Exercice 3 (8 pts)

Dans cet exercice, on s'intéresse au *problème du sous-tableau maximum.* Etant donné un tableau $A[1,\ldots,n]$, le problème du sous-tableau maximum de A est de déterminer :

$$\mathrm{stm}(A) = \max\ \{\sum_{k=i}^{j} A[k] : 1 \leq i \leq j \leq n\}$$

Par exemple, on a $\mathrm{stm}([-3,4,5,-1,2,3,-6,4]) = 13$, valeur que l'on obtient pour le sous-tableau $[4,5,-1,2,3]$. Dans un premier temps, on considère l'algorithme de résolution ci-dessous :

```
pour i = 1 à n faire
    pour j = i à n faire
        temp = 0
        pour k = i à j faire
            temp = temp + A[k]
        si temp > max alors
            max = temp
retourner max
```

Q 3.1 Quelle est la complexité de cet algorithme ?

1/20

$$\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=i}^{j} 1 = \sum_{i=1}^{n}\sum_{j=i}^{n}(j-i+1) = \sum_{i=1}^{n}\sum_{j=1}^{n-i+1} 1$$

$$= \sum_{i=1}^{n}\frac{(n-i+1)(n-i+2)}{2} = \sum_{i=0}^{n-1}\frac{(i+1)(i+2)}{2} \in \Theta(n^3)$$

Q 3.2 Proposer une variante en $\Theta(n^2)$ de cet algorithme.

1/20

```
max = max{A[1],..., A[n]}
pour i = 1 à n faire
    temp = 0
    pour j = i à n faire
        temp = temp + A[j]
        si temp > max alors
            max = temp
retourne max
```

Dans la suite de l'exercice, on s'intéresse à des algorithmes de type diviser pour régner pour ce problème. Soit $m = \lfloor (n+1)/2 \rfloor$, $A_1 = A[1,\ldots,m]$ et $A_2 = A[m+1,\ldots,n]$. Le cas de base est le cas d'un tableau de taille 1, qu'on sait bien sûr résoudre en $\Theta(1)$ (temps constant).

Q 3.3 Soit $\mathrm{stm}_1 = \mathrm{stm}(A_1)$, $\mathrm{stm}_2 = \mathrm{stm}(A_2)$ et stm_3 la valeur maximum d'un sous-tableau qui contient à la fois $A[m]$ et $A[m+1]$. Comment calculer stm_3 efficacement ? Quelle est la complexité en fonction de n ?

1/20

$$stm_3 = \max_{i \leq m} \sum_{k=i}^{m} A[k] + \max_{j \geq m+1} \sum_{k=m+1}^{j} A[k]$$

On calcule $\sum_{k=i}^{m} A[k]$ avec une boucle en $\Theta(n)$

On calcule $\sum_{k=m+1}^{j} A[k]$ avec une boucle en $\Theta(n)$

On somme les deux résultats.

Q 3.4 Dans le cadre d'une méthode diviser pour régner, supposons que l'on a calculé $\max_1$ et $\max_2$ (par des appels récursifs). En vous aidant de la réponse à la question précédente, donner le principe d'une opération de fusion des résultats des deux appels récursifs, et indiquer la complexité de cette opération.

1/20

Pour l'opération de fusion, on a :

$$stm(A) = \max\{stm_1, stm_2, stm_3\}$$

La complexité de la fusion est donc liée au calcul de stm_3, qui se fait en $\Theta(n)$.

Q 3.5 Soit $T(n)$ la complexité de la méthode diviser pour régner pour un tableau de taille n. Donner l'équation de récurrence satisfaite par $T(n)$. A l'aide du théorème maître, en déduire la complexité de la méthode.

1/20

$$T(n) = 2T(n/2) + \Theta(n)$$

Théorème-maître avec $a=2$, $b=2$ et $d=1$

$$\log_b a = 1 = 1 = d$$

$\rightarrow \Theta(n \log n)$

On s'intéresse désormais à une autre approche, toujours de type diviser pour régner, afin de réduire encore la complexité de l'algorithme. Pour ce faire, on définit :

$$\begin{array}{rcl} \text{pref}(A) & = & \max\left\{\sum_{k=1}^{i} A[k] : 1 \leq i \leq n\right\}, \\ \text{suff}(A) & = & \max\left\{\sum_{k=i}^{n} A[k] : 1 \leq i \leq n\right\}, \\ \text{tota}(A) & = & \sum_{k=1}^{n} A[k], \end{array}$$

Autrement dit, pref(A) (resp. suff(A)) est la valeur maximum d'un sous-tableau préfixe (resp. suffixe) de A, et tota(A) est la somme des cases de A.

Q 3.6 Supposons qu'on vise à déterminer non plus simplement stm(A), mais le triplet (stm(A), pref(A), suff(A)). Soit ($\text{stm}_1, \text{pref}_1, \text{suff}_1$) et ($\text{stm}_2, \text{pref}_2, \text{suff}_2$) les triplets retournés par les appels récursifs sur A_1 et A_2. Expliquer pourquoi ce serait faux de faire la fusion en retournant ($\max\{\text{suff}_1 + \text{pref}_2, \text{stm}_1, \text{stm}_2\}, \text{pref}_1, \text{suff}_2$) pour A.

1/20

Ce serait faux car on peut avoir :

$\text{pref}(A_1) \neq \text{pref}(A)$

$\text{suff}(A_2) \neq \text{suff}(A)$

Par exemple :

$$A = \underbrace{\begin{array}{|c|c|}\hline 1 & -2 \\ \hline\end{array}}_{A_1}\underbrace{\begin{array}{|c|c|}\hline 3 & -1 \\ \hline\end{array}}_{A_2}$$

On a $\text{pref}(A_1) = 1$ et $\text{pref}(A) = 2$ (pour $\begin{array}{|c|c|c|}\hline 1 & -2 & 3 \\ \hline\end{array}$)

Q 3.7 Supposons maintenant qu'on vise à déterminer le quadruplet (stm(A), pref(A), suff(A), tota(A)). Soit ($\text{stm}_1, \text{pref}_1, \text{suff}_1, \text{tota}_1$) et ($\text{stm}_2, \text{pref}_2, \text{suff}_2, \text{tota}_2$) les quadruplets retournés par les appels récursifs sur A_1 et A_2. Donner la formule permettant de déterminer le quadruplet pour A à partir de stm_1, pref_1, suff_1, tota_1, stm_2, pref_2, suff_2, tota_2. Quelle est la complexité de cette opération de fusion ?

1/20

La formule est la suivante :

$(\max\{\text{stm}_1, \text{stm}_2, \text{suff}_1 + \text{pref}_2\}, \max\{\text{pref}_1, \text{tota}_1 + \text{pref}_2\}, \max\{\text{suff}_2, \text{suff}_1 + \text{tota}_2\}, \text{tota}_1 + \text{tota}_2)$

La complexité de cette opération de fusion est $\Theta(1)$

Q 3.8 Soit $T(n)$ la complexité de la méthode diviser pour régner correspondante pour un tableau de taille n. Donner l'équation de récurrence satisfaite par $T(n)$. A l'aide du théorème maître, en déduire la complexité de cette nouvelle méthode.

1/20

$T(n) = 2T(n/2) + \Theta(1)$

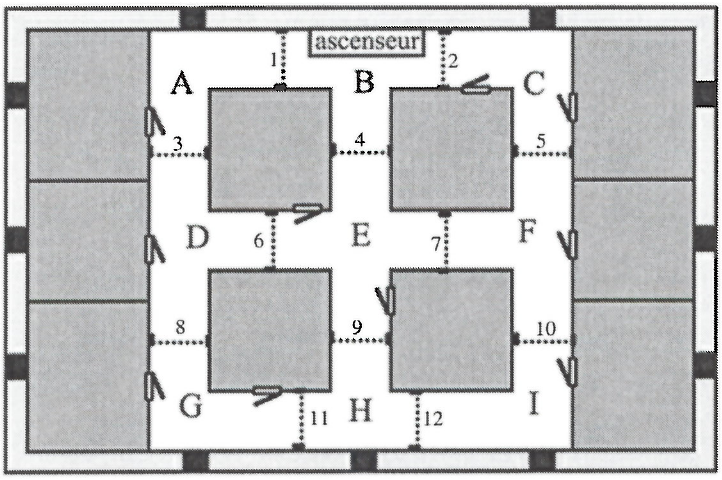
Théorème-maître avec $a=2$, $b=2$ et $d=0$

$\log_b a = 1 > 0 = d$

$\rightarrow \Theta(n)$

Exercice 4 (5 pts)

Une souris-garou s'est échappée de sa cage située dans un des bureaux du Laboratoire de Biologie, pour aller se cacher dans une des bouches d'aération ou derrière une armoire dans un bureau. L'objet de cet exercice est de localiser la souris-garou avant la prochaine pleine lune afin d'éviter qu'un(e) biologiste ne se fasse croquer. Le plan des locaux du laboratoire figure ci-dessous. Les onze carrés noirs dans les murs figurent les bouches d'aération et les rectangles grisés les bureaux (l'ascenseur n'a aucune importance).



Les locaux sont munis de douze capteurs de mouvement. Ils sont représentés par les lignes pointillées sur le plan. A chaque fois qu'on traverse une ligne pointillée (peu importe dans quel sens) le compteur du capteur correspondant est incrémenté. La nuit de l'évasion de la souris-garou, le relevé des compteurs donne les résultats suivants (le laboratoire était désert, il s'agit donc bien à chaque fois de la souris-garou) :

N° du capteur	1	2	3	4	5	6	7	8	9	10	11	12
Relevé du compteur	2	1	3	3	1	3	0	2	2	3	2	1