INF5130 — Devoir 1

Samuele Giraudo

Session 2025-01

Déclaration. *J'affirme, pour ce devoir, n'avoir reçu aucune aide d'aucune forme que ce soit, incluant les aides par des personnes et par des outils génératifs d'intelligence artificielle. Ceci vaut pour toute la période du devoir, depuis sa date de publication initiale à sa date de rendu finale.*

Je comprends que si j'ai bénéficié de telles aides et que je remplis cette déclaration, elle devient fallacieuse. Cela peut entraîner des sanctions prévues en conformité avec ce qui sera décidé par le comité des infractions académiques.

Répondre « J'affirme que tout ceci est exact. » et signer par prénom, nom et code permanent. Si ceci manque, le devoir ne pourra pas être évalué.

Réponse —

Conventions générales. Tout tableau t de longueur n est indicé de 0 à $n-1$. La notation $t[i]$ désigne la case mémoire à l'indice i de t. La fonction Longueur renvoie la longueur du tableau en argument en temps constant. La fonction CréerTableau prend en entrée un entier naturel n et renvoie un tableau de longueur n dont toutes les cases contiennent la valeur 0. La complexité en temps de cette fonction dans le pire et meilleur cas est linéaire en n.

1 Manipulation de sommes

Question 1.1. *(5 points)* — Soit n un entier naturel. Exprimer la quantité

$$\sum_{i=0}^{n}(2i+1) \tag{1}$$

par une expression polynomiale dépendant de n la plus simplifiée possible. Le calcul doit être détaillé et justifié.

Réponse —

Question 1.2. *(5 points)* — Soient α et β deux entiers tels que $\alpha \leq \beta$. Exprimer la quantité

$$\sum_{i=\alpha}^{\beta} i \tag{2}$$

par une expression polynomiale dépendant de α et de β la plus simplifiée possible. Le calcul doit être détaillé et justifié.

Réponse —

Question 1.3. *(5 points)* — Soit n un entier naturel. Exprimer la quantité

$$\sum_{i=0}^{n}\sum_{j=0}^{m}(i+j) \tag{3}$$

par une expression polynomiale dépendant de n et de m la plus simplifiée possible. Le calcul doit être détaillé et justifié.

Réponse —

2 Complexité en temps

Question 2.1. *(5 points)* — Considérons l'algorithme décrit par le pseudo-code suivant.

1. **Fonction** ChercherMot(u, v)
2. $j \leftarrow 0$
3. **Tant que** $j \leq$ Longueur(v) − Longueur(u) **Faire**
4. $i \leftarrow 0$
5. **Tant que** $i \leq$ Longueur(u) − 1 **Et** $u[i] = v[j + i]$ **Faire**
6. $i \leftarrow i + 1$
7. **Fin**
8. **Si** $i =$ Longueur(u) **Alors**
9. **Renvoyer** j
10. **Fin**
11. $j \leftarrow j + 1$
12. **Fin**
13. **Renvoyer** -1
14. **Fin**

Cet algorithme prend en entrée deux tableaux u et v dont les valeurs sont des caractères. Donner la description du problème auquel cet algorithme répond. La réponse doit être justifiée. Elle peut s'appuyer sur des exemples bien choisis qui illustrent le comportement de l'exécution de cet algorithme.

Réponse —

Question 2.2. *(5 points)* — En considérant l'algorithme ChercherMot de la question 2.1, identifier une opération barométrique qui permettra de faire une étude de la complexité en temps de cet algorithme en fonction de n, la longueur de u et m, la longueur de v. La réponse doit être justifiée. Il est possible de répondre à cette question même si la question 2.1 n'a pas été (correctement) traitée.

Réponse —

Question 2.3. *(5 points)* — En considérant l'algorithme ChercherMot de la question 2.1 et l'opération barométrique identifiée dans la réponse à la question 2.2, donner une expression $T(n, m)$ en fonction de n et m (les longueurs respectives des tableaux u et v) qui renseigne sur le nombre d'opérations barométriques exécutées dans le pire cas. La réponse doit être justifiée. Il est possible de répondre à cette question même si la question 2.1 n'a pas été (correctement) traitée.

Réponse —

3 Écriture d'algorithmes

Question 3.1. *(10 points)* — Soit t un tableau de longueur n. Ce tableau t est une *permutation* si l'ensemble des valeurs de t est $\{1, \ldots, n\}$. Ceci est équivalent au fait que toute valeur entre 1 et n apparaît exactement une fois dans t. Écrire un algorithme EST-PERMUTATION qui prend en entrée un tableau t d'entiers et qui renvoie vrai si t est une permutation et faux sinon. Il faut justifier brièvement du fait qu'il répond à ce problème.

Réponse —

Question 3.2. *(10 points)* — Un indice i d'un tableau t d'entiers de longueur $n \geq 1$ est un *maximum local* si pour tout indice j de t tel que $i < j$, $t[i] > t[j]$. Par exemple, le tableau $t = (6, 0, 6, 0, 2, 1)$ est tel que les indices 2, 4 et 5 sont des maximums locaux. Écrire un algorithme NBMAXLOCAUX qui prend en entrée un tableau t d'entiers et qui renvoie le nombre de maximums locaux dans t. En considérant le tableau t de l'exemple précédent, l'algorithme renvoie 3. Il est demandé que la complexité en temps dans le pire cas de l'algorithme proposé soit linéaire en fonction de la longueur de t. Il faut justifier brièvement du fait qu'il répond à ce problème et que sa complexité est bien linéaire.

Réponse —

4 Croissance des fonctions

Question 4.1. *(5 points)* — Soient les fonctions $f : \mathbb{N} \to \mathbb{R}^+$ et $g : \mathbb{N} \to \mathbb{R}^+$ telles que pour tout $n \in \mathbb{N}$, $f(n) = n^4 + 9n^3 - 6n^2 + n - 7$ et $g(n) = n^4$. Démontrer que $f \in \mathcal{O}(g)$ en donnant explicitement des constantes n_0 et c telles que pour tout $n \geq n_0$, $f(n) \leq cg(n)$.

Réponse —

Question 4.2. *(5 points)* — Une relation binaire $\mathcal{R}$ sur un ensemble A est un *préordre* si $\mathcal{R}$ est réflexive et transitive.

Soit la relation binaire $\mathcal{R}_\mathcal{O}$ sur l'ensemble F des fonctions de domaine $\mathbb{N}$ et de codomaine $\mathbb{R}^+$ telle que pour toutes fonctions $f_1, f_2 \in F$, $f_1 \,\mathcal{R}_\mathcal{O}\, f_2$ si $f_1 \in \mathcal{O}(f_2)$. Démontrer que $\mathcal{R}_\mathcal{O}$ est un préordre.

Réponse —

5 Équations de récurrence et approche « diviser pour régner »

Question 5.1. *(5 points)* — En appliquant la méthode de résolution des équations de récurrence aux différences finies, résoudre la récurrence

$$T(n) = T(n-1) + n^2, \quad \text{si } n \geq 1, \tag{4}$$

$$T(0) = 0. \tag{5}$$

L'expression donnée pour $T(n)$ ne doit pas être récursive. En se basant sur l'expression obtenue, démontrer que $T \in \Theta(n^3)$.

Réponse —

Question 5.2. *(10 points)* — Considérons l'algorithme décrit par le pseudo-code suivant.

1. **Fonction** ChercherÉlément(t, x, α, β)
2. **Si** $\alpha > \beta$ **Alors**
3. **Renvoyer** -1
4. **Fin**
5. $m \leftarrow \left\lfloor \frac{\alpha + \beta}{2} \right\rfloor$
6. **Si** $t[m] = x$ **Alors**
7. **Renvoyer** m
8. **Sinon Si** $t[m] > x$ **Alors**
9. **Renvoyer** ChercherÉlément(t, x, α, $m-1$)
10. **Sinon**
11. **Renvoyer** ChercherÉlément(t, x, $m+1$, β)
12. **Fin**
13. **Fin**

Cet algorithme prend en entrée un tableau t dont les valeurs sont des entiers. Ce tableau est de plus trié dans l'ordre croissant de la gauche vers la droite. Il peut contenir plusieurs occurrences d'une même valeur. L'algorithme prend de plus en entrée un entier x, et deux indices α et β tous deux compris entre 0 et la longueur de t moins 1. La fonction ChercherÉlément renvoie un indice i tel que $t[i] = x$ s'il existe et renvoie -1 dans le cas contraire (c'est-à-dire lorsque t ne possède aucune occurrence de x).

Donner, en la justifiant, la fonction de coût $T(n)$ de la forme

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \tag{6}$$

qui mesure la complexité en temps dans le pire cas de cet algorithme, où n est la longueur du tableau t. Il faut ainsi donner les valeurs des constantes a et b, et donner une définition de la fonction $f : \mathbb{N} \to \mathbb{N}$.

Utiliser ensuite le théorème général de résolution des équations de récurrence aux divisions finies pour donner une fonction $g : \mathbb{N} \to \mathbb{R}^+$ telle que $T \in \Theta(g)$.

Réponse —

6 Programmation dynamique

Question 6.1. *(15 points)* — Considérons une grille de dimension $n \times m$ avec $n, m \in \mathbb{N}$. Un *chemin* dans cette grille est une suite de pas *Nord* (noté N, qui est un segment vertical de longueur 1) ou *Est* (noté E, qui est un segment horizontal de longueur 1), débutant à l'origine $(0, 0)$ et arrivant au point (n, m). Un chemin est *valide* s'il reste au dessus du segment connectant l'origine et le point (n, m) de la grille. La notion d'« être au dessus » est prise au sens large : un chemin peut toucher la diagonale tout en restant valide — il faut simplement qu'il ne la traverse pas.

Par exemple, pour $n = 4$ et $m = 9$, le chemin $NENNEEEEENEEE$ est valide tandis que le chemin $NNEEEEEENNEEE$ est invalide. Il est fortement conseillé de dessiner la grille et ces deux chemins pour se familiariser avec ces notions (ces dessins ne font cependant pas partie du rendu attendu).

Proposer un algorithme CheminsValides qui prend en entrées les entiers naturels n et m, et qui renvoie le nombre de chemins valides dans la grille $n \times m$. Il faut utiliser le principe de la programmation dynamique en enregistrant, pour tout point (k, ℓ) de la grille avec $0 \leq k \leq n$ et $0 \leq \ell \leq m$, le nombre de débuts de chemins valides qui commencent en $(0, 0)$ et s'arrêtent en (k, ℓ).

En implantant cet algorithme et en l'exécutant, donner les valeurs renvoyées par CheminsValides(n, n) pour tout $0 \leq n \leq 8$ ainsi que par CheminsValides$(n, 2n)$ pour tout $0 \leq n \leq 8$. Le programme utilisé pour ces calculs ne doit pas faire partie du rendu.

Réponse —