M1.L2: Série d'exercices sur les algorithmes

1 Algorithmes de conversion

A travers quelques exemples d'exécution des algorithmes détaillés dans le cours M1.L2, on commence à s'intéresser à la notion de coût calcul en évaluant le nombre de fois que certains groupes d'instructions sont exécutée pour les données fournies en entrée.

1.1 Conversion virgule fixe vers décimal

Conversion_entier_bin2dec

entrée : *nb de bits en partie fractionnaire* ***nf***
liste P de 32 valeurs 0 ou 1

sortie : *valeur représentée X*

```
X ← 0
B ← 2^(−nf)
Pour i de 1 à 32
    X ← X + P(i) *B
    B ← 2 *B
Sortir X
```

- Pour **nf** valant 2, quelle valeur de X obtient-on pour le motif binaire suivant fourni dans la liste P :
  - Tous les éléments de P sont à zéro sauf P(1) = P(2) = P(4) = P(6) = 1
- Le nombre d'opérations total dépend-il de la valeur de **nf** ?

1.2 Conversion entier décimal vers binaire

Conversion_entier_dec2bin

entrée : *n, entier naturel*
P, liste de 32 valeurs 0 ou 1
sortie : *aucune car l'algo affiche la valeur binaire de n ou un message d'erreur*

```
i ← 1
Répéter
    P(i) = n mod 2
    n ← n / 2
    i ← i + 1
Tant que n ≠ 0  et  i ≤ 32

Si n ≠ 0  et  i > 32
    Afficher : n est trop grand
Sinon
    Pour k de (i -1) à 1
        afficher : P(k)
```

- Pour les valeurs suivantes de **n** indiquer le motif binaire produit par cet algorithme ET le *nombre de passages* dans la boucle conditionnelle :
  - 0
  - 1
  - 2
  - 5
  - 10
  - 33
  - 80
- Peut-on lier le nombre de passage dans la boucle conditionnelle à une propriété du nombre **n** ? Si oui, laquelle ?

1.3 Conversion décimal fractionnaire vers binaire

Conversion_fract_dec2bin

entrée : *a, dans l'intervalle [0, 1[*
sortie : *aucune car l'algo affiche la valeur binaire de a et éventuellement un message d'avertissement*

```
afficher: 0.
i ← 1
Répéter
    a ← 2*a
    b ← PartieEntière( a)
    afficher : b
    a ← a-b
    i ← i + 1
Tant que  a > 0   et   i ≤ 32

Si i > 32
    Afficher : ce résultat est
               une approximation
```

- Convertir les valeurs décimales suivantes vers le binaire à l'aide de cet algorithme :
  - 0.5
  - 0.25
  - 0.375
  - 0.2
- Comme dans l'exercice précédent, existe-il un moyen simple pour anticiper le nombre de passages dans la boucle conditionnelle simplement à partir d'une propriété de la valeur de a ?

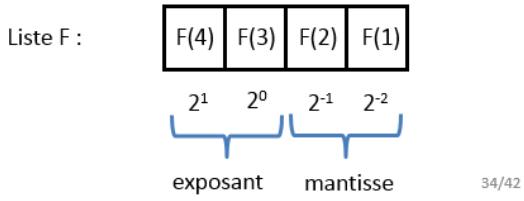
1.4 Conversion virgule flottante du cours M1.L1.3 vers décimal

Conversion_virgule_flottante_vers_dec

entrée : *liste F de 4 valeur binaires*
sortie : *la quantité représentée X*

```
exposant ← 2*F(4) + F(3)
mantisse ← 0.5*F(2) + 0.25*F(1)

Si  exposant = 0
    // forme dénormalisée
    Sortir : 2 * mantisse
Sinon
    // forme normalisée
    Sortir : 2^exposant * (1+mantisse)
```



- Convertir les motifs binaires fournis par la liste F en décimal:
  - F(4) =1 , F(3) =0 , F(2) = 0, F(1)=1
  - F(4) =0 , F(3) =0 , F(2) = 0, F(1)=1
  - F(4) =0 , F(3) =0 , F(2) = 0, F(1)=0

1.5 Conversion d'un nombre décimal vers virgule flottante

Conversion_dec_vers_vflottante

entrée : la quantité décimale X dans l'intervalle [0, 16[
sortie : la liste F du motif binaire du cours M1.L1.3

```
Si  X = 0                                   // zéro est représenté
    Sortir : F ←{0, 0, 0, 0}                // par le motif binaire nul
Sinon Si  X < 2^1                           // forme dénormalisée
    p ←1,  F(4) ←0,  F(3) ←0
Sinon                                       // forme normalisée
    p ←PartieEntière(log2( X ) )
    Convertir l'entier p en binaire
    Ranger le motif binaire du résultat dans F(4) et F(3)

v   ←X / 2^p                                //division sur les réels
m   ←v-PartieEntière( v )                   // obtient la partie fractionnaire

Convertir la partie fractionnaire m en binaire
Ranger le motif binaire du résultat dans F(2) et F(1)

Sortir: F
```

Les parties dans les boites pointillées sont décrites plus haut => conversion d'entier (1.2) et conversion d'une partie fractionnaire (1.3).

- Convertir les valeurs suivantes en précisant si elles sont exactement représentées :
  - 9.5
  - 13
  - 1.25
  - 5
- Calculer l'erreur absolue et l'erreur relative faite sur la représentation de chacune ces grandeurs.
- Vérifier que l'erreur relative est toujours inférieure à la valeur majorante exprimée par la précision **ε** de cette représentation (cf série 1.1).

---

2 Calcul mystérieux 1

Que fait l'algorithme suivant? Essayez par exemple $a = 5$, $b = 7$, et si vous ne voyez pas essayez $a = 10$, $b = 9$. Cet algorithme devrait vous rappeler un exercice de la semaine passée.

devinette1

entrée : a, b deux entiers naturels non nuls
sortie :??

```
c ← 0
Répéter
   Si ( ( b mod 2) = 1)      // si b est impair
      c ← c + a
   b ← b/2                   // division entière
   a ← 2.a
Tant que b > 0

Sortir  c
```

3 Calcul mystérieux 2

Que fait l'algorithme suivant? Essayez par exemple $x = 10, y = 25$, et si vous ne voyez pas essayez $x = 70, y = 42$.

devinette2
entrée : x, y deux entiers naturels non nuls sortie :??

Tant que $x \neq y$
  Si $x > y$
  $x \leftarrow x - y$
  Sinon
  $y \leftarrow y - x$
sortir : x

Note : la science du calcul (Computer Science), en tout cas les algorithmes, existe(nt) depuis bien avant les ordinateurs. Ces deux calculs mystérieux sont vieux de plus de deux milles ans.

---

4 Création d'algorithmes

Soit L une liste d'entiers (pas forcément ordonnée et avec de possibles répétitions), comme par exemple {19, 31, 15, 21}. On pose que la taille n de la liste est une donnée de l'algorithme.

4.1 La plus petite valeur

- Ecrivez un algorithme qui permet de trouver la plus petite valeur de la liste ; exprimer cet algorithme en fonction de la taille de la liste n. On peut par exemple poser arbitrairement que ce minimum est le premier élément de la liste et chercher à vérifier cette hypothèse en le comparant avec les autres éléments de la liste. Si l'hypothèse n'est pas vérifiée pour un élément en particulier, il faut mettre à jour la valeur du minimum. Avec la liste précédente, l'algorithme doit retourner la valeur 15.
- Votre algorithme fonctionne-t-il avec des valeurs ex aequo, comme par exemple avec la liste {15, 31, 15, 21}?
- Combien faut-il de comparaisons en fonction de la taille n de la liste ?

4.2 La plus petite différence

On souhaite maintenant trouver *deux valeurs de la liste qui ont la plus petite différence entre elles.* On peut s'inspirer de la méthode précédente en posant qu'il s'agit des deux premières valeurs de la liste et mettre en place (toutes) les comparaisons pour vérifier et mettre à jour cette paire de valeurs.

Par exemple, si l'algorithme prend en entrée la liste de départ {19, 31, 15, 21}, il doit afficher la paire de nombres (19, 21) car toutes les autres paires possibles ont entre elles une différence plus grande que 2.

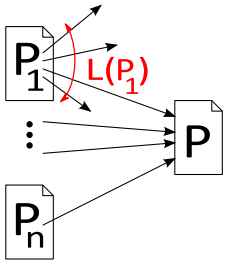
- écrivez un algorithme réalisant ce traitement.
- Comment estimez-vous le nombre total de comparaisons effectuées en fonction de n ?

5 PageRank (optionnel)

Les algorithmes ne sont pas que des abstractions de théoriciens mais peuvent avoir une réelle valeur économique. Par exemple, PageRank (R) est un algorithme, somme toute assez simple, qui a permis à Google de devenir si célèbre. Le but de PageRank est de donner une note à chaque page Web en fonction de la note de chacune des pages qui la citent (c.a.d. qui contiennent un lien vers la page en question.)

5.1 Algorithme

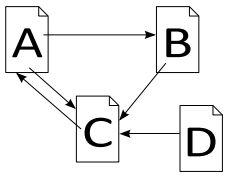
Plus précisément[1], le PageRank PR d'une page P ayant n autres pages P_1, ..., P_n qui la citent, est dé ni par :



$$PR(P) = 0.15 + 0.85 \left(\frac{PR(P_1)}{L(P_1)} + \cdots + \frac{PR(P_n)}{L(P_n)} \right)$$

ou $L(x)$ est le nombre de liens sortant de la page x vers d'autres pages.

Par exemple, si :



la page A cite les pages B et C,
la page B cite la page C,
la page C cite la page A,
la page D cite la page C,
alors on a $L(A) = 2$ et $L(B) = L(C) = L(D) = 1$ (nombre de liens sortant) et

$$\begin{aligned}
\mathrm{PR}(A) &= 0.15 + 0.85\,\mathrm{PR}(C) \\
\mathrm{PR}(B) &= 0.15 + 0.85\,\frac{\mathrm{PR}(A)}{2} \\
\mathrm{PR}(C) &= 0.15 + 0.85\left(\frac{\mathrm{PR}(A)}{2} + \mathrm{PR}(B) + \mathrm{PR}(D)\right) \\
\mathrm{PR}(D) &= 0.15 + 0.85 \times 0 = 0.15
\end{aligned}$$

au départ, les notes PageRank de toutes les pages sont égales à l'inverse du nombre total de pages (0.25 dans l'exemple ci-dessus), ensuite on recalcule la note de chaque page en utilisant les notes précédentes, et on itère comme cela sans arrêt (les notes sont constamment recalculées, le Web évoluant tout le temps).
Pour l'exemple précédent cela donne :

	étape 1	étape 2	étape 3	étape 4	...		...
PR(A)	0.25	0.36250	0.72906	0.70197	...	1.49011	...
PR(B)	0.25	0.25625	0.30406	0.45985	...	0.78330	...
PR(C)	0.25	0.68125	0.64937	0.84580	...	1.57660	...
PR(D)	0.25	0.15	0.15	0.15	...	0.15	...

[1] . Référence : http://infolab.stanford.edu/pub/papers/google.pdf.

5.2 Exercice

Rosa vient de créer son blog personnel et souhaiterait que ses articles atteignent un large public. Pour que les internautes puissent tomber sur son blog, il lui faut être classée le mieux possible par Google. Pour ce faire elle aimerait maximiser son score, tel que donné par l'algorithme PageRank.

Il se trouve que Rosa a quatre amis qui ont déjà une page web. La figure 1 représente les liens entres les pages web des amis de Rosa. Par exemple, le site de Sofien contient un lien vers le site de George, mais il n'y a pas de lien du site de George vers le site de Sofien.

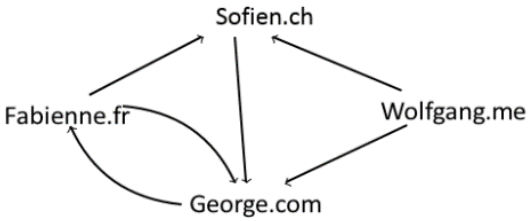


Figure 1 :Liens entre les pages des amis de Rosa.

Rosa décide de demander à un de ses amis d'insérer dans sa page web un lien pointant vers le blog de Rosa. Quel ami Rosa devra-t-elle choisir pour maximiser le score donné par PageRank à son blog?

Indications : commencez par calculer une approximation en 3 étapes du score PageRank de chacune des pages des amis de Rosa. Ecrivez ensuite la forme (équation) du score PageRank de la page de Rosa si un seul ami l'ajoute sur sa page. Conclure.