Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)
Факультет «Робототехника и комплексная автоматизация»
Кафедра «Системы автоматизированного проектирования»

Домашнее задание №2 Часть 1 по дисциплине «Теория вероятностей и математическая статистика»

Вариант 14

Выполнил:
студент группы РК6-36Б
Петраков С.А.

Москва
2020

Оглавление

Задача 1.

Рассматривается извлечение шаров с возвращением из первой корзины (см. исходные данные к ДЗ №1: R1, G1, B1). Выполняется серия из n экспериментов, подсчитывается число k извлечений красных шаров.

R1 = 11

G1 = 10

B1 = 11

1. **Построить графики вероятности P(k). Графики строятся для числа опытов n = 6, 9 и 12 с расчётом вероятностей по формуле Бернулли.**

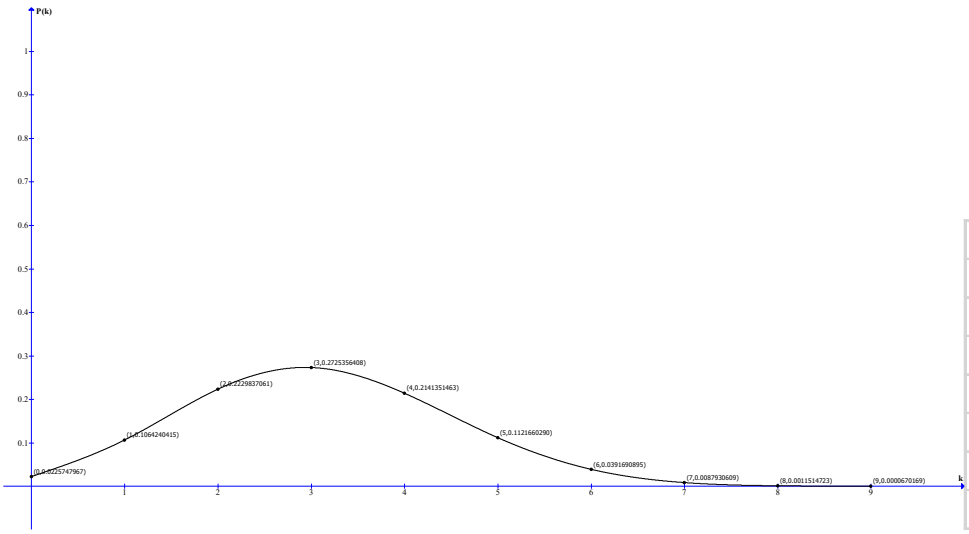
Формула Бернулли:

$P_n(k) = C_n^k p^k q^{n-k}$, где

p – вероятность появления события в одном испытании.

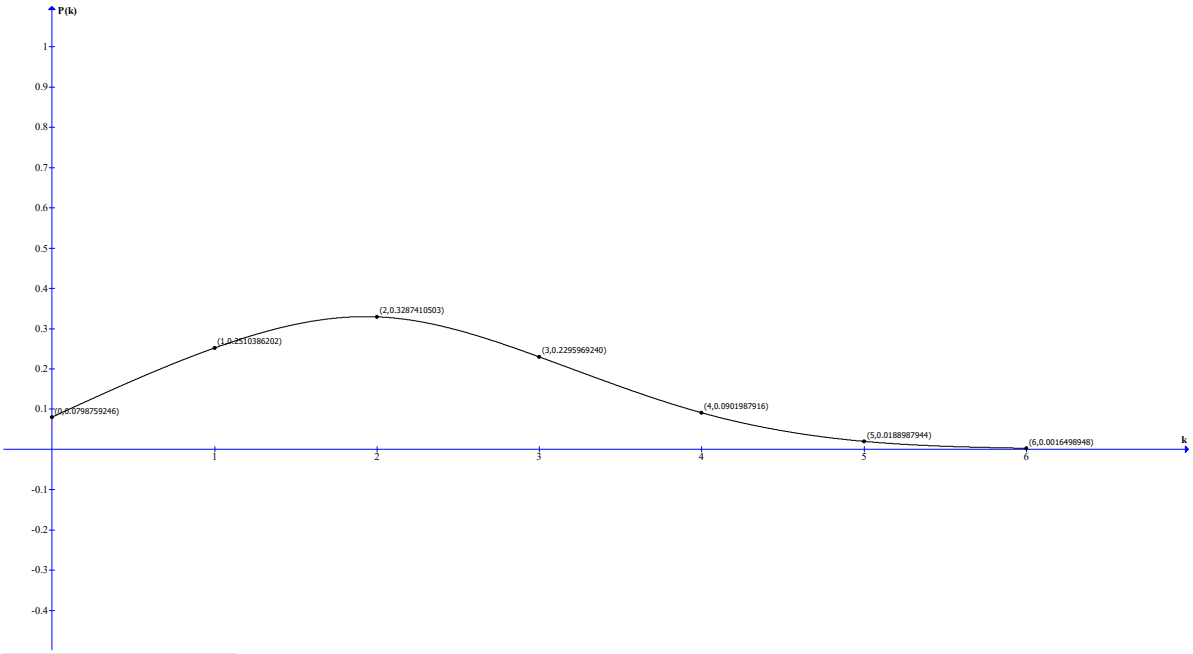
$q = 1 - p$

n=6



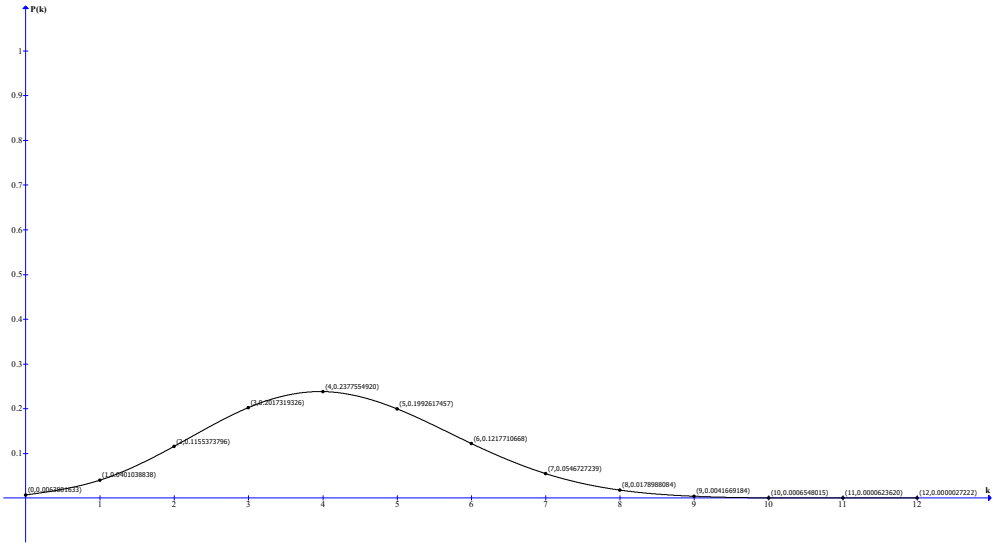
k	P(k)
0	0.0798759246
1	0.2510386202
2	0.3287410503
3	0.2295969240
4	0.0901987916
5	0.0188987944
6	0.0016498948

n=9



k	P(k)
0	0.0225747967
1	0.1064240415
2	0.2229837061
3	0.2725356408
4	0.2141351463
5	0.1121660290
6	0.0391690895
7	0.0087930609
8	0.0011514723
9	0.0000670169

n=12

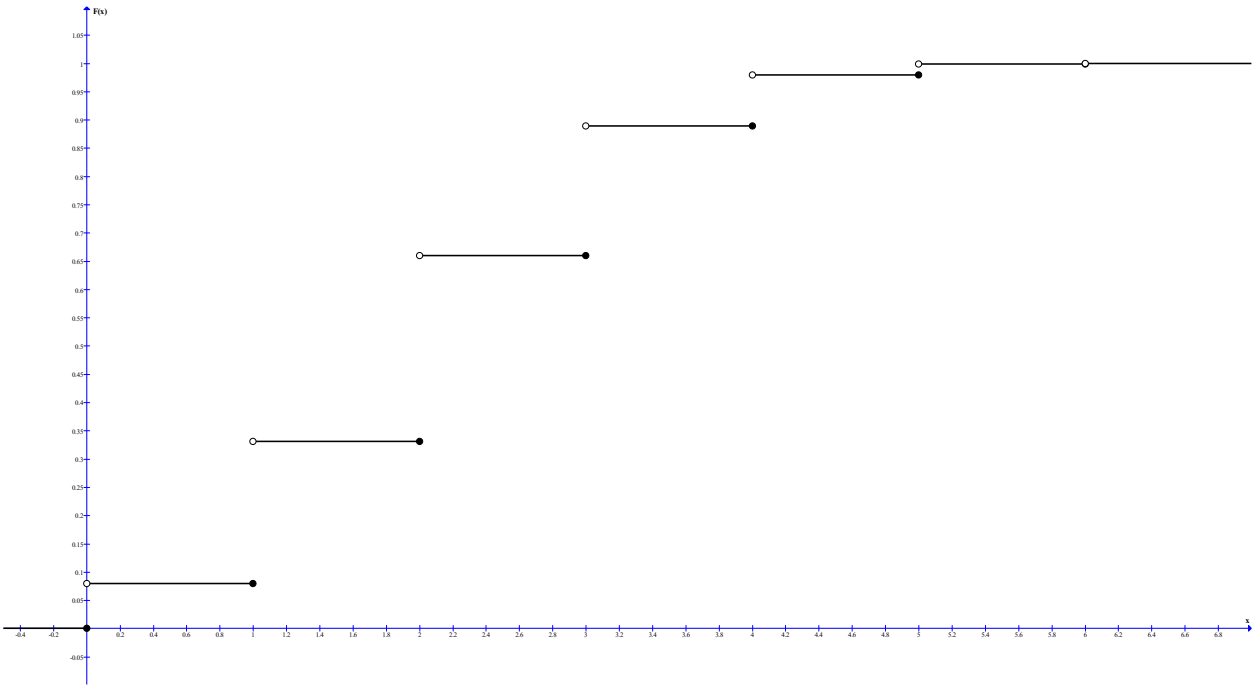


k	P(k)
0	0.0063801633
1	0.0401038838
2	0.1155373796
3	0.2017319326
4	0.2377554920
5	0.1992617457
6	0.1217710668
7	0.0546727239
8	0.0178988084
9	0.0041669184
10	0.0006548015
11	0.0000623620
12	0.0000027222

2. Для n = 6 также строится график функции распределения F(x).

n	P(k)		F(x)
0	0.0798759246		0.0
1	0.2510386202		0.079875925
2	0.3287410503		0.330914545
3	0.2295969240		0.659655595
4	0.0901987916		0.889252519
5	0.0188987944		0.979451311
6	0.0016498948		0.998350105
			1.0

1) $x \leq 0 : F(x) = P(X < 0) = 0$
2) $0 < x \leq 1 : F(x) = P(X < 1) = P(X = 0) = 0.0798759246$
3) $1 < x \leq 2 : F(x) = P(X < 2) = P(X = 0) + P(X = 1) = 0.330914545$
4) …

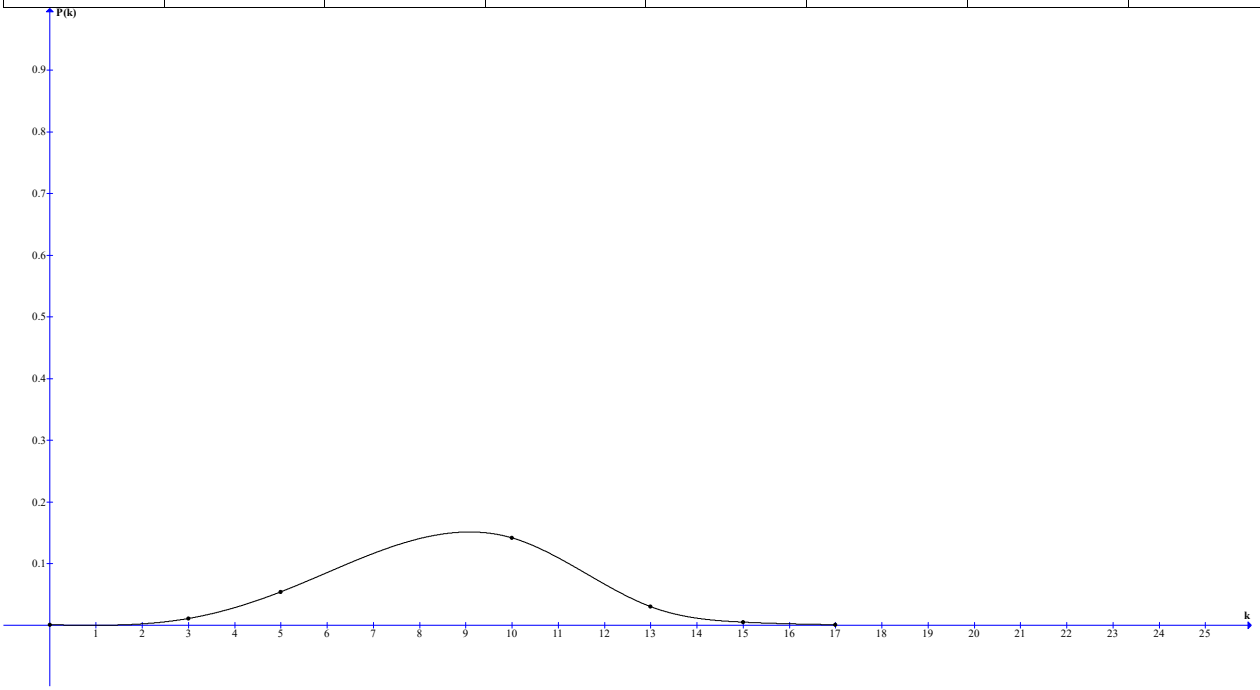


3. **Для n = 25, 50, 100, 200, 400, 1000 строится огибающая графика P(k), при этом для каждого графика рассчитываются не менее 7 точек с использованием локальной теоремы Муавра-Лапласа.**

$$P_n(k) \approx \frac{1}{\sqrt{npq}} * \varphi(\frac{k-np}{\sqrt{npq}})$$

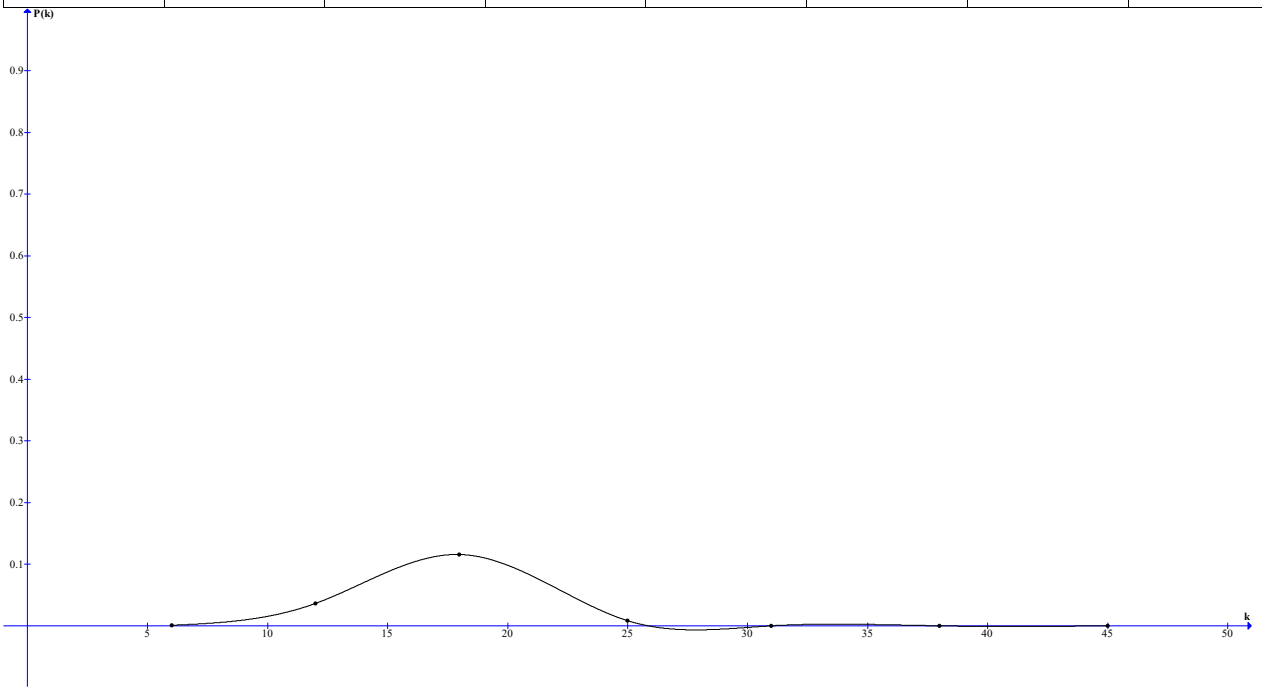
Для n = 25:

k =	0	3	5	10	13	15	17
P(k)	0.000253	0.010359	0.053731	0.141149	0.029771	0.004379	0.000337



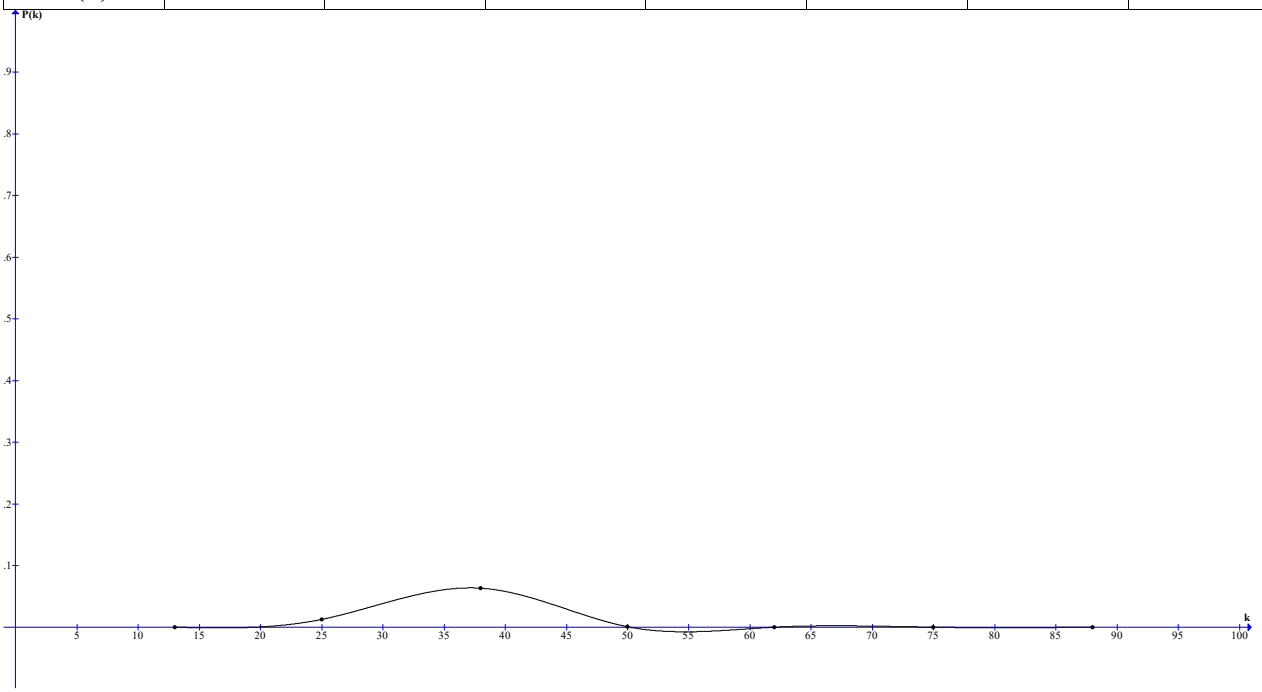
Для n = 50:

k =	6	12	18	25	31	38	45
P(k)	0.000476	0.036296	0.11541	0.007861	0.000030	0.000030	0.000030



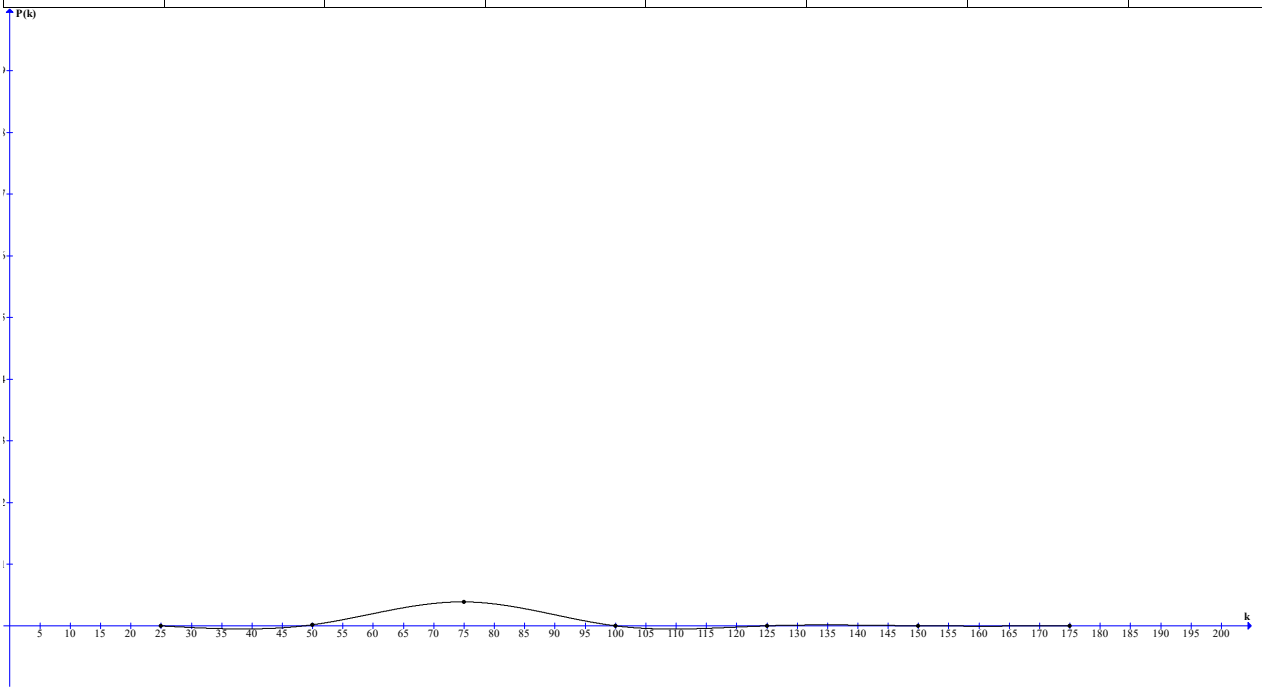
Для n = 100:

k =	13	25	38	50	62	75	88
P(k)	0.000021	0.012064	0.062932	0.000379	0.000021	0.000021	0.000021



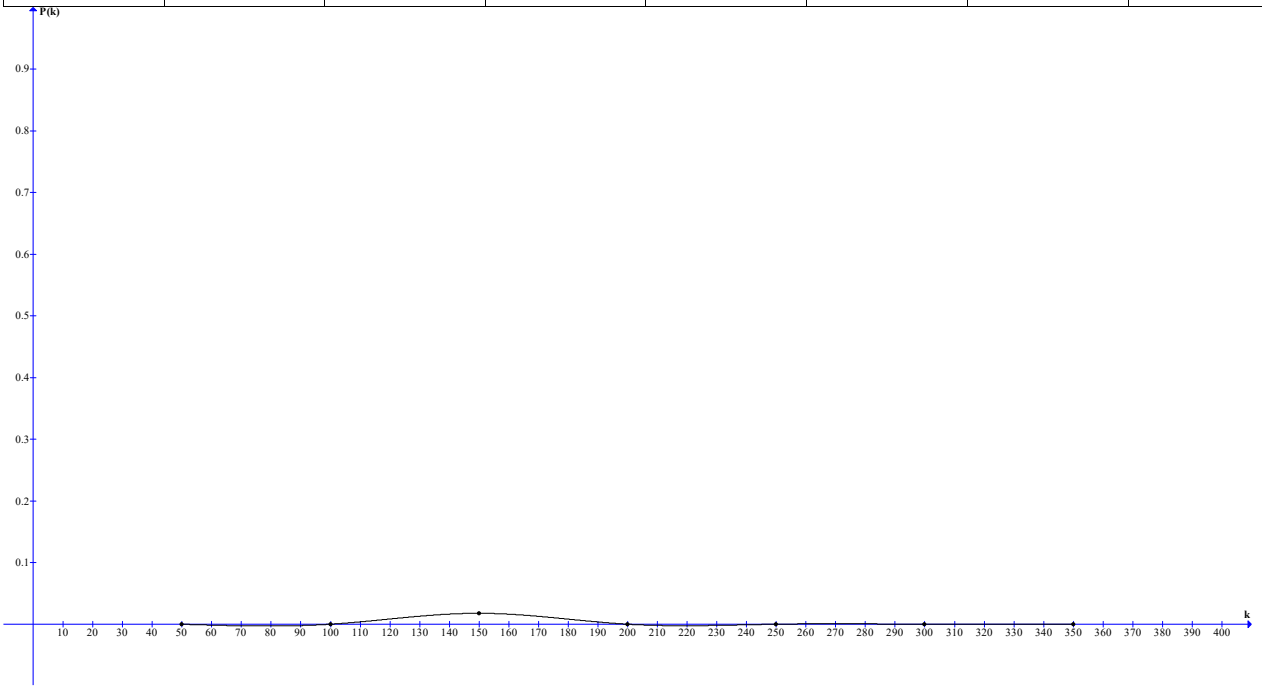
Для n = 200:

k =	25	50	75	100	125	150	175
P(k)	0.000015	0.001206	0.038544	0.000015	0.000015	0.000015	0.000015



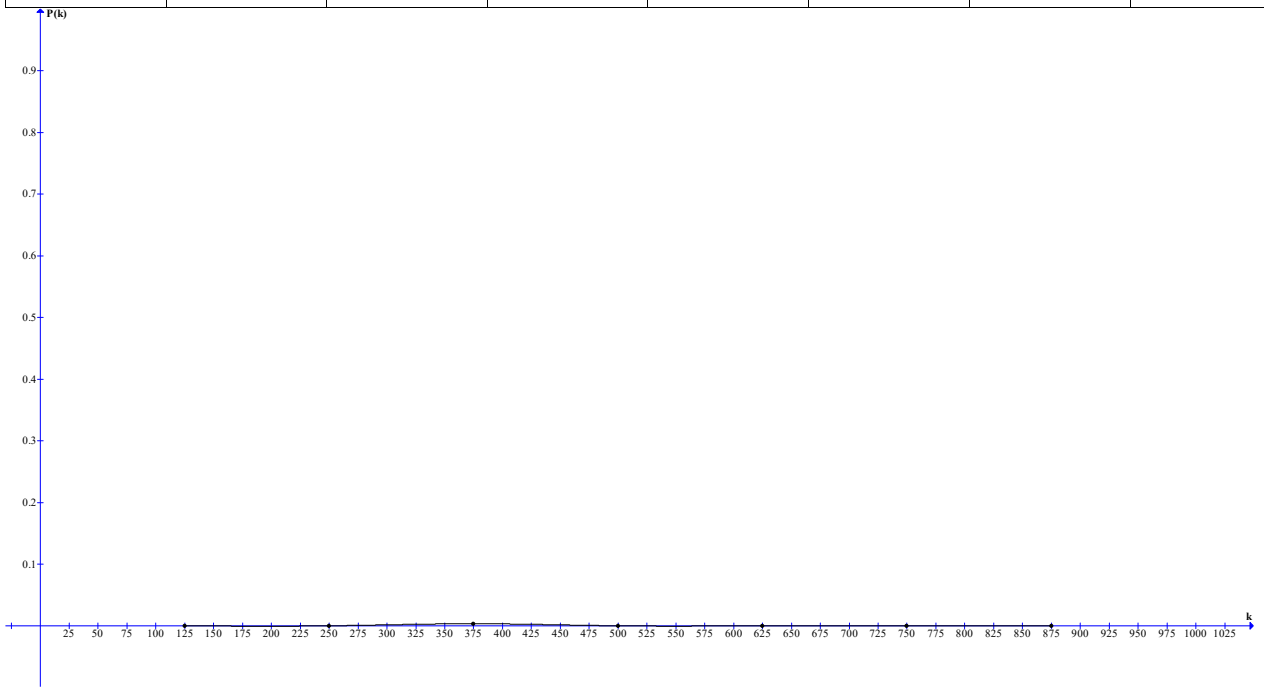
Для n = 400:

k =	50	100	150	200	250	300	350
P(k)	0.000011	0.000021	0.017570	0.000011	0.000011	0.000011	0.000011



Для n = 1000:

k =	125	250	375	500	625	750	875
P(k)	0.000007	0.000007	0.003056	0.000007	0.000007	0.000007	0.000007



4. Построить график вероятности того, что абсолютное число извлечений красных шаров отклонится от математического ожидания не более, чем на R1. При построении графика использовать n = 25, 50, 100.

$$P(|k-M|<11)=2\Phi\left(\frac{11}{\sigma}\right)=2\Phi\left(\frac{11}{\sqrt{D(k)}}\right)=2\Phi\left(\frac{11}{\sqrt{M(k^2)-[M(k)]^2}}\right)$$

$$M(k)=\sum_{i=0}^{n}k_iP(k)_i$$

n=25

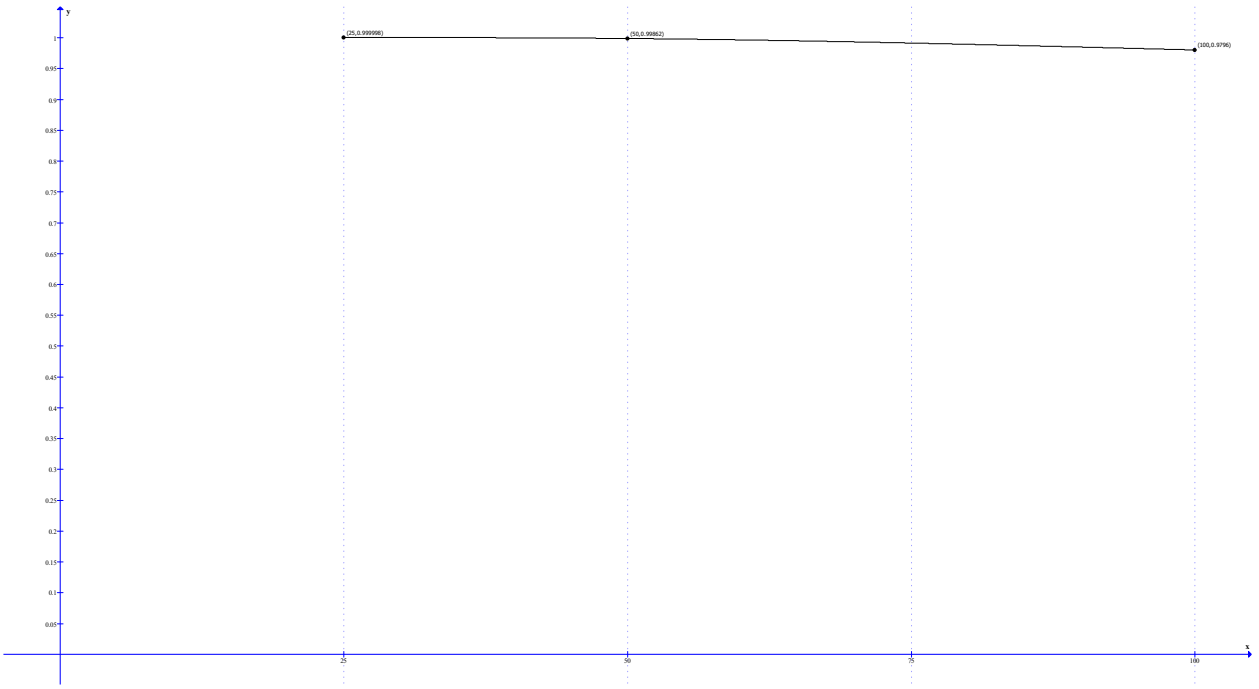
$P(|k-M|<11)=2\Phi(4.631980)=2*0.499999=0.999998$

n=50

$P(|k-M|<11)=0.99862$

n=100

$P(|k-M|<11)=0.9796$



5. **Построить график вероятности того, что относительное число извлечений красных шаров отклонится от математического ожидания не более, чем на R1 / (R1+G1+B1). При построении графика использовать n = 100, 200, 400.**

$$P\left(\left|\frac{m}{n} - M\right| < \frac{R1}{R1 + G1 + B1}\right) = 2\Phi(\frac{\frac{R1}{R1 + G1 + B1}}{\sigma_{\text{отн}}})$$

n=100

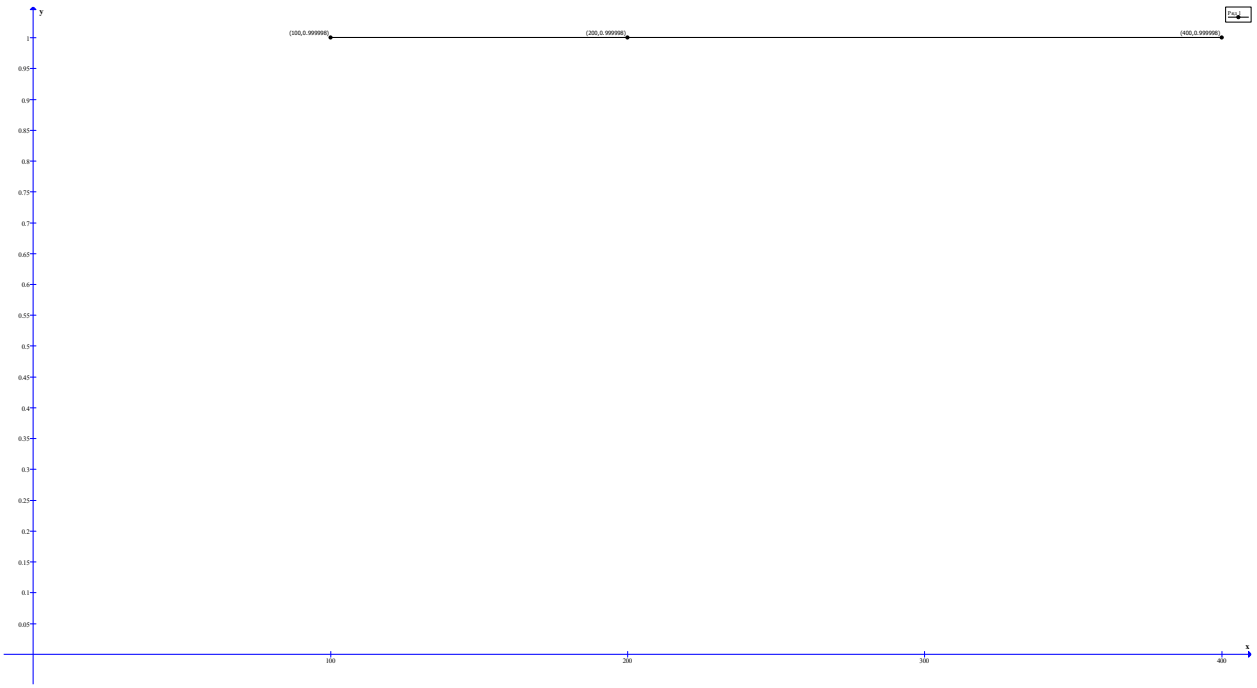
$$P\left(\left|\frac{m}{n} - M\right| < \frac{R1}{R1+G1+B1}\right) = 0.999998$$

n=200

$$P\left(\left|\frac{m}{n} - M\right| < \frac{R1}{R1+G1+B1}\right) = 0.999998$$

n=400

$$P\left(\left|\frac{m}{n} - M\right| < \frac{R1}{R1+G1+B1}\right) = 0.999998$$



6. **Рассчитать допустимый интервал числа успешных испытаний k (симметричный относительно математического ожидания), обеспечивающий попадание в него с вероятностью P = R1 / (R1+G1+B1) при n = 1000.**

A-Событие попадания в интервал

$$P(A) = 2\Phi\left(\frac{\delta}{\sigma}\right) \rightarrow \left(\frac{\delta}{\sigma}\right) = \frac{P(A)}{2} = \frac{R1}{2 * (R1 + G1 + B1)} = \frac{11}{64} \approx 0.171875$$

$$\text{Тогда } \frac{\delta}{\sigma} = 0.45, \sigma = 15.019519$$

$$\text{Получаем } \delta = \sigma * 0.45 = 6.75878355$$

$$M(x) = 343.75$$

Тогда искомый интервал:

$$343.75 - 6.75875355 < k < 343.75 - 6.75875355$$

$$337 < k < 351$$

7. **Построить график зависимости минимально необходимого числа испытаний n, для того, чтобы обеспечить вероятность появления не менее, чем N1=R1+G1+B1 красных шаров с вероятностями P = 0,7; 0,8; 0,9; 0,95.**

$N1 = 32 \qquad P(N1 \leq k \leq n) = \Phi\left(\frac{n-np}{\sqrt{npq}}\right) - \Phi\left(\frac{N1-np}{\sqrt{npq}}\right) = P$

$$\text{P} = \Phi\left(\frac{n - np}{\sqrt{npq}}\right) - \Phi\left(\frac{N1 - np}{\sqrt{npq}}\right) = \Phi\left(\frac{n - n\frac{11}{32}}{\sqrt{n\frac{11}{32} * \frac{21}{32}}}\right) - \Phi\left(\frac{32 - n\frac{11}{32}}{\sqrt{n\frac{11}{32} * \frac{21}{32}}}\right)$$

$$= \Phi\left(\sqrt{n\frac{21}{11}}\right) - \Phi\left(\frac{1024 - 11n}{\sqrt{231n}}\right)$$

При $n \geq 19$, $\sqrt{n\frac{21}{11}} \geq 5 \rightarrow \Phi\left(\sqrt{n\frac{21}{11}}\right) = 0.5$, тогда выражение для P примет вид: $P = 0.5 - \Phi\left(\frac{1024-11n}{\sqrt{231n}}\right)$

- $P = 0.7$

$$0.7 = 0.5 - \Phi\left(\frac{1024 - 11n}{\sqrt{231n}}\right) \rightarrow \Phi\left(\frac{1024 - 11n}{\sqrt{231n}}\right) = -0.2$$

Ближайшее $\Phi\left(\frac{1024 - 11n}{\sqrt{231n}}\right) = 0.2019$, тогда $\frac{1024 - 11n}{\sqrt{231n}} = -0.53$

Тогда наименьшее $\boldsymbol{n = 100}$

- $P = 0.8$

$$0.7 = 0.5 - \Phi\left(\frac{1024 - 11n}{\sqrt{231n}}\right) \rightarrow \Phi\left(\frac{1024 - 11n}{\sqrt{231n}}\right) = -0.3$$

Ближайшее $\Phi\left(\frac{1024 - 11n}{\sqrt{231n}}\right) = 0.3023$, тогда $\frac{1024 - 11n}{\sqrt{231n}} = -0.85$

Тогда наименьшее $\boldsymbol{n = 105}$

- $P = 0.9$

$$0.7 = 0.5 - \Phi\left(\frac{1024 - 11n}{\sqrt{231n}}\right) \rightarrow \Phi\left(\frac{1024 - 11n}{\sqrt{231n}}\right) = -0.4$$

Ближайшее $\Phi\left(\frac{1024 - 11n}{\sqrt{231n}}\right) = 0.4015$, тогда $\frac{1024 - 11n}{\sqrt{231n}} = -1.29$

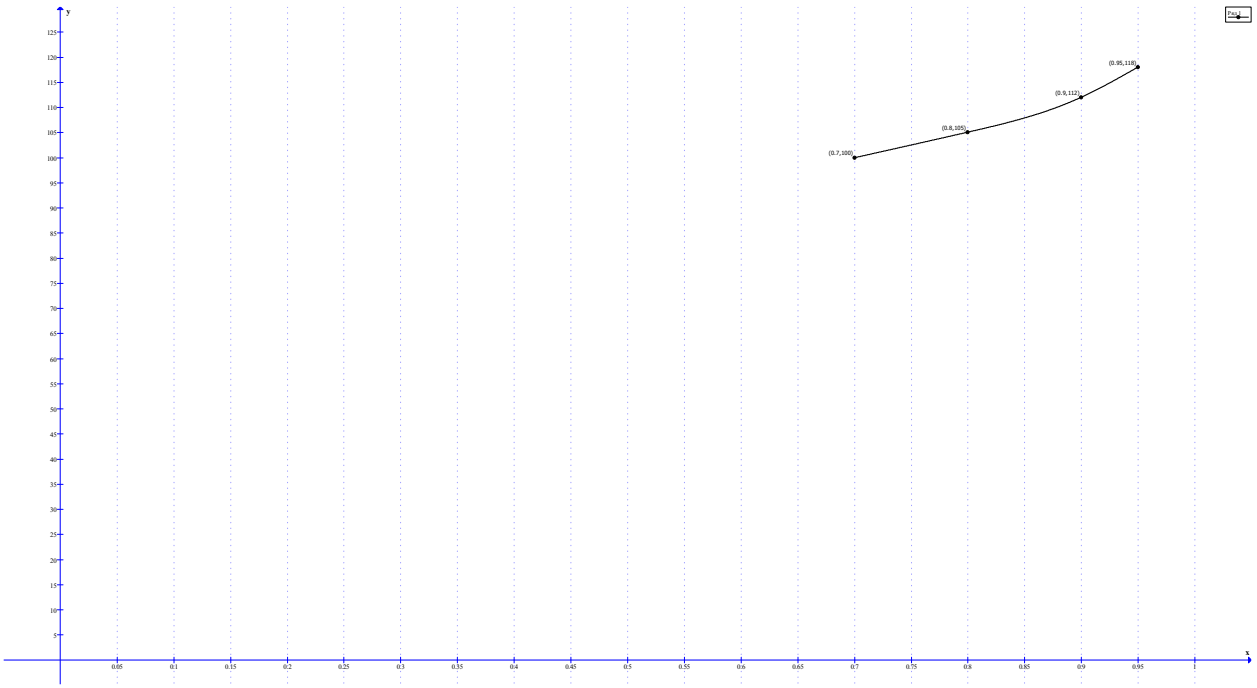
Тогда наименьшее $\boldsymbol{n = 112}$

- $P = 0.95$

$$0.7 = 0.5 - \Phi\left(\frac{1024 - 11n}{\sqrt{231n}}\right) \rightarrow \Phi\left(\frac{1024 - 11n}{\sqrt{231n}}\right) = -0.45$$

Ближайшее $\Phi\left(\frac{1024 - 11n}{\sqrt{231n}}\right) = 0.4505$, тогда $\frac{1024 - 11n}{\sqrt{231n}} = -1.65$

Тогда наименьшее $\boldsymbol{n = 118}$


y
x
Data
(0.7, 100)
(0.8, 105)
(0.9, 112)
(0.95, 118)

Задача 2.

Рассматривается извлечение шаров без возвращения из второй корзины (см. исходные данные к ДЗ №1: R2, G2, B2). Выполняется серия из n=G2+B2 экспериментов, подсчитывается число k извлечений красных шаров.

R2 = 10

G2 = 9

B2 = 10

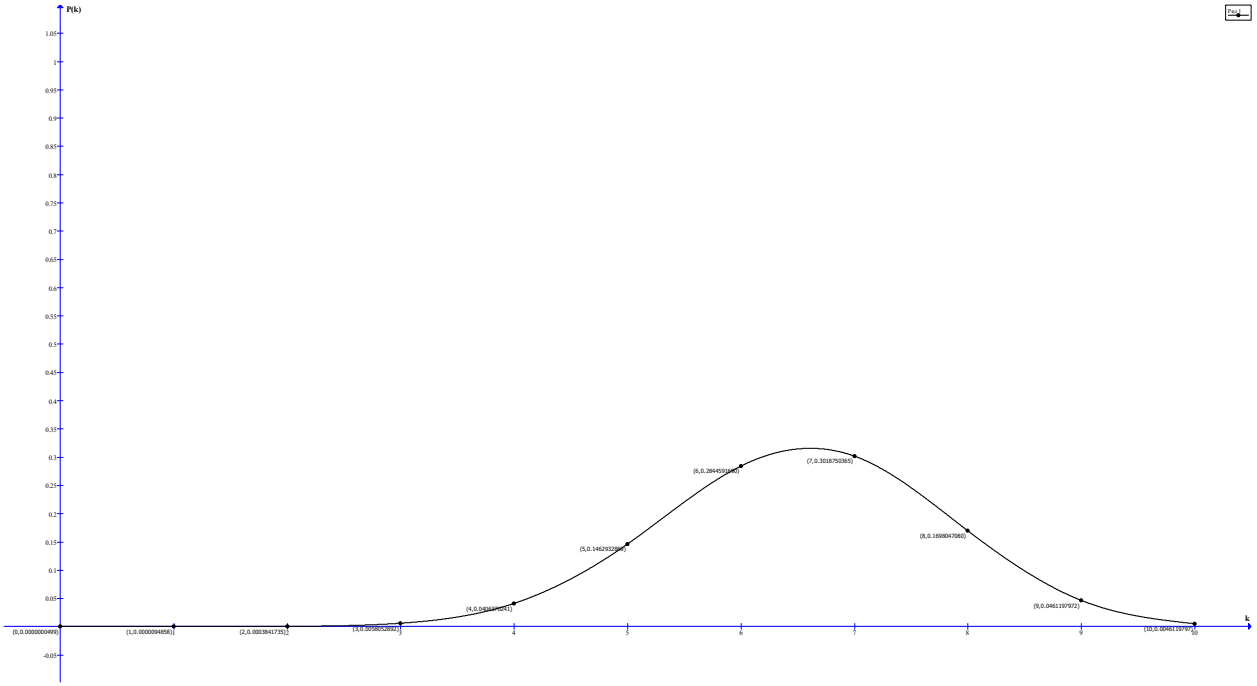
n=19

N=29

1. Построить график вероятности P(k).

$$P(k) = \frac{C_{R2}^{k} * C_{N-R2}^{n-k}}{C_{N}^{n}}$$

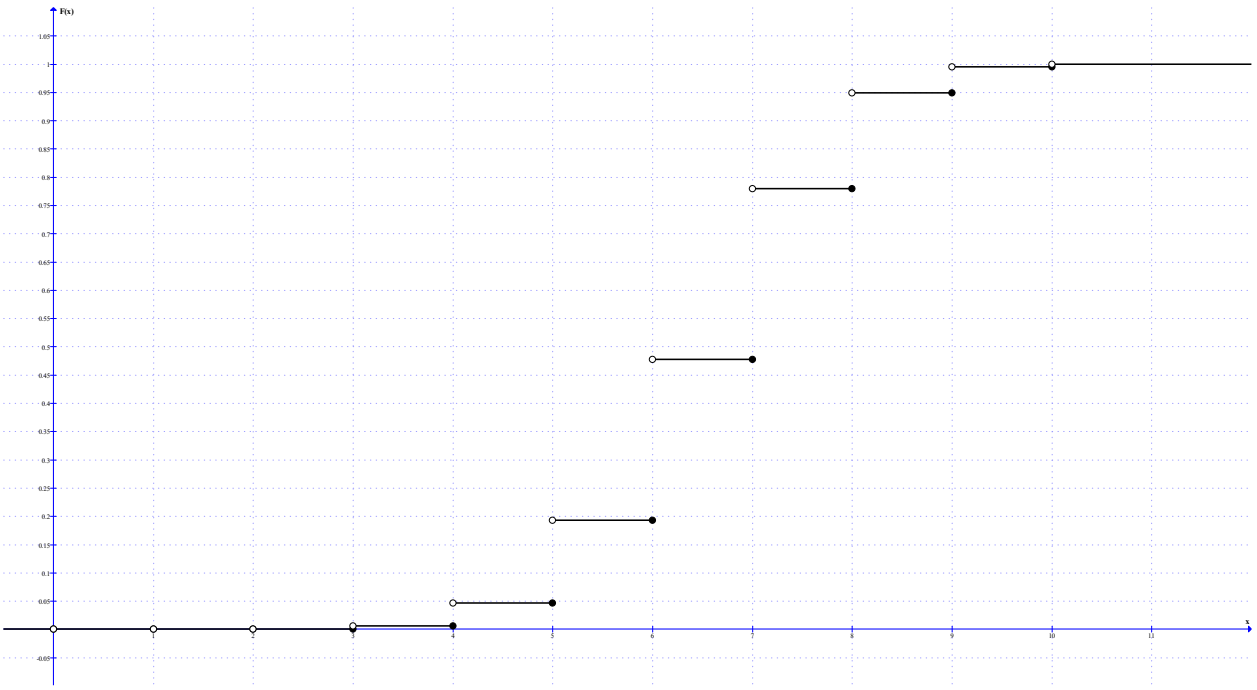
k	P(k)
0	0,0000000499
1	0,0000094858
2	0,0003841735
3	0,0058052892
4	0,0406370241
5	0,1462932869
6	0,2844591690
7	0,3018750365
8	0,1698047080
9	0,0461197972
10	0,0046119797



2. Построить график функции распределения F(x).

1) $x \leq 0 : F(x) = P(X < 0) = 0$
2) $0 < x \leq 1 : F(x) = P(X < 1) = P(X = 0) = 0.0000000499$
3) $1 < x \leq 2 : F(x) = P(X < 2) = P(X = 0) + P(X = 1) = 0{,}0000095357$

k	P(k)		F(x)
0	0,0000000499		0
1	0,0000094858		0,0000000499
2	0,0003841735		0,0000095357
3	0,0058052892		0,0003937092
4	0,0406370241		0,0061989984
5	0,1462932869		0,0468360225
6	0,2844591690		0,1931293094
7	0,3018750365		0,4775884784
8	0,1698047080		0,7794635149
9	0,0461197972		0,9492682229
10	0,0046119797		0,9953880201
			1,0000000000



3. Рассчитать математическое ожидание числа извлечённых красных шаров k.

$$M(k) = \sum_{i=0}^{n} k_i P(k)_i$$

$$M(k) = 6.505604$$

4. **Рассчитать дисперсию числа извлечённых красных шаров k.**

$$D(k) = M(k^2) - \big(M(k)\big)^2$$

$$D(k) = 1.674042$$

Задача 3.

Рассматривается извлечение шаров без возвращения из третьей корзины (см. исходные данные к ДЗ №1: R3, G3, B3). Выполняется серия из k экспериментов, которая прекращается, когда извлечены все R3 красных шаров.

R3 = 9

G3 = 11

B3 = 5

N=R3+G3+B3=25

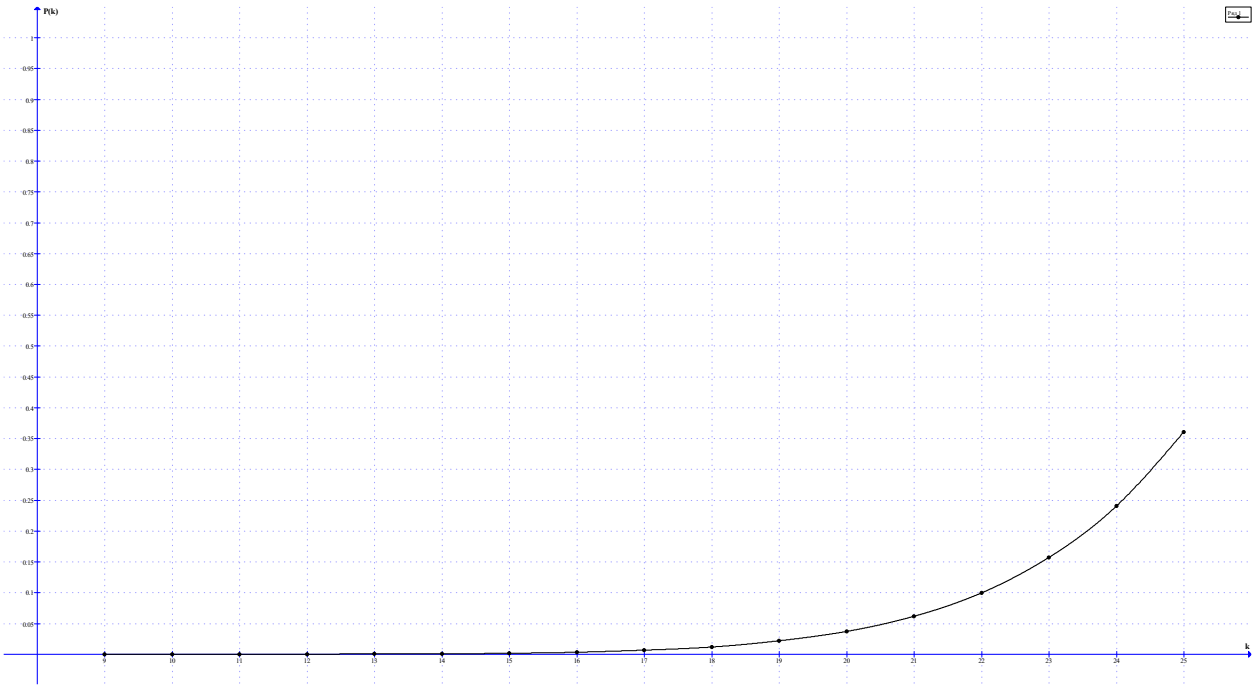
1. Рассчитать значения P(k).

$$P(k) = \frac{C_n^k * C_{N-R3}^{n-k}}{C_N^n}$$

$$P(k) = \frac{C_{R3}^{R3} * C_{N-R3}^{0}}{C_N^{R3}} * C_{k-1}^{k-R3} = \frac{C_{k-1}^{k-R3}}{C_N^{R3}}$$

$$R3 \leq k \leq N$$

k	P(k)
9	0,0000004895
10	0,0000044053
11	0,0000220267
12	0,0000807646
13	0,0002422937
14	0,0006299637
15	0,0014699152
16	0,0031498183
17	0,0062996366
18	0,0118993135
19	0,0214187643
20	0,0369960474
21	0,0616600791
22	0,0996047431
23	0,1565217391
24	0,2400000000
25	0,3600000000



2. **Рассчитать математическое ожидание числа извлечений k.**

$$M(k) = \sum_{i=0}^{n} k_i P(k)_i$$

$$M(k) = 23.4$$

3. **Рассчитать дисперсию числа извлечений k.**

$$D(k) = M(k^2) - \big(M(k)\big)^2$$

$$D(k) = 3.403636$$

Программа

```
#include <iostream>
#include <iomanip>
using namespace std;
#pragma region Utility
long double fact(int N)
{
	long double result = 1;
	for (int i = 1; i <= N; i++) {
		result = result * i;
	}
	return result;
}
long double Combinations(int n, int k)
{

	if (k == 0 || k == n) { return 1; }
	if (k == 1 || k == n - 1) { return n; }

	int numeratorStart = k > n / 2 ? k + 1 : (n - k) + 1;
	int denomAmount = k > n / 2 ? n - k : k;

	int amount = denomAmount > (n - numeratorStart) ? n - numeratorStart + 1 :
denomAmount;
	long double result = 1;
	int a = 1, b = numeratorStart;
	for (int i = 1; i <= amount; ++i)
	{
		result *= b++;
```

```
            result /= a++;
        }

        return result;
}

#pragma endregion

#pragma region Probabilities
long double Bernulli(long double p, int n, int k)
{
        return Combinations(n, k) * pow(p, k) * pow(1 - p, n - k);
}
long double ProbabilityWithoutReturn(int n,int k,int N,int r)
{
        return Combinations(r,k)* Combinations(N-r, n-k)/ Combinations(N,n);
}

long double ProbabilityWithoutReturnToR(int r, int N, int k)
{
        return Combinations(k-1, k-r) / Combinations(N, r);
}



#pragma endregion

#pragma region MathExpectings
long double MathExpecting(long double p, int n)
```

```
{
	long double result = 0;
	for (int i = 0; i <= n; i++)
	{
		result = result+ Bernulli(p, n, i) * i;
	}
	return result;
}
long double MathExpectingSqr(long double p, int n)
{
	long double result = 0;
	for (int i = 0; i <= n; i++)
	{
		result = result + Bernulli(p, n, i) * i * i;
	}
	return result;
}
long double MathExpectingRelative(long double p, int n)
{
	long double result = 0;
	for (int i = 0; i <= n; i++)
	{
		result = result + Bernulli(p, n, i) * i/n;
	}
	return result;
}
long double MathExpectingRelativeSqr(long double p, int n)
{
	long double result = 0;
```

```
	for (int i = 0; i <= n; i++)
	{
		result = result + Bernulli(p, n, i) * i*i/((long double)n* (long double)n);
	}
	return result;
}

long double MathExpectingWithoutReturn(int r, int N, int n)
{
	long double result = 0.0;

	for (int i = 0; i < r; i++)
	{
		result += ProbabilityWithoutReturn(n,i,N,r)*(long double)i;
	}

	return result;

}
long double MathExpectingWithoutReturnSqr(int r, int N, int n)
{
	long double result = 0.0;

	for (int i = 0; i < r; i++)
	{
		result += ProbabilityWithoutReturn(n, i, N, r) * (long double)i* (long double)i;
	}
```

```
        return result;

}

long double MathExpectingWithoutReturnToR(int r, int N)
{
        long double result = 0;
        for (int i = r; i <= N; i++)
        {
                result += ProbabilityWithoutReturnToR(r, N, i) * (long double)i;
        }
        return result;
}
long double MathExpectingWithoutReturnToRSqr(int r, int N)
{
        long double result = 0;
        for (int i = r; i <= N; i++)
        {
                result += ProbabilityWithoutReturnToR(r, N, i) * (long double)i*
(long double)i;
        }
        return result;
}
#pragma endregion

#pragma region Dispersions
long double Dispersion(long double p, int n) {
        return MathExpectingSqr(p, n) - pow(MathExpecting(p, n), 2);
```

```
}

long double DispersionRelative(long double p, int n) {
	return MathExpectingRelativeSqr(p, n) - pow(MathExpectingRelative(p, n), 2);
}

long double DispersionWithoutReturn(int r, int N, int n) {
	return MathExpectingWithoutReturnSqr(r, N,n) - pow(MathExpectingWithoutReturn(r, N, n), 2);
}
long double DispersionWithoutReturntoR(int r, int N) {
	return MathExpectingWithoutReturnToRSqr(r, N) - pow(MathExpectingWithoutReturnToR(r, N), 2);
}
#pragma endregion

#pragma region Deviation
long double StdDeviation(long double p, int n)
{
	return sqrtl(Dispersion(p, n));
}long double StdDeviationRelative(long double p, int n)
{
	return sqrtl(DispersionRelative(p, n));
}
#pragma endregion

#pragma region Punkt1
void Number1Point1(int r, int g, int b, int n)
{
```

```
	long double p = (long double)r / ((long double)r + (long double)g + (long double)b);
	for (int i = 0; i <= n; i++)
	{
		cout << "P(" << i << ") = " << setprecision(10)<< Bernulli(p, n, i)<<endl;
	}
}
void Number1Point3(int r, int g, int b, int n)
{
	int k;
	long double p = (long double)r / ((long double)r + (long double)g + (long double)b);
	long double q = 1 - p;
	long double fi;
	while (1)
	{
		cout << "Enter k = ";
		cin >> k;
		long double forFi = (k - n * p) / sqrtl(n * q * p);
		cout << "ForFi = " << forFi << endl;
		cout << "Enter fi = ";
		cin >> fi;
		cout << "P(" << k << ") = " << fi / sqrtl(n * q * p)<<endl<<endl;
	}
}
void Number1Point4(int r, int g, int b, int n)
{
	long double p = (long double)r / ((long double)r + (long double)g + (long double)b);
```

```
        long double q = 1 - p;
        long double F;
        long double forF;
        //long double forF = MathExpecting(p,n)/StdDeviation(p,n)ж
                forF =r / StdDeviation(p, 25);
        cout << "ForF = " << forF << endl;
        cout << "Enter F = ";
        cin >> F;
        cout << "for n = 25"  << ": " << 2*F << endl << endl;

 forF = r / StdDeviation(p, 50);
        cout << "ForF = " << forF << endl;
        cout << "Enter F = ";
        cin >> F;
        cout << "for n = 50"  << ": " << 2 * F << endl << endl;

 forF = r / StdDeviation(p, 100);
        cout << "ForF = " << forF << endl;
        cout << "Enter F = ";
        cin >> F;
        cout << "for n = 100"  << ": " << 2 * F << endl << endl;
}
void Number1Point5(int r, int g, int b, int n)
{
        long double p = (long double)r / ((long double)r + (long double)g + (long
double)b);
        long double q = 1 - p;
        long double F;
        long double forF;
```

```
	forF = p / StdDeviationRelative(p, 100);
	cout << "ForF = " << forF << endl;
	cout << "Enter F = ";
	cin >> F;
	cout << "for n = 100" << ": " << 2 * F << endl << endl;

	forF = p / StdDeviationRelative(p, 200);
	cout << "ForF = " << forF << endl;
	cout << "Enter F = ";
	cin >> F;
	cout << "for n = 200" << ": " << 2 * F << endl << endl;

	forF = p / StdDeviationRelative(p, 400);
	cout << "ForF = " << forF << endl;
	cout << "Enter F = ";
	cin >> F;
	cout << "for n = 400" << ": " << 2 * F << endl << endl;
}
void Number1Point6(int r, int g, int b, int n)
{
	long double p = (long double)r / ((long double)r + (long double)g + (long double)b);
	cout <<"Deviation: " <<StdDeviation(p, n)<<endl;
	cout << "MathExpecting: " << MathExpecting(p, n) << endl;
}
void Number1Point7(int r, int g, int b, int n)
{
	long double N1 = (long double)r + (long double)g + (long double)b;
```

```
        long double phi;
        long double i=19;
        for(int i=19;i<=n;i++)
        {
                phi = (1024-11* (long double)i)/sqrtl(231* (long double)i);
                cout << i << ": " << phi << endl;
        }
}
#pragma endregion

#pragma region Punkt2
void Number2Point1(int r, int g, int b, int n)
{
        std::cout << "P(k)=...\n";
        for (int i = 0; i <= r; i++)
        {
                std::cout << i << "\t" <<setprecision(10)<<
ProbabilityWithoutReturn(n,i,r+g+b,r)<<endl;
        }
}
void Number2Point3(int r, int g, int b, int n)
{
        cout << MathExpectingWithoutReturn(r, r + g + b, n) << endl;
}
void Number2Point4(int r, int g, int b, int n)
{
        cout << DispersionWithoutReturn(r, r + g + b, n) << endl;
}
#pragma endregion
```

```
#pragma region Punkt3
void Number3Point1(int r, int g, int b, int n)
{
	std::cout << "P(k)=...\n";
	for (int i = r; i <= (r+g+b); i++)
	{
		std::cout << i << "\t" << setprecision(10) << 
ProbabilityWithoutReturnToR(r, r + g + b, i) << endl;
	}
}
void Number3Point2(int r, int g, int b, int n)
{
	std::cout << "M(k) = " << setprecision(10) << 
MathExpectingWithoutReturnToR(r, r + g + b) << endl;;
}
void Number3Point3(int r, int g, int b, int n)
{
	cout << DispersionWithoutReturntoR(r, r + g + b) << endl;
}
#pragma endregion

int main()
{
#pragma region input
	cout.setf(ios::fixed);
	cout << "Enter R,G,B,n:"<<endl;

	int R=0, G=0, B=0, n=1000;
```

```
        cout << "R= ";
        cin >> R;
        cout << "G= ";
        cin >> G;
        cout << "B= ";
        cin >> B;
        cout << "n= ";
        //cin >> n; //For punkt1
        //n = G + B;//FOr punkt2
        //Nothing for punkt3
        cout << n << endl;
        cout << "End of enter" << endl << endl;

#pragma endregion

        Number1Point6(R, G, B, n);
        return 0;
}
```