Université USTHB – Bab-Ezzouar Bab-Ezzouar, 27 Septembre 2017

Faculté de l’Electronique et de l’Informatique, Département de l’Informatique Année universitaire 2017/2018

1ère année Master Informatique, Semestre 1 Semestre 1

Module : Conception et Complexité des Algorithmes

-------------------------------------------------------------------------------------------------------------------------------------------------------

**Série de Travaux Dirigés n° 1**

**Complexité des algorithmes**

**Exercice 1**

**Rappel :** Un nombre entier naturel n est premier s’il n’a que 2 diviseurs : le nombre 1 et le nombre n lui-même.

1- Développer un algorithme itératif, noté A1, qui permet de déterminer si un nombre entier naturel n est premier (on suppose : n>=2). On doit utiliser l’opérateur modulo qui donne le reste de la division de n par i, i variant de 2 jusqu’à (n-1). Une variable de type booléen, notée premier, permettra d’indiquer l’arrêt des tests au cas où n n’est pas premier.

2- Calculer la complexité temporelle asymptotique en notation de Landau O (Grand O), notée CT(n), de cet algorithme en précisant le cas exact, le meilleur cas et/ou le pire cas. Pour cela, on doit calculer le nombre d’instructions exécutées par l’algorithme.

3- On cherche à améliorer le temps d’exécution de l’algorithme précédent en utilisant certaines des propriétés des nombres premiers.

3.1- **Propriété 1 :** Tout diviseur i d’un nombre entier naturel n vérifie la relation :

, sauf pour .

Développer un 2ème algorithme, noté A2, en tenant compte de cette propriété et refaire la question 2.

3.2- **Propriété 2 :** Un nombre entier naturel pair n n’est pas premier, sauf pour .

Développer un 3ème algorithme, noté A3, en tenant compte de cette propriété et refaire la question 2.

3.3- Développer un 4ème algorithme, noté A4, en utilisant en hybridant (càd., conjointement) les 2 propriétés précédentes et refaire la question 2.

3.4- **Propriété 3 :** Les diviseurs d’un nombre entier naturel n sont pour la moitié et pour l’autre moitié compris entre .

Développer un 5ème algorithme, noté A5, en tenant compte de cette propriété et refaire la question 2.

3.5- Développer un 6ème algorithme, noté A6, en utilisant en hybridant (càd., conjointement) les 2 propriétés 2 et 3 et refaire la question 2.

4- Calculer le nombre d’instructions exécutées par chacun des 6 algorithmes pour la valeur suivante de : .

5- Regrouper dans un tableau comparatif les résultats obtenus (complexités temporelles et nombres d’opérations exécutées) pour les 6 algorithmes.

6- Quel est le gain des 5 algorithmes A2, A2, …, A5 par rapport à l’algorithme initial A1.

7- Que peut-on conclure ?

**Exercice 2**

On suppose un ordinateur dont les instructions ont un temps d’exécution uniforme égal à :

.

On suppose aussi différents algorithmes dont les nombres d’instructions exécutées sont représentés respectivement par les fonctions suivantes de la variable :

.

1- Calculer les temps d’exécution de chacun de ces algorithmes pour les valeurs suivantes de . Représenter les résultats dans un tableau comparatif.

2- Classer ces algorithmes en fonction de leurs complexités asymptotiques.

3-On souhaite améliorer les temps d’exécution de ces algorithmes. Pour cela, on utilise des ordinateurs plus rapides.

3.1- On dispose d’un ordinateur 1000 fois plus rapide que le premier ordinateur, donc son temps d’exécution d’une instruction est : .

Refaire la question 1. Quel est l’impact de l’amélioration de la vitesse de l’ordinateur sur les temps d’exécution des 6 algorithmes précédents ?

3.2- On suppose, cette fois, qu’on a à notre disposition un ordinateur 1.000.000 (1 million) de fois plus rapide que le premier ordinateur, donc son temps d’exécution d’une instruction est :

.

Refaire la question 1. Quel est l’impact de l’amélioration de la vitesse de l’ordinateur sur les temps d’exécution des 6 algorithmes précédents ?

3.3- Est-ce que l’amélioration de la vitesse des ordinateurs permettra à l’avenir d’exécuter les algorithmes (et donc de résoudre les problèmes) en un temps raisonnable ?

**Ind :** *Le terme* ***"temps raisonnable"*** *signifie ici un temps à échelle humaine. Celle-ci (l’échelle) peut être variable selon les besoins. On peut attendre quelques jours, quelques mois ou même quelques années pour des problèmes spéciaux tout en continuant à chercher de meilleures solutions; mais on ne peut pas attendre plus que* ***"quelques fractions de seconde"*** *pour corriger la trajectoire d’un avion ou d’un engin spatial, ou pour contrer un tir de missile par le tir d’un contre missile. En tout état de cause, un temps de réponse de l’ordre des siècles n’a aucun sens pratique pour les êtres humains.*

**Exercice 3**

1- Développer un algorithme qui permet de déterminer le plus grand commun diviseur qu’on note habituellement "pgcd" de 2 entiers naturels non nuls . On doit utiliser la méthode de Euclide. L‘algorithme doit être écrit sous la forme d’une fonction notée qui retourne le .

**Ind :** On peut utiliser l’opérateur modulo qui donne le reste de la division .

2- Calculer la complexité temporelle asymptotique en notation de Landau O (Grand O), notée CT(n), de cet algorithme en précisant le cas exact, le meilleur cas et/ou le pire cas. Pour cela, on doit calculer le nombre d’instructions exécutées par l’algorithme.

**Exercice 4**

On dit qu’un mot est un "palindrome" si ce mot se lit de la même manière de la droite vers la gauche ou de la gauche vers la droite. Par exemple, les mots suivant sont des palindromes :

"radar", "elle", "aya", "afifa", "anina", "aziza", "ada", "php", …

1- Développer un algorithme itératif ou récursif qui permet de déterminer si un mot est un palindrome. L‘algorithme doit être écrit sous la forme d’une fonction notée qui retourne la valeur .

2- Calculer la complexité temporelle asymptotique en notation de Landau O (Grand O), notée CT(n), de cet algorithme en précisant le cas exact, le meilleur cas et/ou le pire cas. Pour cela, on doit calculer le nombre d’instructions exécutées par l’algorithme.

**Partie II : Machines de Turing**

**Exercice 5**

On dit qu’un mot est un "palindrome" si ce mot se lit de la même manière de la droite vers la gauche ou de la gauche vers la droite. Par exemple, les mots suivant sont des palindromes :

"radar", "elle", "aya", "afifa", "anina", "aziza", "ada", "php", "1001", "001100", …

On considère ici l’alphabet . Un mot palindrome se présente sous la forme ww-1 où w est un mot et w-1 est l’image miroir de w.

1- Développer une machine de Turing à 1 ruban qui permet de reconnaître les mots palindromes. Le résultat en sortie doit être égal à 1 si le mot en entrée est un palindrome et égal à 0 sinon. Calculer les complexités temporelle et spatiale de cette machine.

2- Refaire la question précédente avec une machine de Turing à 2 rubans.

**Exercice 6**

On considère le langage L défini sur l’alphabet :

1- Développer un algorithme itératif ou récursif qui permet de reconnaître les mots du langage . Calculer les complexités temporelle et spatiale de cet algorithme.

2- Développer une machine de Turing qui permet de reconnaître les mots du langage . Le résultat en sortie doit être égal à 1 si le mot en entrée est correct et égal à 0 sinon. Calculer les complexités temporelle et spatiale de cette machine.

**Ind :** Le nombre de rubans de la machine de Turing est au choix du concepteur.

**Exercice 7 - Machine de Turing pour l’addition de 2 nombres binaires**

On souhaite concevoir une machine de Turing qui permet d’effectuer l’addition de 2 nombres binaires. On suppose que les 2 nombre ont la même longueur n.

1- Illustrer la solution sur l’exemple : .

2- Développer une machine de Turing qui permet d’effectuer cette addition. Calculer les complexités temporelle et spatiale de cette machine.

**Ind :** Le nombre de rubans de la machine de Turing est au choix du concepteur.