.

Diagrammes de décisions binaires

Épreuve pratique d'algorithmique et de programmation
Concours commun des écoles normales supérieures

Durée de l'épreuve: 3 heures 30 minutes

Juillet 2009

ATTENTION !

N'oubliez en aucun cas de recopier votre u_0
à l'emplacement prévu sur votre fiche réponse

Important.

Sur votre table est indiqué un numéro u_0 qui servira d'entrée à vos programmes. Les réponses attendues sont généralement courtes et doivent être données sur la fiche réponse fournie à la fin du sujet. À la fin du sujet, vous trouverez en fait deux fiches réponses. La première est un exemple des réponses attendues pour un $\widetilde{u_0}$ particulier (précisé sur cette même fiche et que nous notons avec un tilde pour éviter toute confusion!). Cette fiche est destinée à vous aider à vérifier le résultat de vos programmes en les testant avec $\widetilde{u_0}$ au lieu de u_0. Vous indiquerez vos réponses (correspondant à votre u_0) sur la seconde et vous la remettrez à l'examinateur à la fin de l'épreuve.

En ce qui concerne la partie orale de l'examen, lorsque la description d'un algorithme est demandée, vous devez présenter son fonctionnement de façon schématique, courte et précise. Vous ne devez en aucun cas recopier le code de vos procédures!
Quand on demande la complexité en temps ou en mémoire d'un algorithme en fonction d'un paramètre n, on demande l'ordre de grandeur en fonction du paramètre, par exemple: $O(n^2)$, $O(n \log n)$,...
Il est recommandé de commencer par lancer vos programmes sur de petites valeurs des paramètres et de ***tester vos programmes sur des petits exemples que vous aurez résolus préalablement à la main ou bien à l'aide de la fiche réponse type fournie en annexe***. Enfin, il est recommandé de lire l'intégralité du sujet avant de commencer afin d'effectuer les bons choix de structures de données dès le début.

1 Introduction

Dans ce sujet on va s'intéresser aux formules logiques simples (sans quantificateur universel, ou existentiel). On va développer des algorithmes permettant de compter le nombre de valuations des variables booléennes apparaissant dans une formule qui permettent de satisfaire la dite formule. Ceci nous permettra finalement de compter le nombre de solutions pour un problème bien connu (n dames).

Définition 1 (Ensemble des booléens) *On note $\mathcal{B}$ l'ensemble des booléens. Cet ensemble est réduit à deux éléments appelés valeurs de vérité et notés respectivement Vrai et Faux ou plus communément 1 (Vrai) et 0 (Faux).*

Définition 2 (Fonction booléenne) *Une fonction booléenne f de n variables est une fonction de $\mathcal{B}^n$ à valeurs dans $\mathcal{B}$. Une telle fonction f est entièrement définie par la donnée d'une table associant à chacune des 2^n valeurs possibles pour le n-uplet $(x_1, \ldots, x_n) \in \mathcal{B}^n$ la valeur prise par f en ce point, à savoir $f(x_1, \ldots, x_n)$. On appelle une telle table, la table de vérité associée à f.*

Question à développer pendant l'oral : Combien existe-t-il de fonctions booléennes différentes à n variables ?

Définition 3 (Opérateur OU logique.) *On notera $\vee$ la fonction booléenne de deux variables définie par la table de vérité 1(a). On appellera cette fonction OU logique, car elle est prend la valeur Vrai si et seulement si son premier **ou** son second argument est égal à Vrai.*

Définition 4 (Opérateur ET logique.) *On notera $\wedge$ la fonction booléenne de deux variables définie par la table de vérité 1(b). On appellera cette fonction ET logique, car elle est prend la valeur Vrai si et seulement si son premier argument **et** son second sont égaux à Vrai.*

Définition 5 (Opérateur NON logique.) *On notera $\neg$ la fonction booléenne d'une variable définie par la table de vérité 1(c). On appellera cette fonction NON logique car elle prend la valeur Vrai si et seulement si son argument **n'**est **pas** égal à Vrai.*

$\vee$	0	1
0	0	1
1	1	1

(a) OU logique

$\wedge$	0	1
0	0	0
1	0	1

(b) ET logique

	$\neg$
0	1
1	0

(c) NON logique

FIG. 1 – Tables de vérité des opérateurs logiques OU, ET et NON

Définition 6 (Algèbre des booléens) *L'ensemble $\mathcal{B}$ muni des deux lois de composition internes $\wedge$ et $\vee$ forme une algèbre, c'est-à-dire que les propriétés suivantes sont vérifiées :*
- ***Associativité*** $(a \wedge b) \wedge c = a \wedge (b \wedge c)$ *et* $(a \vee b) \vee c = a \vee (b \vee c)$
- ***Commutativité*** $a \wedge b = b \wedge a$ *et* $a \vee b = b \vee a$

- ***Éléments neutres*** $a \vee 0 = 0 \vee a = a$ *et* $a \wedge 1 = 1 \wedge a = a$
- ***Éléments absorbants*** $a \vee 1 = 1 \vee a = 1$ *et* $a \wedge 0 = 0 \wedge a = 0$
- ***Distributivité*** $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ *et* $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

Définition 7 (Formule de logique booléenne) *On définit $\mathcal{F}_n$ l'ensemble des formules de logique booléennes à n variables $(x_1, \ldots, x_n)$ de la manière suivante :*
- *Les valeurs de vérité 0 et 1 sont des éléments de $\mathcal{F}_n$.*
- *Les n variables booléennes $\{x_1, \ldots, x_n\}$ appartiennent à $\mathcal{F}_n$.*
- *Si t_1 et t_2 sont deux éléments de $\mathcal{F}_n$ alors :*
  1. *$t_1 \wedge t_2$ est un élément de $\mathcal{F}_n$.*
  2. *$t_1 \vee t_2$ est un élément de $\mathcal{F}_n$.*
  3. *$\neg t_1$ est un élément de $\mathcal{F}_n$.*

Définition 8 (Substitution d'une variable) *Soit $F \in \mathcal{F}_n$ une formule booléenne des variables $x_1, \ldots, x_n$. La substitution de la variable x_j par la valeur de vérité v dans la formule F, opération notée $F[x_j \leftarrow v]$ est l'opération qui consiste à remplacer l'ensemble des occurrences de la variable x_j dans la formule F par la valeur de vérité v. Elle crée une nouvelle formule de logique booléenne des variables $x_1, \ldots, x_{j-1}, x_{j+1}, \ldots x_n$.*

Définition 9 (Valuation d'une formule logique) *Soit F une formule booléenne des variables $x_1, \ldots, x_n$. Une valuation $\mathcal{V} = \{x_1 \leftarrow v_1, \ldots, x_n \leftarrow v_n\}$ est la donnée d'un ensemble de substitution associées à chacune des variables $x_1, \ldots, x_n$. Une telle valuation peut être appliquée à la formule F, ce que l'on notera $F[\mathcal{V}]$. Elle est équivalente à l'application de chacune substitution $x_i \leftarrow v_i$ à F, c'est-à-dire $F[\mathcal{V}] = F[x_1 \leftarrow v_1][x_2 \leftarrow v_2] \ldots [x_n \leftarrow v_n]$. $F[\mathcal{V}]$ est donc une valeur de vérité, puisque l'ensemble des variables apparaissant initialement dans F ont été substituées.*

Définition 10 (Forme normale disjonctive) *Une formule booléenne $F(x_1, \ldots, x_n)$ est dite sous forme normale disjonctive si et seulement si elle s'écrit de la manière suivante :*

$$\begin{array}{rcl} F(x_1, \ldots, x_n) & = & \bigvee_{i=1}^{t} T_i \\ \text{avec } T_i & = & \bigwedge_{j=1}^{n} \delta_i(j) \\ \text{et } \delta_i(j) & = & 1 \text{ ou } x_j \text{ ou } \neg x_j. \end{array}$$

C'est donc la disjonction de $t \geq 1$ termes où chaque terme est lui-même la conjonction d'un certains nombre de littéraux. Chaque littéral est l'unique occurrence d'une des variables de la formule, ou de sa négation.

Question à développer pendant l'oral : Vous expliquerez comment à une formule de logique booléenne peut être associée une fonction booléenne telle que la formule et la fonction correspondent sur toutes les valuations possibles des variables. Inversement vous expliquerez comment on peut associer une formule de logique booléenne à une fonction booléenne (sous les mêmes conditions).
À partir de maintenant, et par abus de langage on utilisera indifféremment les termes formule de logique booléenne et fonction booléenne.

Définition 11 (Tautologie) *Une formule de logique booléenne est qualifiée de tautologie si et seulement si pour l'ensemble des valuations possibles de ses variables, elle prend la valeur Vrai.*

Définition 12 (Satisfiabilité) *Soit $F(x_1, \ldots, x_n)$ une formule de logique booléenne de n variables. On dit que F est satisfiable si et seulement si il existe au moins une valuation de ses variables pour laquelle la formule F prend la valeur de vérité Vrai.*

2 Génération aléatoire de formules logiques

Considérons la suite d'entiers (u_k) définie pour $k \geq 0$ par :

$$u_k = \begin{cases} \textit{votre } \mathtt{u_0} \textit{ (à reporter sur votre fiche)} & \text{si } k = 0 \\ 15\,091 \times u_{k-1} \mod 64\,007 & \text{si } k \geq 1 \end{cases}$$

Question 1 *Que valent :* ***a)*** u_{10} ***b)*** u_{100} ***c)*** u_{1000}.

Définition 13 *Pour $n \in \mathbb{N}^*$, $m \in \mathbb{N}^*$ et $p \in [0, 1]$, on note $F_{n,m,p}(x_1, \ldots, x_n)$ la formule de logique booléenne (ou indifféremment la fonction booléenne) à n variables définie par :*

- $F_{n,m,p}(x_1, \ldots, x_n) = \bigvee_{i=1}^{m} T_i(x_1, \ldots, x_n)$.
- $\forall\, i \in [1, m] \quad T_i(x_1, \ldots, x_n) = \bigwedge_{j=1}^{n} \delta(x_j, (i-1)n + j, p)$
- $\delta(x, k, p) = \begin{cases} \neg x \textit{ si } 0 \leq u_k < \dfrac{64007 \times p}{2} \\ x \textit{ si } \dfrac{64007 \times p}{2} \leq u_k < 64007 \times p \\ 1 \textit{ sinon.} \end{cases}$

Remarquez que les formules de logique ainsi générées sont déjà sous forme normale disjonctive. Cependant vous vous attacherez à concevoir des structures de données permettant de construire des formules sous une forme quelconque. On souhaite pouvoir évaluer la valeur de vérité prise par une formule logique pour une valuation donnée de ses variables. Écrivez une fonction réalisant cette tâche.

Question 2 *On note $\mathbb{V}_n$ la n-valuation $(0, 0, 1 \ldots, 1)$. Indiquer la valeur de vérité prise en ce point par les fonctions booléennes suivantes :*
a) $F_{5,1,1/2}$ ***b)*** $F_{10,2,1/3}$ ***c)*** $F_{20,20,1/3}$

On souhaite pouvoir calculer la nouvelle formule logique obtenue par substitution d'une variable par une valeur de vérité à partir d'une formule donnée en entrée. Implémentez une fonction réalisant une telle tâche. Elle prendra comme paramètres la formule sur laquelle doit s'appliquer la substitution, la variable à substituer, ainsi que la valeur de vérité qui lui sera substituée. On appliquera par ailleurs les simplifications suivantes :

- Dans une disjonction de termes, un terme se réduisant à la valeur de vérité faux ne contribue pas au résultat de la disjonction. Inversement, un terme qui se réduit à la valeur de vérité vrai rend la disjonction globalement égale à vrai.
- Dans une conjonction de littéraux, un littéral dont la valeur de vérité vaut vrai n'apparaît plus. Inversement un littéral dont la valeur de vérité vaut faux rend la conjonction égale à faux.

Question 3 *Appliquez cette fonction afin de substituer la variable x_1 par la valeur de vérité 0 (on appliquera les simplifications précédemment présentées), calculez le nombre de termes restants dans la forme normale disjonctive pour les exemples suivants :*
a) $F_{5,1,1/2}$ ***b)*** $F_{10,2,1/3}$ ***c)*** $F_{20,20,1/3}$

Une première question pertinente concernant une formule de logique consiste à déterminer une valuation permettant de satisfaire la dite formule. Ce problème est connu dans la littérature sous le nom de AnySat. Une seconde question intéressante est de connaître le nombre de valuations satisfaisant la dite formule. Ce problème est connu dans la littérature sous le nom de #Sat. Une méthode naïve pour résoudre ces deux problèmes consiste à énumérer l'ensemble des valuations possibles pour la formule concernée et à l'évaluer pour chacune d'entre elles. Pour résoudre AnySat, on s'arrête dès que l'on a trouvé une valuation convenable. Pour résoudre #Sat on compte simplement le nombre de valuation satisfaisant la formule concernée.
On souhaite résoudre ces deux problèmes. Pour cela commencez par implémenter une fonction permettant d'énumérer l'ensemble des valuations à n variables. On ordonnera ces valuations selon l'ordre lexicographique strict $\prec$ défini par :

$$\begin{cases} 0 \prec 1. \\ \forall x, y \in \mathcal{B} \quad x \preceq y \Leftrightarrow (x \prec y) \text{ ou } (x = y). \\ \forall\, x, y \in \mathcal{B}^n \quad x \prec y \Leftrightarrow \exists\, i \in \{1, \ldots, n\}, \forall\, j < i \quad x[j] \preceq y[j] \text{ et } x[i] \prec y[i]. \end{cases}$$

Question à développer pendant l'oral : Vous exposerez l'algorithme que vous avez développé. Vous en évaluerez la complexité.

Question 4 *Pour les valeurs de n et k suivantes, donnez la kième valuation à n variables dans l'ordre strict $\prec$:*
a) $n = 5, k = u_{1000} \mod 32$ ***b)*** $n = 8, k = u_{2000} \mod 256$ ***c)*** $n = 12, k = u_{3000} \mod 4096$

Question 5 *Résolvez le problème AnySat pour les propositions suivantes. On retournera la plus petite valuation satisfaisant la formule (au sens de l'ordre $\prec$) :*
a) $F_{5,1,1/2}$ ***b)*** $F_{10,2,1/3}$ ***c)*** $F_{20,20,1/3}$

Question 6 *Résolvez le problème #Sat pour les propositions suivantes :*
a) $F_{5,1,1/2}$ ***b)*** $F_{10,2,1/3}$ ***c)*** $F_{20,20,1/3}$

L'efficacité de cette méthode n'est pas satisfaisante. Nous allons développer dans la suite des structures de données plus complexes permettant de répondre plus efficacement aux problèmes soulevés précédemment.

3 Table de hachage

Nous allons avoir besoin de structures de données plus efficaces que les simples listes afin de retrouver efficacement des éléments possédant certaines caractéristiques. Il existe de nombreuses structures de données adaptées à ce genre de problématique. Nous allons opter pour des tables de hachage qui présentent un compromis intéressant entre performance et difficulté d'implémentation. Une table de hachage est une structure de données qui permet d'associer un ensemble d'éléments à des clés. Nous considérerons ici des tables

de hachage dont les clés sont des n-uplet d'entiers positifs (n est fixé pour une table donnée). Les éléments pourront être d'un type quelconque mais on fera l'hypothèse que tous ont le même type. Une table de hachage est caractérisée par un paramètre p fixé à sa création. Ce paramètre correspond à la taille d'un tableau alloué à la création de la table de hachage. Chaque élément du tableau est une liste (initialement vide) qui servira à stocker un ou plusieurs des éléments stockés dans la table de hachage. Plus précisément chacune de ces listes stocke des couples formés d'une clé et de l'élément qui lui est associé. La figure 2 illustre un exemple de table de hachage pour lequel $p = 4$ et qui contient 6 éléments.

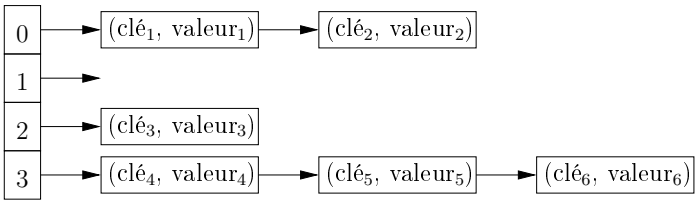


FIG. 2 – Une table de hachage ($p = 4$). La seconde liste est vide.

Soit T une table de hachage stockant des éléments de type t. T supporte deux opérations :

1. **put**(clé, élément) : insère dans la table un élément.
2. t **get**(clé) : retourne l'élément associé à la clé passée en paramètre (s'il existe, sinon une valeur par défaut indiquant qu'il n'y a aucun élément associé à la clé).

Techniquement la fonction **put** est implémentée de la manière suivante. Le n-uplet que représente la clé est tout d'abord transformé en un entier i compris entre 0 et $p - 1$. À l'index i dans le tableau associé à T, correspond une liste. On parcourt cette liste afin de vérifier s'il y a déjà un élément associé à la clé passée en paramètre. Deux cas peuvent survenir :

1. il n'y a aucun élément dans la liste correspondant à la clé passée en paramètre. Dans ce cas, on insère un nouveau couple formé de la clé et de l'élément associé dans la liste concernée.
2. sinon il y a déjà un élément associé à la clé passée en paramètre. Dans ce cas, on remplace l'élément précédemment associé à cette clé par celui passé en paramètre.

La fonction **get** fonctionne de manière analogue. Elle calcule de la même manière l'index i associé à la clé puis recherche dans la liste correspondante s'il existe un élément correspondant à la clé passée en paramètre et le retourne.
Il nous reste à expliquer comment on dérive un index dans le tableau depuis une clé. On définit pour cela la fonction h_n qui associe à une clé $c = (x_1, \ldots, x_n)$ un index défini de la manière suivante :

$$h_n(c) = \sum_{k=1}^{n} u_{x_k} \mod p.$$

Question 7 *Quelles valeurs renvoient la fonction h_n pour les n-uplets suivants et pour $p = 100$:*
a) $[u_0, u_{10}, u_{20}]$ ***b)*** $[u_{10}, u_{20}, u_{30}]$ ***c)*** $[u_0, u_{10}, u_{20}, u_{30}]$

Implémentez une table de hachage permettant de stocker des éléments de type quelconque, et dont les clés sont des n-uplets d'entiers. Afin de rendre l'implémentation suffisamment générale pour ne pas avoir à la modifier en fonction de n, on pourra coder un n-uplet d'entiers comme une liste d'entiers.
Afin de tester votre implémentation vous insérerez t éléments dans une liste de hachage pour laquelle $p = 100$. Pour $1 \leq k \leq t$, la $k$ième clé sera un couple d'entiers de la forme (u_{2k}, u_{2k+1}). La valeur associée sera simplement k.

Question 8 *Donnez les longueurs minimale, moyenne, maximale des p listes composant la table de hachage après avoir insérer le nombre d'éléments indiqué dans une table de hachage initialement vide.*
a) 100 ***b)*** 1000 ***c)*** 10000

Question à développer pendant l'oral : Commentez les résultats obtenus à la question précédente. Que cela vous inspire-t-il ? Quel gain peut-on espérer lorsque le nombre d'éléments insérés dans la table de hachage n'est pas trop grand par rapport à p ? Quelle amélioration envisageriez-vous lorsque le nombre d'éléments insérés est largement supérieur à p ?

4 Diagrammes de décision binaire

Définition 14 (Diagramme de décision binaire) *Soit F une formule booléenne de n variables $x_1, \ldots, x_n$. Un diagramme de décision binaire $\mathcal{B}$ associé à la formule F est une structure de données arborescente. Elle capture l'ensemble des valuations associées à la formule F. Comme son nom l'indique il s'agit d'un arbre binaire. Les feuilles de l'arbre correspondent à des valeurs de vérité (0 ou 1). À chaque nœud ν interne est associé une variable booléenne, notée $Var(\nu)$. Ce nœud possède deux fils. On notera le fils gauche (resp. droit) $\nu \to 0$ (resp. $\nu \to 1$). Un chemin π partant de la racine r de $\mathcal{B}$ jusqu'à une feuille f vérifie les propriétés suivantes :*

$$\begin{cases} \pi = [\nu_1 = r, \ldots, \nu_l = f] \text{ avec } 1 \leq l \leq n+1 \\ \forall i, j \in [1, l] \quad i \neq j \Rightarrow Var(\nu_i) \neq Var(\nu_j). \end{cases}$$

L'arbre est donc de hauteur au plus $n+1$, et les variables qui apparaissent le long de tout chemin sont deux à deux distinctes. Par ailleurs, $\mathcal{B}$ vérifie la propriété récursive suivante : soit $x_r = Var(r)$ la variable associée à la racine de $\mathcal{B}$, le sous-arbre $r \to 0$ (resp. $r \to 1$) associé à la racine de $\mathcal{B}$ est un diagramme de décision binaire associé à la formule $F[x_r \leftarrow 0]$ (resp. $F[x_r \leftarrow 1]$).

La figure 3 illustre un diagramme de décision binaire pour la formule de logique $(x_1 \wedge x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$. Notez que le long de tous les chemins de la racine aux feuilles, les variables x_1, x_2 et x_3 n'apparaissent qu'au plus une fois.

Propriété 15 *Un chemin π de la racine r à une feuille f constitue une valuation $\mathcal{V}$ de la formule logique F. Si le chemin est de longueur exactement n chacune des variables de F apparaît une et une seule fois le long du chemin π avec une valeur de vérité donnée. La feuille f correspond à la valeur de vérité de la formule pour la valuation $\mathcal{V}$, c'est-à-dire $F[\mathcal{V}]$. Dans le cas où le chemin est de longueur strictement plus petite que n, seul*

un sous-ensemble des variables de la formule apparaissent le long du chemin. Le chemin π est donc associé à une valuation partielle. Ceci ne peut arriver que si les valeurs de vérité associées aux variables manquantes n'influencent pas la valeur de vérité prise par la formule. Il s'agit là d'une optimisation permettant de compresser la taille du diagramme pour la formule considérée.

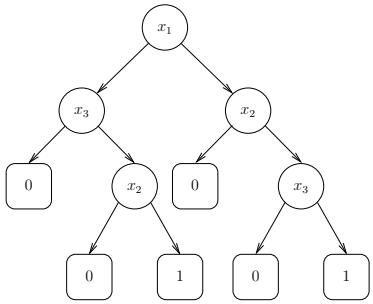


FIG. 3 – Un diagramme de décision binaire associé à la formule $(x_1 \wedge x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$.

Notez que sur la figure 3 le chemin correspondant à la valuation partielle $[x_1 \leftarrow 1][x_2 \leftarrow 0]$ se termine sans que la variable x_3 n'apparaisse. Cela traduit bien le fait que la valuation de x_3 dans ce cas n'a aucune importance. Qu'elle vaille 0 ou 1 la formule s'évalue à 0.

Définition 16 (Diagramme de décision binaire ordonné) *Soit F une formule booléenne de n variables $x_1, \ldots, x_n$. Un diagramme de décision binaire ordonné associé à F est un diagramme de décision binaire associé à F vérifiant la propriété supplémentaire suivante. Le long de tout chemin partant de la racine à une feuille, l'ordre dans lequel apparaissent les variables associées aux nœuds est unique. Plus formellement, soient deux chemins $\pi_1 = [\nu_1^1 = r, \ldots, \nu_{l_1}^1 = f_1]$ et $\pi_2 = [\nu_1^2 = r, \ldots, \nu_{l_2}^2 = f_2]$ alors :*

$$\forall\, i_1, i_2, j_1, j_2 \quad \textit{tels que} \quad (i_1 < j_1) \textit{ et } Var(\nu_{i_1}^1) = Var(\nu_{i_2}^2) \textit{ et } Var(\nu_{j_1}^1) = Var(\nu_{j_2}^2) \Rightarrow i_2 < j_2.$$

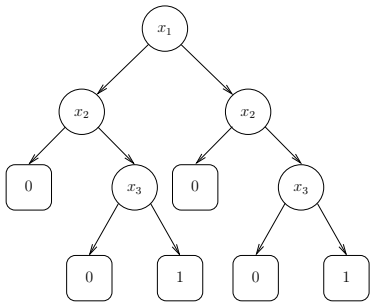


FIG. 4 – Un diagramme de décision binaire ordonné associé à la formule $(x_1 \wedge x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$.

Définition 17 (Ordre canonique) *Pour n variables $(x_1, \ldots, x_n)$ on appellera ordre canonique noté $<$ l'ordre des variables suivant : $x_1 < x_2 < \ldots < x_n$.*

La figure 4 illustre un diagramme de décision binaire ordonné pour l'ordre canonique associé à la formule de $(x_1 \wedge x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$.

Définition 18 (Diagrammes de décision binaire isomorphes) *Soient $\mathcal{B}_1$ et $\mathcal{B}_2$ deux diagrammes de décision binaire. $\mathcal{B}_1$ et $\mathcal{B}_2$ sont isomorphes, ce que l'on notera $\mathcal{B}_1 \equiv \mathcal{B}_2$, si et seulement si :*

- *$\mathcal{B}_1$ et $\mathcal{B}_2$ sont réduits à un unique nœud feuille correspondant à la même valeur de vérité (0 ou 1).*
- *ou $\mathcal{B}_1$ et $\mathcal{B}_2$ sont deux arbres tels que leurs racines respectives r_1 et r_2 vérifient :*
  - *$Var(r_1) = Var(r_2)$*
  - *$r_1 \rightarrow 0 \equiv r_2 \rightarrow 0$*
  - *$r_1 \rightarrow 1 \equiv r_2 \rightarrow 1$*

Question à développer pendant l'oral : Développez (sur le papier) un algorithme qui décide si deux arbres de décision binaires ordonnés sont isomorphes. Quelle est la complexité en pire cas de cet algorithme pour des arbres de hauteur n ?

Définition 19 (Diagramme de décision binaire ordonné et réduit) *Soit F une formule booléenne de n variables $x_1, \ldots, x_n$. Un diagramme de décision binaire ordonné et réduit associé à F est un diagramme de décision binaire ordonné associé à F vérifiant la propriété supplémentaire suivante :*

1. *tous les sous-arbres isomorphes entre eux n'apparaissent qu'une seule fois dans l'arbre.*
2. *tous les nœuds dont les deux fils sont isomorphes sont éliminés.*

Le diagramme obtenu dépend clairement de l'ordre d'apparition des variables de la racine aux feuilles.

La figure 5 illustre le diagramme de décision binaire ordonné et réduit pour l'ordre canonique associé à la formule de $(x_1 \wedge x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$. Remarquez comment le fait que dans la formule concernée, la valeur de la variable x_1 n'a aucune influence a été détecté.

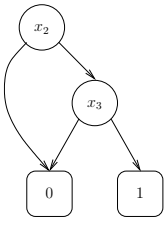


FIG. 5 – Le diagramme de décision binaire ordonné et réduit (pour l'ordre canonique) associé à la formule $(x_1 \wedge x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$.

Question à développer pendant l'oral : Vous justifierez les raisons qui expliquent pourquoi pour une fonction booléenne donnée, il existe un unique diagramme de décision

binaire ordonné et réduit lui correspondant, dès lors qu'un ordre sur les variables a été choisi.

On souhaite construire un diagramme de décision binaire et ordonné et réduit (pour l'ordre canonique) à partir d'une formule logique donnée en entrée. Si l'on veut obtenir une procédure efficace réalisant cette tâche, cela nécessite de pouvoir tester rapidement si deux sous-arbres sont isomorphes. Pour cela, nous allons adopter la stratégie suivante. Chaque nœud de l'arbre se verra attribué un identifiant unique (un entier). Les deux feuilles possibles correspondant aux deux valeurs de vérité (0 et 1) auront par convention les identifiants 0 et 1. La procédure permettant de décider si deux sous-arbres sont isomorphes se verra donc ramenée à la comparaison :

1. des variables associées aux deux racines ;
2. des identifiants des fils gauche et droit des deux arbres.

Cela nécessite cependant de pouvoir retrouver rapidement si un nœud possédant certaines caractéristiques (variable associée à la racine, identifiants des fils gauche et droit) existe déjà dans l'arbre. Implémentez une fonction réalisant une telle recherche. À cet effet on utilisera une table de hachage [1] dont la clé est le triplet constitué de :

1. le numéro de la variable associée à la racine ;
2. l'identifiant associé au fils gauche ;
3. l'identifiant associé au fils droit.

Une fois cette fonction réalisée, implémentez une fonction permettant la construction d'un nouveau nœud dans l'arbre. Cette fonction devra attribuer un nouvel identifiant au nœud, lorsque celui-ci n'existait pas déjà dans le diagramme de décision puis l'insérer dans le diagramme, ou bien retourner l'identifiant du nœud si celui-ci existait déjà. On en déduira un algorithme permettant la construction d'un diagramme de décision binaire ordonné et réduit à partir d'une formule de logique booléenne.

Question à développer pendant l'oral : Vous exposerez rapidement l'algorithme que vous avez développé.

Question 9 *Vous calculerez le diagramme de décision ordonné et réduit pour l'ordre canonique des formules suivantes. Vous donnerez le nombre de nœuds de l'arbre obtenu* [2] *:*
a) $F_{25,10,9/10}$ ***b)*** $F_{25,20,9/10}$ ***c)*** $F_{25,30,9/10}$

5 Calculs de satisfiabilité

5.1 Problème AnySat

Écrivez une fonction résolvant le problème AnySat et qui retourne la plus petite valuation au sens de l'ordre $\prec$ en utilisant le diagramme de décision binaire ordonné et réduit.

Question 10 *Vous résoudrez le problème AnySat pour les fonctions suivantes :*
a) $F_{25,10,9/10}$ ***b)*** $F_{25,20,9/10}$ ***c)*** $F_{25,30,9/10}$

[1] On pourra prendre $p = 10000$.
[2] L'arbre de la figure 5 possède 4 nœuds.

(a) Solution 1 (b) Solution 2

FIG. 6 – Les deux solutions au problème des quatre dames.

5.2 Problème #Sat

Soit F la formule de logique $(x_1 \wedge x_2 \wedge x_3 \wedge x_4) \vee (x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4) \vee (\neg x_1 \wedge x_2 \wedge x_3 \wedge x_4) \vee (\neg x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4)$. Calculez l'arbre de décision binaire ordonné et réduit associé à F lorsque l'on prend comme ordre des variables l'ordre canonique.

Question à développer pendant l'oral : Résolvez (manuellement) le problème #Sat pour cette formule. Vous exposerez le raisonnement que vous avez suivi.

En déduire une procédure qui sur l'entrée d'un diagramme de décision binaire réduit et ordonné (par l'ordre canonique) résout le problème #Sat. Vous penserez à vérifier l'égalité entre le résultat retourné par votre fonction et celui obtenu manuellement sur l'exemple précédent ainsi que les résultats obtenus à la question 6.

Question 11 *Vous résoudrez le problème #Sat pour les fonctions suivantes :*
a) $F_{25,10,9/10}$ ***b)*** $F_{25,20,9/10}$ ***c)*** $F_{25,30,9/10}$

Question à développer pendant l'oral : Vous justifierez pourquoi la complexité en temps de cet algorithme est toujours meilleure que celle de l'algorithme naïf que vous avez développé à la question 6.

6 Application au jeu des n-dames

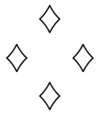
Le but du problème des $n-$dames, est de placer n dames d'un jeu d'échecs sur un échiquier de $n \times n$ cases sans que les dames ne puissent se menacer mutuellement, conformément aux règles du jeu d'échecs[3] (la couleur des pièces étant ignorée). Par conséquent, deux dames ne devraient jamais partager la même rangée, colonne, ou diagonale. L'instance de ce problème pour $n = 4$ possède deux solutions (si l'on ne tient pas compte d'éventuelles symétrie). Ces deux solutions sont illustrées par la figure 6. Attention notez bien que sur cette figure la couleur des pièces n'a été respectée que pour des raisons de lisibilité, dans le problème que l'on se propose d'étudier, elle n'intervient pas.
On souhaite compter le nombre solutions distinctes de ce problème pour des valeurs de n variant de 1 à 7. Pour cela on se propose de modéliser une solution du problème comme une valuation satisfaisant une formule de logique donnée.

[3]Une dame peut se déplacer sur la ligne, la colonne ainsi que les deux diagonales issues de sa position. Elle menace donc toute pièce située sur la même ligne, colonne ou diagonales qu'elle.

Proposez une formule de logique dont les valuations la satisfaisant sont des codages d'une solution au jeu des n-dames. On pourra introduire des variables booléennes du type $x_{i,j}$ qui valent vrai si et seulement si une dame est positionnée en position (i, j) sur l'échiquier.

Question à développer pendant l'oral : Vous développerez à l'oral la construction de cette formule de logique.

Question 12 *Donnez le nombre de solutions au jeu des n-dame pour n variant de* 5 *à* 8.



Fiche réponse type: Diagrammes de décisions binaires

$\widetilde{u_0}$: **5**

Question 1

a) **u10 = 55 175**

b) **u100 = 35 004**

c) **u1000 = 47 928**

Question 2

a) **1**

b) **0**

c) **1**

Question 3

a) **1**

b) **1**

c) **17**

Question 4

a) **[1,1,0,0,0]**

b) **[1,1,0,0,1,0,1,1]**

c) **[1,0,0,1,1,0,1,1,0,1,1,1]**

Question 5

a) **[0,0,0,0,0]**

b) **[0,1,0,1,1,0,0,0,0,0]**

c) **[12*0,1,0,0,0,1,0,0,0]**

Question 6

a) **8**

b) **192**

c) **353598**

Question 7

a) **81**

b) **73**

c) **37**

Question 8

a) **min=0 moy=1 max=4**

b) **min=3 moy=10 max=19**

c) **min=71 avg=100 max=124**

Question 9

a) **990**

b) **1842**

c) **2546**

Question 10

a) **[7*0,1,4*0,1,0,4*1,0,1,1,0,1,1,1]**

b) **idem a)**

c) **idem a)**

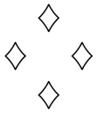
Question 11

a) **104**

b) **238**

c) **304**

Question 12



Fiche réponse: Diagrammes de décisions binaires

Nom, prénom, u_0: .....

Question 1

a)

b)

c)

Question 2

a)

b)

c)

Question 3

a)

b)

c)

Question 4

a)

b)

c)

Question 5

a)

b)

c)

Question 6

a)

b)

c)

Question 7

a)

b)

c)

Question 8

a)

b)

c)

Question 9

a)

b)

c)

Question 10

a)

b)

c)

Question 11

a)

b)

c)

Question 12