Document de travail pour le TD1

Rappel : la récursivité dans le module Info21

- x^n $n!$ C_n^k *et suite de Fibonacci, triangle de Pascal*
- *recherche dichotomique*
- *listes*
- *parents et ancêtres, chaînes binaires sans « 11 »*
- *tours de Hanoi*

Question 1

Vérifiez que vous saurez programmer en TP l'algorithme d'Euclide étendu (pgcd et coefficients de Bézout).
Idem pour le crible d'Eratosthène.

Question 2

Dessinez la courbe $y=1/x$ pour $x \in [1/2, 5]$. Et en déduire :

$$1.083 \approx 13/12 = 1/2 + 1/3 + 1/4 < \mathbf{\log 4} < 1 + 1/2 + 1/3 = 11/6 \approx 1.833$$

Déduisez-en une formule pour encadrer $\log x$.
Comment peut-on améliorer la précision ?

Rappel (vu en cours) : $\int_{t=1}^{t=x} 1/t\, dt$ est l'aire sous l'arc de l'hyperbole ($y=1/x$)
$H(k)=1 + 1/2 + 1/3 + \ldots + 1/k$ est la série harmonique, tronquée au terme k.
On en déduit que pour un grand entier n, $\log n \approx H_n$.
En fait $\gamma = \lim_{n\to\infty} H_n - \log n \approx 0.577$ est appelée constante d'Euler.

Question 3

D'après le petit théorème de Fermat, si p est un entier premier, alors pour tout $k \in \mathbb{N}$ on a :

$$1 = k^{p-1} \text{ modulo } p$$

Un test probabiliste calcule k^{p-1} modulo p pour un certain nombre (disons de l'ordre de $\log p$) de valeurs entières aléatoires de k (dans [1, p-1]); si pour tous les tests, $1 = k^{p-1}$ modulo p, le nombre p est "probablement" premier. Sinon (pour un des tests, $1 \neq k^{p-1}$ modulo p), le nombre p n'est sûrement pas premier.
Que pensez-vous du cas où $k=1$?
Comment pouvez-vous optimiser le calcul de k^{p-1} pour $k > 1$?

Remarque : $a^d = a^{d \bmod \Phi(p)}$ modulo p pour tout entier (premier ou non premier), où $\Phi(p)$ est l'indicatrice d'Euler ;
$\Phi(p)$ est le nombre d'entiers dans $[1,p-1]$ et premiers avec p (leur PGCD avec p vaut 1). Si p est premier, $\Phi(p)=p-1$.

Remarque : les nombres de Carmichael passent tous ces tests, mais ne sont pas premiers. Il vaut mieux utiliser le test probabiliste de primalité de Solovay-Strassen, qui utilise aussi l'exponentiation rapide, ou le test de Miller-Rabin.

Question 4

La suite de Fibonacci est définie par : $F_0=0$; $F_1=1$;
$F_n = F_{n-1} + F_{n-2}$, *pour* $n>1$

Proposez un calcul récursif simple de cette suite.

Construisez l'arbre des premiers appels récursifs (jusqu'à *n* valant 4 ou 5) et calculez le nombre d'appels sur 0 et 1. Vérifiez que pour toute valeur de *n*, le nombre d'appels de F1 est égal à la valeur de la suite de Fibonacci sur *n*.

Pour obtenir une certaine efficacité, on veut conserver et réutiliser des résultats intermédiaires. L'idée est de passer à du calcul matriciel, comme illustré ci-contre.

$$\begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}$$
$$(F_n , F_{n-1}) = (F_{n-1} , F_{n-2})$$

Quelle matrice doit être utilisée ?

Remarque : Le calcul matriciel de (Fn, Fn-1) sera donc fait comme indiqué ci-contre : le calcul de la suite de Fibonacci consiste à calculer une puissance de matrice.
Il sera possible d'utiliser une méthode de calcul de puissance rapide similaire à celle vue dans la question précédente sur les entiers.

Ce calcul matriciel sera programmé plus tard.

$$\begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}^2$$
$$(F_n , F_{n-1}) = (F_{n-2} , F_{n-3})$$

$$\begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}^3$$
$$(F_n , F_{n-1}) = (F_{n-3} , F_{n-4})$$

$$\ldots \quad \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}^{n-1}$$
$$(F_n , F_{n-1}) = (F_1 , F_0)$$

Question 5
On note *fi* et *fi'* les deux racines de l'équation $x^2-x-1=0$:

$$fi = (1+\sqrt{5})/2 \quad \text{et} \quad fi' = (1-\sqrt{5})/2$$

Démontrez par récurrence sur *n* que

$$Fib(n) = (fi^n - fi'^n)/\sqrt{5}$$

Rappel : pour cette démonstration par récurrence, vous devez vérifier l'égalité sur 0 et 1 puis en prenant comme hypothèse l'égalité pour *n* et *n-1*, démontrer l'égalité pour *n+1*.

Question 6
Proposez une solution récursive au problème des tours de Hanoï : déplacement d'une tour *Départ* à une tour *Finale* de *n* galets de tailles croissantes ($n \geq 3$). Une tour *Intermédiaire* est fournie. Les règles de déplacement sont les suivantes : 1) déplacement d'un seul galet à la fois, 2) déplacement entre deux tours uniquement, 3) aucun galet ne peut être déposé sur un galet plus petit.
Vous devrez simplement afficher la séquence des déplacements à faire (sous la forme I → F[1]).

1 En Java, codage unicode pour afficher la flèche : *" \u2192 "* dans une instruction *System.out.println() ;*

Ouverture sur les prochains TD

- *tas binaire sous forme de tableau (en récursif)*
- *tri naïf, tri par fusion, tri rapide, tri par tas (sur des tableaux, en récursif)*
- *Fibonacci (calcul matriciel)*
- *courbe de Von Koch, courbe du dragon*

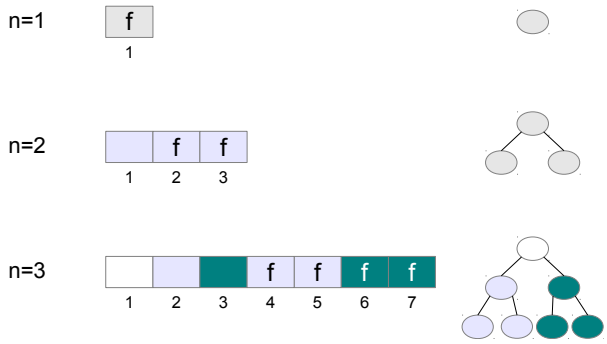
Vocabulaire à revoir : arbre ordonné, tas binaire, tas maximum, tas minimum, tri par tas (heapsort).

Programmation - Pour travailler sur les tris, préparez une classe proposant :

- plusieurs initialisations aléatoires[2] d'un tableau d'entiers (sans aucune contrainte, croissant, décroissant) ;
- l'affichage d'un tableau (sans vous préoccuper de la mise en page sur écran, les éléments étant simplement séparés par quelques espaces).

Travail sur papier - Pour travailler sur les tas, considérez l'utilisation décrite ci-contre de tableaux dont 1) chaque élément d'indice *i* est lié aux éléments d'indices *2i* et *2i+1* et dont 2) les derniers éléments sont appelés *feuilles*.
Calculez en fonction de *n* (qui est appelée *profondeur* du tas) :

- la longueur du tableau ;
- la limite entre les éléments qui sont des feuilles et ceux qui n'en sont pas (i.e., l'indice du dernier élément qui n'est pas une feuille).



Programmation - Ajoutez à votre classe (tableau d'entiers) :

- un traitement récursif *propMax(i)* permettant d'assurer qu'à partir de l'indice *i* chaque élément est supérieur à ses deux éléments liés (d'indices *2i* et *2i+1*)[3] ;
- un traitement itératif permettant d'assurer la propriété *propMax(i)* sur tout le tableau : vous obtenez un tas maximum ;
- les traitements permettant d'obtenir de façon similaire un tas minimum (chaque élément d'indice *i* est inférieur à ses deux éléments liés, d'indices *2i* et *2i+1*).

Programmation - Pour travailler sur la version matricielle de Fibonacci, préparez une classe proposant, sur des entiers longs :

- le produit de matrices 2 lignes, 2 colonnes ;
- le produit d'un vecteur (1 ligne, 2 colonnes) par une matrice 2 lignes, 2 colonnes ;
- l'affichage d'une matrice (2 lignes, 2 colonnes) et d'un vecteur (1 ligne, 2 colonnes) : sans vous préoccuper de la mise en page sur écran, les éléments d'une même ligne étant simplement séparés par quelques espaces).

2 En Java, utiliser *generateur.nextInt(1000);* où *generateur* est déclarée par *Random generateur = new Random();*
3 Il ne s'agit pas d'un arbre binaire **ordonné** : il n'y a aucune contrainte sur l'ordre relatif des éléments d'indices *2i* et *2i+1*