Corrigé série 5

Andrey Martinez Cruz

1 Exercise 1

1.1 Question A

Sachant que pour chaque problème il manque toujours un nombre entre 0 et n, il y a donc $n + 1$ variantes du problèmes dû que 0 est aussi un élément potentiellement manquant dans le tableau.

1.2 Question B

La ligne 5 est plus précisément la comparaison entre i et n peut être pris comme opération barométrique, car la comparaison à cette ligne s'exécute $n + 1$ fois et les instructions dans le boucle seulement au plus n fois. Pour le reste des instructions en dehors de la boucle, elles s'exécutent en temps constant.

1.3 Question C

Dans le pire des cas : $\Theta(n)$.
Dans le meilleur des cas : $\Theta(1)$

1.4 Question D

Cas	tableau	Nombre manquant
Pire cas	$[0, 1, 2, 3, 4, ..., n-1]$	n
Meilleur cas	$[1, 2, 3, 4, 5, ..., n]$	0

1.5 Question E

1. **Fonction** TrouverNombreManquant (t, *bas*,*haut*)
2. Si $bas = haut$ alors
3. Si bas = $t[bas]$ alors
4. Renvoyer bas + 1
5. Sinon

6. Renvoyer bas
7. Fin Si
8. milieu$\leftarrow \lfloor \frac{bas+haut}{2} \rfloor$
9. Si milieu < t[milieu] alors
10. Renvoyer TrouverNombreManquant (t,bas,mileu)
11. Fin Si
12. Renvoyer TrouverNombreManquant (t,milieu + 1, haut)
13. Fin Fonction

Complexité temporelle : $\Theta(n)$

1.6 Question F

Si le tableau n'était pas trié, l'algorithme à la précédente question ne serait pas exacte, car l'algorithme assume que les éléments sont triées et donc se base sur des comparaisons plus grand ou plus petit. Donc, l'algorithme nous donnerait le mauvais résultat, car si le tableau n'est pas trié, il n'y a aucune garantit qu'une certaine partie du tableau est bien placé.

1.7 Question G

1. **Fonction** TrouverNombreManquant (t,n)
2. somme $\leftarrow \frac{n(n+1)}{2}$
3. Pour $i \leftarrow 1$ haut de n faire
4. somme $\leftarrow$ somme - t[i]
5. Fin Pour
6. Renvoyer somme
7. Fin Fonction

2 Exercise 2

2.0.1 Question A

Voici le travail effectué à chaque niveau de l'arbre :

1. Niveau 0 : 64
2. Niveau 1 : 3×32
3. Niveau 2 : 9×16
4. Niveau 3 : 27×8
5. Niveau 4 : 81×4

6. Niveau 5 : 243×2
7. Niveau 6 : 729×1

Le travail total est la sommation du travail à tous les niveaux qui est 2059.

2.1 Question B

Le nombre de noeuds dans l'arbre correspond aux nombres d'appels récursifs exécutés dans l'arbre qui est représentable par la somme suivante :

$$\sum_{i=0}^{\log_2 n} 3^i$$

Pour $n = 2048$, on a le résultat suivant :

$$\sum_{i=0}^{\log_2 2048} 3^i = \frac{1 - 3^{12}}{1 - 3} = 1093$$

2.2 Question C

$T(n) \in \Theta(n^{\log_2 3})$

3 Exercise 3

3.1 Question A

Algorithme possible :

1. **Fonction** ExisteI(tab, bas, haut)
2. Si bas = haut alors
3. Renvoyer tab[bas] = bas
4. Fin Si
5. milieu $\leftarrow \lfloor \frac{bas+haut}{2} \rfloor$
6. Si tab[milieu] < milieu
7. Renvoyer ExisteI(tab, milieu + 1, haut)
8. Fin si
9. Renvoyer ExisteI(tab, bas, milieu)
10. Fin Fonction

3.2 Question B

L'algorithme est juste, car la logique est que si la borne inférieur (bas) et la borne supérieur (haut) ne sont pas égale, alors il serait possible qu'il existe un nombre satisfaisant les propriétés du problème se trouvent entre les deux. En effet, tous les éléments avant la borne inférieur et après la borne supérieur sont considérés comme décalé comparé à leurs indices respectifs et le milieu permet de déterminer vers quelle partie du tableau qu'il faudrait regarder. Ce qui fait que lorsque la borne supérieur et inférieur se croisent, alors il y a deux scénarios possibles :

1. Si la borne inférieur est égale à l'élément du tableau à l'indice bas, alors il existe un nombre qui satisfait les contraintes du problème.
2. Si la borne inférieur n'est pas égale à l'élément du tableau à l'indice bas, alors il n'existe pas de nombre qui satisfait les contraintes du problème.

De plus, vu que les éléments sont distincts, il n'a pas de "milieu" entre deux éléments adjacents et ni même de possibilité qu'un élément en double soit sur la bonne case.

3.3 Question C

Si le tableau n'était pas trié, l'algorithme à la préédente question ne serait pas exacte, car l'algorithme assume que les éléments sont triées et donc se base sur des comparaisons plus grand ou plus petit pour conclure quel partie du tableau est décalé comparé à son indice. Donc, l'algorithme nous donnerait le mauvais résultat, car si le tableau n'est pas trié, il n'y a aucune garantit de savoir quel élément est décalé avec sa position dans le tableau.

4 Exercise 4

1. $T(n) \in \Theta(n^{\log_5 2})$
2. $T(n) \in \Theta(n \log n)$
3. $T(n) \in \Theta(n^{\log_2 3})$