TD/TP Systèmes de numération et représentation des nombres

1 Quelques ordres de grandeur

On utilise un alphabet comportant c caractères.

1. Combien y a-t-il de mots différents ayant une longueur n ?
   Solution : c^n
2. Combien y a-t-il de mots ayant une longueur au plus égale à n ?
   Solution : $\sum_{i=0}^{n} c^i$ en comptant le mot vide
3. Application : combien y a-t-il d'octets différents (mots de 8 bits) ? Quel est le nombre minimum de bits nécessaires pour avoir 1000 mots différents ?
   Solution : Il y a $2^8 = 256$ octets différents ; Le nb minimum de bits pour 1000 mots différents est 10
4. On considère un livre comportant 300 pages. Chaque page comporte 30 lignes de 50 caractères chacune.
   Solution : $300 * 30 * 50 = 450.000$
   (a) Si l'alphabet utilisé comporte 128 caractères, combien de livres différents peut-on écrire ?
   Solution : $L = 128^{450000}$
   (b) Quelle est la probabilité qu'un singe tapant au hasard sur les touches écrive un livre donné de ce format ?
   Solution : $1/L$
   (c) Combien de bits utiliserait un ordinateur pour mémoriser le livre ?
   Solution : $8 * 450000 = 3.600.000$

2 Les informatoks

Sur la lointaine planète Infok les Informatoks jouent à un jeu très populaire. Mais dans chaque région d'Infok où se joue un match, les Informatoks ont la mauvaise habitude de ne pas utiliser la même technique d'affichage des scores. Néanmoins les scores doivent apparaître sur toute la planète. L'objectif de cet exercice est d'aider les Informatoks chargés des affichages à s'y retrouver.

Infok est composée de quatre régions :

- Les Binoks n'utilisent que des lampes éteintes ou allumées pour afficher les résultats. Ils sont donc en base 2.
- Les Octoks, toujours étourdis, ont malencontreusement perdu les chiffres 8 et 9. Ils sont donc en base 8.
- Les Hexoks, réputés pour leur avarice, ne veulent utiliser que deux caractères pour afficher les scores, ainsi la base 16 leur suffit.
- Enfin les Décoks, qui sont la honte des Informatoks, n'utilisent que des chiffres dans une bête base 10.

Question 1

Avant tout, il faut aider les Informatoks à optimiser leurs calculs. Leur rappeler le moyen *le plus simple* pour passer :

1. d'une base 10 à une base 2
   Solution : Divisions successives par 2 jusqu'au reste nul. Lecture de droite à gauche et de bas en haut
2. d'une base 2 à une base 8
   Solution : Paquets de 3 bits en partant de la droite
3. d'une base 8 à une base 16
   Solution : Convertir en base 2 puis ensuite convertir en base 16.
4. d'une base 16 à une base 10
   Solution : $\sigma_{i=0}^{n} \alpha_i * 16^i$
5. d'une base 2 à une base 10
   Solution : $\sigma_{i=0}^{n} \alpha_i * 2^i$

Question 2

Il y a eu un match dans chaque région ; afficher les scores pour les autres régions :

1. chez les Binoks : score 10110 à 111101,
   Solution : Base 8 : 26 à 75; Base 16 : 16 à 3D; Base 10 : 22 à 61
2. chez les Octoks : score 53 à 102
   Solution : Base 2 : 101011 à 1000010; Base 10 : 43 à 66; Base 16 : 2B à 42
3. chez les Hexoks : score 1E à 39
   Solution : Base 2 : 11110 à 111001; Base 8 : 36 à 71; Base 10 : 30 à 57
4. chez les Décoks : score 172 à 240
   Solution : Base 2 : 10101100 à 11110000; Base 8 : 254 à 360 ; Base 16 : AC à F0

Question 3

Un indice dans l'énoncé permet de déterminer le nombre de points maximum que l'on peut marquer dans ce sport. Quel est-il ?
Solution : Les scores en base 16 peuvent être écrits avec seulement 2 chiffres Le nombre de point maximum est donc $(FF)_{16} = 16^2 - 1 = (255)_{10}$

Question 4

Lors d'un match particulièrement serré, les supporters n'étaient pas d'accord sur le total des scores des mi-temps et donc sur le vainqueur. Donner le score final et le vainqueur des différents matchs (Les opérations seront à effectuer dans la base correspondant au score, en évitant de revenir à la base honteuse des Décoks !).

Région	Equipe	1ère mi-temps	2ème mi-temps	— total —
Binoks	Biclars	1001011	11001	**1100100**
	Bihell	101110	10001001	**10110111**
Octoks	Octeurs	156	75	**253**
	Ocarinas	23	134	**157**
Hexoks	Hexoux	90	BB	**14B**
	Hextoir	F1	76	**167**

3 Le système bibinaire

Parce que seize peut s'écrire 2^{2^2}, et puisque l'on parle de binaire pour la base 2, Boby Lapointe estimait qu'on pouvait parler de *Bi-Binaire* pour la base 4, et de *Bi-Bi-Binaire* pour la base 16, terme qu'il abrègea en **Bibi**.

A partir de ce postulat, Boby Lapointe inventa la notation et la prononciation de seize chiffres. A l'aide de quatre consonnes et de quatre voyelles, on obtient les seize combinaisons nécessaires :

HO	HA	HE	HI	BO	BA	BE	BI	KO	KA	KE	KI	DO	DA	DE	DI
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

La figure ci-après indique le moyen de conversion du décimal vers le bibinaire, en passant par le binaire et l'héxadécimal.

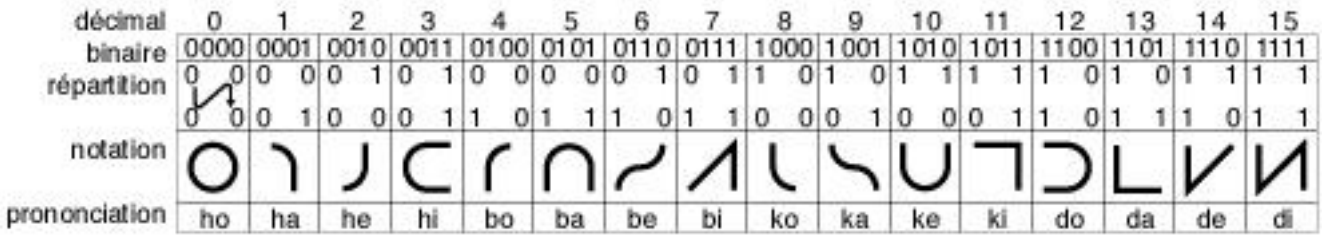

Figure 1: Illustration du système bibinaire par Boby

Pour définir un nombre, il suffit d'énumérer les chiffres (hexadécimaux) qui le composent. Par exemple, le nombre 2000 se traduit 7D0 en hexadécimal, et ainsi BIDAHO en Bibi.

1. Quel jour sommes-nous en bibinaire ?
   Solution : $27/09/2010 \Rightarrow 1B/9/7DA_{16} \Rightarrow$ HAKI-KA-BIDAKE

2. Convertir les nombres suivants de la base 10 au bibinaire, et vice-versa :

Décimal	— Bibinaire —
0	**HO**
3	**HI**
89	**BAKA**
4875	**HAHIHOKI**
1048577	**HAHOHOHOHOHA**

Bibinaire	— Décimal —
DODO	**204**
KOKA	**137**
BIBADE	**1886**
HAHIKIDO	**5052**
KIHADIDODO	**729036**

3. Calculs en bibinaire :

   (a) Calculer DO+BE, puis HAHE+HA. Poser alors l'addition DODO+BEBE et la calculer à partir des deux sommes précédentes.
   Solution : DO+BE = HAHE ; HAHE+HA = HAHI

$$\begin{aligned} \text{DODO} + \text{BEBE} &= (\text{DO} + \text{BE}) \times 16_{10} + \text{DO} + \text{BE} \\ &= \text{HAHE} \times 16_{10} + \text{HAHE} \\ &= (\text{HAHE} + \text{HA}) \times 16_{10} + \text{HE} \\ &= \text{HAHI} \times 16_{10} + \text{HE} \\ &= \text{HAHIHE} \end{aligned}$$

   (b) Calculer BE*DO, puis BOKO+BO. Poser alors la multiplication DODO*BEBE et la calculer à partir des deux résultats précédents.
   Solution : BE $\times$ DO = BOKO ; BOKO+BO = BODO
   DODO*BEBE=BAHABOKO

4. On admet que HAHAHA = 273_{10}. En déduire rapidement les valeurs de HAHAHAHA et de HAHAHAHAHA en justifiant la méthode utilisée. Qu'en est-il du calcul de HIHIHIHIHI à partir de HIHIHI ? Généraliser cette méthode à toute base β.
   Solution : Rajouter un HA = rajouter une puissance de 16 = décalage vers la gauche et ajout de HA=$1 = 16^0$ en position droite. Soit $(x)_\beta = \sum_{i=0}^{n} a\beta^i$. Alors cela revient à calculer $\sum_{i=0}^{n+1} a\beta^i = \sum_{i=0}^{n} a\beta^i + a\beta^{n+1} = (\beta \sum_{i=1}^{n} a\beta^i) + \beta^0$.

5. **Programmation Python (conversions et opérations en bibibinaire–décimal)**

   (a) Programmer une fonction qui lit au clavier une mot composé des syllabes du système bibibinaire, puis calcule et renvoie le nombre décimal correspondant. Tester cette fonction.

   (b) Programmer et tester la fonction inverse.

   (c) Programmer et tester la fonction qui additionne deux nombres en bibibinaire et retourne le résultat en bibibinaire.

   (d) Idem avec la multiplication.

4 Un peu plus de sérieux : quelques propriétés

1. Montrer que 10101 est divisible par 111 dans tout système de numération. Exprimer le quotient au moyen de la base.
   Solution : $10101 = 1 + \beta^2 + \beta^4$ et $111 = 1 + \beta + \beta^2$. Or $1 + \beta^2 + \beta^4 = (1 + \beta + \beta^2)(1 - \beta + \beta^2)$. Le quotient est donc $C_\beta = \beta^2 - \beta + 1$ sécrivant en base β : $\beta - 1.1$. Par exemple si $\beta = 3, (10101)_3 = (1 + 9 + 81)_{10} = (91)_{10}$ et $(111)_3 = (1 + 3 + 9)_{10} = (13)_{10}$ et $C = (21)_3 = (6 + 1)_{10} = 7_{10}$ et on vérifie qu'en base 10, $7 * 13 = 91$.

2. Montrer que dans tout système de numération de base $\beta > 3$; le nombre 1331 est un cube.
   Solution : En base β, $1331 = 1 + 3\beta + 3\beta^2 + \beta^3 = (1 + \beta)^3$

3. Existe-t-il un système de numération dans lequel le produit de 24 par 42 s'écrit 1401 ? Si oui, lequel ?
   Solution : $24 * 42 = (4 + 2\beta)(2 + 4\beta)$ et $1401 = 1 + 4\beta^2 + \beta^3$. On doit donc résoudre l'équation $(4 + 2\beta)(2 + 4\beta) = 1 + 4\beta^2 + \beta^3$ avec $\beta > 4$, dont la seule solution est $\beta = 7 > 4$

4. Existe-t-il une base dans laquelle le nombre 276 est un carré ?
   Solution : On doit résoudre l'équation $x^2 = 2\beta^2 + 7\beta + 6$ avec $\beta > 7$ et $x = \sum_{i=0}^{n} a_i\beta^i$. Il y a une infinité de solutions, les premières étant 23, 839, 28559, etc. Par exemple avec $\beta = 23$, $(276)_{23} = (1225)_{10}$ est le carré de $(1C)_{23} = (35)_{10}$

5. On pose $x = 1000\ldots01_2$ (n digits 0 encadrés par deux digits 1). Comment s'écrivent x^2 et x^3 en binaire ?
Solution : $x = (1 + 2^{n+1})$ si n est le nombre de 0 au milieu. Alors si $n > 0$, $x^2 = 1 + 2^{n+2} + 2^{2n+2}$ ce qui donne la représentation binaire 1000...00010000...0001 avec $n + 1$ zéros à droite et $n - 1$ zéros à gauche. Si $n = 0$ alors $x = 11_2$ et $x^2 = 11_2 \times 11_2 = 1001_2$.

6. Démontrer que $(1111000001)_2$ est un carré. Quelle est sa racine carrée ?
Solution : $(1111000001)_2 = (961)_{10} = (1 + 6\beta + 9\beta^2)_{\beta=10} = (1 + 3\beta)^2_{\beta=10}$ donc carré de $(31)_{10} = (11111)_2$

7. On note x le nombre réel qui s'écrit $x = 0, \alpha\alpha\alpha\alpha\alpha\alpha\alpha\alpha\alpha\alpha\alpha\alpha\ldots$ en base β (où α est un digit de la base β). Montrer que x est rationnel et déterminer une fraction qui le représente. Que peut-on dire de plus si $\alpha = \beta - 1$?
Solution : On compare $\alpha + x$ à βx : $\alpha + x = \alpha, \alpha\alpha\alpha... = \beta x$ donc $x = \frac{\alpha}{\beta - 1}$. Si $\alpha = \beta - 1$ alors $x = 1$

8. Pour pousuivre la question précédente : on note $y = 0, \alpha\gamma\alpha\gamma\alpha\gamma\alpha\gamma\alpha\gamma\alpha\gamma\ldots$ en base β. Montrer que y est rationnel et déterminer une fraction qui le représente.
Solution : Idem : $\beta^2 y = y + (\alpha\beta + \gamma)$ donc $y = \frac{\alpha\beta + \gamma}{\beta^2 - 1}$

5 L'algorithme Gentleman

Que fait ce programme ?

```
float a=1.0, b=1.0;
tant que ( (((a+1.0) - a) - 1.0) == 0.0 ) faire a <-- 2.0 * a
tant que ( (((a+b) - a) - b) != 0.0 ) faire b <-- b + 1
afficher b
```

Solution :
Programme Python :
a=1.0 b=1.0 i=0 while (((a+1.0) - a) - 1.0) == 0.0: a = 2.0 * a while (((a+b) - a) - b) != 0.0: b = b + 1 print b

Durant la première boucle de l'algorithme, a grandit jusqu'à devenir supérieur à B^m (ici la boucle s'arête quand $a = 2^{m+1} = 2^{53}$) avec B la base (ici deux) et m la largeur de la mantisse (ici 52). A ce moment, le codage de a dans la machine est :

signe (1 bit) exposant (11 bits) mantisse (52 bits)
$\overbrace{0}$ $\overbrace{00000110101}$ $\overbrace{10000\ldots0000}$

La deuxième boucle va s'arêter quand b sera égal à B. En effet, quand b est plus petit que B, le digit de poids fort de b est de poids plus faible que le dernier zéro de a dans la mantisse. L'algorithme affiche donc la base du système de numération du processeur

6 Phénomène de compensation (élimination)

Les phénomènes de compensation se produisent lorsque l'on tente d'effectuer des soustractions de valeurs très voisines. Ils peuvent aboutir à des pertes importantes de précision. Nous cherchons ici à calculer la valeur de e^{-10} en utilisant la série :

$$e^{-10} \approx \sum_{k=0}^{n} (-1)^k \frac{10^k}{k!}$$

Nous rappelons qu'en python, on peut affiner l'affichage des réels grâce aux paramètres d'affichage. Ainsi, pour afficher le réel `x` avec 2 chiffres avant la virgule et 20 chiffres après, on écrit :

```
print '%2.15f' % x
```

- Ecrire une fonction `approcheDirecte(n)` qui calcule et renvoie le terme de rang n de la série précédente. Pour cela, on ne cherchera pas à calculer directement $k!$, mais on doit remarquer que

$$\frac{10^k}{k!} = \frac{10^{k-1}}{(k-1)!} \frac{10}{k}$$

Solution :

```
k = 1
res = 0.0
term = 1.0
i = 100
while (k < i):
        res += term
        term *= -10.0 / k
        k+=1
print '%2.15f' % res
```

En déduire une valeur approchée de e^{-10} par cette technique. Quelle précision obtient-on sachant que $e^{-10} \approx 4,539992976.10^{-5}$?

- Ecrire une autre fonction `approcheIndirecte(n)` qui calcule et renvoie une valeur approchée de e^{-10} en remarquant que $e^{-10} = 1/e^{10}$ avec :

$$e^{10} = \sum_{k=0}^{n} \frac{10^k}{k!}$$

En déduire une valeur approchée de e^{-10} par cette technique. Quelle précision obtient-on cette fois-ci ?

Solution :

```
k = 1
res = 0.0
term = 1.0
i = 100
while (k < i):
        res += term
        term *= 10.0 / k
        k+=1
res = 1/res
print '%2.15f' % res
```

- En étudiant la simple différence entre les deux séries, et en considérant le nombre de chiffres significatifs de la mantisse des réels, expliquer le gain en précision constaté.

7 Phénomène de Rump (1988)

Ecrire une fonction `rump(a,b)` où `a` et `b` sont des entiers, qui calcule la valeur réelle :

$$\text{rump}(a,b) = 333,75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5,5b^8 + \frac{a}{2b}$$

Théoriquement, $f(77617, 33096) = -0.8273960599$. Programmer et tester ce résultat en pratique.

Solution :

```
def rump(a,b):
        return 333.75*(b**6)+(a**2)*((11*(a**2)*(b**2)-(b**6)-121*(b**4)-2)
               +5.5*(b**8)+a/(2.0*b))

print '%2.15f' % rump(77617,33096)
```

8 Instabilité numérique

8.1 Suite récurrente

Ecrire une fonction `suite(n)` qui calcule le $n^{\text{ème}}$ terme de la suite (un réel) :

$$u_0 = 2 \text{ et } u_1 = -4 \text{ et } u_{n+1} = 111 - \frac{1130}{u_n} + \frac{3000}{u_n u_{n-1}}$$

Théoriquement, cette suite converge vers 6. Vérifier alors le résultat en prenant des valeurs approchées des données initiales, avec des précisions de 10^{-7} : constate-t-on une amélioration du résultat ?

Solution :

```
def suite(n):
        u_prec=2.0
        u=-4.0
        i=0
        while(i<n) :
                print u_prec
                temp = u
                u = 111.0-1130.0/u+3000.0/(u*u_prec)
                i+=1
                u_prec=temp
        return u_prec

suite(25)
```

8.2 Intégration par la méthode des rectangles

Soit $\int_\alpha^\beta f(x)dx$ à calculer, $f : [a, b] \mapsto \mathbb{R}$ étant une fonction continue. Si l'on choisit une subdivision $\alpha = \alpha_0 < \alpha_1 < \cdots < \alpha_n = \beta$, alors :

$$\int_\alpha^\beta f(x)dx = \sum_{i=0}^{n-1} \int_{\alpha_i}^{\alpha_{i+1}} f(x)dx$$

La méthode des rectangles approxime l'intégrale comme suit :

$$\int_\alpha^\beta f(x)dx \approx \sum_{i=0}^{n-1} (\alpha_{i+1} - \alpha_i) f(\mu_i) \text{ avec } \mu_i = \frac{\alpha_i + \alpha_{i+1}}{2}, \forall i \in [0, n-1]$$

Programmer la fonction `integrale(n)` qui calcule et renvoie le réel $\int_1^2 \frac{dx}{x}$, n étant le nombre de subdivisions de l'intervalle. La valeur exacte de $\int_1^2 \frac{dx}{x}$ est $\log 2$. Etablir la précision de `integrale` pour 7 valeurs de n (de 10 à 10^6). Théoriquement, plus le nombre de subdivisions est grand, plus le calcul est long mais précis. Expliquer le phénomène constaté en pratique.

Solution : $\ln(2) = 0.693147180559945309417232121458176568075500134360255\ldots$

```
def integrale(p):
        res = 0.0
        x=1.0+1.0/(2.0*p)
        for i in range(p):
                res+=1.0/(p*x)
                x+=1.0/p
        return res

for i in range(8):
        a=integrale(10**i)
        print a
```

9 Une preuve de théorème

- Soit $x = b_n b_{n-1} b_{n-2} \cdots b_2 b_1 b_0$ écrit dans la base β.
- Si $b_n \neq 0$, on note $n + 1 = N_\beta(x)$ le nombre de chiffres nécessaires pour exprimer x dans la base β.

Montrer ce théorème. vu en cours : $N_\beta(x)$ est le plus petit entier strictement supérieur à $\text{Log}_\beta(x)$.

Solution : Puisque $b_n \neq 0$, nous avons $b_n \geq 1$, et les trois nombres écrits ci-dessous sont rangés en ordre croissant :

$$\begin{array}{c|cccccccc} \beta^n & & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ x & & b_n & b_{n-1} & b_{n-2} & \dots & b_2 & b_1 & b_0 \\ \beta^{n+1} & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{array}$$

Par conséquent, $\beta^n \leq x < \beta^{n+1}$ et $\log_\beta(\beta^n) \leq \log_\beta(x) < \log_\beta(\beta^{n+1})$. On en déduit que $n \leq \log_\beta(x) < n+1$. Donc $(n+1)$ est strictement supérieur à $\log_\beta(x)$, et comme n ne l'est pas, $(n+1)$ est le plus petit entier strictement supérieur à $\log_\beta(x)$.