Algorithmique 4: Contrôle continu

- SUJET A -

- Durée : 1 heure - Notes de cours autorisées uniquement -
Novembre 2023

Le barême est indicatif, il permet juste de comparer l'importance des exercices.
Toutes les réponses doivent être claires et justifiées.
Le sujet est à rendre avec votre copie.

Exercice 1. *Complexité d'algorithmes* **(3 points)**
Calculer la complexité de chacun des algorithmes suivants. L'instruction <op_elem> désigne n'importe quelle opération élémentaire, qui a complexité $O(1)$ par définition.

```
ALGO1(n) :
Pour i = 0 à n−1 :
    Pour j = 0 à n−1 :
        Pour k = 0 à j :
            <op_elem>
Pour i = 0 à n−1 :
    <op_elem>
```

```
ALGO2(n) :
<op_elem>
Tant que n > 1 :
    n ← n/3
    <op_elem>
<op_elem>
```

```
ALGO3(n) :
Si n ≤ 1 : Renvoyer 0
<op_elem>
ALGO3(⌈n/3⌉)
ALGO3(⌈n/3⌉)
Pour i = 0 à n :
    <op_elem>
```

Exercice 2. *Pile avec vidange* **(5.5 points)**
On dispose d'une pile P avec les opérations EMPILER(x, P) et DÉPILER qui respectivement empile l'élément x sur P et dépile P et que l'on suppose s'exécuter en temps c, constant. On souhaite maintenant disposer d'une méthode VIDEPILE qui vide la pile P sur laquelle elle est appelée.

1. Ecrire l'algorithme de VIDEPILE. On pourra utiliser l'appel ESTVIDE(P) qui renvoie VRAI si P est vide et FAUX sinon.
2. Calculer le temps d'exécution de VIDEPILE, en fonction de c et du nombre k d'éléments contenu dans la pile au moment de l'appel.
3. On effectue maintenant n opérations EMPILER, DÉPILER ou VIDEPILE à partir d'une pile vide. Le but de la question est de montrer que le coût amorti de chacune de ces opérations est constant.
   (a) On veut utiliser d'abord la méthode par agrégat. On note n_E le nombre d'appels à EMPILER. En distinguant les éléments empilés jamais dépilés, ceux dépilés par DÉPILER et ceux dépiler par VIDEPILE, montrer que le temps total des n opérations est majoré par $2.c.n_E$. Conclure.
   (b) On utilise ensuite la méthode comptable. On va verser sur 'le compte' $2.c$ par appel à EMPILER et 0 pour les autres opérations. En montrant, par récurrence sur le nombre d'opérations effectuées, que le montant du compte est toujours supérieur ou égal à deux fois le nombre d'éléments dans la pile, conclure que la complexité amortie de chaque opération est constante.

Exercice 3. *Déménagement* **(11.5 points)**
Lors d'un déménagement, n objets de poids $T_{[0]}, \ldots, T_{[n-1]}$ telles que $0 < T_{[i]} < 100$ pour tout i doivent être transportés par m véhicules de poids de charge maximum 100. On veut agencer les n objets dans les m véhicules : un véhicule peut contenir plusieurs objets, mais la somme des poids des objets contenus dans un véhicule ne doit pas dépasser 100.
Exemple. Supposons qu'on a 7 objets, de poids 20, 50, 40, 70, 10, 30 et 60, et 3 véhicules. Alors une solution est $\{20, 70\}$, $\{50, 40, 10\}$ et $\{30, 60\}$. En revanche, si on n'a que 2 véhicules, il n'y a pas de solution.

1. **(a)** Pourquoi dans l'exemple fourni, il est impossible de ranger tous les objets dans 2 véhicules ?

   (b) Donner une condition nécessaire sur m en fonction de la somme des poids des objets, pour qu'il y ait une solution.

   (c) En considérant trois objets, de poids bien choisis, montrer que la condition n'est pas suffisante.

On résout le problème par recherche exhaustive. Une *tentative* consiste à placer chaque objet dans un véhicule. Une tentative est *valide* si aucun véhicule ne contient des objets dont la somme des poids dépasse 100. On numérote les véhicules de 0 à $m-1$. Une tentative est représentée par un n-uplets S, autrement dit un tableau de taille n, d'entiers entre 0 et $m-1$, tel que $S[i]=j$ si l'objet i va dans le véhicule j.

2. **(a)** Écrire un algorithme EstValide(T,S) qui teste si S est une tentative valide.

   (b) Quelle est la complexité de votre algorithme EstValide ?

3. On veut parcourir toutes les tentatives possibles, c'est-à-dire tous les n-uplets S possibles.

   (a) Combien y a-t-il de n-uplets S à parcourir ? Dans l'ordre lexicographique (vu en cours), quel est le premier n-uplet ? Et le dernier ?

   (b) Écrire un algorithme Suivant qui prend en entrée un n-uplet S et m, et renvoie le n-uplet suivant dans l'ordre lexicographique. La fonction renvoie « Fini » si S est le dernier n-uplet.

   (c) Quelle est la complexité dans le pire des cas de l'algorithme Suivant ?

4. **(a)** Déduire des questions précédentes un algorithme Déménagement qui prend en entrée un tableau T de taille n et un entier m et renvoie Vrai si on peut charger les n objets de poids $T_{[0]}, \ldots, T_{[n-1]}$ dans m véhicules de poids de charge maximum 100.

   (b) Analyser la complexité de l'algorithme Déménagement.

   (c) Supposons qu'on n'ait plus m en entrée. Décrire un algorithme pour trouver le nombre minimal m de véhicules nécessaires pour déménager les n objets.

5. Écrire une version *backtrack* de l'algorithme Déménagement.