format	signe	exposant décalé	décalage	mantisse	signification (nombre normalisé)
32 bits	1 bit	8 bits	$2^{8-1}-1=127$	23 bits	$(-1)^{\text{signe}} \times 1,\underbrace{\cdots}_{\text{mantisse}} \times 2^{\text{exposant décalé}-127}$
64 bits	1 bit	11 bits	$2^{11-1}-1=1023$	52 bits	$(-1)^{\text{signe}} \times 1,\underbrace{\cdots}_{\text{mantisse}} \times 2^{\text{exposant décalé}-1023}$
9 bits	1 bit	4 bits	$2^{4-1}-1=7$	4 bits	$(-1)^{\text{signe}} \times 1,\underbrace{\cdots}_{\text{mantisse}} \times 2^{\text{exposant décalé}-7}$

On rappelle qu'un *nombre normalisé* a son exposant décalé qui n'est ni $0\cdots0$, ni $1\cdots1$.

Exercice 15. *Quelques représentations.* On considère la représentation sur 9 bits donnée plus haut. À quoi sont égaux les nombres suivants ?

1. 110010000
2. 000111010
3. 011101111
4. 101111010

Corrigé.

1. 110010000 est négatif. Son exposant décalé vaut $\overline{1001}^2 = 9$. Sa mantisse vaut $\overline{1.0000}^2 = 1$. Le nombre est donc $-2^{9-7} = -4$.
2. 000111010 est positif. Son exposant décalé vaut $\overline{0011}^2 = 3$, et sa mantisse $\overline{1.1010}^2 = 1 + 1/2 + 1/8 = 1.625$. Le nombre est donc $1.625 \times 2^{3-7} = -0.1015625$.
3. 011101111 vaut $2^7(1 + 1/2 + 1/4 + 1/8 + 1/16) = 248$.
4. 101111010 vaut $-(1 + 1/2 + 1/8) = -1.625$.

Exercice 16. *Représentations de dyadiques.* Donnez la représentation des dyadiques suivants sur 9 bits. On garantit qu'on peut les représenter de manière exacte.

1. 16.0
2. 0.3125
3. -8.5

Corrigé.

1. $16 = 2^4 = 2^{11-7}$. Ce nombre est représenté en flottant sur 9 bits par 010110000.
2. $0.3125 = 2^{-2} \times 1.25 = 2^{-2} \times (1 + 1/4) = 2^{5-7} \times (1 + 1/4)$, est représenté par 001010100.
3. $-8.5 = -(2^3 + 2^{-1}) = -2^3(1 + 2^{-4}) = -2^{10-7}(1 + 2^{-4})$ est représenté par 110100001.

Exercice 17. *Approximation.* Donner la représentation sur 9 bits du flottant le plus proche de π. (On pourra s'aider d'une calculatrice...)

Corrigé. Il s'agit de $3.125 = 2^1 + 2^0 + 2^{-3} = \overline{1.1001}^2 \times 2^1 = \overline{1.1001}^2 \times 2^{8-7}$, qui est représenté par 010000101.

Exercice 18. *Nombres représentables normalisés.* On considère une représentation avec 1 bit de signe, e bits d'exposant et m bits de mantisse.

1. Combien de nombres normalisés peut-on représenter ?
2. Quel est le plus grand nombre que l'on peut représenter ? Le plus petit ?
3. Quel est le plus petit nombre représentable strictement supérieur à 1 ? Le plus petit strictement positif normalisé ?

Corrigé.

1. Les m bits de mantisse peuvent être choisis sans contraite, ainsi que le bit de signe : cela fait 2^{m+1} choix. Les e bits d'exposants ne peuvent être tous nuls ou tous égaux à 1, on obtient donc $2^e - 2$ choix. Au total : $2^{m+1}(2^e - 2)$.
2. Le plus grand nombre est un normalisé, obtenu avec des bits de mantisse tous égaux à 1, et un exposant décalé égal à $2^e - 2$ (tous les bits à 1 sauf le dernier). Le nombre est donc $2^{2^e-2-D} \times \sum_{i=0}^{m} 2^{-i}$. Le décalage vaut $2^{e-1}-1$, et $\sum_{i=0}^{m} 2^{-i} = \frac{1-2^{-(m+1)}}{1-2^{-1}} = 2 - 2^{-m}$. D'où le nombre maximal représentable : $2^{2^{e-1}-1}(2 - 2^{-m}) = 2^{2^{e-1}}(1 - 2^{-m-1})$ (environ 1.8×10^{308} sur 64 bits). Le plus petit nombre représentable est son opposé.
3. 1 est obtenu avec des bits de mantisse tous nuls, le « flottant suivant » est obtenu en prenant le dernier bit de mantisse égal à 1, soit $1 + 2^{-m}$. Le plus petit normalisé positif est obtenu en prenant les bits d'exposant tous nuls sauf le dernier, et la mantisse nulle. On obtient $2^{1-D} = 2^{2-2^{e-1}}$.

On rappelle maintenant ce que sont les nombres *dénormalisés*. Ceux-ci ont leur exposant décalé égal à $0\cdots0$. Si la mantisse est nulle, le nombre vaut zéro (il y a alors un zéro positif et un zéro négatif), sinon l'interprétation est

$$(-1)^{\text{signe}} \times 0,\underbrace{\cdots}_{\text{mantisse}} \times 2^{-2^{\text{taille de l'exposant décalé}-1}+2}$$

En d'autres termes, l'interprétation est la même que pour les normalisés (car l'exposant décalé vaut 0), mais le décalage est réduit de 1.

Exercice 19. *Quelques nombres dénormalisés*. On considère les nombre dénormalisés suivants, sur 9 bits. À quoi sont ils égaux ?

1. 000001111 ***2.*** 100000101 ***3.*** 000000001

Corrigé. Dans tous les cas, l'interprétation est $(-1)^S \times \overline{0.M_1M_2M_3M_4}^2 \times 2^{-6}$, avec les M_i les bits de mantisse et S le bit de signe. On obtient donc :

1. $2^{-6}(1/2+1/4+1/8+1/16)$ dans le premier cas ;

2. $-2^{-6}(1/4+1/16)$ dans le second ;

3. 2^{-10} dans le troisième.

Exercice 20. *Nombres dénormalisés représentables*. On considère une représentation avec 1 bit de signe, e bits d'exposant et m bits de mantisse.

1. Combien de nombres dénormalisés peut-on représenter ?

2. Quel est le plus grand nombre dénormalisé que l'on peut représenter ? Quelle est sa différence avec le plus petit normalisé positif ? Justifier la valeur différente du décalage.

3. Quel est le plus petit nombre dénormalisé strictement positif ?

Corrigé.

1. L'exposant est fixé, les bits de mantisse et le signe sont libres : il y en a 2^{m+1}.

2. Le plus grand est $2^{2-2^{e-1}}\sum_{i=1}^{m}2^{-i}$. Le plus petit normalisé positif est $2^{2-2^{e-1}}$, la différence est seulement de $2^{2-2^{e-1}-m}$. Avec le décalage standard, il y aurait eu un « trou » entre normalisés et dénormalisés !

3. Le plus petit nombre dénormalisé strictement positif est obtenu avec des bits de mantisse tous nuls, sauf le dernier. On obtient donc $2^{2-2^{e-1}}2^{-m} = 2^{2-2^{e-1}-m}$. Remarque : sur 64 bits, on trouve $2^{-1074} \simeq 5\cdot10^{-324}$, d'où le comportement suivant :

```
>>> 2**-1074
5e-324
>>> 2**-1074/2
0.0
```

Exercice 21. *Comparaison de flottants*. Montrer que pour comparer deux flottants positifs, il suffit de les comparer bit à bit. (on excluera le cas où l'un au moins des exposants décalés est $1\cdots1$)

Corrigé. Évident !

Exercice 22. *Nombres non représentables*. Les nombres non décimaux ne sont pas représentables exactement avec des flottants. Proposer des nombres entiers ou décimaux non entiers qui ne le sont pas non plus dans les cas suivants :

— parce qu'ils sont trop grands ou trop petits.

— pour des raisons de précision.

Corrigé.

— $2^{2^{e-1}}$ est un entier (donc dyadique) trop grand pour être représentable en flottant (tout juste !).

— $1/10$ est un décimal qui n'est pas dyadique, il n'est pas représenté de manière exacte sur les flottants.

Lorsque l'exposant décalé d'un flottant est égal à $11\cdots1$, on parle de NAN (Not A Number). Les NAN sont utilisés pour signaler des opérations non valides (par exemple le calcul de $\sqrt{-1}$). Une exception : si la mantisse est nulle, le flottant représente $+\infty$ ou $-\infty$ suivant son signe. En C par exemple, le calcul de $1/0$ produit $+\infty$ (on obtient une erreur en Python). Les infinis sont utilisés pour représenter des résultats de calculs trop grands en valeur absolue.