Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №2

З предмету: «Теорія ймовірності»

Виконав: студент групи ІО-22

Щербина Микита

Київ 2013р.

Завдання: Створити генератор послідовності чисел підпорядкований біноміальному розподілу

Згенерувати послідовність довжиною 5000 та знайти для неї математичне очікування та дисперсію.

Значення математичного очікування має бути рівним .

Значення дисперсії

Лістинг програми

**import** java.math.BigDecimal;

**import** java.math.BigInteger;

**import** java.util.Random;

/\*\*

\* биномиальное распределение

\*

\* **@author** Nikita

\*

\*/

**public** **class** BinomialDistribution {

**private** BigDecimal[] F;// интегральная функция

**private** Random random = **new** Random();// генератор случайных чисел

/\*\*

\* Конструктор

\*

\* **@param** n

\* - количество

\* **@param** p

\* - вероятность

\*/

**public** BinomialDistribution(**int** n, **double** p) {

BigDecimal bigP = **new** BigDecimal(p);

BigDecimal bigQ = BigDecimal.*ONE*.subtract(bigP);

BigInteger[] c = **new** BigInteger[n + 1];// значение коефициентов с

BigDecimal[] p\_i = **new** BigDecimal[n + 1];// значение р в степени i

BigDecimal[] q\_i = **new** BigDecimal[n + 1];// значение q в степени n-i

// вычисление q в степени n - i

q\_i[n] = BigDecimal.*ONE*;

q\_i[n - 1] = bigQ;

**for** (**int** i = n - 2; i > -1; i--) {

q\_i[i] = q\_i[i + 1].multiply(bigQ);

}

0

// вычисление p в степени i

p\_i[0] = BigDecimal.*ONE*;

p\_i[1] = bigP;

**for** (**int** i = 2; i < p\_i.length; i++) {

p\_i[i] = p\_i[i - 1].multiply(bigP);

}

// вычисление коефициентов с

BigInteger[] bigNumber = **new** BigInteger[n + 1];

**for** (**int** i = 0; i < bigNumber.length; i++)

bigNumber[i] = **new** BigInteger(Integer.*toString*(i));

c[0] = bigNumber[1];

c[1] = bigNumber[n];

**int** k = n / 2;

**if** (n % 2 == 1)

k++;

**for** (**int** i = 2; i <= k; i++)

c[i] = c[i - 1].multiply(bigNumber[n - i + 1]).divide(bigNumber[i]);

**for** (**int** i = k + 1; i < c.length; i++) {

c[i] = c[n - i];

}

// вычисление вероятностей по формуле Бернули

BigDecimal[] pX = **new** BigDecimal[n + 1];

**for** (**int** i = 0; i < pX.length; i++) {

BigDecimal bigDecimalC = **new** BigDecimal(c[i]);

pX[i] = bigDecimalC.multiply(p\_i[i].multiply(q\_i[i]));

}

// вычисление интергральной функции F(x)

F = **new** BigDecimal[n + 2];

F[0] = pX[0];

**for** (**int** i = 1; i < pX.length; i++) {

F[i] = F[i - 1].add(pX[i]);

}

}

/\*\*

\* выдает следующее число

\*

\* **@return**

\*/

**public** **int** nextNumber() {

**double** dr = random.nextDouble();

BigDecimal R = **new** BigDecimal(dr);

**int** result = 0;

**if** (R.compareTo(F[0]) >= 0) {

**int** left = 0;

**int** right = F.length - 1;

**int** m = -1;

**while** (right - left > 1) {

m = (left + right) / 2;

**int** compare = F[m].compareTo(R);

**if** (compare < 0) {

left = m;

} **else** {

**if** (compare > 0) {

right = m;

} **else** {

left = m;

**break**;

}

}

}

result = left + 1;

}

**return** result;

}

}

**import** java.math.BigDecimal;

**public** **class** Main {

**static** **int** *countNumber* = 5000;// длина последовательности

**static** **int**[] *x* = **new** **int**[*countNumber*];// генерируемая последовательность

**static** **int** *n* = 10;// количество проводимых испытаний

**static** **double** *p* = 0.7;// вероятность успеха в одной испытании

**static** **double** *m* = *n* \* *p*;// ожидаемое значение математического ожидания

**static** **double** *d* = *n* \* *p* \* (1 - *p*);// ожидаемое значение дисперсии

/\*\*

\* математическое ожидание

\*

\* **@param** x

\* - входная последовательность чисел

\* **@return**

\*/

**static** **double** mathematicalExpectation(**int**[] x) {

**double** summa = 0;

**for** (**int** i = 0; i < x.length; i++) {

summa += x[i];

}

**double** result = summa / x.length;

**return** result;

}

/\*\*

\* дисперсия

\*

\* **@param** x

\* - входная последовательность чисел

\* **@return**

\*/

**static** **double** dispersion(**int**[] x) {

**double** m = *mathematicalExpectation*(x);

**double** summa = 0;

**for** (**int** i = 0; i < x.length; i++) {

summa += (x[i] - m) \* (x[i] - m);

}

**double** result = summa / x.length;

**return** result;

}

**public** **static** **void** main(String[] args) {

BinomialDistribution bd = **new** BinomialDistribution(*n*, *p*);

**for** (**int** i = 0; i < *x*.length; i++) {

*x*[i] = bd.nextNumber();

}

System.*out*.println("Входные данные");

System.*out*.println("n = "+Integer.*toString*(*n*));

System.*out*.println("p = "+*p*);

System.*out*.println("Результаты");

System.*out*.println("m = " + *mathematicalExpectation*(*x*));

System.*out*.println("d = " + *dispersion*(*x*));

System.*out*.println("Ожидаемые результаты");

System.*out*.println("m = " + *m*);

System.*out*.println("d = " + *d*);

}

}

Результати роботи програми:

Входные данные

n = 10

p = 0.7

Результаты

m = 7.0246

d = 2.1455948400000113

Ожидаемые результаты

m = 7.0

d = 2.1000000000000005

Висновок: з результатів тесту видно, що відхилення отриманих результатів від очікуваних лежить у допустимій межі.