TD #8 — Nombres flottants (avec solutions)

Exercice 1. *Racine carrée entière*

On suppose vouloir calculer des *racines carrées entières* $\lfloor \sqrt{x} \rfloor$ d'entiers $x \in [\![0, 2^{64} - 1]\!]$.

1. Expliquez pourquoi il n'est pas envisageable d'utiliser des nombres flottants « à double précision » (avec mantisse (effective) de 53 bits) pour effectuer ce calcul (par exemple comme la composition des fonctions `int_of_float`, `sqrt` et `float_of_int` en OCaml).
   Demandez-moi pour un indice.

   Avec de tels nombres flottants, l'on a par exemple que 2^{62} et $2^{62} - 1$ ont la même représentation flottante (dont la mantisse est nulle), ce qui implique que `sqrt` s'évaluera identiquement sur les approximations flottantes de ces deux nombres. Or $\lfloor \sqrt{2^{62}} \rfloor = 2^{31}$ et $\lfloor \sqrt{2^{62} - 1} \rfloor < 2^{31}$; il est donc impossible d'obtenir un résultat correct pour les deux calculs.

Exercice 2. *Tri par base pour nombres flottants*

On rappelle quelques aspects de la représentation des nombres flottants sur 64 bits (« double précision ») IEEE754, qui sont les seuls « nombres flottants » que l'on considérera dans tous l'exercice. On utilisera pour cela (entre autres) des *masques* binaires (notés en hexadécimal) pour désigner des ensembles de bits.

— Le bit de signe (indiquant si le nombre est positif (bit à zéro) ou négatif (bit à un)) est le bit « le plus significatif », c'est à dire celui « le plus à gauche », ou d'indice 63. Il correspond au masque : `0x8000000000000000`.
— Les 11 bits d'exposant sont les bits les plus significatifs suivant (d'indices 62 à 52), qui correspondent au masque : `0x7FF0000000000000`. La valeur e de l'exposant quand non nul est donnée par l'entier que ces bits représentent en base deux (bit le plus significatif « à gauche ») auquel on soustrait 1023 (c'est une représentation *biaisée* de nombres relatifs).
— Les 52 bits restant représentent la mantisse, dont la valeur est donnée par 2^{-52} fois l'entier qu'ils représentent en base deux (bit le plus significatif « à gauche ») plus 2^{52} (le bit « gratuit »), hors cas particuliers que nous ignorerons. Le masque correspondant est : `0x000FFFFFFFFFFFFF`.

De plus on a que :

— Un nombre dont tous les bits de l'exposant sont à un représente $\pm\infty$ (en fonction du bit de signe) si tous les bits de la mantisse sont à zéro, et un `NaN` sinon.
— Un nombre dont tous les bits sont à zéro sauf éventuellement le bit de signe représente ± 0 en fonction de la valeur de ce dernier.
— Les représentations sont normalisées : l'unique représentation d'un nombre non nul (fini) représentable est celle d'exposant minimal, et sa valeur est $\pm 2^e \times m$, avec m la valeur représentée par la mantisse comme ci-dessus.

Exemples

— Une mantisse `0x0000000000004` représente $2^{-52} \times (2^{52} + 2^2)$, soit $1 + 2^{-50}$.
— Un exposant `0x431` représente l'exposant $1073 - 1023 = 50$.
— Un nombre flottant de représentation binaire `0x4310000000000004` représente le nombre $2^{50} \times (1 + 2^{-50}) = 2^{50} + 1$.

Dans tout ce qui suit, soit x un nombre réel représentable exactement par un nombre flottant, on note $\rho(x)$ l'entier $\in [\![0, 2^{64} - 1]\!]$ obtenu en interprétant sa représentation binaire (comme ci-dessus) comme l'écriture en base deux d'un entier naturel. Dans le cas particulier de $x = 0$, on note $\rho^+(x)$ et $\rho^-(x)$ ses représentations positives et négatives.

1. Montrez que l'on peut avoir $x \in \mathbb{Z}$ et $x \neq \rho(x)$.

On a par exemple que pour $x = -0$, $\rho(x) = 2^{63} \neq -0$.

2. Soit x, y deux nombres réels représentables exactement par des flottants non négatifs (où l'on considère `-0.0` comme étant négatif), montrez que $\rho(x)$ et $\rho(y)$ se comparent identiquement à x et y (autrement dit, que $x = y \Rightarrow \rho(x) = \rho(y)$, $x < y \Rightarrow \rho(x) < \rho(y)$, $x > y \Rightarrow \rho(x) > \rho(y)$).

Par les propriétés de la représentation IEEE754, l'on a notamment que :

- si l'exposant e de la représentation de x est supérieur à celui e' de la représentation de y, alors $\rho(x) > \rho(y)$ et $x > y$. En effet, la valeur minimale que peut représenter une mantisse est 1, et la valeur maximale μ est $2^{-52} \times (2^{52} + \sum_{i=0}^{51} 2^i = 2^{-52} \times (2^{52} + 2^{52} - 1) < 2$, (elle s'écrit aussi $1 + \sum_{i=1}^{52} 2^{-i}$), donc notamment $2^{e+1} \times 1 > 2^e \times \mu$.
- si ces deux exposants sont égaux, $\rho(x)$ est inférieur, égal ou supérieur à $\rho(y)$ en fonction de si la valeur représentée par sa mantisse est inférieure, égale ou supérieure à celle de y, et il en va de même pour x et y.

On conclut en observant que dans le format IEEE754 l'exposant est stocké dans les bits de poids fort, et la mantisse dans les bits de poids faible.

3. Cela reste-t'il vrai si l'on considère également des nombres négatifs ? Si non, que faut-il prendre en compte ?

C'est évidemment faux sans adaptation pour les nombres négatifs, puisque pour tout x négatif et y positif l'on a $x \leq y$ et $\rho(x) > \rho(y)$ (puisque le bit de signe est à `1` pour les nombres négatifs, et est stocké dans le bit de poids fort).
De plus, par la même analyse qu'à la question précédente on a que pour x, y tous deux négatifs $\rho(x)$ et $\rho(y)$ comparent *les valeurs absolues* de x et y, et donc « à l'inverse » de x et y.
Une fois ces deux aspects pris en compte, il est à nouveau possible de comparer deux flottants (représentant des nombres finis) par simple lecture de leurs représentations binaires.

4. Déduire de ce qui précède que la comparaison de deux nombres flottants (représentant des nombres finis) peut se faire essentiellement « sans calcul »

Cf. la réponse à la question précédente, et le fait que comparer deux nombres entiers écrits en base deux se fait par comparaison des chiffres, et donc « sans calcul ».

5. En déduire également qu'il est possible de trier un tableau de tels nombres en temps linéaire en la longueur du tableau (on pourra se contenter de détailler le cas de nombres non négatifs).

Par ce qui précède, il suffit dans ce cas de trier le tableau en interprétant les représentations binaires des flottants comme des entiers. Puisque ces entiers sont taille fixe, ceci peut se faire en temps linéaire en leur nombre avec un tri par base (par ex. en base 2^8 ou 2^{16}).
Les nombres négatifs peuvent également être traités efficacement et avec élégance, *cf.* par ex. : https://www.codercorner.com/RadixSortRevisited.htm.

6. Cela reste-t'il possible si certains nombres flottants représentent des nombres infinis ?

Oui, il n'y a pas grand chose à adapter : les représentations de l'infini ont un exposant de valeur maximale, et seront donc bien triés par rapport aux nombres finis si l'on procède comme ci-dessus.

7. Cela reste-t'il possible si certains nombres flottants ne représentent pas des nombres ?

La question n'a que partiellement du sens puisque les `NaN` IEEE754 ne sont pas ordonnables avec les autres nombres flottants (notamment, un `NaN` est différent de tout autre nombre flottant, y compris lui-même). Cependant on peut remarquer qu'au bit de signe près, lorsqu'elle est interprétée comme un entier une représentation binaire de `NaN` est de strictement supérieure à celle de l'infini de même signe (notamment $\rho(\texttt{NaN}) > \rho(\infty)$). Si l'on applique un tri par base sans modification, les `NaN`s seront donc naturellement « triés » aux extrémités du tableau, ce qui peut éventuellement être pratique.

Exercice 3. *Un exercice de Jean-Baptiste Bianquis*

À votre grand bonheur, vous avez reçu pour Noël une balance d'excellente qualité : elle offre trois chiffres (décimaux) de précision, et ce autant pour des masses de l'ordre du gramme que de l'ordre de la tonne. Vous décidez d'utiliser cette balance pour mesurer la masse m_h de votre hamster h. On suppose pour simplifier que m_h est de l'ordre de 100 grammes (un gros hamster, d'après Wikipédia).

1. Si h accepte de monter docilement sur la balance et d'y rester le temps qu'elle fasse sa mesure, avec quelle précision obtiendrez-vous m_h ?

La précision absolue de la mesure sera au demi-gramme : la masse affichée (au gramme près) sera au plus supérieure ou inférieure à celle de h de 0.5g. On pourra suivre l'évolution de la masse de h dans ses moindres détails !

2. h (qui est d'une intelligence assez rare pour un rongeur) vous soupçonne, à tort ou à raison, de vouloir utiliser cette pesée pour justifier une mise au régime. Il descend donc immédiatement de la balance à chaque fois que vous l'y posez, et ce avant que la mesure n'ait été faite. Vous décidez alors de le peser indirectement : vous vous pesez une première fois avec h dans la main, puis une deuxième fois sans h, et vous faites la différence. Avec quelle précision obtenez-vous m_h (par ex. avec un arrondi vers les pairs) ?

La première affichera votre masse à la centaine de gramme près (ou au kg près...), et aura donc une précision absolue de de $50\,g$ (ou $\frac{1}{2}$ kilogramme...). Cependant, la seconde pesée ne permettra pas de déterminer m_h avec une précision meilleure que $100\,g$ (éventuellement légèrement mieux en fonction de votre masse et des choix d'arrondis effectués par la balance).
Pour s'en convaincre considérons que votre masse réelle est de $59.95\,kg$, ce que la balance affiche comme (par exemple) $60.0\ kg$. Si h pèse quant à lui exactement $100\,g$, la masse totale mesurée par la balance lors de la seconde pesée est $60.05\,kg$, ce qui (pour une règle d'arrondi réaliste, qui en cas d'équidistance arrondi vers le nombre pair le plus proche) est également affiché comme $60.0\,kg$. La seule information fournie par cette seconde pesée est alors que $m_h \leq 100$g. Cependant, si $m_h > 100\,g$, il est *certain* que la seconde pesée donnera un résultat différent de la première.