CSC_3IN01_TA - Devoir maison

19 novembre 2024

François Pessaux

Énoncé

- **Lisez attentivement les consignes et tout le sujet avant de commencer.**

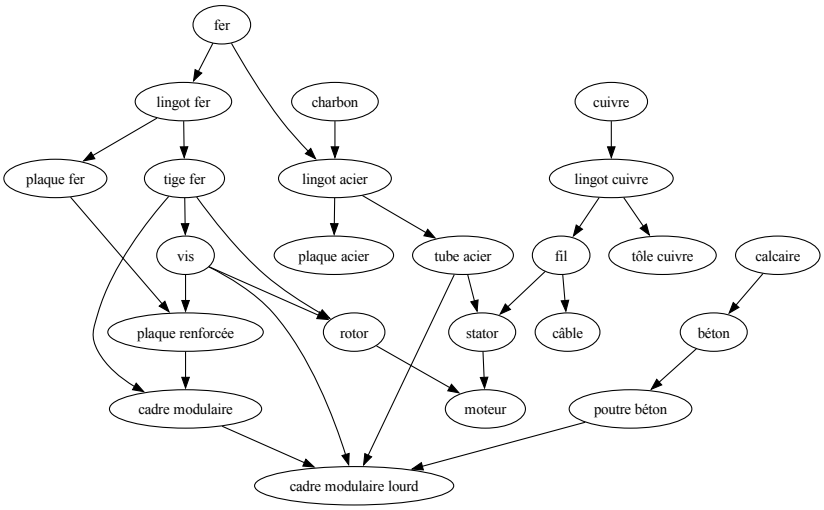
- Lorsqu'il vous est demandé que votre programme réponde en affichant « **Yes** » ou « **No** », il ne doit **rien** afficher d'autre, et **pas** « Oui » ou « Yes. » ou « no » ou « La réponse est : no ». Donc, pensez à retirer vos affichages de test / debug, sinon les tests automatiques **échoueront** sans pitié.

- Libre à vous de ne pas faire de devoir maison.
- Le but est de vous rendre compte de **votre niveau** par rapport à l'attendu de l'examen final.
- La correction sera **totalement** automatique et ne prendra pas en compte les questions de réflexion préalable.
- Ces questions ne sont pas là pour gagner des points mais pour **structurer votre pensée** et **réussir** ensuite le reste.
- La « récompense » sera de 0 à 2 pts rajoutés à la note d'examen final en fonction
  - des tests réussis,
  - de métriques (automatiques) relatives à la structure de votre programme,
  - de la preuve de « bonne volonté ».
- Vous pouvez truander les règles du jeu, vous aurez vos 2 pts, mais tant pis, vous ne saurez pas vous situer.

- Pour rendre votre devoir, vous devrez créer une **archive** contenant **tous** les fichiers **sources** que vous avez écrits (`.c`, `.h`). **N'incluez pas** d'exécutables dans l'archive, le mail pourrait la considérer comme un attachement dangereux et le supprimer. Le nom de cette archive devra avoir la structure suivante :
  `nom_prenom.zip` ou `.tgz` (selon l'outil d'archivage que vous utilisez).
- Vous devrez **m'envoyer** cette archive par mail.
- L'exercice est calibré pour 2h environ et vous devrez le rendre **au plus tard le dimanche 01 décembre 2024 à 23h59** (ce qui vous laisse environ une semaine et demie).

1 Chaîne de production

On se place dans le contexte d'une chaîne de production extrêmement simplifiée où l'on construit des produits à partir d'autres produits. Ainsi, par exemple, à partir de « fer » on produit des « lingots ». À partir de « lingots » on produit des « tiges » ou des « plaques ». À partir de « tiges » on produit des « vis ». À partir de « plaques » et de « vis » on produit des « plaques renforcées », etc.[1] Le schéma ci-dessous donne un exemple de telle chaîne de production.



Dans cette chaîne logistique, il **n'y a pas moyen de déconstruire** un produit. Dit autrement, on ne peut pas refaire des tiges à partir de vis.

But macroscopique du programme : On souhaite déterminer si deux produits sont liés par une relation de dépendance dans une chaîne de production donnée. C'est par exemple le cas entre « rotor » et « lingot de fer » puisqu'il y a besoin de lingots de fer pour fabriquer des rotors. Par contre, ce n'est pas le cas entre « charbon » et « rotor ».

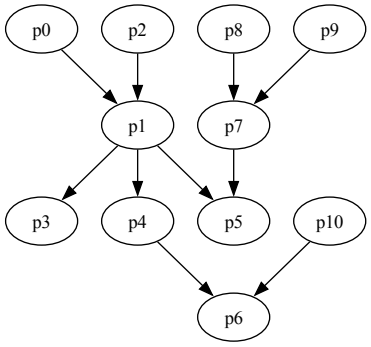
Description de la chaîne de production : Les produits ne seront pas désignés par des noms « alphabétiques » mais par des **numéros** débutant à **0** (pour vous simplifier la vie). Les relations « matière première / produit construit » seront obtenues depuis un fichier texte contenant :

1. le nombre de produits connus dans la chaîne (que l'on appellera *size*),
2. des lignes composées au choix de :
   - id_1 donne id_2
   - id_1 requiert id_2

où id_1 et id_2 représentent les « noms » (donc les identifiants) des produits concernés.

Ainsi, la chaîne de production de la partie gauche ci-dessous sera représentée indifféremment par les deux fichiers texte (`ex1.dat` et `ex1_1.dat` joints dans l'archive de l'énoncé) de la partie droite. Le premier fichier n'utilise que la relation `"donne"`, le second utilise également la relation `"requiert"`. Notez bien que dans la représentation « graphique », les produits sont nommés `p0`, `p1`, etc. mais dans les fichiers de données, le « `p` » n'apparaît pas (c'est l'outil qui a permis de générer cette représentation graphique qui l'a rajouté, donc cela ne vous concerne pas).

1. Toute ressemblance avec un jeu existant ou ayant existé serait purement fortuite.



Fichier ex1.dat

```
11
0 donne 1
2 donne 1
1 donne 3
1 donne 4
1 donne 5
4 donne 6
7 donne 5
9 donne 7
8 donne 7
10 donne 6
```

Fichier ex1_1.dat

```
 11
1 requiert 0
2 donne 1
1 donne 3
4 requiert 1
1 donne 5
4 donne 6
5 requiert 7
9 donne 7
7 requiert 8
10 donne 6
```

Nommage : Le fichier source de ce programme devra s'appeler `factory.c`.

Format d'entrée : Votre programme prendra en arguments de **ligne de commande**, et **dans cet ordre** : le nom du fichier décrivant la chaîne de production, le « nom » du premier produit concerné, le « nom » du second produit concerné.

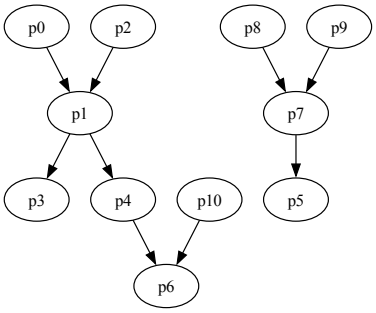
Format de sortie :

- S'il existe une relation de dépendance entre les deux produits, la **première** ligne d'affichage contiendra, séparés par **un espace**, les « noms » des produits participant à ce chemin de dépendance. Le « sens» d'affichage du chemin **n'importe pas** (par exemple, sur la chaîne précédente, il existe une dépendance entre 0 et 6 : on pourra obtenir l'affichage `0␣1␣4␣6` ou bien `6␣4␣1␣0` (selon ce qui vous arrange pour votre algorithme). Un **retour à la ligne** final viendra compléter ce premier affichage.

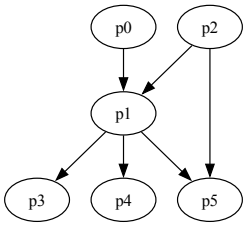
  La **dernière ligne** d'affichage devra être : « `Lien.` » La ligne sera terminée par un **retour à la ligne**.
- Sinon, la **dernière ligne** d'affichage devra être : « `Pas de lien.` » La ligne sera terminée par un **retour à la ligne**.

Jeux de données : Des jeux de données vous sont donnés à titre d'exemples dont les représentations graphiques sont les suivantes (`ex1.dat` et `ex1_1.dat` ont déjà été présentés ci-dessus) :

`ex2.dat`



`ex3.dat`



Question 1.1

Comment peut-on « informatiquement » représenter une telle chaîne de production ?

Question 1.2

À quelle(s) condition(s) une chaîne de production (au sens abstrait, pas au sens du contenu d'un éventuel fichier qui contiendrait sa description) est-elle correcte ?

Dans toute la suite, on considérera **par hypothèse** que les chaînes de production traitées **sont correctes**.

Question 1.3

Qu'est-ce que cela implique sur la « topologie » de la structure de données permettant de représenter une chaîne de production ?

Question 1.4

Identifiez les grandes étapes du programme, sans descendre dans le détail des fonctions que vous allez écrire. Cela représentera en quelque sorte votre futur `main`.

Question 1.5

Donnez deux avantages de découper le programme que vous allez écrire en plusieurs fonctions.

Question 1.6

On s'intéresse maintenant à la fonction de chargement d'une chaîne de production. Quels sont ses domaines d'entrée et de sortie ?

Question 1.7

Identifiez les grandes étapes de la fonction de chargement d'une chaîne de production. La gestion des cas de fichier mal formé est **hors sujet**. De même, on ne vous demande **pas** de vérifier le succès des **lectures** dans le fichier.

Question 1.8

Expliquez algorithmiquement de quelle manière vous allez explorer la chaîne de production pour répondre à l'existence d'une dépendance entre deux produits p_1 et p_2.

Question 1.9

Maintenant, on change légèrement la question à laquelle répondre, qui devient « existe-il une relation de dépendance **suffisante** entre 2 produits » (l'un suffit à produire l'autre) ? Sur le premier schéma, identifiez :

- une relation de dépendance **suffisante** non triviale entre 2 produits (i.e. pas une simple dépendance directe comme entre « plaque de fer » et « fer »),
- un exemple où votre parcours précédent ne fonctionnerait pas.

Question 1.10

Transformez votre réflexion algorithmique en un programme effectif. **Respectez les formats d'entrée et de sortie** spécifiés en introduction de cet exercice.

Le fichier `runtests.sh` qui vous est fourni dans l'archive vous permet de tester votre programme. Il l'exécute et compare les résultats obtenus par rapport aux résultats attendus, en affichant un rapport dans le terminal.

Vous pouvez le copier dans votre répertoire de travail sur le serveur Linux `info1` et le lancer depuis le terminal. Vous devrez toutefois modifier la ligne 7 pour remplacer

`"/home/moi/mon/repertoire/de/devoir"`

par le chemin du répertoire où vous l'avez copié et où se trouve votre fichier source et les fichiers d'exemple. Pour trouver ce chemin, allez dans le répertoire et recopiez le chemin affiché par la commande `pwd`.
Ensuite, dans un terminal, invoquez `./runtests.sh` pour lancer les tests. Même sans lancer le script, vous pouvez lire dedans les résultats attendus en fonction des arguments passés.

—— Fin du sujet ——