Utbm mt42 A12 **Examen final** **Durée 2h**

Exercice 1 *Logique*

1.1 *Equivalence logique*

Deux formules du calcul des propositions F_1 et F_2 sont dites équivalentes, et on note $F_1 \equiv F_2$, si $F_1 \models F_2$ et $F_2 \models F_1$, où $\models$ dénote la conséquence logique.

- Montrez que si $F_1 \equiv F_2$ alors F_1 et F_2 ont la même table de vérité.

1.2 *Forme normale disjonctive*

Un monôme conjonctif est une conjonction de littéraux, un littéral étant soit une variable, soit une variable complémentée. Par exemple, $p \wedge \neg q \wedge r$ et $\neg p \wedge q$ sont des monômes conjonctifs.

Une formule du calcul des propositions F est en forme normale disjontive (*FND*) si F s'écrit comme une disjonction de monômes conjonctifs.

Par exemple, la formule:

$$(\neg p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge \neg r) \vee (p \wedge r)$$

est en *FND*.

En revanche,

$$\neg(p \wedge \neg q) \vee (\neg r \wedge \neg p)$$

n'est pas en *FND*, à cause de la négation qui précède le premier monôme.

Etant donnée une formule F_1 du calcul des propositions, il existe toujours une formule F_2 en *FND* telle que $F_1 \equiv F_2$.

- Proposez une méthode pour, étant donnée $F1$, trouver F_2 en *FND* et telle que $F_1 \equiv F_2$.
  Suggestion : Utiliser la table de vérité ou autre représentation semblable.
- Appliquer votre méthode à la formule $\neg(p \rightarrow q) \wedge (q \rightarrow (\neg r \wedge p))$ pour trouver une formule équivalente en *FND*.

1.3 *FND et satisfiabilité*

Il est très facile de savoir si une formule en *FND* est satisfiable, en examinant ses monômes conjonctifs.

- Proposez une condition nécessaire et suffisante pour qu'une formule en *FND* soit satisfiable, en justifiant votre idée.
- Montrez que si $F_1 \equiv F_2$, alors F_1 est satisfiable si et seulement si F_2 est satisfiable.
- A partir de tout ce qui précède, proposez les étapes d'une méthode de décision de la satisfiabilité qui s'applique à toute formule du calcul de propositions.

.../...

Exercice 2 *Etude de complexité: aspects pratiques*

On considère trois algorithmes $algo1\,(n \rightarrow res)$, $algo2\,(n \rightarrow res)$ et $algo3\,(n \rightarrow res)$ fournissant un même résultat res lorsqu'ils travaillent sur des données de taille n.

On suppose qu'une étude théorique préalable a prouvé que leurs temps de traitement d'une donnée de taille n, définis respectivement par $T_1(n)$, $T_2(n)$ et $T_3(n)$, vérifient $T_1(n) \in \Theta(n)$, $T_2(n) \in \Theta(n\ln(n))$ et $T_3(n) \in \Theta(n^2)$.

On suppose enfin que pour les trois algorithmes concernés on a: $T_1(100) = T_2(100) = T_3(100) = 1$".

2.1 Déduire des hypothèses précédentes les temps d'exécution présumés de $T_1(n')$, $T_2(n')$ et $T_3(n')$ en fonction de l'entier n' supposé "grand".

Application numérique:

Fournir $T_1(n')$, $T_2(n')$ et $T_3(n')$ pour $n' = 15000$.

2.2 On suppose disposer, lors d'un processus temps réel, d'un temps de calcul disponible de A secondes.

a) Caractériser, sous les hypothèses précédentes, les tailles maximales n_1, n_2 et n_3 des données que pourront traiter respectivement les trois algoritmes concernés, dans le temps imparti A.

b) *Application numérique:*

Déterminer les tailles maximales n_1, n_2 et n_3 pour $A = 5$".

2.3 Un complément d'étude expérimentale sur les trois algorithmes considérés a en fait permis de mettre en évidence que $T_1(n) \simeq 3000n$, $T_2(n) \simeq 170n\ln(n)$ et $T_3(n) \simeq 10n^2$.

a) Résoudre dans $\mathbb{N}$ l'inéquation $T_1(n) \leq T_3(n)$. Représenter les fonctions T_1 et T_3 sur un même graphique. Que pourra en déduire l'informaticien utilisateur des algorithmes $algo1\,(\,)$ et $algo3\,(\,)$?

b) Mêmes questions adaptées, pour la comparaison des algorithmes $algo1\,(\,)$ et $algo2\,(\,)$.

Exercice 3 *Etude de complexité d'un algorithme récursif*

N.B: On pourra admettre tout résultat intermédiaire pour poursuivre la résolution de l'exercice.

On considère un algorithme récursif $quicksort\,(n)$ capable de trier des données de taille n, où n est élément de $\mathbb{N}^*$. On note $T(n)$ le nombre de comparaisons élémentaires nécessaires. L'analyse détaillée de l'algorithme $quicksort\,(\,)$ a permis de montrer que:

$$\begin{array}{|ll|}\hline & T(0) = T(1) = 0; \qquad T(2) = 1 \\ \forall n \geq 2 & T(n) = n - 1 + \frac{1}{n}\sum_{r=1}^{n}\left[T(r-1) + T(n-r)\right] \\ \hline\end{array}$$

3.1 Montrer que: $\forall n \geq 2 \quad T(n) = n - 1 + \frac{2}{n}\sum_{r=1}^{n-1} T(r)$.

.../...

3.2 On pose $v_1 = 0$ et si $n \geq 2$, $v_n = \sum_{r=1}^{n-1} T(r)$.

a) Montrer que :

$$\forall n \geq 1 \quad T(n) = v_{n+1} - v_n \quad \text{et} \quad v_{n+1} - \frac{n+2}{n} v_n = n - 1.$$

b) Montrer que:

$$\forall n \geq 1 \quad \frac{v_{n+1}}{(n+1)(n+2)} - \frac{v_n}{n(n+1)} = \frac{3}{n+2} - \frac{2}{n+1}.$$

c) En déduire que :

$$\forall n \geq 3 \quad T(n) = -4n + 2(n+1)\left(\sum_{r=1}^{n} \frac{1}{r}\right).$$

3.3 *Lemmes techniques:*

N.B: Le résultat final pourra être admis pour atteindre la conclusion de l'exercice.

Soit n quelconque de $\mathbb{N}^*$.

- Montrer que : $\quad \frac{1}{n+1} \leq \int_n^{n+1} \frac{dx}{x} \leq \frac{1}{n}$;
- En déduire que : $\quad \int_1^{n+1} \frac{dx}{x} \leq \left(\sum_{r=1}^{n} \frac{1}{r}\right) \leq \int_1^n \frac{dx}{x} + 1$;
- puis que : $\quad \ln(n+1) \leq \left(\sum_{r=1}^{n} \frac{1}{r}\right) \leq \ln(n) + 1$;
- et enfin :

$$\left(\sum_{r=1}^{n} \frac{1}{r}\right) \in \Theta\left(\ln(n)\right).$$

3.5 En déduire la complexité du tri étudié.