Epreuve finale Mars 2021 - Master TC - Département Informatique - Faculté FSEA - Université Oran1

Algorithmique Avancée et Complexité

N.B./ Réponse de l'exercice 1 dans une copie. Exercices 2&3 dans une deuxième copie.

Exercice 1 (Tris / 10 pts)

Q1) (4p) Dérouler le tri rapide sur le tableau [9, 8, 7, 6, 5, 4].

Q2) (3p) Démontrer la complexité du tri-rapide sur un tableau décroissant de longueur *n*, en explicitant l'équation récurrente et sa résolution.

Q3) (3p) Soit l'algorithme du tri-fusion d'un tableau $A[1..n]$. Donner l'algorithme de l'étape du *combiner* : la fusion de deux tableaux triés $A[i..k]$ et $A[k+1..j]$. Puis, démontrer sa complexité.

Exercice 2 (Structures de données / 7 pts)

Q1) Soit l'ensemble de clés $T = \{9, 8, 7, 6, 5, 4, 3\}$:

a. (2p) Dessiner des arbres binaires de recherche de hauteur 2, 3, 4, 5, 6 de *T*.

b. (1p) Justifier, lesquels de ces six arbres, sont des arbres AVL.

Q2) Soit une nouvelle structure d'un arbre binaire de recherche, dite DZ, ayant la propriété suivante :

Un arbre binaire de recherche est un arbre DZ si, pour n'importe lequel de ses nœuds, la différence absolue de hauteur entre ses deux fils diffère d'au plus ***deux****.*

(N.B./ La structure DZ permet un facteur d'équilibrage (hauteur sous-arbre droit moins hauteur sous-arbre gauche) plus tolérant ''-2, -1, 0, 1, 2'' alors que AVL permet ''-1, 0, 1'')

a. (2p) Est-ce que les 7 algorithmes des arbres AVL restent valables sur les arbres DZ en justifiant ?

b. (2p) Formuler les équations récurrentes nécessaires pour démontrer que les arbres DZ sont équilibrés.

Exercice 3 (Programmation dynamique – Algorithmes gloutons / 3 pts)

Q1) (2pt) Soit le problème de la recherche du parenthésage optimal d'une suite de n matrices $A_1 * A_2 * \ldots * A_n$. Soit $P(n)$ le nombre de parenthésages de n matrices. Exprimer l'équation récurrente qui régit le nombre de parenthésages $P(n)$, ainsi qu'une caractérisation de sa solution.

Q2) (1pt) Pourquoi on fait appel à un algorithme glouton ?
Quand est-ce qu'un algorithme glouton donne un résultat exact ? Donner un exemple.

AAA — EF - Mars 2021

Exercice 1

Q2) Complexité tri-rapide d'un tableau croissant :

Le partitionnement du tri rapide sur chaque sous-tableau de taille n produit un seul appel récursif sur n-1.
On obtient la récurrence :

$$T(n) = T(n-1) + \Theta(n)$$

D'où
$$= T(n-2) + \Theta(n-1) + \Theta(n)$$
$$\vdots$$
$$= \Theta\left(\sum_{k=1..n} k\right)$$
$$= \Theta(n^2).$$

Q3)

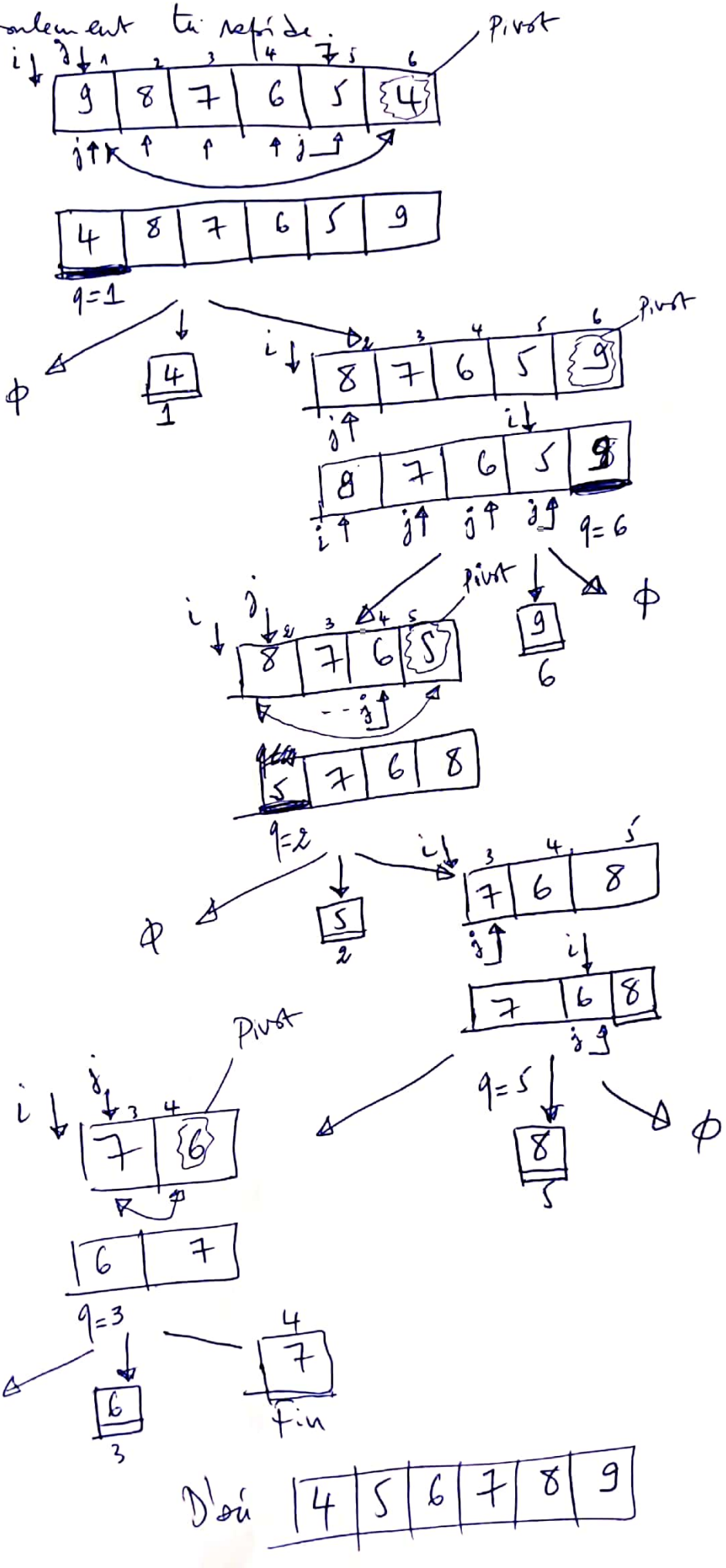
Algorithme Fusion (A[1..n], i, k, j)
Entrée : A[i..k-1], A[k..j] sont triés.
Soit B : tableau d'entiers de taille j-i+1.

```
de1 ← i ; vers1 ← k-1 ; de2 ← k ; vers2 ← j ;
s ← 1;
for a ← 1 to (j-i+1) do
    if (de2 > vers2) then
        B[s++] ← A[de1++]
    else if (de1 > vers1) then
        B[s++] ← A[de2++]
    else if (A[de1] <= A[de2]) then
        B[s++] ← A[de1++]
    else B[s++] ← A[de2++]
    endif
end for
for a ← 1 to j-i+1 do
    A[i+a-1] ← B[a]
end for
```

Complexité : Theta(n) avec deux boucles simples qui itèrent n fois sur une fusion de deux tableaux de n cases au total.

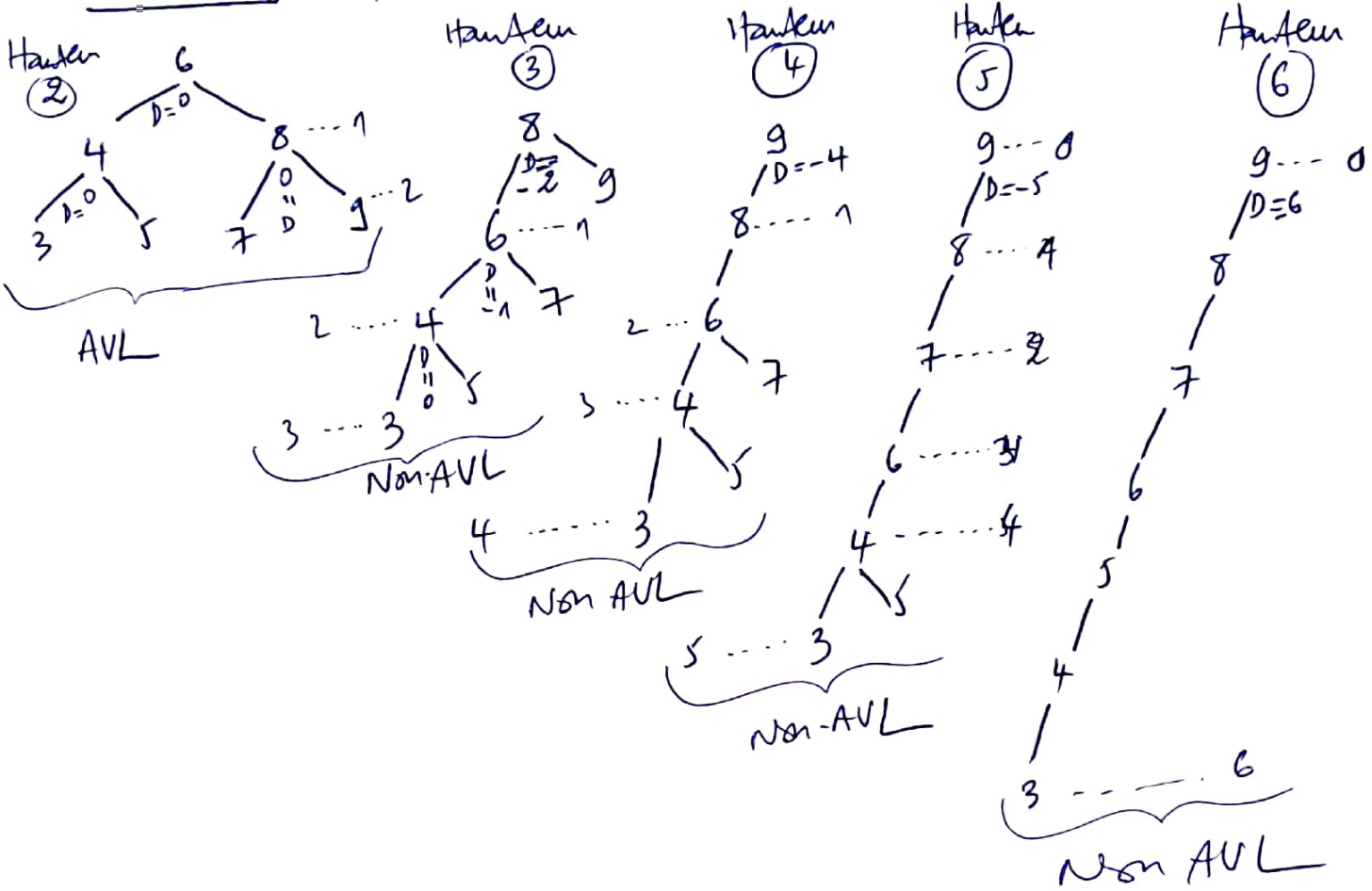
AAC — Examen - Mars 2021

Exercice 1 : Q1) Déroulement du tri rapide.



EP — AAC — Mars 2024

Exercice 2 | Q1) a. et b.



Uniquement l'arbre de Hauteur 2 est AVL.

Tous les autres 3, 4, 5, 6 ne sont pas AVL.

Car le facteur de déséquilibre (D = Hauteur droite − Hauteur gauche)

est $\begin{cases} D = 0 \text{ sur tous les nœud de l'arbre de hauteur 2.} \\ |D| > 1 \text{ sur au moins un nœud de chacun} \\ \text{des arbres de Hauteur 3, 4, 5 et 6.} \end{cases}$

EF — AAC — Mars 2021

Exercice 2 Q2) a) Les 5 algorithmes Recherche, Min, Max, Successeur et Prédécesseur restent valables car ils ne changent pas la structure de l'arbre D2.

Par contre les deux algorithmes "insérer" et "supprimer" ne sont plus valables car ils peuvent provoquer un déséquilibre avec un facteur = 2 ou -2, qui ne peuvent être résolus par une double rotation.

b) Les équations récurrentes :

Proposition : La hauteur d'un arbre D2 est en $\Theta(\lg(n))$?

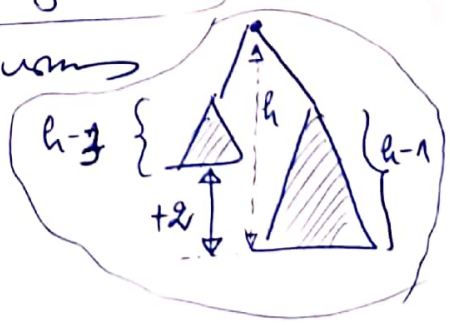
Preuve : Nous avons deux cas limites : a) soit chaque nœud est parfaitement équilibré ⇒ facteur de déséquilibre égal à zéro partout.

b) Soit l'arbre D2 est déséquilibré partout avec un facteur = 2.

⇒ Cas a) L'arbre est complet, d'où $n \leq \sum_{i=0}^{h} 2^i \Rightarrow n \leq 2^{h+1} - 1$

$\Rightarrow \lg_2(n+1) \leq 1+h \Rightarrow \boxed{h \geq \lg_2(n+1)}$

⇒ Cas b) Partout dans l'arbre D2, nous avons



Dans ce cas, soit $U(h)$ le nombre de nœuds.

$$U_h = U_{h-1} + U_{h-3} + 1$$
$$U_0 = 1,\ U_1 = 2,\ U_2 = 3$$

Exercice 3

Q1) $P(n) = \begin{cases} 1 & \text{si } n = 1 \\ \sum_{k=1..n-1} P(k) \cdot P(n-k) & \text{si } n \geq 2 \end{cases}$

Caractérisation de la solution $P(n) = \Omega(4^n / n^{3/2})$

Q2) a) On fait appel aux algorithmes gloutons quand toutes les variantes de l'algorithmique Diviser pour régner, à savoir par programmation dynamique et autres, sont trop contenues.

b) L'un algorithme glouton choisit toujours un seul sous-problème à résoudre en utilisant une heuristique pour choisir le sous-problème le plus intéressant menant le plus vers la solution exacte/optimale.
L'algorithme glouton est exact quand l'heuristique réussit toujours à choisir le sous-problème menant vers la solution exacte.
Un exemple : Algorithme de Huffman de compression de fichiers avec la méthode par préfixe. (Codages Préfixes).