Université Pierre et Marie Curie
Master 2 IFMA
Programmation en C++

Examen du 16 décembre 2015

Les deux premières questions consistent à écrire deux classes permettant de manipuler les matrices de taille n et les chaîne de Markov à valeur dans $\{0, \dots, n-1\}$.
Les questions 3 et 4 sont des utilisations de ces classes. Elles doivent donc être traitées après les questions 1 et 2, mais peuvent être traitées indépendamment l'une de l'autre.

1. Dans un fichier `matrice.hpp`, écrire un modèle ("template") de classe `matrice`, indexé par un entier `n`, permettant de manipuler les matrices de taille `n`. Cette classe devra contenir :
   — En champ privé, une variable `coeff` de type `double*`, pointant vers un tableau de taille `n*n`. Ce tableau représentera les coefficients de la matrice.
   — Deux surcharges de l'opérateur `()` ayant pour prototypes respectifs
   ```
   double& operator() (int i, int j)
   ```
   et
   ```
   double operator() (int i, int j) const
   ```
   permettant de lire et d'écrire dans les coefficients de la matrice.
   — Un constructeur par défaut créant une variable correspondant à la matrice identité.
   — Une surcharge des opérateurs `*` et `*=` pour calculer le produit de deux matrices.
   Si nécessaire, on récrira le constructeur par copie, le destructeur et l'opérateur `=` pour cette classe.
2. Dans un fichier `markov.hpp`, écrire un modèle (template) de classe `markov`, indexé par un entier `n` permettant de manipuler les chaînes de Markov dont l'espace d'état est $\{0, \dots, n-1\}$. Cette classe devra contenir :
   — Comme champs privés, une variable de type `matrice<n>`, correspondant à la matrice de transition [1] de la chaîne et une variable `position` de type `int`, correspondant à la position actuelle de la chaîne.
   — Un constructeur prenant en argument un entier, correspondant à la valeur initiale de la variable `position`. L'entier fourni prendra par défaut la valeur `0`.
   — Une surcharge de l'opérateur `=`, de prototype
   ```
   void operator= (int k)
   ```
   permettant de modifier la valeur de `position`.
   — Deux méthodes `noyau` et `pos` renvoyant respectivement les valeurs des variables `P` et `position`.
   — Une méthode `modif_coeff` de prototype
   ```
   void modif_coeff(int i, int j, double p)
   ```
   qui a pour effet d'affecter la valeur `p` au coefficient en `(i,j)` de la matrice de transition et modifie le coefficient en `(i,i)` de sorte à ce que la matrice reste une matrice de transition (si cela est possible ; dans le cas contraire, la matrice est laissée inchangée).

1. On rappelle qu'une matrice de transition est une matrice P dont les coefficients $0 \leq P_{i,j} \leq 1$ vérifient $\sum_j P_{i,j} = 1$ pour tout i (autrement dit, les lignes de P sont des probabilités). La coefficient $P_{i,j}$ correspond à $\mathbb{P}(X_{n+1} = j | X_n = i)$.

— Une surcharge de l'opérateur `++`, correspondant à une itération de la chaîne[2]. Si nécessaire, on récrira le constructeur par copie, le destructeur et l'opérateur `=` pour cette classe.

3. On considère la marche aléatoire biaisée sur $\{0, \dots, N\}$. Plus précisément on regarde le noyau de transition P défini par

$$P_{i,j} = \begin{cases} p & \text{si } i \in \{0, \dots, N-1\} \text{ et } j = i+1, \\ 1-p & \text{si } i \in \{1, \dots, N\} \text{ et } j = i-1, \\ 1-p & \text{si } i = j = 0, \\ p & \text{si } i = j = N, \\ 0 & \text{sinon.} \end{cases}$$

On prendra pour paramètres $N = 100$ et $p = 1/3$.

(a) Écrire un programme `prog3a.cpp` affichant les 300 premiers pas de cette marche partant de $X_0 = 50$.

(b) Écrire un programme `prog3.cpp` qui calcule un histogramme de 10000 simulations de la variable X_{1024} (partant de $X_0 = 50$) et la première ligne de la matrice[3] P^{1024}.

4. On considère la chaîne de Markov correspondant à la marche aléatoire symétrique sur l'ensemble $\{0, \dots, N\}$ arrêtée en 0 et en N. Plus précisément, la matrice de transition de cette chaîne est donnée par

$$P_{i,j} = \begin{cases} 1/2 & \text{si } i \in \{1, \dots, N-1\}, j = i \pm 1, \\ 1 & \text{si } i = j = 0 \text{ ou } i = j = N, \\ 0 & \text{sinon.} \end{cases}$$

Une trajectoire de cette chaîne finit toujours par être constante, égale à 0 ou à N. On prendra dans cette question la valeur $N = 20$.

(a) Écrire un programme `prog4a.cpp` qui calcule un histogramme du temps d'atteinte de $\{0, N\}$ partant de $N/2 = 10$, pour 1000 simulations.

(b) Écrire un programme `prog4b.cpp` qui calcule en fonction de la condition initiale la probabilité d'être absorbé en 0 (plutôt qu'en N). On utilisera une méthode de Monte Carlo avec 1000 simulations indépendantes pour chaque condition initiale.

2. Pour simuler une variable X avec $\mathbb{P}(X = i) = p_i$, $i \geq 0$, on simule une variable U uniforme sur $[0, 1]$ et on cherche le plus petit i tel que $p_0 + \ldots + p_i > U$. Le i obtenu est une réalisation de X.

3. Le choix de $1024 = 2^{10}$ au lieu de 1000 n'est pas innocent.