M1 Informatique
X1II030

Université de Nantes
2019/2020

Nom(s):	Prénom(s):	Groupe:

Contrôle continu
Complexité et Algorithmes

Novembre 2019

Durée: 1h30. Documents de CM/TD autorisés, calculatrices interdites. Le barème est indicatif. Dans tout l'énoncé, les complexités sont demandées sous la forme "notation de Landau", et les tableaux sont indicés à partir de 1.

Exercice 1 (Questions diverses – 4 points)

On suppose avoir deux algorithmes A_1 et A_2. Si la taille des données est n:

- A_1 est au mieux en $\Omega(\log n)$ et au pire en $O((\log n)^2)$
- A_2 est au mieux en $O(1)$ et au pire en $O(n \log n)$

On se donne maintenant les 2 algorithmes suivants, appelés Fromage et Dessert.

```
Algorithme Fromage
  Début
    int i:=1;

    Tant que i<=n faire
      Exécuter A1;
      i:=i+2;
    Fin TantQue

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

```
Algorithme Dessert
  Début
    int k:=1;
    Tant que k<=n faire
      Exécuter A2;
      Exécuter A1;
      k:=2*k;
    Fin TantQue
  Fin
```

1. Quelles sont les complexités au mieux et au pire de Fromage ?

2. Quelles sont les complexités au mieux et au pire de Dessert ?

3. Pourquoi les problèmes NP-complets sont-ils dans NP ?

4. Soit Pb_1 et Pb_2 deux problèmes de décision, et supposons que l'on a démontré $\text{Pb}_1 \geq_P \text{Pb}_2$ (c'est-à-dire que Pb_1 se réduit polynomialement en Pb_2). Si Pb_2 est dans P, que dire de Pb_1 ? Pourquoi ?

Exercice 2 (Calcul de la plus grande somme – 8 points)

On se donne un tableau T, contenant n entiers, strictement positifs ou strictement négatifs (0 n'appartient donc pas à T). On supposera que T contient au moins un entier positif et un entier négatif.

Le problème MAX-SOMME qu'on se pose est le suivant: trouver la valeur de $S_{i,j}$ *la plus grande* dans le tableau, sachant que $S_{i,j} = \Sigma_{k=i}^{j} T[k]$ est la somme des éléments consécutifs de T compris entre $T[i]$ et $T[j]$ ($T[i]$ et $T[j]$ inclus).

Par exemple, si $T = [2, -3, \underline{15, 7, -8, 11}, -22, 14, 6]$, alors $S_{3,6} = 25$ est la plus grande somme (elle correspond à la partie soulignée).
Dans cet exercice, on cherche à définir un algorithme qui prend en entrée le tableau T et qui résout MAX-SOMME en renvoyant la somme maximale recherchée. On supposera que le tableau T d'entiers qui satisfait aux conditions du problème est représenté par le type `tab-entiers`, et qu'il contient n valeurs.
On affirme que MAX-SOMME est dans `P`, et plus précisément qu'il existe un algorithme, que l'on appellera `max-somme1`, dont la complexité au pire est polynomiale.

1. Écrire en pseudo-code l'algorithme `max-somme1`.

2. Montrer que `max-somme1` est correct, et donner, en la justifiant, la complexité au pire de `max-somme1`.

Voici un autre algorithme, récursif, qui résout MAX-SOMME. Cet algorithme est appelé `max-somme2`. On précise la signification des trois fonctions suivantes, qui s'exécutent chacune en temps constant:

- `pei`(x) (pour "partie entière inférieure") renvoie l'entier $y \leq x$ qui est le plus proche de x (par exemple, `pei`$(7.5)= 7$).
- `max3`(p, q, r) renvoie le maximum des trois entiers p, q et r
- `max`(p, q) renvoie le maximum des deux entiers p et q

```
int max-somme2(T: tab-entiers; a,b: int){
 var i,somme,ama,bma,resultat:int;

    Si a=b alors
      return max(0,T[a]);
    FinSi

    c:=pei((a+b)/2);
    ama:=0;
    somme:=0;
    i:=c;

    Tant que i>=a faire{
       somme:=somme+T[i];
       ama=max(ama,somme);
       i:=i-1;
    }

    bma:=0;
    somme:=0;
    i:=c+1;

    Tant que i<=b faire{
       somme:=somme+T[i];
       bma=max(bma,somme);
       i:=i+1;
    }

   resultat:=max3(ama+bma,max-somme2(T,a,c),max-somme2(T,c+1,b));

   return resultat;
}
```

3. Si $T = [2, -3, 15, 7, -8, 11, -22, 14, 6]$, $a = 1$ et $b = 9$ (9 étant le nombre d'éléments de T), que valent `ama` et `bma` avant le premier appel récursif ?

4. Au vu de la question précédente, que représente, en général, la valeur `ama+bma` lorsqu'on appelle `max-somme2` avec les indices a et b ?

5. Montrer que `max-somme2` est correct (lorsqu'on l'appelle avec $a = 1$ et $b = n$ où n est le nombre d'éléments de T).

6. En utilisant le Master Theorem, donner la complexité au pire de `max-somme2` (lorsqu'on l'appelle avec $a = 1$ et $b = n$).

Voici un troisième algorithme, appelé `max-somme3`, qui résout MAX-SOMME, et dont on admettra qu'il est correct.

```
int max-somme3(T: tab-entiers; n: int){
var p,q,i: int;

    p:=0;
    q:=0;
    i:=1;
    Tant que i<=n faire{
        q:=max(q+T[i],0);
        p:=max(p,q);
        i:=i+1;
    }

    return p;
}
```

7. Remplir le tableau ci-dessous, qui représente l'évolution des variables p et q lors de l'exécution de `max-somme3` sur le tableau T suivant : $T = [2, -3, 15, 7, -8, 11, -22, -5, 6]$ avec $n = 9$, et indiquer la valeur retournée.

i	1	2	3	4	5	6	7	8	9
q									
p									

Valeur retournée:

8. Que représente p dans l'algorithme `max-somme3` ? Que représente q ?

9. Donner, en la justifiant, la complexité au pire de `max-somme3`.

Exercice 3 (Le problème 3-SAT – 8 points)

On rappelle les définitions des problèmes SAT et 3-SAT ci-dessous.

> 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 ?

> 3-SAT
> *Instance* : Une formule booléenne Φ' en FNC, construite à partir d'un ensemble $X' = \{x'_1, x'_2 \dots x'_p\}$ de p variables, et dans laquelle chaque clause contient 3 variables.
> *Question* : Existe-t-il une affectation (`Vrai`/`Faux`) des variables de X' qui rend Φ' satisfiable ?

1. Montrer que 3-SAT est dans NP.

On veut maintenant montrer que SAT se réduit vers 3-SAT (SAT $\geq_P$ 3-SAT). Pour cela, partant d'une instance quelconque I de SAT, on construit une instance I_3 de 3-SAT de la manière suivante:

(a) Pour toute clause $C = (a)$ de I contenant une seule variable a, on crée dans I_3 la clause $C' = (a \vee a \vee a)$.

(b) Pour toute clause $C = (a \vee b)$ de I contenant deux variables a et b, on crée dans I_3 la clause $C' = (a \vee b \vee b)$.

(c) Pour toute clause C de I contenant trois variables, on garde la même clause dans I_3 (donc $C' = C$).

(d) Pour toute clause $C = (a_1 \vee a_2 \vee \ldots \vee a_k)$ de I contenant $k \geq 4$ variables, on crée dans I_3 l'expression $E'_C = (a_1 \vee a_2 \vee y_1) \wedge (\overline{y_1} \vee a_3 \vee y_2) \wedge (\overline{y_2} \vee a_4 \vee y_3) \dots (\overline{y}_{k-4} \vee a_{k-2} \vee y_{k-3}) \wedge (\overline{y}_{k-3} \vee a_{k-1} \vee a_k)$, où les y_i sont de nouvelles variables.

(e) La formule I_3 relie les clauses et expressions listées ci-dessus par des "ET" logiques.

2. Supposons que $I = (x_1 \vee \overline{x_2}) \wedge (x_1 \vee x_2 \vee \overline{x_3} \vee \overline{x_4} \vee x_5) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_5})$. Que vaut l'instance I_3 ?

3. Montrer que si une clause C à $k \leq 3$ variables est satisfaite dans I, alors sa clause correspondante C' est satisfaite dans I_3.

Soit $C = (a_1 \vee a_2 \vee \ldots \vee a_k)$ une clause à $k \geq 4$ variables dans I et E'_C son expression correspondante dans I_3. Supposons que C soit satisfaite. Il existe donc au moins une variable, que l'on appellera a_i, qui rend C vraie.

4. Montrer que l'affectation $y_1, y_2 \dots y_{i-2} =$`Vrai` et $y_{i-1}, y_i \dots y_{k-3} =$`Faux` satisfait E'_C.

5. Que déduit-on des Questions 3. et 4. ?

Une clause C' de I_3 construite à partir d'une clause C à $k \leq 3$ variables de I est appelée une *petite* clause.

6. Montrer que si une petite clause C' est satisfaite dans I_3, alors la clause C dont elle provient est satisfaite dans I.

7. Supposons que tous les a_i sont à `Faux` dans une expression E'_C. Montrer que E'_C ne peut jamais être satisfaite.

8. En déduire que si une expression E'_C est satisfaite dans I_3, alors la clause C dont elle provient est satisfaite dans I.

9. Que déduit-on des Questions 6. et 8. ?

10. Que déduit-on des Questions 5. et 9. ?

11. Que déduit-on des Questions 1. et 10. ?