INF5130 - Algorithmique

Série d'exercices - 3

Exercice 1. La fonction suivante prend en entrée une portion de tableau contenue entre l'indice B_I et l'indice B_S et retourne deux nombres entiers (les indices des tableaux commencent à 1). Les éléments du tableau sont eux-mêmes des entiers. Vous pouvez supposer que les éléments du tableau sont tous différents les uns des autres et que le nombre d'éléments de la portion de tableau est pair et plus grand que 0.

```
1:  fonction BIDON(Tab : tableau, B_I, B_S : indices) : deux entiers
2:      si Tab[B_I] ≥ Tab[B_I + 1] alors
3:          w ← Tab[B_I]
4:          x ← Tab[B_I + 1]
5:      sinon
6:          w ← Tab[B_I + 1]
7:          x ← Tab[B_I]
8:      fin si
9:      si B_S ≤ B_I + 1 alors
10:         n_1 ← w
11:         n_2 ← x
12:     sinon
13:         y, z ← BIDON(Tab, B_I + 2, B_S)
14:         si w ≥ y alors
15:             n_1 ← w
16:         sinon
17:             n_1 ← y
18:         fin si
19:         si x ≤ z alors
20:             n_2 ← x
21:         sinon
22:             n_2 ← z
23:         fin si
24:     fin si
25:     retourner n_1, n_2
26: fin fonction
```

(a) Supposons que $B_I = 1$, $B_S = 6$ et que Tab contient les éléments $-7, 2, -3, 5, 0, 1$. Faites la trace de l'appel BIDON(Tab, B_I, B_S) : pour chaque appel à BIDON() dans la pile d'exécution, donnez les valeurs de B_I, B_S et les valeurs retournées n_1, n_2, si ces dernières ne sont pas encore connues, mettez "?".

(b) Décrivez ce que fait BIDON() en général.

(c) Soit $T(n)$ le nombre de comparaisons effectuées par un appel de BIDON() lorsque le nombre d'éléments de la portion de tableau est égal à n. La comparaison $B_S \leq B_I + 1$ ne compte pas... Écrivez l'équation de récurrence satisfaite par $T(n)$.

(d) Résolvez l'équation de récurrence que vous avez trouvée en (c). Vous pouvez la résoudre de manière exacte.

Exercice 2. Voici un algorithme qui permet de résoudre le problème des tours de Hanoi.

```
1: procedure DEPLACER(n : entier naturel, T_1, T_2, T_3 : indice de tour [1, 2, 3])
2:     si n = 1 alors
3:         AFFICHER(Deplacer un disque de la tour T_1 à la tour T_2)
4:     sinon
5:         DELACER(n − 1, T_1, T_3, T_2)
6:         AFFICHER(Deplacer un disque de la tour T_1 à la tour T_2)
7:         DELACER(n − 1, T_3, T_2, T_1)
8:     fin si
9: fin procedure
```

DEPLACER$(n, T_1, T_2, T-3)$ a pour effet d'afficher tous les déplacements requis pour enfiler n disques sur la tour T_2. Au départ les disques sont enfilés sur la tour T_1.

(a) Soit $T(n)$ le nombre de déplacements effectués, ou, ce qui revient au même, le nombre de messages affichés par la procédure DEPLACER() lorsqu'elle est appelée avec n pour paramètre. En supposant que l'affichage d'un message prend un temps égal à 1, écrivez l'équation de récurrence satisfaite par $T(n)$.

(b) Résolvez l'équation de récurrence trouvée en (a). Vous pouvez la résoudre de maniàre exacte.

Exercice 3. Soit Tab un tableau de nombres entiers tous différents les uns des autres dont les indices sont compris entre 1 et n. On suppose que les éléments de Tab sont en ordre croissant et on désire résoudre le problème suivant :

Existe-t-il un indice i tel que $Tab[i] = i$?

Écrivez une fonction booléenne qui retourne la valeur *VRAI* si et seulement si un indice i vérifiant le problàme ci-dessus existe dans le tableau Tab. Votre fonction doit prendre un temps proportionnel à $\log(n)$ dans le pire des cas.

Exercice 4. Écrivez un algorithme efficace pour calculer la racine carrée entière d'un entier n supérieur ou égal à 2. La racine carrée entière de n est le plus grand entier positif r tel que $r^2 \leq n$.

Par *algorithme efficace*, on entend un algorithme dont le temps d'exécution est proportionnel à $\log(n)$.

Exercice 5. Le but de cet exercice est de résoudre des équations de récurrence de la forme,

$$T(n) = T(n-a) + T(a) + n$$

où a est un nombre naturel positif.

(a) Résolvez cette équation de manière exacte dans le cas où a est égal à 1. Vous pouvez supposer que $T(1)$ est une constante connue.

(b) Résolvez cette équation de manière exacte dans le cas où a est égal à 2. Vous pouvez supposer que $T(2)$ et $T(1)$ sont des constantes connues.

(c) Pouvez-vous "deviner" la complexité de la solution lorsque a est un nombre quelconque mais fixe ?