Méthodes de Conception d'Algorithmes

C. TRABELSI & E. MARTINS & M. FRANÇOIS

TP2 -- Fibonacci / Exponentiation / EBP

A. Calcul de la suite de Fibonacci

Pour rappel la suite de Fibonacci est définie par la formule :

$$F(n) = \begin{cases} 0 \text{ si } n = 0 \\ 1 \text{ si } n = 1 \\ F(n-1) + F(n-2) \text{ pour } n > 1 \end{cases}$$

► **1.** Écrire un programme (vu en TD) qui tant qu'il y a des nombres sur le flux d'entrée :
- récupère un nombre N positif ou nul ;
- calcule le $N^{\text{ième}}$ rang de la suite de Fibonacci via un algorithme récursif ;
- affiche "FIBONACCI(XX) = YY" où XX est le nombre N et YY la valeur de la suite associée, suivi d'un retour à la ligne.

Le prototype de la fonction de Fibonacci devra être :

```
unsigned int FIBONACCI_REC(unsigned int N)
```

► **2.** Testez votre programme pour différentes valeurs de N. Retrouver la valeur limite de N pour le type `unsigned int`.

► **3.** Modifier le type de retour de la fonction `FIBONACCI_REC`, afin de retourner des entiers beaucoup plus grands.

► **4.** Écrire la version itérative en utilisant une boucle. Le prototype de la fonction devra être :

```
unsigned int FIBONACCI_ITER(unsigned int N)
```

► **5.** Testez votre programme pour différentes valeurs de N. Retrouver la valeur limite de N pour le type `unsigned long long`.

► **6.** Laquelle des deux fonctions est la plus efficace ? Pourquoi ?

B. Exponentiation modulaire

a) Algorithme naïf

Le but ici est de calculer $g^e \mod n$, où g et e sont deux entiers positifs ou nuls et n est un entier strictement positif.

► **1.** Écrire une fonction récursive terminale qui retourne la valeur de $g^e \mod n$.

► **2.** Écrire la version itérative.

b) Utilisation de la représentation binaire de e

On considère qu'en base 2 l'exposant e s'écrit :

$$d_{k-1}, d_{k-2}, \ldots, d_1, d_0$$

avec $d_{k-1} \neq 0$ représentant le bit de poids fort de e.

► **3.** Écrire une fonction itérative retournant la valeur de $g^e \mod n$, en adoptant la stratégie **Droite-Gauche** sur le parcours des bits de e.
On considère que les données se trouvent dans un fichier. Par exemple pour $g = 2, e = 654, n = 7$ on a le fichier suivant :

```
2 654 7
10
1 0 1 0 0 0 1 1 1 0
```

où la première ligne indique les valeurs de g, e et n, la seconde donne le nombre de bits nécessaire pour le codage de e et la dernière donne les bits correspondants.

► **4.** Donner la version **Gauche-Droite** de cet algorithme.

NB : vous pouvez utiliser ce lien pour convertir et récupérer directement la représentation binaire d'un nombre.
`http://www.binaryconvert.com/`

C. Expressions Bien Parenthésées (EBP)

On considère une chaîne de caractères constituées uniquement de parenthèses ouvrantes et fermantes.

Écrire une fonction de "parsing", qui prenne en argument une chaîne de caractères composée de parenthèses ouvrantes et fermantes et qui teste si cette chaîne est une EBP. La chaîne de caractères sera donnée par l'utilisateur en ligne de commande.