **copier sur vos camarades**.

On considère une population de n candidats Prologin situés dans un amphithéâtre à l'épreuve régionale de l'École polytechnique. Chaque candidat est identifié par un *ID* unique (représenté par un entier) et possède des *voisins* : les autres candidats dans son champ de vision, reliés à lui par des *liaisons*. Dans tout le problème, on considérera que tous les candidats exécutent les mêmes algorithmes.

Par exemple, pour dire bonjour, on pourra considérer l'algorithme suivant :

Algorithme HelloWorld

```
Pour tout voisin dans mes voisins
  Dire "Bonjour !"
```

Et, comme vous êtes polis, j'imagine que vous aurez aussi implémenté cette fonction, exécutée à chaque fois qu'un candidat reçoit un bonjour :

Algorithme onBonjour [1]

```
Répondre "Bonjour !"
```

Ainsi, on entendra $2n$ fois « Bonjour ! » dans la salle [3].

Il peut être utile d'élire le *leader* de la salle. Il s'agit du candidat qui a le plus grand ID.

Question 1 (2 points)

> Écrire un algorithme qui détermine l'identité du *leader* et la propage à tous les candidats.

Il est donc maintenant temps de passer aux choses sérieuses. Supposons que *chaque* candidat implémente l'algorithme suivant :

Algorithme Tricher

```
Choisir un voisin au hasard.
Copier sur lui.
```

Question 2 (3 points)

> Montrer qu'il existe forcément un cycle, c'est-à-dire une boucle $c_1 \to c_2 \to \ldots \to c_n \to c_1$, où $\to$ représente la relation « copier sur ».

1. Cet algorithme pourra être remplacé à loisir par un `Répondre "T'es qui ?"`, qui à son tour donnera lieu à l'implémentation d'une fonction onTEsQui [2] renvoyant le ID de la personne courante, par exemple.
2. Ceci n'a rien à voir avec Montesquieu, ne faites pas l'enfant.
3. N. B. – Les organisateurs Prologin ne disent pas bonjour. Ils ont autre chose à faire.

Vous vous rendez compte que vous n'arriverez pas à grand-chose ainsi, et décidez de convenir d'un réseau de communication non orienté sans cycle qui vous permettra de vous faire passer des messages pendant l'épreuve. Vous connaissez la liste de vos voisins ainsi que votre distance à chacun d'entre eux (on considère que toutes les distances sont différentes).

Envoyer un message à un candidat sur une distance plus longue risque de vous faire griller auprès des organisateurs Prologin, qui, entre deux pains au chocolat, surveillent un peu la salle. Vous cherchez donc à obtenir l'arbre [4] couvrant [5] de poids minimal [6]. Il s'agira donc de déterminer progressivement, pour toutes les liaisons, lesquelles seront :

– *dangereuses* : ne figureront pas dans l'arbre final ;
– *intimes* : figureront dans l'arbre final.

Initialement, toutes les liaisons sont dites *banales*.

Question 3 (2 points)

> Supposons qu'un tel arbre soit découvert, mais qu'un candidat quitte la salle pour aller à son entretien. Comment déterminer le nouvel arbre de poids minimal ? Il n'est pas demandé d'écrire un algorithme pour cette question.

Dans un premier temps, on considère un monde idéal où vous avez eu le temps de préparer un plan en discutant sur Google+ [7] au préalable et donc, vous savez que vous devez effectuer votre tâche toutes les 10 minutes.

On appelle *équipe* un ensemble de k gens reliés par $k-1$ liaisons. On s'attachera à ce que tous les membres d'une équipe en connaissent l'identité du *leader*. Le principe est le suivant : au départ (à 9 h), chaque candidat est une équipe. Dans les 10 minutes, chaque équipe va déterminer le voisin le plus proche d'un membre de l'équipe qui ne soit pas dans l'équipe, noté *cible* (algorithme Stalker), et lui envoyer un *poke* [8]. Au bout de 10 minutes (chaque candidat a alors déterminé son voisin le plus proche), si deux candidats se sont pokés mutuellement, ils ~~tombent amoureux~~ fusionnent leurs équipes (algorithme Fuuusion), et le nouveau *leader* doit propager son influence à l'ancienne équipe. Puis, à nouveau on démarre une étape de Stalker, jusqu'à 9 h 20 où une fusion est opérée, et ainsi de suite. Cet algorithme garantit d'obtenir l'unique [9] arbre couvrant de poids minimal recherché.

Question 4 (2 points)

> À quelle heure au plus tard l'arbre sera-t-il construit, en fonction de n ?

La fonction Stalker se fait en trois temps. Dans un premier temps, le *leader* réveille d'abord son équipe en diffusant un message relayé de voisin en voisin. Ensuite, chaque candidat membre de l'équipe va être amené à tester différents voisins banals, ne sachant pas s'ils font déjà partie de son équipe ; il devra utiliser la commande LolTKi ? qui demande au candidat voisin de décliner l'identité du *leader* de son équipe. Enfin, chaque candidat rétropropage [10] ses recherches au *leader*, qui détermine la cible à poker.

4. C'est-à-dire sans cycle.
5. C'est-à-dire qu'entre deux candidats quelconques il existera toujours une chaîne de voisins.
6. C'est-à-dire que la somme des distances des liaisons choisies sera minimale.
7. Désolé, Facebook n'est pas (encore ?) un sponsor du concours.
8. Ou une gomme, ou autre projectile susceptible d'attirer son attention.
9. Attendez, on y vient.
10. Du verbe *rétropropager* qui n'existe pas.

Question 5 (7 points)

> Implémenter les algorithmes Stalker, Fuuusion et onLolTKi.

On suppose à présent qu'aucun plan est établi à l'avance, et donc les opérations de Fuuusion [11] sont menées en même temps que les opérations de Stalker. Ainsi, on munit les équipes d'un *niveau*, tel que :

- lorsque deux équipes de niveau k sont fusionnées, on obtient une équipe de niveau $k+1$;
- lorsqu'il s'agit de deux équipes de niveaux $k < k'$, on obtient une équipe de niveau k' (absorption).

Il peut arriver que, lorsqu'un candidat effectue une requête LolTKi ? à un voisin, celui-ci ne soit pas encore averti de la fusion en cours de leurs équipes [12]. Le candidat attend alors la réponse de son voisin.

Question 6 (2 points)

> Comment vous aider des niveaux des équipes pour modifier l'algorithme onLolTKi en conséquence ?

Il faut alors s'interroger sur ce qui se passe lorsque une opération de Fuuusion (absorption) est en cours selon que le candidat cible de l'équipe absorbante a déjà ou pas encore effectué la phase de rétropropagation de la fonction Stalker.

Question 7 (6 points)

> Que deviennent alors les fonctions Stalker et Fuuusion ?

Question bonus 8 (4 points)

⋆ Prouvez que votre algorithme des deux questions précédentes termine [13].

Question bonus 9 (3 points)

Prouvez l'unicité de l'arbre couvrant minimal, sachant que les distances entre deux candidats sont deux à deux distinctes.

Question bonus 10 (3 points)

Que se passe-t-il si des organisateurs Prologin se dissimulent parmi les candidats et communiquent de fausses informations au sein du réseau ?

Question bonus 11 (1 point)

Quel est l'acteur actuellement en lice pour interpréter Alan Turing dans le film biographique *The Imitation Game* ?

Le sujet est sur 24 points, et les questions bonus rapportent au total 11 points, plus 1 point de présentation.

11. Ha ha, énorme, cette césure involontaire.
12. On est de la même équipe, **mais tu ne le sais pas encore**.
13. Indication : aidez-vous des niveaux.