Задача 1

Условие. В городе с населением в $n+1$ человек некто узнаёт новость. Он передаёт её первому встречному, тот — ещё одному и т.д. На каждом шагу впервые узнавший новость может сообщить её любому из n человек с одинаковыми вероятностями.
Найти вероятность того, что в продолжение r единиц времени

1. Новость не возвратится к человеку, который узнал её первым.
2. Новость не будет никем повторена.

Решить ту же задачу в предположении, что на каждом шагу новость сообщается группе из N случайно выбранных людей.

Решение. В первом случае задачи решение довольно просто, а для $N \neq 1$ решить я её не могу.
В случае $N = 1$ новость в любой момент времени передаёт не более 1 человека (1, если он получил её в прошлый момент времени впервые, 0, если новость пришла к тому, кто её уже знал).

а. Посчитаем вероятность, что новость вернётся к первому человеку.
В первый момент времени второй человек узнаёт новость. Потом он с вероятностью $\frac{1}{n}$ говорит её первому, на чём всё заканчивается. В противном случае он передаёт новость третьему... k-тый человек имеет вероятность $\frac{1}{n}$ передать её первому, $\frac{k-2}{n}$ — передать её не-первому человеку, который новость уже знает и $\frac{n-k+1}{n}$ — передать новость новому (k+1-му человеку). Итого вероятность того, что новость вернётся к первому, составляет

$$\frac{1}{n} + \frac{n-1}{n}\left(\frac{1}{n} + \frac{n-2}{n}\left(\frac{1}{n} + \frac{n-3}{n}\left(\frac{1}{n} + \cdots \frac{n-r+2}{n}\left(\frac{1}{n}\right)\cdots\right)\right)\right)$$

Это можно упростить до:

$$\frac{1}{n} + \frac{n-1}{n}\frac{1}{n} + \frac{(n-1)(n-2)}{n^2}\frac{1}{n} + \cdots + \frac{(n-1)(n-2)\cdots(n-r+2)}{n^{r-2}}\frac{1}{n}$$

Как это просуммировать, правда, я не знаю. Ответом является разность единицы и этой величины.

б. Не очень понятно, что имеется в виду под «повторена», если новость сообщает только впервые её услышавший. Вероятно, имеется в виду, что никто не услышит новость дважды. Тогда нам подходит ситуация, когда k-тый человек передаёт новость любому из $n-k$ не слышавших её, то есть искомая вероятность равна

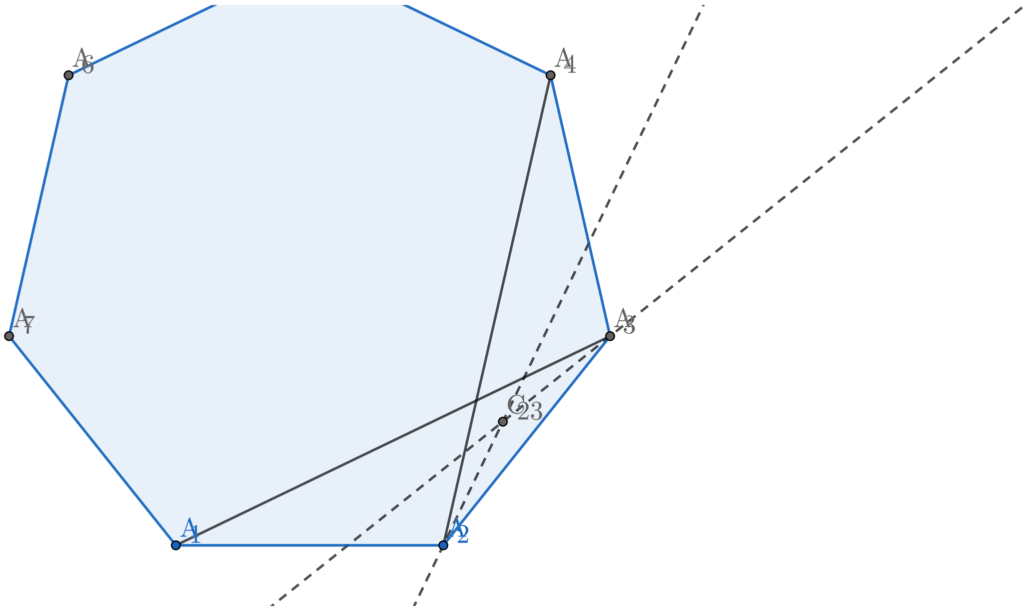
$$\frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \ldots \cdot \frac{n-r+1}{n}$$

Как это упростить, я всё ещё понятия не имею.

Задача 2.

Условие. Случайная точка A имеет равномерное распределение в правильном n-угольнике. Найти вероятность P_n, что точка A находится ближе к границе многоугольника, чем к его диагоналям. Найти числа C, α, что

$$P_n = Cn^{\alpha}(1 + o(1))$$



Решение. Пусть $A_1 \cdots A_n$ — искомый многоугольник. Нам нужно посчитать площадь той части, где точки ближе к сторонам, чем к диагоналям. Несложно заметить, что граница, разделяющая точки, которые ближе к одной прямой, чем к другой — биссектриса угла между ними. Т.е., если обратить внимание на рисунок выше, точки, которые ближе к A_2A_3, чем к A_1A_3 находятся «ниже» биссектрисы угла $A_1A_3A_2$ (т.е. «ниже» прямой A_3C_{23}).
Несложно заметить, что в треугольнике $A_2C_{23}A_3$ находятся точки, которые ближе к A_2A_3, чем к **любой** из диагоналей. И нигде в другом месте такие точки не находятся. То есть всё, что нам остаётся, — найти площадь этого треугольника, умножить её на n (потому что около каждой стороны есть такой) и поделить полученное на площадь многоугольника.
Пусть сторона многоугольника равна 1. Его площадь тогда равна

$$\frac{n}{4} \cot \frac{\pi}{n}$$

Теперь давайте посчитаем площадь треугольника $A_2C_{23}A_3$. Он, как несложно заметить, равнобедренный, а его основание — 1. Если посчитать углы, с площадью можно будет справиться.
Рассмотрим $\triangle A_1A_2A_3$. Он равнобедренный и в нём $\angle A_1A_2A_3 = \frac{\pi(n-2)}{n}$, а значит $\angle A_2A_3A_1 = \angle A_3A_1A_2 = \frac{\pi}{n}$. Следовательно $\angle C_{23}A_3A_2 = \frac{\pi}{2n}$, и аналогично $\angle C_{23}A_2A_3 = \frac{\pi}{2n}$. А отсюда $\angle A_2C_{23}C_3 = \frac{\pi(n-1)}{n}$. По формуле площади треугольника через три угла и сторону

$$S_{\triangle A_2C_{23}A_3} = \frac{\left(\sin \frac{\pi}{2n}\right)^2}{2 \sin \frac{\pi(n-1)}{n}}$$

Итого ответом к задаче является

$$\frac{n \frac{\left(\sin \frac{\pi}{2n}\right)^2}{2 \sin \frac{\pi(n-1)}{n}}}{\frac{n}{4} \cot \frac{\pi}{n}} = \frac{2\left(\sin \frac{\pi}{2n}\right)^2}{\sin \frac{\pi(n-1)}{n} \cot \frac{\pi}{n}} = \frac{1 - \cos \frac{\pi}{n}}{\sin\left(\pi - \frac{\pi}{n}\right) \cot \frac{\pi}{n}} = \frac{1 - \cos \frac{\pi}{n}}{\cos \frac{\pi}{n}} = \frac{1}{\cos \frac{\pi}{n}} - 1$$

Осталось только оценить P_n:

$$\lim_{x\to\infty}\frac{\frac{1}{\cos\frac{\pi}{x}}-1}{Cx^\alpha}=1\overset{\hat{o}}{\Leftrightarrow}\lim_{x\to\infty}\frac{-\frac{\pi\tan\frac{\pi}{x}}{x^2\cos\frac{\pi}{x}}}{C\alpha x^{\alpha-1}}=1\Leftrightarrow\lim_{x\to\infty}\frac{-\frac{\pi\tan\frac{\pi}{x}}{\cos\frac{\pi}{x}}}{C\alpha x^{\alpha+1}}=1\overset{\hat{o}}{\Leftrightarrow}$$
$$\overset{\hat{o}}{\Leftrightarrow}\lim_{x\to\infty}\frac{\frac{\pi^2\left(\tan^2\frac{\pi}{x}+\frac{1}{\cos^2\frac{\pi}{x}}\right)}{x^2\cos\frac{\pi}{x}}}{C\alpha(\alpha+1)x^\alpha}=1\Leftrightarrow\lim_{x\to\infty}\frac{\frac{\pi^2\left(\tan^2\frac{\pi}{x}+\frac{1}{\cos^2\frac{\pi}{x}}\right)}{\cos\frac{\pi}{x}}}{C\alpha(\alpha+1)x^{\alpha+2}}=1\Leftrightarrow$$
$$\Leftrightarrow\begin{cases}\dfrac{\pi^2}{C\alpha(\alpha+1)}=1\\ \alpha+2=0\end{cases}\Leftrightarrow\begin{cases}C=\dfrac{\pi^2}{2}\\ \alpha=-2\end{cases}$$

Ответ: $P_n=\frac{\pi^2}{2}n^{-2}(1+o(1))$.

Задача 3.

Условие. Введем события $A_i=\{X=i\}$, $B_i=\{Y=i\}$, $i\geqslant 0$. Известно, что для любых $i\geqslant 0$ и $j\geqslant 0$ события A_i и B_i независимы и

$$P(X=i)=e^{-\lambda}\frac{\lambda^i}{i!}\qquad\lambda>0$$
$$P(Y=j)=e^{-\mu}\frac{\mu^i}{i!}\qquad\mu>0$$

Найти $P(X=i\mid X+Y=j)$.

Трактовка условия. Для начала давайте поймём, что такое i и j, исходя из этого условия. На мой взгляд, это неотрицательное **целое** число т.к.

$$\sum_{i=0}^{\infty}P(X=i)=e^{-\lambda}\underbrace{\sum_{i=0}^{\infty}\frac{\lambda^i}{i!}}_{\text{Ряд Тейлора } e^\lambda}=1$$

Решение. По определению условной вероятности

$$P(X=i\mid X+Y=j)=\frac{P(X+Y=j\wedge X=i)}{P(X+Y=j)}=\frac{P(Y=j-i\wedge X=i)}{P(X+Y=j)}$$

Отсюда сразу видно, что если $j<i$ или $i<0$, то искомая условная вероятность — ноль, а если $j<0$, то не определена. Числитель этой дроби понятно какой, а вот знаменатель надо посчитать. Зная, что i и j, целые (и неотрицательные), разобьём $\{X+Y=j\}$ на следующие попарно несовместные события:

0. $\{X=0\wedge Y=j\}$.
1. $\{X=1\wedge Y=j-1\}$.

...

j. $\{X=j\wedge Y=0\}$.

Вероятности их соотвественно равны

0.
$$e^{-\lambda}\frac{\lambda^0}{0!}e^{-\mu}\frac{\mu^j}{j!}$$

1.
$$e^{-\lambda}\frac{\lambda^1}{1!}e^{-\mu}\frac{\mu^{j-1}}{(j-1)!}$$

...

j.
$$e^{-\lambda}\frac{\lambda^j}{j!}e^{-\mu}\frac{\mu^0}{0!}$$

Поскольку эти события несовместны, а их объединение равно $\{X+Y=j\}$, надо лишь сложить искомые вероятности.
$$\sum_{i=0}^{j} e^{-\lambda}\frac{\lambda^i}{i!}e^{-\mu}\frac{\mu^{j-i}}{(j-i)!} = \frac{e^{-\lambda-\mu}}{j!}\sum_{i=0}^{j}\frac{j!}{i!(j-i)!}\lambda^i\mu^{j-i} = \frac{(\lambda+\mu)^j}{j!\cdot e^{\lambda+\mu}}$$

Осталось лись поделить $P(X=i \wedge Y=j-i)$ на это.
Ответ: $\binom{j}{i}\dfrac{\lambda^i\mu^{j-i}}{(\lambda+\mu)^j}$.

Задача 4.

Условие. Рассмотрите схемы Бернулли при $n\in\{10,100,1000,10000\}$ и $p\in\{0.001,0.01,0.1,0.25,0.5\}$ и рассчитайте точные вероятности (где это возможно) $P\left(S_n\in\left[\frac{n}{2}-\sqrt{npq};\frac{n}{2}+\sqrt{npq}\right]\right)$, S_n — количество успехов в n испытаниях, и приближенную с помощью одной из предельных теорем. Сравните точные и приближенные вероятности.
Объясните результаты.

Решение. Для начала, это очень просто оценить «*с помощью одной из предельных теорем*». Согласно интегральной теореме Муавра — Лапласа,

$$P\left(x_1\sqrt{npq}+np \leqslant S_n \leqslant x_2\sqrt{npq}+np\right) \approx \frac{1}{\sqrt{2\pi}}\int\limits_{x_1}^{x_2} e^{-\frac{t^2}{2}}\ \mathrm{d}t$$

Несложно вывести формулу

$$x_1 = -1+\frac{(1-2p)\sqrt{npq}}{2p(1-p)} \qquad x_2 = 1+\frac{(1-2p)\sqrt{npq}}{2p(1-p)}$$

n	p	x_1	x_2	$\frac{1}{\sqrt{2\pi}}\int\limits_{x_1}^{x_2} e^{-\frac{t^2}{2}}\ \mathrm{d}t$
10	0.001	$-1+\frac{499\sqrt{1110}}{333}$	$1+\frac{499\sqrt{1110}}{333}$	5.48×10^{-523}
100	0.001	$-1+\frac{4990\sqrt{111}}{333}$	$1+\frac{4990\sqrt{111}}{333}$	8.99×10^{-5348}
1000	0.001	$-1+\frac{4990\sqrt{1110}}{333}$	$1+\frac{4990\sqrt{1110}}{333}$	1.27×10^{-53911}
10000	0.001	$-1+\frac{49900\sqrt{111}}{333}$	$1+\frac{49900\sqrt{111}}{333}$	2.48×10^{-540559}
10	0.01	$-1+\frac{49\sqrt{110}}{33}$	$1+\frac{49\sqrt{110}}{33}$	8.29×10^{-49}
100	0.01	$-1+\frac{490\sqrt{11}}{33}$	$1+\frac{490\sqrt{11}}{33}$	1.13×10^{-508}
1000	0.01	$-1+\frac{490\sqrt{110}}{33}$	$1+\frac{490\sqrt{110}}{33}$	1.15×10^{-5202}
10000	0.01	$-1+\frac{4900\sqrt{11}}{33}$	$1+\frac{4900\sqrt{11}}{33}$	3.02×10^{-52454}
10	0.1	$-1+\frac{4\sqrt{10}}{3}$	$1+\frac{4\sqrt{10}}{3}$	2.59×10^{-4}
100	0.1	$\frac{37}{3}$	$\frac{43}{3}$	1.2×10^{-35}
1000	0.1	$-1+\frac{40\sqrt{10}}{3}$	$1+\frac{40\sqrt{10}}{3}$	4.38×10^{-371}
10000	0.1	$\frac{397}{3}$	$\frac{403}{3}$	2.36×10^{-3806}
10	0.25	$-1+\frac{\sqrt{30}}{3}$	$1+\frac{\sqrt{30}}{3}$	8.1×10^{-2}
100	0.25	$-1+\frac{10\sqrt{3}}{3}$	$1+\frac{10\sqrt{3}}{3}$	3.61×10^{-7}
1000	0.25	$-1+\frac{10\sqrt{30}}{3}$	$1+\frac{10\sqrt{30}}{3}$	1.96×10^{-63}
10000	0.25	$-1+\frac{100\sqrt{3}}{3}$	$1+\frac{100\sqrt{3}}{3}$	3.02×10^{-702}
Сколько угодно	0.5	-1	1	2.7×10^{-1}

Для $n = 10$ посчитаем искомые вероятности явно.

p	Подходящие исходы	Вероятность
0.001	$\{5\}$	$\binom{10}{5}0.001^5 0.999^5 = \frac{63}{250000} \approx 2.5\times10^{-4}$
0.01	$\{5\}$	$\binom{10}{5}0.01^5 0.99^5 =$ много цифр $\approx 2.4\times10^{-2}$
0.1	$\{5\}$	$\binom{10}{5}0.1^5 0.9^5 =$ много цифр $\approx 1.5\times10^{-3}$
0.25	$\{4; 5; 6\}$	$\binom{10}{4}0.25^4 0.75^6 + \binom{10}{5}0.25^5 0.75^5 + \binom{10}{6}0.25^6 0.75^4 = \frac{28917}{131072} \approx 2.2\times10^{-1}$
0.5	$\{4; 5; 6\}$	$\binom{10}{4}0.5^4 0.5^6 + \binom{10}{5}0.5^5 0.5^5 + \binom{10}{6}0.5^6 0.5^4 = \frac{21}{32} \approx 6.6\times10^{-1}$

Разумеется, для $n = 10$ результаты сходятся с оценкой довольно плохо т.к. она нормально работает только при больших n.

Симуляция. Для симуляции этого процесса мной был написан следующий код на языке C++ стандарта 20:

```
#include <algorithm>
#include <array>
#include <cmath>
#include <concepts>
#include <format>
#include <iostream>
#include <mutex>
#include <random>
#include <thread>
#include <vector>

template <class T>
concept copy_constructible =
    std::constructible_from<T, T&> && std::convertible_to<T&, T>
    && std::constructible_from<T, const T&> && std::convertible_to<const T&, T>
```

```
    && std::constructible_from<T, const T> && std::convertible_to<const T, T>;

template <class T>
concept copy_assignable =
    std::assignable_from<T&, T&> && std::assignable_from<T&, const T&>
    && std::assignable_from<T&, const T>;

template <class P, class D>
concept distribution_param_type =
    copy_constructible<typename D::param_type> && copy_assignable<typename D::param_type>
    && std::equality_comparable<P>
    && requires {
           typename P::distribution_type;
           requires std::same_as<typename P::distribution_type, D>;
       };

template <class D>
concept random_number_distribution =
    copy_constructible<D> && copy_assignable<D> && std::equality_comparable<D>
    && requires(
            D d,
            const D x,
            typename D::param_type p,
            std::ranlux24 g,
            std::basic_istream<char> is,
            std::basic_ostream<char> os) {
           typename D::result_type;
           requires std::integral<typename D::result_type>
                   || std::floating_point<typename D::result_type>;
           typename D::param_type;
           requires distribution_param_type<typename D::param_type, D>;
           D();
           D(p);
           {
               d.reset()
           } -> std::same_as<void>;
           {
               x.param()
           } -> std::same_as<typename D::param_type>;
           {
               d.param(p)
           } -> std::same_as<void>;
           {
               d(g)
           } -> std::same_as<typename D::result_type>;
           {
               d(g, p)
           } -> std::same_as<typename D::result_type>;
           {
               x.min()
           } -> std::same_as<typename D::result_type>;
           {
               x.max()
           } -> std::same_as<typename D::result_type>;
```

```
            {
                os << d
            } -> std::same_as<decltype(os)&>;
            {
                is >> d
            } -> std::same_as<decltype(is)&>;
        };

static_assert(random_number_distribution<std::binomial_distribution<std::size_t>>);

template <std::totally_ordered value_type>
class in_range
{
private:
    value_type from, to;

public:
    in_range(const value_type& from, const value_type& to) : from(from), to(to) {}

    bool operator()(const value_type& value) const
    {
        return from <= value && value < to;
    }
};

template <random_number_distribution distribution,
          std::predicate<typename distribution::result_type> predicate>
std::uint64_t variable_in_range(
        std::uniform_random_bit_generator auto& gen,
        distribution& distr,
        const predicate& p,
        std::uint64_t series)
{
    std::uint64_t count = 0;
    while (series-- > 0)
    {
        if (p(distr(gen)))
        {
            count++;
        }
    }
    return count;
}

int main()
{
    std::cout.precision(20);

    std::array<std::uint64_t, 4> ns = {
        {10, 100, 1'000, 10'000}
    };
    std::array<double, 5> ps = {
        {0.001, 0.01, 0.1, 0.25, 0.5}
    };
```

```
    std::vector<std::thread> threads;
    std::mutex in_lock;

    for (std::size_t i = 0; i < ns.size(); i++)
    {
        std::uint64_t n = ns[i];
        for (std::size_t j = 0; j < ps.size(); j++)
        {
            double p = ps[j];

            threads.emplace_back(
            [i, j, n, p, &in_lock]() -> void
            {
                std::mt19937_64 engine;
                std::uint64_t series = 1'000'000'000;
                std::uint64_t lower_bound =
                static_cast<std::uint64_t>(std::ceil(n / 2.0L - std::sqrt(n * p * (1.0L - p))));
                std::uint64_t upper_bound =
                static_cast<std::uint64_t>(std::ceil(n / 2.0L + std::sqrt(n * p * (1.0L - p))));
                std::binomial_distribution distr(n, p);
                std::uint64_t positive =
                        variable_in_range(engine, distr, in_range(lower_bound, upper_bound), series);

                std::unique_lock lck(in_lock);
                std::format_to(
                        std::ostream_iterator<char>(std::cout),
                        "{:>6} {:>6.3}:    {:>10}/{:<10}\n",
                        n,
                        p,
                        positive,
                        series);
                std::cout << std::flush;
            });
        }
    }

    for (std::thread& thread : threads)
    {
        thread.join();
    }
    return 0;
}
```

Он запускает биномиальное распределение с параметрами n и p несколько (миллион) раз, и проверяет, сколько раз значение лежало в диапазоне $\left[\frac{n}{2} - \sqrt{npq}; \frac{n}{2} + \sqrt{npq}\right]$ (точнее, $\left[\left\lceil \frac{n}{2} - \sqrt{npq}\right\rceil; \left\lceil \frac{n}{2} + \sqrt{npq}\right\rceil\right)$, что для целых чисел то же самое). Делается всё это в несколько потоков.
Результатом данной программы является следующее:

```
   100  0.001:             0/1000000000
    10   0.01:            22/1000000000
    10  0.001:             0/1000000000
    10    0.5:     656231930/1000000000
    10    0.1:       1486671/1000000000
```

```
   10   0.25:      220634338/1000000000
 1000  0.001:              0/1000000000
  100   0.01:              0/1000000000
 1000   0.01:              0/1000000000
  100    0.1:              0/1000000000
  100    0.5:      680280034/1000000000
10000  0.001:              0/1000000000
10000   0.01:              0/1000000000
 1000    0.1:              0/1000000000
10000   0.25:              0/1000000000
 1000    0.5:      673057269/1000000000
 1000   0.25:              0/1000000000
  100   0.25:           4278/1000000000
10000    0.1:              0/1000000000
10000    0.5:      682658421/1000000000
```

Почему, например, при $n = 1000$, $p = 0.25$ количество позитивных результатов равен нулю? Вероятно, потому что не повезло.