M1 Informatique | Université de Nantes
X1II030 | 2017/2018

Contrôle continu
Complexité et Algorithmes

Novembre 2017

Durée : 1h30. Documents de CM/TD autorisés. Il est important de toujours *bien justifier vos réponses. Le barème est indicatif. De plus, dans tout l'énoncé, sauf mention contraire :*
- *Les complexités sont demandées sous la forme "notation de Landau"*
- *Les tableaux et matrices sont indicés à partir de 1*

Exercice 1 (Analyse d'algorithmes – 3 points)

On suppose avoir deux algorithmes A_1 et A_2. Si la taille des données est n :
- A_1 est en $\Omega((\log n)^2)$ et en $O(n)$
- A_2 est en $\Omega(\sqrt{n})$ et en $O(n)$

On se donne maintenant les 2 algorithmes suivants, appelés X et Y.

```
Algorithme X
  Début
    Pour i de 1 à n*log(n) faire
      Exécuter A1;
    Fin Pour

    Pour i de 1 à n faire
      Exécuter A2;
    Fin Pour
  Fin
```

```
Algorithme Y
  Début
    int k:=1;
    Tant que k<=n faire
      Exécuter A2;
      k:=3*k;
    FinTantQue

    Pour i de 1 à n faire
      Si est_pair(i) alors
        Exécuter A1;
      Fin Si
    Fin Pour
  Fin
```

La fonction `est_pair(int p)` renvoie `Vrai` si `p` est pair, et `Faux` sinon. Elle s'exécute en $O(1)$.

1. Quelles sont les complexités au mieux et au pire de X ? Justifier.
2. Quelles sont les complexités au mieux et au pire de Y ? Justifier.

On se donne maintenant la fonction ci-dessous.

```
boolean anotheralgo(T:tableau d'entiers ; deb:entier, fin:entier)
   Début
     int p,q;

     Si deb >= fin alors
       return True;
     FinSi

     p:=(2deb+fin)/3;
     q:=(deb+2fin)/3;

     return (anotheralgo(T,deb,p) && anotheralgo(T,q,fin));
   Fin
```

3. En supposant `deb` $= 1$ et `fin` $= 100$, quelles sont les valeurs de `p` et `q` à la première itération ?
4. Donner, en détaillant vos calculs, la complexité de `anotheralgo` quand celui-ci est appelé sur un tableau T de taille n, avec `deb` $= 1$ et `fin` $= n$.

Exercice 2 (Questions de cours – 2 points)

1. Pourquoi les problèmes NP-complets sont-ils dans NP ?
2. Pourquoi les problèmes NP-complets sont-ils considérés comme les plus difficiles de la classe NP ?

Voici la définition du problème de décision MIN-COL-TREES. On rappelle qu'un coloration *propre* des sommets d'un graphe est une coloration dans laquelle deux sommets voisins ne peuvent pas avoir la même couleur.

> MIN-COL-TREES
> *Instance* : Un arbre $T = (V, E)$.
> *Solution* : Une coloration propre des sommets de T.
> *Mesure* : Le nombre de couleurs utilisées.

3. Montrer que le problème MIN-COL-TREES est dans P. Pour cela, décrire en français (pas de pseudo-code) un algorithme qui résout MIN-COL-TREES, montrer qu'il est correct, et indiquer sa complexité.

Exercice 3 (Calories – 8 points)

On possède une matrice carrée C à k lignes et k colonnes, contenant des informations sur le nombre de calories de k aliments différents. Chaque case $C[i][j]$ de C (qui se lit "ligne i, colonne j") contient une des trois valeurs +, = ou − selon la règle suivante :

(a) $C[i][j]$ = '+' si l'aliment i a strictement plus de calories que l'aliment j.
(b) $C[i][j]$ = '=' si i et j ont le même nombre de calories.
(c) $C[i][j]$ = '−' si i a strictement moins de calories que j.

On se donne la fonction `analyse` ci-après, où le type `mat` représente les matrices contenant les valeurs −, = et +.

```
int analyse(C:mat; k:int){
  var i,j:int; var b:boolean;

  i:=1; b:=FAUX;

  Tant que (i<=k) et (b=FAUX) faire
     b:=VRAI;
     j:=1;

     Tant que (j<=k) et (b=VRAI)faire
        Si C[i][j]='-' alors
           b=FAUX;
        FinSi

        j:=j+1;
     FinTtque

     i:=i+1;
  FinTtque

  return (i-1);
}
```

1. Simuler `analyse`(C,k), où C est la matrice ci-dessous, et $k = 5$.

C	1	2	3	4	5
1	=	+	−	=	+
2	−	=	−	−	+
3	+	+	=	+	+
4	=	+	−	=	+
5	−	−	−	−	=

2. A quelle question répond `analyse` (on demande ici une réponse très précise) ? Justifier.
3. Quelle est la complexité au mieux de `analyse` ? Quelle est la forme des données correspondante ?
4. Quelle est la complexité au pire de `analyse` ? Quelle est la forme des données correspondante ?

On souhaite maintenant écrire une fonction `analyse-rapide`, qui fait exactement la même chose qu'`analyse`, mais dont la complexité au pire est **linéaire**. Pour cela, on s'inspire de l'idée représentée dans la matrice C' (page suivante), où *seules les valeurs entourées ont été lues* pour déterminer le résultat.

C'	1	2	3	4	5	6
1	=	+	=	–	–	–
2	–	=	–	–	–	–
3	=	+	=	–	–	–
4	+	+	+	=	–	=
5	+	+	+	+	=	+
6	+	+	+	=	–	=

5. Indiquer en français la méthode à suivre.
6. Écrire en pseudo-code la fonction `analyse-rapide`.
7. Démontrer que l'algorithme `analyse-rapide` est correct.
8. Quelle est la complexité au mieux de `analyse-rapide` ? Quelle conclusion peut-on en tirer ?
9. Démontrer que `analyse-rapide` est *optimal*, c'est-à-dire que tout algorithme répondant à la question aura une complexité au moins égale à celle de `analyse-rapide`.

Exercice 4 (Le problème NOT-ALL-EQUAL-SAT (ou NAE-SAT) - 7 points)

On rappelle la définition du problème de décision SAT.

> SAT
> *Instance* : Une formule booléenne Φ en forme normale conjonctive (FNC), construite à partir d'un ensemble $X = \{x_1, x_2 \dots x_n\}$ de n variables.
> *Question* : Existe-t-il une affectation (`Vrai`/`Faux`) des variables de X qui rend Φ satisfiable ?

On dit qu'une affectation `Vrai`/`Faux` à des variables rend une clause *hybride* si cette clause contient au moins un élément qui s'évalue à `Vrai` **et** un autre qui s'évalue à `Faux`. Par exemple, si $C = (x_1 \vee \overline{x_2} \vee \overline{x_3})$, alors :

(a) $(x_1, x_2, x_3) = (\texttt{Vrai}, \texttt{Vrai}, \texttt{Vrai})$ rend $C = (\texttt{V} \vee \texttt{F} \vee \texttt{F})$ hybride ;

(b) $(x_1, x_2, x_3) = (\texttt{Faux}, \texttt{Vrai}, \texttt{Vrai})$ ne rend pas $C = (\texttt{F} \vee \texttt{F} \vee \texttt{F})$ hybride.

Soit maintenant le problème de décision suivant, appelé NOT-ALL-EQUAL-SAT (ou NAE-SAT).

> NOT-ALL-EQUAL-SAT (NAE-SAT)
> *Instance* : Une formule booléenne Φ' en FNC, construite à partir d'un ensemble $X' = \{x'_1, x'_2 \dots x'_n\}$ de n variables.
> *Question* : Existe-t-il une affectation (`Vrai`/`Faux`) des variables de X' qui rend Φ' satisfiable **et** pour laquelle chaque clause est hybride ?

1. Quelle est la réponse à NAE-SAT sur l'expression suivante : $\Phi_1 = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3})$?
2. Quelle est la réponse à NAE-SAT sur l'expression suivante : $\Phi_2 = (x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3})$?
3. Montrer que NAE-SAT est dans `NP`.

On veut montrer que NAE-SAT est `NP`-complet, en réduisant depuis SAT. Soit Φ une instance quelconque de SAT construite à partir d'un ensemble X de variables. On crée alors une instance Φ' de NAE-SAT comme suit :

— on rajoute une nouvelle variable z à l'ensemble X (donc $X' = X \cup \{z\}$)

— Φ' est obtenue à partir de Φ en rajoutant le littéral z à chacune des clauses de Φ

4. Illustrer la réduction sur l'expression Φ_2 de la Question 2.

Supposons d'abord que la réponse à SAT soit OUI, c'est-à-dire qu'il existe une affection `Vrai`/`Faux` des variables de X qui satisfait Φ.

5. Indiquer une affectation `Vrai`/`Faux` des variables de X' qui satisfait Φ' pour le problème NAE-SAT. Justifier.

Supposons maintenant que la réponse à NAE-SAT soit OUI, c'est-à-dire qu'il existe une affection `Vrai`/`Faux` des variables de X' qui satisfait Φ', et supposons que dans cette affectation, $z = \texttt{Faux}$.

6. Indiquer une affectation `Vrai`/`Faux` des variables de X qui satisfait Φ pour le problème `SAT`. Justifier.

Supposons maintenant que la réponse à NAE-SAT soit OUI, c'est-à-dire qu'il existe une affection `Vrai`/`Faux` des variables de X' qui satisfait Φ', et supposons que dans cette affectation, $z = \texttt{Vrai}$.

7. Indiquer une affectation `Vrai`/`Faux` des variables de X qui satisfait Φ pour le problème SAT. Justifier.
8. Donner toutes les raisons pour lesquelles NAE-SAT est `NP`-complet.