Devoir Maison n°01

4 novembre 2023

Exercice 1

On rappelle qu'une fonction f est une fonction polynomiale de degré inférieur ou égal à n si elle est de la forme $f(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n = \sum_{i=0}^{n} a_ix^i$.

1) Pour i et a_i donnés, que vaut la dérivée de $f_1 : x \mapsto a_ix^i$? Pour $a_1, ..., a_n \in \mathbb{R}$ donnés, que vaut la dérivée de $f_2 : x \mapsto \sum_{i=0}^{n} a_ix^i$?

2) Ecrire une fonction en OCaml qui prend en entrée une liste de coefficients réels (sous la forme de flottants) $a_0, ..., a_n$ et un réel x_0, et renvoie la valeur de $f(x_0)$, où $f(x) = \sum_{i=0}^{n} a_ix^i$. On considérera que la liste vide correspond au polynôme nul. (On pourra utiliser une fonction auxiliaire récursive)

3) Ecrire une fonction en OCaml qui prend en entrée une liste de coefficients réels (sous la forme de flottants) $a_0, ..., a_n$ d'une fonction polynomiale $f : x \mapsto \sum_{i=0}^{n} a_ix^i$, et qui renvoie la liste de coefficients de la fonction polynomiale dérivée f'.

4) La *descente en gradient* est une méthode permettant, pour une application de $\mathbb{R}^n$ dans $\mathbb{R}$, d'*approcher un minimum local* de cette fonction. Le principe est, partant d'un point X dans $\mathbb{R}^n$, de se déplacer à partir de X dans le sens opposé du gradient (dont les coordonnées sont les dérivées selon chacune des directions dans $\mathbb{R}^n$), avec un facteur $\alpha > 0$, puis on réitère l'opération à partir du nouveau point un certain nombre de fois. Au fur et à mesure que l'on s'approche d'un minimum local, la norme du gradient baisse et le déplacement vers le minimum se ralentit. Si l'on atteint le minimum local, le gradient s'annule et la suite stationne.

En particulier, pour une fonction f de $\mathbb{R}$ dans $\mathbb{R}$, la descente en gradient à partir d'un point x_0 est donnée par $x_{n+1} = x_n - \alpha f'(x_n)$.

Ecrire une fonction en OCaml qui prend en entrée la liste des coefficients $a_0, ..., a_n$ d'une fonction polynomiale $f : x \mapsto \sum_{i=0}^{n} a_ix^i$, une valeur initiale x_0, un pas α et un nombre d'itérations n, et renvoie le couple $(x_n, f(x_n))$. (On pourra utiliser les fonctions définies précédemment. x_n est l'approximation du point en lequel f atteint un minimum local ; $f(x_n)$ est l'approximation

de ce minimum)

5) La *méthode de Newton* est une méthode permettant d'approcher les racines d'une fonction de $\mathbb{R}$ dans $\mathbb{R}$. Le principe est de partir d'un point x_0 et de calculer à chaque étape $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$.

Ecrire une fonction en OCaml qui prend en entrée la liste des coefficients $a_0, ..., a_n$ d'une fonction polynomiale $f : x \mapsto \sum_{i=0}^{n} a_i x^i$, une valeur initiale x_0 et un nombre d'itérations n et renvoie la valeur de x_n.

Exercice 2

On rappelle le principe du jeu des Tours de Hanoi : on dispose de trois tours (à gauche, au milieu et à droite). Sur l'une des trois tours se trouvent n disques de tailles différentes, le plus petit au sommet et le plus grand en bas. On cherche à déplacer la totalité des disques sur une des deux autres tours, avec les règles suivantes :

-A chaque étape, on déplace d'une tour à l'autre un unique disque (celui qui est au sommet) ;

-On peut placer un disque uniquement sur un disque de taille supérieure ou dans une tour vide.

1) Définir un type `position` dont les constructeurs sont `Centre`, `Gauche` et `Droite`.

2) Ecrire une fonction récursive `deplacements_hanoi` qui prend en entrée un nombre n de disques, trois positions correspondant aux tours de départ, d'intermédiaire et d'arrivée, et qui renvoie la liste des déplacements de disques à effectuer pour déplacer les n disques de la tour de départ à la tour d'arrivée.

Etant donné qu'à chaque déplacement, c'est le disque au sommet de la tour de départ qui est placé sur la tour d'arrivée, on pourra représenter un déplacement par un élément de type `position * position`, et la liste des déplacements par une liste de type `position * position list`.

3) On considère que pour une tour de Hanoi de taille n, les disques sont numérotés par taille croissante de 1 à n. L'état du jeu à un moment donné peut donc être défini par la donnée pour chacune des tours (gauche, centre et droite) de la liste des disques (représentés par des entiers) situés dessus.

Définir un type `etat` à l'aide d'un enregistrement, de champs `gauche`, `centre` et `droite` et dont les valeurs sont des listes d'entiers (correspondant à la liste des disques à chaque position, rangées par ordre croissant).

4) Définir une fonction `etats_hanoi` qui prend en entrée un entier n, des positions de départ, intermédiaire et arrivée, et qui renvoie la liste des états du jeu (éléments de type `etat`) au cours du déplacement. On pourra s'aider de la fonction précédente.

5) On note $C(n)$ le nombre d'appels récursifs dans l'appel de `hanoi` sur un jeu de taille n. Calculer $C(0)$, $C(1)$, $C(2)$, $C(3)$ et $C(4)$.

6) Calculer $C(n+1)$ en fonction de $C(n)$: soit (u_n) la suite définie par $u_n = C(n)$, quelle est la nature de la suite (u_n) ?

7) En déduire l'expression de $C(n)$ en fonction de n. Quel est l'ordre de grandeur de la complexité temporelle de `hanoi` ?

Exercice 3

Partie A

On considère la fonction de tri par insertion définie dans le TP 03 (exercice 25).

1) Montrer que la fonction termine.

2) Montrer que la fonction est correcte.

Partie B

On considère la fonction de tri fusion définie dans le TP 03 (exercice 26).

1) Montrer que la fonction termine.

2) Montrer que la fonction est correcte.