**Algorithmique / Travaux pratiques**

S13 – Générateur de nombres pseudo-aléatoires





18.05.2015

Adriano De Almeida Silva – T-1f

Alex Travasso – T-1f

1. RndTriangle.java

package s13;

import java.util.Random;

import javafx.geometry.Point2D;

public class RndTriangle {

public static void main(String[] args) {

int nbOfExperiments = 1000000;

Random r = new Random();

if (args.length > 0)

nbOfExperiments = Integer.parseInt(args[0]);

System.out.println(rndTriangleAvgArea(r, nbOfExperiments));

}

// ============================================================

public static double rndTriangleAvgArea(Random r, int nbOfExperiments) {

double rslt = 0;

int cmpt = 0;

while (cmpt <= nbOfExperiments) {

Point2D ptn1 = new Point2D(r.nextDouble(), r.nextDouble());

Point2D ptn2 = new Point2D(r.nextDouble(), r.nextDouble());

Point2D ptn3 = new Point2D(r.nextDouble(), r.nextDouble());

rslt += 0.5 \* Math.abs(ptn1.getX() \* (ptn2.getY() - ptn3.getY())

+ ptn2.getX() \* (ptn3.getY() - ptn1.getY()) + ptn3.getX()

\* (ptn1.getY() - ptn2.getY()));

cmpt++;

}

System.out.println("la surface moyenne d'un triangle de coordonnées tirées entre 0 et 1 est de:");

return rslt / nbOfExperiments;

}

}



1. RndFlipCoin.java

package s13;

import java.util.Random;

public class RndFlipCoin {

public static void main(String[] args) {

int nbOfExperiments = 100000;

int n = 100, m = 27;

Random r = new Random();

*testFlipCoin*(r, m, n, nbOfExperiments);

}

// ------------------------------------------------------------

public static boolean flipCoin(Random r, int m, int n) {

// une méthode qui tire un booléen suivant la probabilité m/n

return r.nextDouble() < (double)m/n;

}

// ============================================================

static void testFlipCoin(Random r, int m, int n, int nbOfTests) {

int nbOfTrue = 0;

for (int i = 0; i < nbOfTests; i++) {

if (*flipCoin*(r, m, n))

nbOfTrue++;

}

System.*out*.println("There was " + nbOfTrue + " TRUE among " + nbOfTests

+ " flips of coin");

}

}



1. RndLinear.java

package s13;

import java.util.Random;

public class RndLinear {

public static void main(String[] args) {

int nbOfExperiments = 100000;

int n = 10;

Random r = new Random();

*testLinear*(r, n, nbOfExperiments);

}

public static int rndLinear(Random r, int n) {

int m = (n \* (n + 1)) / 2;

int x = r.nextInt(m)+1;

int value = 0;

int result;

for (result = 0; value < x; result++) {

value += result + 1;

}

return result;

}

// ============================================================

static void testLinear(Random r, int n, int nbOfExperiments) {

int[] t = new int[n + 1];

for (int i = 0; i < nbOfExperiments; i++)

t[*rndLinear*(r, n)]++;

System.*out*.println(0 + " : " + t[0]);

for (int i = 1; i < n + 1; i++)

System.*out*.println(i + " : " + (double) t[i] / nbOfExperiments);

}

}



1. RndMajority.java

*hasMajority()*

**public** **static** **boolean** hasMajority(Random r, **int**[] t, **double** risk) {

**for** (**int** i = 0; i < Math.*log*(1 / risk); i++) {

**int** val = r.nextInt(t.length);

**int** w = t[val];

**int** cmpt = 0;

**for** (**int** j = 0; j < t.length; j++) {

**if** (w == t[j]) {

cmpt++;

}

}

**if** (cmpt > t.length / 2) {

**return** **true**;

}

}

**return** **false**;

}



*Facultatif : afin d’avoir un algorithme déterministe, il suffit de supprimer la boucle for de la méthode.*

1. RndWalk.java

package s13;

import java.util.Random;

public class RndWalk {

public static void main(String [] args) {

int nbOfExperiments = 100000;

int n=20;

int leftChoicePercentage = 50; //non-used

Random r = new Random();

System.*out*.println(*rndWalkMirrorAvgLength*(r, n, leftChoicePercentage,

nbOfExperiments));

}

//============================================================

static double rndWalkMirrorAvgLength(Random r, int pointToReach,

int leftChoicePercentage,

int nbOfExperiments) {

int x, nbOfSteps =0;

int total=0;

for (int i=0; i<nbOfExperiments; i++) {

x=0; nbOfSteps=0;

while(x!=pointToReach) {

int e = r.nextInt(100);

//droite

if(e < 50){

x++;

}

//gauche

else if(x != 0){

x--;

}

nbOfSteps++;

}

total+= nbOfSteps;

}

return total/(double)nbOfExperiments;

}

}

A) Formule = N\*(N+1)

pour n = 20 (50 / 50)



B) Formule = pas résolu

Pour n = 20 (55 / 45)



C) Formule = pas résolu

Pour n = 20 (45 / 55)

