КИЇВСЬКИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

**Лабораторна робота №2**

З теорії ймовірності та математичної статистики

**Тема**: «Генератор послідовності чисел підпорядкований певному розподілу»

Виконав:

студент ІІ курсу ФІОТ

група ІО-32

Попенко Руслан

Перевірив:

Марковський О. П.

Київ-2014

1. **Завдання.**

**Створити генератор чисел підпорядкований біноміальному розподілу**

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

Значення математичного очікування має бути рівним .

Значення дисперсії

Значення середньоквадратичного відхилення

1. **Лістинг програми**

**Клас Біноміального розподілу**

**package** lab2;

**import** java.math.BigInteger;

**import** java.util.Random;

/\*\*

\*

\* **@author** Ruslan Popenko

\*

\*/

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

**private** **BigInteger** **factorial**(**int** n) {

**BigInteger** **ret** = **BigInteger**.***ONE***;

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

ret = ret.multiply(**BigInteger**.*valueOf*(i));

**return** ret;

}

**private** **int** **Cnk**(**int** n, **int** k) {

**int** **buf** = n - k;

**BigInteger** **BI** = factorial(n).divide(

factorial(k).multiply(factorial(buf)));

**return** BI.intValue();

}

**private** **double** **Bernoulli**(**double** p, **int** n, **int** k) {

**double** **cnk** = (**double**) Cnk(n, k);

**double** **p\_k** = **Math**.*pow*(p, (**double**) k);

**double** **one** = 1.0;

**double** **oneMp** = one - p;

**double** **nMk** = (**double**) (n - k);

**double** **power** = **Math**.*pow*(oneMp, nMk);

**return** cnk \* p\_k \* power;

}

**private** **Random** random = **new** Random();

**private** **double** **randomInRange**(**double** min, **double** max) {

**double** **range** = max - min;

**double** **scaled** = random.nextDouble() \* range;

**double** **shifted** = scaled + min;

**return** shifted;

}

**private** **int** **gen**(**double**[] bern2) {

**double** **rand** = randomInRange(0.0, 1.0);

**if** (bern2[0] > rand) {

**return** 0;

} **else** {

**if** (bern2[bern2.length - 1] < rand) {

**return** bern2.length - 1;

} **else** {

**for** (**int** **i** = 0; i < bern2.length; i++) {

**if** (bern2[i] > rand) {

**return** i;

}

}

}

}

**return** **Integer**.***MAX\_VALUE***;

}

**public** **double** **MathematicalExpectation**(**int**[] vector) {

**int** **s** = 0;

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

s += vector[i];

}

**double** **buf** = (**double**) vector.length;

**return** s / buf;

}

**public** **double** **Dispersion**(**int**[] vector) {

**double** **me** = MathematicalExpectation(vector);

**double** **s** = 0.0;

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

**double** **buf** = (**double**) vector[i];

s += (buf - me) \* (buf - me);

}

**double** **buf** = (**double**) vector.length;

**return** s / buf;

}

**public** **double** **StandardDeviation**(**int**[] vector) {

**return** **Math**.*pow*(Dispersion(vector), 0.5);

}

**public** **int** Sequence[];

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

Sequence = **new** **int**[size];

**double** **bern1**[] = **new** **double**[n + 1];

**double** **bern2**[] = **new** **double**[n + 1];

**for** (**int** **i** = 0; i < bern1.length; i++) {

bern1[i] = Bernoulli(p, n, i);

}

bern2[0] = bern1[0];

**for** (**int** **i** = 1; i < bern2.length; i++) {

bern2[i] = bern2[i - 1] + bern1[i];

}

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

Sequence[i] = gen(bern2);

}

}

}

**Клас Лабараторна робота 2 (перевірка правильної роботи)**

**package** lab2;

/\*\*

\*

\* **@author** Ruslan Popenko

\*

\*/

**public** **class** **Lab2** {

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

**double** **p**=0.7;

**int** **n**=10;

**int** **length**=5000;

**BinomialDistribution** **BM**=**new** BinomialDistribution(p, n, length);

**System**.***out***.println("Математичне очікування");

**System**.***out***.println(BM.MathematicalExpectation(BM.Sequence));

**System**.***out***.println("Дисперсія");

**System**.***out***.println(BM.Dispersion(BM.Sequence));

**System**.***out***.println("Середнє квадратичне відхилення");

**System**.***out***.println(BM.StandardDeviation(BM.Sequence));

**System**.***out***.println("По формулі");

**System**.***out***.println("m=np="+(**double**) n\*p);

**double** **D**=(**double**) n\*p\*(1-p);

**System**.***out***.println("D=np(1-p)="+D);

**double** **sigma**=**Math**.*pow*(D, 0.5);

**System**.***out***.println("СКВ="+sigma);

}

}