Concours Commun INP (CCINP), France

Session 2016, filière MP

Informatique

Durée : 4 heures

Ce que traite le sujet

- Partie I : logique et calcul des propositions
- Partie II : algorithmique et programmation

Sujet reproduit sans retouche à partir de la page suivante.

Résumé des parties, chapitres de cours liés et autres années :
prepahelp.com/annales/ccinp/mp/2016/info

Prepa Help ne revendique aucun droit sur le sujet, propriété de l'organisateur du concours.

CONCOURS COMMUNS
POLYTECHNIQUES

EPREUVE SPECIFIQUE - FILIERE MP

INFORMATIQUE

Jeudi 5 mai : 14 h - 18 h

N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

Les calculatrices sont interdites

Les deux parties qui composent ce sujet sont indépendantes et peuvent être traitées par les candidats dans un ordre quelconque.

Partie I. Logique et calcul des propositions

Nous nous intéressons dans cet exercice à l'étude de quelques propriétés de la logique propositionnelle tri-valuée. En plus des deux valeurs classiques VRAI ($\top$) et FAUX ($\bot$) que peut prendre une expression, la logique propositionnelle tri-valuée introduit une troisième valeur INDETERMINE (?).
$\mathcal{V}$ est l'ensemble des variables propositionnelles et $\mathcal{F}$ l'ensemble des formules construites sur $\mathcal{V}$. Pour $A, B \in \mathcal{V}$, les tables de vérité des opérateurs classiques dans cette logique propositionnelle sont les suivantes :

(a) $A \wedge B$

A	B	$A \wedge B$
$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\top$	?	?
$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$
$\bot$	?	$\bot$
?	$\top$	?
?	$\bot$	$\bot$
?	?	?

(b) $A \vee B$

A	B	$A \vee B$
$\top$	$\top$	$\top$
$\top$	$\bot$	$\top$
$\top$	?	$\top$
$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\bot$
$\bot$	?	?
?	$\top$	$\top$
?	$\bot$	?
?	?	?

(c) $A \Rightarrow B$

A	B	$A \Rightarrow B$
$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\top$	?	?
$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\top$
$\bot$	?	$\top$
?	$\top$	$\top$
?	$\bot$	?
?	?	$\top$

(d) $\neg A$

A	$\neg A$
$\top$	$\bot$
$\bot$	$\top$
?	?

Définitions

Définition 1 Tri-valuation.
Une tri-valuation est une fonction $f : \mathcal{V} \to \{\top, \bot, ?\}$.

On étend alors de manière usuelle la notion de tri-valuation sur l'ensemble des formules :

Définition 2 Une tri-valuation sur l'ensemble des formules est une fonction $\hat{f} : \mathcal{F} \to \{\top, \bot, ?\}$.

Définition 3 Une tri-valuation $\hat{f}$ satisfait une formule ϕ si $\hat{f}(\phi) = \top$. On notera alors $\hat{f} \vdash_3 \phi$.

Définition 4 Formule.
Une formule ϕ est :
- une conséquence d'un ensemble de formules $\mathcal{X}$ si toute interprétation qui satisfait toutes les formules de $\mathcal{X}$ satisfait ϕ. On notera dans ce cas $\mathcal{X} \Vdash_3 \phi$;
- une tautologie si pour toute tri-valuation $\hat{f}$, $\hat{f}(\phi) = \top$. On notera dans ce cas $\Vdash_3 \phi$.

Questions

I.1. Montrer que $A \vee \neg A$ n'est pas une tautologie.

I.2. Proposer alors une tautologie simple dans cette logique.

Posons $\top = 1, \bot = 0$ et $? = 0,5$.

I.3. Proposer un calcul simple permettant de trouver la table de vérité de $A \wedge B$ en fonction de A et B. Même question pour $A \vee B$.

I.4. En logique bi-valuée classique, les propositions $\neg A \vee B$ et $A \Rightarrow B$ sont équivalentes. Qu'en est-il dans le cadre de la logique propositionelle tri-valuée ?

I.5. En écrivant les tables de vérité, indiquer si les propositions $\neg B \Rightarrow \neg A$ et $A \Rightarrow B$ sont équivalentes.

I.6. Donner la table de vérité de la proposition $((A \Rightarrow B) \wedge ((\neg A) \Rightarrow B)) \Rightarrow B$. Cette proposition est-elle une tautologie ?

Un nouvel opérateur d'implication, noté $\rightarrow$, est alors défini, dont la table de vérité est la suivante :

A	B	$A \rightarrow B$
$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\top$	$?$	$?$
$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\top$
$\bot$	$?$	$\top$
$?$	$\top$	$\top$
$?$	$\bot$	$?$
$?$	$?$	$?$

I.7. $A \rightarrow A$ est-elle une tautologie ?

I.8. Montrer qu'il n'existe aucune tautologie en utilisant uniquement cette définition de l'implication.

I.9. La proposition suivante est-elle une tautologie :

“ $(\{A\} \Vdash_3 B)$ est équivalent à $(\Vdash_3 A \rightarrow B)$ ” ?

On définit alors un type `FormuleLogique` représentant les formules de la manière suivante :

```
type FormuleLogique
|Vrai (* Constante Vrai*)
|Faux (* Constante Faux*)
|Indétermine (* Constante Indeterminé*)
|Var of string (* Variable propositionnelle*)
|Non of FormuleLogique (* Négation d'une formule*)
|Et of FormuleLogique*FormuleLogique (* conjonction de deux formules*)
|Ou of FormuleLogique*FormuleLogique (* disjonction de deux formules*)
|Implique of FormuleLogique*FormuleLogique (* implication*)
```

I.10. Avec la représentation précédente, écrire en CaML la formule :

$$((A \Rightarrow B) \wedge ((\neg A) \Rightarrow B)) \Rightarrow B \ .$$

I.11. Écrire alors une fonction récursive CaML `lectureFormule`, prenant en argument une formule et renvoyant une chaîne de caractères spécifiant comment un lecteur lirait la formule. Ainsi, par exemple, pour $\phi = A \wedge (\neg B)$, `lectureFormule` ϕ renvoie `A et non B`.

Partie II. Algorithmique et programmation

Le calcul de flots dans des graphes permet de modéliser et de répondre à une très large classe de problèmes, dont l'interprétation correspond à la circulation de flux physiques sur un réseau : distribution électrique, acheminement de paquets de données sur Internet, ...

Il s'agit en général d'acheminer la plus grande quantité possible de "matière" (courant, données, ...) entre une source s et une destination t. Les liens permettant d'acheminer les flux ont une capacité limitée et il n'y a ni perte ni création de "matière" lors de l'acheminement : pour chaque noeud intermédiaire du réseau, le flux entrant (ce qui arrive) doit être égal au flux sortant (ce qui repart).

Dans la suite de ce problème, nous nous intéressons à une modélisation de ces problèmes de flot par des graphes, dits réseaux de transport. Nous détaillons un algorithme de recherche d'un flot maximal, que nous relions à d'autres notions sur les graphes, telles que les coupes ou les couplages.

Définitions et premières propriétés

Dans toute la suite, nous considérons un graphe pondéré $G = (V_G, E_G)$ orienté, où V_G est l'ensemble des sommets de G et E_G l'ensemble des arcs de G. $|V|$ désigne le cardinal de V_G, c'est-à-dire le nombre de sommets.

Définition 5 Réseau de transport.
G est un réseau de transport si :
- il existe dans V_G deux sommets particuliers, la source s_G et le puits t_G ;
- il existe une fonction $c : V_G \times V_G \to \mathbb{R}^+ \cup \{\infty\}$, appelée capacité et évaluant les arcs $(u, v) \in E_G$ de G.

Dans cette définition, ∞ est un élément absorbant pour tous les opérateurs arithmétiques, sauf l'inversion.

Définition 6 Flot.
Un flot de G est une fonction $x : V_G \times V_G \to \mathbb{R}$, qui associe à chaque arc (u, v) de E_G une quantité de flot $x(u, v)$.

On définit pour x une fonction bilan $\mathcal{B} : V_G \to \mathbb{R}$, telle que :

$$\forall y \in V_G \quad \mathcal{B}(y) = \sum_{\substack{(u,v) \in E_G, \\ u = y}} x(u, v) \; - \sum_{\substack{(v,u) \in E_G, \\ u = y}} x(v, u) \; .$$

Le flot x est alors de valeur ν si :

$$\begin{cases} \mathcal{B}(s_G) &= \nu \\ \mathcal{B}(t_G) &= -\nu \\ \forall u \in V_G \backslash \{s_G, t_G\} \quad \mathcal{B}(u) &= 0 \end{cases} \tag{1}$$

et

$$\forall (u, v) \in E_G \quad 0 \leqslant x(u, v) \leqslant c(u, v) \tag{2}$$

II.1. Donner une interprétation des équations (1) et (2).

Le problème du *flot maximum* consiste à maximiser ν en respectant les contraintes imposées par (1) et (2).

Nous supposons dans la suite de ce problème que :

— G ne contient aucun circuit orienté de s_G vers t_G composé uniquement d'arcs de capacité infinie ;
— G est un graphe symétrique, ce qui est possible car nous autorisons les arcs à avoir une capacité nulle. Seuls les arcs dont la capacité est strictement positive seront représentés.

La valeur maximum d'un flot peut être reliée avec une autre caractéristique du réseau de transport G.

Définition 7 s,t-coupe.
Soit $S \subset V_G$ et $T = V_G \backslash S$ tels que $s_G \in S$ et $t_G \in T$. Les arcs dont l'origine est dans S et l'extrémité terminale dans T forment une s,t-coupe de G. On note $c(S,T) = \sum_{\substack{(u,v) \in E_G, \\ u \in S, v \in T}} c(u,v)$ la capacité de la coupe.

Définition 8 Coupe minimale.
Une coupe minimale est une coupe de capacité minimale parmi toutes les s,t-coupe de G.

II.2. Soit x un flot de valeur ν dans G et (S,T) une s,t-coupe de G. Montrer que :

$$\nu \leqslant c(S,T).$$

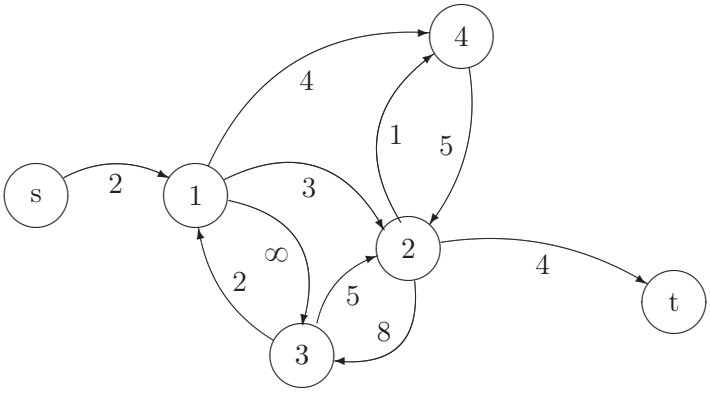
Indication : faire la somme des relations (1) pour tous les sommets de S.

Définition 9 Flot maximal.
Soit G un réseau de transport et x un flot sur G. Si la valeur du flot x est égale à la capacité d'une coupe minimale de G, alors le flot est maximal. A l'inverse, si une coupe de G a une capacité égale à la valeur d'un flot maximal, alors cette coupe est minimale.

Cette relation sera utilisée pour stopper dans de bonnes conditions un algorithme itératif de maximisation de flot.

Modélisation

Dans la suite, un graphe sera représenté par sa matrice d'adjacence A, $A_{u,v}$ représentant la capacité de l'arc (u,v). Le réseau de transport G suivant sera utilisé pour répondre aux questions de ce paragraphe :



II.3. Proposer en CAML des types `reseau_transport` et `flot` permettant de modéliser un réseau de transport et un flot. Préciser en particulier comment indiquer que deux sommets ne sont pas reliés dans le graphe et comment modéliser une capacité infinie.
Déclarer alors le réseau de transport G.

II.4. Écrire en CAML une fonction `flot_admissible:reseau_transport→flot→bool`, qui teste qu'un flot vérifie les équations (1) et (2) pour le réseau de transport passé en paramètre.

Algorithme de Ford et Fulkerson

Algorithme 1: schéma général de l'algorithme de Ford et Fulkerson

Entrées:
Un réseau de transport $G = (V_G, E_G)$
Sorties:
Un flot x
pour *tout* $(u, v) \in E_G$ **faire**
  $x(u, v) \leftarrow 0$
tant que *il existe un chemin améliorant* P *de* s_G *à* t_G **faire**
  Calculer α, l'augmentation maximum sur le chemin P
  $x(u, v) \leftarrow x(u, v) + \alpha$ pour tous les arcs (u, v) pris dans le sens direct
  $x(u, v) \leftarrow x(u, v) - \alpha$ pour tous les arcs (u, v) pris dans le sens contraire
retourner x

Soit G un réseau de transport et x un flot.

Définition 10 Chemin dans un graphe.
Un chemin d'origine u et d'extrémité v dans un graphe $G = (V_G, E_G)$ est défini par une suite finie d'arcs consécutifs, reliant u à v.

Plusieurs chemins peuvent exister entre deux sommets de G.

II.5. Définir une fonction récursive CaML `chemin(g,u,v)` qui donne la liste des sommets successifs dans le graphe G pour passer de u à v. Dans le cas où plusieurs chemins existent, la fonction doit retourner un de ces chemins. Il pourra être pertinent d'utiliser une structure de pile, à définir, pour mémoriser le chemin.

Définition 11 Chemin améliorant.
Un chemin P de s_G à t_G dans G est dit améliorant de valeur α lorsqu'il construit un flot x' à partir de x tel que :
- si l'arc $(u, v) \in P$ est pris dans le sens direct, son flot peut être augmenté de α unités : $x'(u, v) = x(u, v) + \alpha$ et il faut $x'(u, v) \leqslant c(u, v)$,
- si l'arc $(u, v) \in P$ est pris dans le sens contraire, son flot peut être diminué de α unités : $x'(u, v) = x(u, v) - \alpha$ et il faut $x'(u, v) \geqslant 0$.

II.6. Montrer que x' est toujours un flot de G. Donner la valeur de x' en fonction de x.

II.7. Soit P un chemin améliorant. Pour un arc $(u, v) \in P$, donner la capacité maximum d'augmentation α, suivant qu'il soit parcouru dans le sens direct ou contraire, pour que x' reste un flot du réseau de transport G. En déduire, pour P donné, l'augmentation maximale possible de P pour que x' reste un flot de G. L'algorithme de Ford et Fulkerson, décrit dans l'algorithme 1, reprend l'idée de chemin améliorant pour construire un flot maximum dans un réseau de transport.

II.8. Démontrer que l'algorithme de Ford et Fulkerson s'arrête effectivement.

Graphes des résidus et algorithme d'étiquetage

Tel qu'il est décrit en algorithme 1, l'algorithme n'est pas exploitable, en raison de sa complexité élevée.

II.9. Si $|V|$ est le nombre de sommets du réseau de transport et $|E|$ est le nombre d'arcs, évaluer la complexité de l'algorithme de Ford et Fulkerson, dans le cas où les capacités sont entières et bornées par $C < \infty$.

L'algorithme 1 présente un principe général qui a donné lieu à de nombreuses variantes en vue d'en diminuer la complexité. Les questions suivantes traitent l'une de ces variantes.

Définissons alors un graphe des résidus permettant de construire une variante efficace de l'algorithme de Ford et Fulkerson.

Définition 12 Graphe des résidus.
Soit $G = (V_G, E_G)$ un réseau de transport, muni d'une fonction capacité c. Soit x un flot dans G. Le graphe des résidus de G, noté $G[x]$, possède les mêmes sommets et arcs que G. Chaque arc (u, v) est valué par un résidu $r(u, v)$, donné par :

$$\forall (u,v) \in E_G \quad r(u,v) = c(u,v) - x(u,v) + x(v,u) \text{ et } r(v,u) = c(v,u) - x(v,u) + x(u,v).$$

II.10. Définir en CaML un type `residu` permettant de modéliser un graphe des résidus.

II.11. Écrire une fonction CAML `graphe_residu:reseau_transport→ flot→ residu` qui, à partir d'un réseau de transport G et d'un flot x, retourne un graphe des résidus de type `residu` spécifiant le graphe des résidus entre le réseau de transport G et le flot x. Cette fonction fera impérativement appel à une fonction récursive.

Utilisons alors $G[x]$ pour construire un algorithme dérivant de Ford et Fulkerson, appelé algorithme d'étiquetage (algorithme 2), qui est étudié dans la suite de ce problème. Dans cet algorithme, la constante `NULL` désigne l'absence de valeur associée à la variable correspondante.

Algorithme 2: algorithme d'étiquetage

Entrées:
Un réseau de transport $G = (V_G, E_G)$
Sorties:
Un flot x
pour *tout* $(u, v) \in E_G$ **faire**
  $x(u, v) \leftarrow 0$
Marque[t]← vrai
tant que *Marque[t]* **faire**
  pour tous les $u \in V_G$ **faire**
    Marque[u]← faux
    Pred[u] ← NULL
  Marque[s]←vrai
  $S \leftarrow \{s\}$
  ********** Phase 1 **********
  tant que $S \neq \emptyset$ *et Non(Marque[t])* **faire**
    Choisir $u \in S$
    $S \leftarrow S \backslash \{u\}$
    pour tous les $(u, v) \in G[x]$ **faire**
      si $r(u, v) > 0$ *et Non(Marque[v])* **alors**
        Pred[v]← u
        Marque[v]←vrai
        $S \leftarrow S \cup \{v\}$
  ********** Phase 2 **********
  si *Marque[t]* **alors**
    $P \leftarrow \{(u, v)/Pred[v] = u\}$
    $\alpha \leftarrow min\,\{r(u, v)/(u, v) \in P\}$
    `MiseAJour`$((u, v), P, x, \alpha)$
retourner x

II.12. Montrer que l'on a toujours $r(u, v) \geqslant 0$.

II.13. Montrer, à l'aide d'un exemple simple d'un réseau de transport à deux sommets, qu'il est possible d'avoir $r(u, v) > c(u, v)$.

Il n'est donc pas toujours possible d'ajouter le résidu au flot de l'arc correspondant.

II.14. Soit un chemin améliorant P de α dans $G[x]$, empruntant l'arc (u, v), de capacité $c(u, v)$ et de flot $x(u, v)$, tels que $x(u, v) \leqslant c(u, v)$ mais $x(u, v) + \alpha \geqslant c(u, v)$. Proposer une stratégie d'amélioration de P permettant d'augmenter la valeur du flot x de la quantité α, tout en conservant les contraintes (2). En déduire une interprétation du résidu. A quelle étape de l'algorithme 2 correspond cette stratégie ?

Nous nous intéressons maintenant à la phase 2 de l'algorithme 2.

II.15. Si la condition `Marque[t]` est vérifiée, que cela implique-t-il sur la recherche de chemins dans le réseau de transport ? Que représentent alors P et α ?

L'algorithme 2 poursuit son exécution tant qu'un chemin améliorant de s_G à t_G dans $G[x]$ existe. À l'arrêt, notons S l'ensemble des sommets marqués et $T = V_G \backslash S$.

II.16. Montrer que (S, T) est une s,t-coupe de G et que pour tout couple $(u, v) \in S \times T$, $r(u, v) = 0$. En déduire que :

(i). pour tout arc (u, v) de S vers T, $x(u, v) = c(u, v)$;

(ii). pour tout arc (u, v) de T vers S, $x(u, v) = 0$.

II.17. Soit x un flot de valeur ν dans G. En utilisant la question **II.2.**, montrer alors que :

$$\nu = c(S, T).$$

La valeur d'un flot maximum de s_G à t_G est donc égale à la plus petite capacité d'une s,t-coupe.

Application : problème de couplage

Soit $G = (V_G, E_G)$ un graphe non-orienté et (A, B) une partition de V_G.

Définition 13 Couplage.
Un couplage de G est un ensemble d'arêtes $E'_G \subset E_G$, tel que les arêtes de E'_G soient deux à deux non adjacentes.

En considérant une partition (A, B) de V_G, un couplage de G peut aussi être vu comme un ensemble d'arêtes $E'_G \subset E_G$, tel que toute arête de E'_G relie un sommet dans A à un sommet de B.

Dans la suite, nous recherchons un couplage de G qui contient un maximum d'arêtes (couplage maximum).

II.18. Montrer que ce problème peut se modéliser comme un problème de flot. Définir le réseau de transport correspondant et le flot associé.

II.19. Proposer une implémentation récursive en CAML de la recherche d'un couplage maximum.

Fin de l'énoncé

IMPRIMERIE NATIONALE – 16 1218 – D'après documents fournis

Travailler les chapitres de ce sujet

Prepa Help suit le programme de MP chapitre par chapitre : cours détaillé, méthodes, pièges, exercices et problèmes corrigés pas à pas, tirés des concours. L'essai gratuit donne accès aux chapitres offerts de ta filière, sans carte bancaire.

Chapitres liés à ce sujet

- Logique, ensembles et applications
- Langage OCaml : expressions, types, filtrage, récursivité
- Complexité des algorithmes itératifs et récursifs
- Spécifier, tester, prouver : terminaison et correction

Ouvrir l'essai gratuit

Scanne le code ou clique sur le bouton.

prepahelp.com/#essai

Une question sur le concours ou sur ta filière : **écrire sur WhatsApp**

Les autres années et épreuves : prepahelp.com/annales/ccinp/mp