Commencé le	Monday 9 December 2024, 12:00
État	Terminé
Terminé le	Monday 9 December 2024, 12:36
Temps mis	36 min 51 s
Points	2,00/5,00
Note	**8,00** sur 20,00 (**40**%)

Description

(FR) Rappel du théorème général.
Pour une récurrence du type $T(n) = aT(n/b + \mathrm{O}(1)) + f(n)$ avec $a \geq 1,\ b > 1$:

- si $f(n) = \mathrm{O}(n^{(\log_b a)-\varepsilon})$ pour un $\varepsilon > 0$, alors $T(n) = \Theta(n^{\log_b a})$;
- si $f(n) = \Theta(n^{\log_b a})$, alors $T(n) = \Theta(n^{\log_b a} \log n)$;
- si $f(n) = \Omega(n^{(\log_b a)+\varepsilon})$ pour un $\varepsilon > 0$, et de plus $af(n/b) \leq cf(n)$ pour un $c < 1$ et toutes les grandes valeurs de n, alors $T(n) = \Theta(f(n))$.

(EN) Master Theorem.
For a recurrence equation of the form $T(n) = aT(n/b + \mathrm{O}(1)) + f(n)$ where $a \geq 1,\ b > 1$:

- if $f(n) = \mathrm{O}(n^{(\log_b a)-\varepsilon})$ for some $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$;
- if $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$;
- if $f(n) = \Omega(n^{(\log_b a)+\varepsilon})$ for some $\varepsilon > 0$, and additionally $af(n/b) \leq cf(n)$ for some $c < 1$ and all large values n, then $T(n) = \Theta(f(n))$.

Question **1**

Partiellement correct

Note de 0,67 sur 2,00

(FR) Dans la liste d'équations qui suivent, lesquelles ont pour solution $T(n) = \Theta(n \log n)$?
(EN) In the following list of recurrence equations which ones have for solution $T(n) = \Theta(n \log n)$?

Veuillez choisir au moins une réponse.

- [ ] $T(n) = 3T(n/3) + \Theta(\sqrt{n})$
- [ ] $T(n) = 2T(n/3) + \Theta(\sqrt{n})$
- [x] $T(n) = 2T(n/3) + \Theta(n \log n)$ ✔
- [ ] $T(n) = 2T(n/2) + \Theta(\sqrt{n})$
- [x] $T(n) = 2T(n/2) + \Theta(n)$ ✔
- [ ] \(T(n)=3T(n/2)+\Theta (n)\)
- [ ] \(T(n)=3T(n/2)+\Theta (\sqrt {n})\)
- [x] \(T(n)=3T(n/3)+\Theta (n\log n)\) ✖
- [ ] \(T(n)=2T(n/3)+\Theta (n)\)
- [x] \(T(n)=3T(n/3)+\Theta (n)\) ✔
- [ ] \(T(n)=3T(n/2)+\Theta (n\log n)\)
- [x] \(T(n)=2T(n/2)+\Theta (n\log n)\) ✖

Les réponses correctes sont : \(T(n)=2T(n/2)+\Theta (n)\)
, \(T(n)=3T(n/3)+\Theta (n)\)
, \(T(n)=2T(n/3)+\Theta (n\log n)\)

Question **2**

Correct

Note de 1,00 sur 1,00

$T(n)=\begin{cases}1 & \text{if } n\in\{1,2,3\}\\T(\lfloor n/2\rfloor) + T(\lfloor n/4\rfloor) +n & \text{if }n\ge 4\end{cases}$

(FR) À quelle classe de complexité appartient la fonction $T(n)$ solution de l'équation ci-dessus ?

(EN) What is the complexity class of the function $T(n)$ satisfying the above recurrence equation?

Veuillez choisir au moins une réponse.

- [ ] $T(n)=\Theta(n^3)$
- [ ] $T(n)=\Theta(1)$
- [ ] $T(n)=\Theta(\log n)$
- [ ] $T(n)=\Theta(n\log n)$
- [ ] $T(n)=\Theta(n^2)$
- [x] $T(n)=\Theta(n)$
- [ ] $T(n)=\Theta(n^2\log n)$
- [ ] $T(n)=\Theta((\log n)^2)$

Votre réponse est correcte.

This equation cannot be solved with the Master Theorem in its present form. Because of the term $+n$ in the recursive case, it is clear that $T(n)=\Omega(n)$. If we neglect the term $T(n/4)$ corresponding to the smallest recursion , the Master Theorem applies and gives a solution of $\Theta(n)$. So it is plausible that $T(n)$ could be $\Theta(n)$ even with this term is not ignored. We just have to prove $T(n)=O(n)$ by induction.

Let's consider the hypothesis H_n that $T(n)\le cn$ for some constant c we'll determine later.

H_1,H_2,H_3 are true if $c\ge 1$.

For $n>3$, let's assume $H_1,...,H_{n-1}$. Then $T(n)= T(\lfloor n/2\rfloor) + T(\lfloor n/4\rfloor) +n \le \frac{3cn}{4} + n$. It follows that $T(n)\le cn - (\frac{cn}{4}-n)$ and therefore $T(n)\le cn$ if $c\ge 4$.

So for c=4, our induction demonstrates that $T(n)\le 4n$ for any $n\ge 1$. This implies $T(n)=O(n)$.

Since we have both $T(n)=O(n)$ and $T(n)=\Omega(n)$, we conclude that $T(n)=\Theta(n)$.

La réponse correcte est :
$T(n)=\Theta(n)$

Question **3**

Terminé

Non noté

```python
def hanoi(n, source, target, auxiliary):
    if n == 1:
        print("Move disk 1 from", source, "to", target)
        return
    hanoi(n-1, source, auxiliary, target)
    print("Move disk", n, "from", source, "to", target)
    hanoi(n-1, auxiliary, target, source)

# Example usage
hanoi(3, 'A', 'C', 'B')
```

(FR) La fonction récursive `hanoi` ci-dessus résoud le problème des tours de Hanoï avec n disques. Le problème en lui-même n'est pas important.
Indiquez combien d'appels à `print` sont effectués en fonctions de $n>0$ lorsqu'on execute `hanoi(\(n\),’A’,’B’,’C’)`?
(EN) The above recursive function (`hanoi`) solves the Towers of Hanoi problem for n disks. The problem itself is not important.
Specify how many calls to `print` are made as a function of n when `hanoi(\(n\),’A’,’B’,’C’)` is run.

Veuillez choisir une réponse.

- ○ $n\lfloor \log _2 n\rfloor$
- ○ $2n+1$
- ○ $2n-1$
- ○ $n-1$
- ○ $(n+1)^2$
- ○ $(n-1)^2$
- ○ 2^{n+1}
- ○ $n\lceil \log _2 n\rceil$
- ○ 2^n
- ○ $2n$
- ○ n^2
- ◉ 2^{n-1}
- ○ $n+1$
- ○ n

La réponse correcte est :

2^n-1

Description

```
def is_safe(board, row, col, n):
    # Check if there's a queen in the same column
    for i in range(row):
        if board[i][col] == 1:
            return False

    # Check upper left diagonal
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    # Check upper right diagonal
    for i, j in zip(range(row, -1, -1), range(col, n)):
        if board[i][j] == 1:
            return False

    return True

def solve_n_queens_util(board, row, n):
    if row == n:
        return True

    for col in range(n):
        if is_safe(board, row, col, n):
            board[row][col] = 1
            res = solve_n_queens_util(board, row + 1, n)
            if res:
                return res
            board[row][col] = 0
    return False

def solve_n_queens(n):
    board = [[0] * n for _ in range(n)]
    if solve_n_queens_util(board, 0, n):
        return board
    return None
```

(FR) L'algorithme ci-dessus cherche à placer \(n\) reines sur un échiquier de taille \(n\times n\) de façon à ce que deux reines ne partagent pas une même ligne, colonne, ou diagonale. Quand une telle position est trouvée, elle est retournée sous la forme d'un tableau de taille \(n\times n\) où les \(1\) représentent des cases occupées par les reines, et les \(0\) des cases vides.
(EN) The above algorithm attempt to place \(n\) queens on a chess board of size \(n\times n\) is such a way that two queen never share a row, column, or diagonal. When such a position is found, it is returned as a \(n\times n\) array in which \(0\) denotes an empty square, and \(1\) denotes a queen.

Question **4**

Partiellement correct

Note de 0,33 sur 1,00

(FR) Quelle est la complexité de la fonction `is_safe`, en fonction de \(n\)? Cochez toutes les réponse correctes.
(EN) What is the complexity of function `is_safe`, as a function of \(n\)? Select all correct answers.

Veuillez choisir au moins une réponse.

- [ ] \(O(n^2)\)
- [x] \(O(n)\) ✔
- [ ] \(\Theta (n)\)
- [ ] \(\Theta (\log n)\)
- [ ] \(O(1)\)
- [ ] \(\Theta (1)\)
- [ ] \(\Theta (n^2)\)
- [ ] \(O(\log n)\)

Les réponses correctes sont : \(\Theta (n)\)
, \(O(n)\)
, \(O(n^2)\)

Question **5**

Terminé

Non noté

(FR) Lors d'un appel de `solve_n_queens(n)`, combien fait-on d'appels à `solve_n_queens_util`, en fonction de \(n\)? Cochez toutes les réponse correctes.
(EN) In one call to `solve_n_queens(n)`, how many calls to `solve_n_queens_util` do we do, as a function of \(n\)? Select all correct answers.

Veuillez choisir au moins une réponse.

- [x] \(\Theta (1)\)
- [ ] \(\Theta (n!)\)
- [ ] \(\Theta (\log n)\)
- [ ] \(O(n^n)\)
- [ ] \(O(n^2)\)
- [ ] \(O(n)\)
- [ ] \(O(n!)\)
- [ ] \(O(\log n)\)
- [ ] \(\Theta (n^2)\)
- [ ] \(\Theta (n^n)\)
- [ ] \(O(1)\)
- [ ] \(\Theta (n)\)

Les réponses correctes sont : \(O(n!)\)
, \(O(n^n)\)

Question **6**

Incorrect

Note de 0,00 sur 1,00

(FR) On dispose d'un tableau $A[1..n]$ non-trié contenant n entiers, et dont les indices commencent à 1, et on vient d'exécuter l'algorithme QuickSelect, vu en cours, pour y trouver la valeur de rang k. L'algorithme retourne cette valeur, mais ignorons là pour la suite. Cette question concerne l'état du tableau A après l'exécution de QuickSelect pour trouver le rang k. Parmis les propositions qui suivent, indiquez toutes celles que sont justes.
(EN) We have an unsorted array $A[1..n]$ containing n integers, with indices starting at 1, and we have just executed the QuickSelect algorithm, seen in the course, to find the value of rank k. The algorithm returns this value, but let's discard it. This question concerns the state of the array A after executing QuickSelect to find the rank k. Among the following propositions, indicate all those that are correct.

Veuillez choisir au moins une réponse.

- [ ] Si je veux chercher un autre rang, je peux le trouver en temps constant / If I want to find another rank, I can find it in constant time.
- [x] A n'est pas nécessairement trié / A isn't necessarily sorted ✔
- [x] La position de la valeur de rang k dépend des autres valeurs présentes dans le tableau / the position of the value of rank k depends on the other values present in the array ✖
- [ ] A est forcément trié / A is necessarily sorted
- [ ] La valeur de rang k se trouve en $A[1]$ / the value of rank k is at $A[1]$
- [ ] La valeur de rang k se trouve en $A[k]$ / the value of rank k is at $A[k]$
- [ ] Si je veux chercher un autre rang, je peux tirer partie de la nouvelle organisation de A pour faire moins de travail / If I want to find another rank, I can benefit from the new order of A to perform less work.
- [x] Une moitiée du tableau A est triée / Half of the array A is sorted ✖
- [ ] La valeur de rang k se trouve en $A[n]$ / the value of rank k is at $A[n]$

Les réponses correctes sont : A n'est pas nécessairement trié / A isn't necessarily sorted
, La valeur de rang k se trouve en $A[k]$ / the value of rank k is at $A[k]$
, Si je veux chercher un autre rang, je peux tirer partie de la nouvelle organisation de A pour faire moins de travail / If I want to find another rank, I can benefit from the new order of A to perform less work.

◄ CPXA

Aller à...

CPXA ►