Solutions du midterm

Question 1. (3 points)

a) (1 point) La sortie de l'algorithme vaut 3.

b) (1 point) En général, la sortie de l'algorithme vaut $\log_2(n)$ lorsque n est une puissance de 2.

c) (1 point) La complexité de l'algorithme est un $\Theta(n)$ (quoi qu'il arrive, la boucle sera exécutée jusqu'au bout).

Question 2. (3 points)

a) (1 point) La sortie de l'algorithme vaut $\max(a, b)$.

b) (1 point) La sortie de l'algorithme vaut $\mathrm{pgdc}(a, b)$.

c) (1 point) $\textbf{algo2}(2^n, 2^{n-1}) = \textbf{algo2}(2^n - 2^{n-1}, 2^{n-1}) = \textbf{algo2}(2^{n-1}, 2^{n-1}) = 2^{n-1}$. Donc le nombre d'opérations effectuées est un $\Theta(1)$.

Question 3. (3 points)

a) (1 point) Plusieurs solutions sont possibles: $S = \{1, 2, 3, 5, 6\}$, $S = \{1, 2, 4, 5, 6\}$ ou $S = \{2, 3, 4, 5, 6\}$.

[Alternativement, les solutions écrites sous la forme $S = \{-4, +12, -3, +7, +8\}$ etc. sont aussi acceptées ici.]

b) (1 point) Oui, le problème est dans P: il suffit de parcourir une fois la liste L et de retenir tous les nombres positifs dans le produit, ainsi que le plus grand nombre *pair* possible de nombres négatifs (tout en excluant les nombres nuls).

c) (1 point) Oui, le problème est dans NP, car il est dans P.

Question 4. (4 points)

a) (1 point) 1001 x 1001 = 1010001 (en décimal, 9 x 9 = 81).

b) (1 point) Si x est représenté sur n bits, il faut environ $2n$ bits pour représenter x^2.

c) (2 points) Il y a 14 scores finaux possibles (7-0, 7-1, 7-2, etc., ainsi que 0-7, 1-7, 2-7, etc.), donc 4 bits suffisent pour représenter tous les scores finaux possibles.

Question 5. (6 points)

a) (3 points) Voici une solution possible:

algo
entrée : *liste L de nombres entiers positifs, de taille n, nombre entier positif x* sortie : *rang r de x dans la liste L (0 si $x \notin L$)*

```
s ⟵ non
Pour i allant de 1 à n
    Si L(i) = x
        s ⟵ oui
Si s = non
    Sortir : 0
r ⟵ 1
Pour i allant de 1 à n
    Si L(i) > x
        r ⟵ r + 1
Sortir : r
```

b) (3 points) Voici une solution possible:

algo
entrée : *liste L de nombres entiers positifs, de taille n, nombre entier positif x* sortie : *rang r de x dans la liste L (0 si $x \notin L$)*

```
Si recherche_dichotomique(L, n, x) = non
    Sortir : 0
a ⟵ 1
b ⟵ n
Tant que b > a
    m ⟵ ⌊(a+b)/2⌋
    Si L(m) = x
        Sortir : m
    Si L(m) > x
        a ⟵ m + 1
    Sinon
        b ⟵ m − 1
Si L(a) = x
    Sortir : a
Si L(b) = x
    Sortir : b
```

La complexité temporelle de cet algorithme est un $\Theta(\log_2(n))$.

Question 6. (6 points)

a) (3 points) Voici une solution possible:

algo
entrée : *liste L de nombres entiers positifs, de taille n* sortie : *oui/non*

$s \longleftarrow 0$
Pour *k allant de 1 à n*
    $s \longleftarrow s + L(n+1-k)$
    Si $s \geq 0$
        Sortir : *oui*
Sortir : *non*

b) (3 points) Voici une solution possible:

algo
entrée : *liste L de nombres entiers positifs, de taille n* sortie : *oui/non*

Si $n = 1$
    Si $L(1) \geq 0$
        Sortir : *oui*
    Sinon
        Sortir : *non*
Si $L(n) \geq 0$
    Sortir : *oui*
$M \longleftarrow L(1:n-2) \oplus (L(n-1) + L(n))$
Sortir : algo$(M, n-1)$