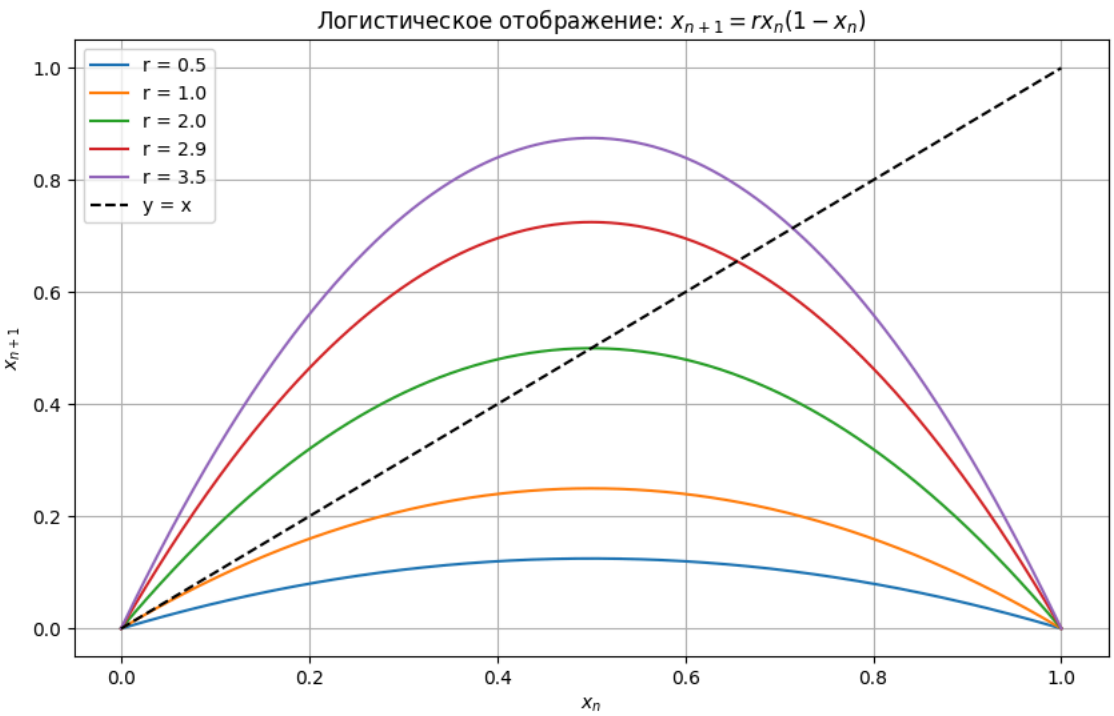
Easy: Доказательство ограниченности

Докажем, что для логистического отображения $x_{n+1} = rx_n(1 - x_n)$ при условиях $r \in [0, 1]$ и $0 < x_0 < 1$ выполняется $0 < x_n < 1$ для всех $n \in \mathbb{N}$.

Доказательство методом математической индукции:

1. **База:** при $n = 0$ имеем $0 < x_0 < 1$ по условию.
2. **Предположение:** пусть для некоторого $n = k$ выполняется $0 < x_k < 1$.
3. **Шаг:** докажем для $n = k + 1$:
$$x_{k+1} = rx_k(1 - x_k)$$
Так как $x_k > 0$ и $1 - x_k > 0$, то $x_k(1 - x_k) > 0$. При $r > 0$ имеем $x_{k+1} > 0$.
Функция $f(x) = x(1 - x)$ на интервале $(0, 1)$ достигает максимума в точке $x = 0.5$:
$$f(0.5) = 0.25$$
Следовательно, $x_k(1 - x_k) \leq 0.25$.
Тогда: $x_{k+1} = r \cdot x_k(1 - x_k) \leq 1 \cdot 0.25 = 0.25 < 1$.
4. По принципу математической индукции утверждение доказано.

```
# Easy: Графики логистического отображения
import numpy as np
import matplotlib.pyplot as plt

def logistic(x, r):
    return r * x * (1 - x)

x = np.linspace(0, 1, 1000)
r_values = [0.5, 1.0, 2.0, 2.9, 3.5]

plt.figure(figsize=(10, 6))
for r in r_values:
    plt.plot(x, logistic(x, r), label=f'r = {r}')

plt.plot(x, x, 'k--', label='y = x')
plt.title('Логистическое отображение: $x_{n+1} = r x_n (1 - x_n)$')
plt.xlabel('$x_n$')
plt.ylabel('$x_{n+1}$')
plt.legend()
plt.grid()
plt.show()
```



Анализ влияния параметра r

По построенным графикам можно сделать следующие выводы:

1. **При r ∈ [0, 1):**
   - График функции целиком лежит **ниже** биссектрисы $y = x$
   - Единственная неподвижная точка: $x^* = 0$
   - Так как $x_{n+1} < x_n$ для всех $x_n \in (0, 1)$, последовательность монотонно убывает
   - Предел: $\lim\limits_{n \to \infty} x_n = 0$
2. **При r = 1:**
   - График касается биссектрисы в точке $x = 0$
   - Последовательность также убывает к 0
3. **При r ∈ (1, 3]:**
   - График пересекает биссектрису в двух точках: $x_1^* = 0$ и $x_2^* = 1 - \frac{1}{r}$
   - Точка $x_1^* = 0$ становится неустойчивой
   - Точка $x_2^* = 1 - \frac{1}{r}$ — устойчивая неподвижная точка
   - При r чуть больше 1 наблюдается **монотонная сходимость** к x_2^*
   - При r ближе к 3 наблюдается **колебательная сходимость** к x_2^*
4. **При r > 3:**
   - Неподвижная точка x_2^* теряет устойчивость
   - Возникают **циклы** (период 2, затем 4, 8, ...)
   - При $r \approx 3.57$ начинается **хаотичное поведение**

Easy: Анализ отображения варианта N=3

Формула варианта:

$$x_{n+1} = r x_n (1 - x_n)(3 - x_n)$$

Допустимый диапазон параметра:

$$r \in \left[0; \frac{27}{2(7\sqrt{7} - 10)}\right] \approx [0; 1.585]$$

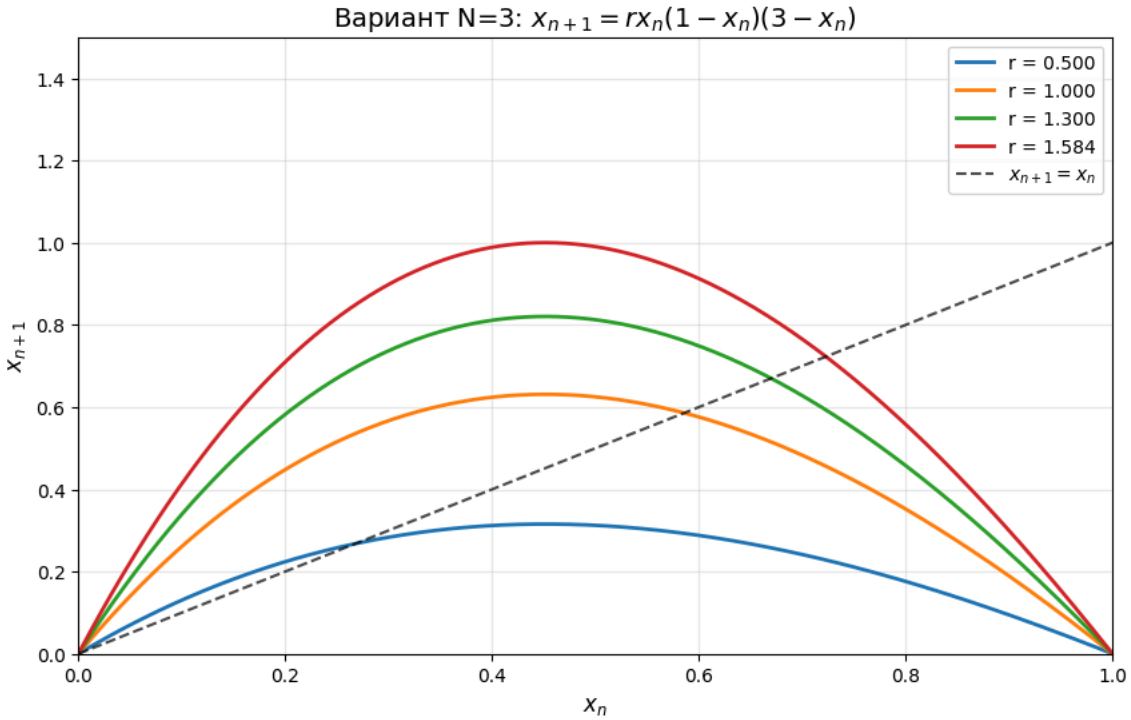
Задание:

1. Построить графики зависимости x_{n+1} от x_n для нескольких значений r
2. Сравнить с логистическим отображением
3. Объяснить причины сходств и различий

```
import numpy as np
import matplotlib.pyplot as plt
import math

# Функция для варианта N=3
def variant_map(x, r):
    """Вариант N=3: x_{n+1} = r * x_n * (1 - x_n) * (3 - x_n)"""
    return r * x * (1 - x) * (3 - x)

# Вычисление максимального r
r_max = 27 / (2 * (7 * math.sqrt(7) - 10))
print(f"Диапазон параметра r: [0, {r_max:.3f}]")

# Значения r для построения (равномерно распределенные)
r_values = [0.5, 1.0, 1.3, r_max]  # 4 различных значения

# Построение графиков
x = np.linspace(0, 1, 1000)
plt.figure(figsize=(10, 6))

for r in r_values:
    y = variant_map(x, r)
    plt.plot(x, y, label=f'r = {r:.3f}', linewidth=2)

# Биссектриса для сравнения
plt.plot(x, x, 'k--', label='$x_{n+1} = x_n$', linewidth=1.5, alpha=0.7)

# Настройки графика
plt.title('Вариант N=3: $x_{n+1} = r x_n (1 - x_n)(3 - x_n)$', fontsize=14)
plt.xlabel('$x_n$', fontsize=12)
plt.ylabel('$x_{n+1}$', fontsize=12)
```

```
    plt.legend(fontsize=10)
    plt.grid(True, alpha=0.3)
    plt.xlim(0, 1)
    plt.ylim(0, 1.5)
    plt.show()
```

```
Диапазон параметра r: [0, 1.584]
```



Выводы о сходстве и различии поведения

Сходства:

1. **Базовая динамика:** Оба отображения показывают:
   - Убывание к 0 при малых r ($r < 1$)
   - Появление нетривиальных неподвижных точек при увеличении r
   - Возможность сложного поведения при больших r
2. **Неподвижные точки:** В обоих случаях:
   - $x = 0$ всегда неподвижная точка
   - Количество неподвижных точек увеличивается с ростом r
3. **Бифуркации:** Оба демонстрируют бифуркационное поведение

Различия:

1. **Математическая форма:**
   - Логистическое: $x_{n+1} = rx_n(1 - x_n)$ (квадратичное)
   - Вариант N=3: $x_{n+1} = rx_n(1 - x_n)(3 - x_n)$ (кубическое)
2. **Форма графика:**
   - Логистическое: симметричная парабола
   - Вариант N=3: несимметричная кривая
3. **Количественные характеристики:**

Параметр	Логистическое	Вариант N=3
Максимум при	$x = 0.5$	$x \approx 0.423$
Диапазон r	$[0, 4]$	$[0, 1.585]$
Макс. значение	$r/4$	$\approx 0.385r$

Причины сходств:

- Оба отображения моделируют одну физическую реальность: популяционную динамику
- Оба учитывают ограниченность ресурсов (член $1 - x$)
- Оба являются нелинейными системами с параметром управления

Причины различий:

1. **Кубическая vs квадратичная нелинейность:** Вариант N=3 содержит дополнительную степень свободы
2. **Дополнительный множитель $(3 - x)$:** Может соответствовать:
   - Миграционным процессам
   - Кооперативному поведению
   - Другим формам биологического взаимодействия
3. **Физические предположения:** Разные модели делают разные допущения о взаимодействии особей

Общий вывод:

Параметр r в обоих отображениях играет схожую роль: определяет "силу" роста популяции. Однако из-за разной математической структуры количественное поведение систем различается. Вариант N=3 представляет более сложную и, возможно, более реалистичную модель, но за счёт усложнения анализа.

Normal: Задание 1 - Неподвижные точки логистического отображения

Дано: Логистическое отображение $x_{n+1} = rx_n(1 - x_n)$

1. Найти все неподвижные точки:

Решаем $x = rx(1 - x)$:

$$x - rx(1-x) = 0$$
$$x[1 - r(1-x)] = 0$$
$$x(1 - r + rx) = 0$$

Решения:

1. $x_1^* = 0$ (всегда существует)
2. Из $1 - r + rx = 0$: $x_2^* = 1 - \frac{1}{r}$ (при $r \neq 0$)

2. Количество неподвижных точек в зависимости от r:

Диапазон r	Количество точек	Объяснение
$r = 0$	1	Уравнение $x = 0$
$0 < r < 1$	1	$x_2^* < 0$ вне $[0, 1]$
$r = 1$	1 (кратная)	$x_2^* = 0$ совпадает с x_1^*
$r > 1$	2	$0 < x_2^* < 1$

3. Максимальное количество неподвижных точек: Максимум **2** точки. Причина: уравнение $x = rx(1 - x)$ является квадратным относительно x после преобразования.

Normal: Задание 2 - Монотонность и предел при $r \in (0, 1]$

Дано: $x_0 \in (0, 1)$, $r \in (0, 1]$, $x_{n+1} = rx_n(1 - x_n)$

Доказательство монотонного убывания:

Рассмотрим разность:

$$x_{n+1} - x_n = rx_n(1 - x_n) - x_n = x_n[r(1 - x_n) - 1]$$

Анализируем:

1. $x_n > 0$ (из $x_0 > 0$ и сохранения положительности)
2. При $r \in (0, 1]$ и $x_n \in (0, 1)$:

$$r(1 - x_n) \leq 1 \cdot (1 - x_n) < 1 \Rightarrow r(1 - x_n) - 1 < 0$$

Следовательно:

$$x_{n+1} - x_n < 0 \Rightarrow x_{n+1} < x_n$$

Последовательность **монотонно убывает**.

Существование предела:

1. Последовательность ограничена снизу: $0 < x_n$
2. Последовательность монотонно убывает
3. По теореме Вейерштрасса: предел существует

Пусть $\lim_{n \to \infty} x_n = L$. Переходим к пределу:

$$L = rL(1 - L)$$
$$L[1 - r(1 - L)] = 0$$

Возможные пределы:

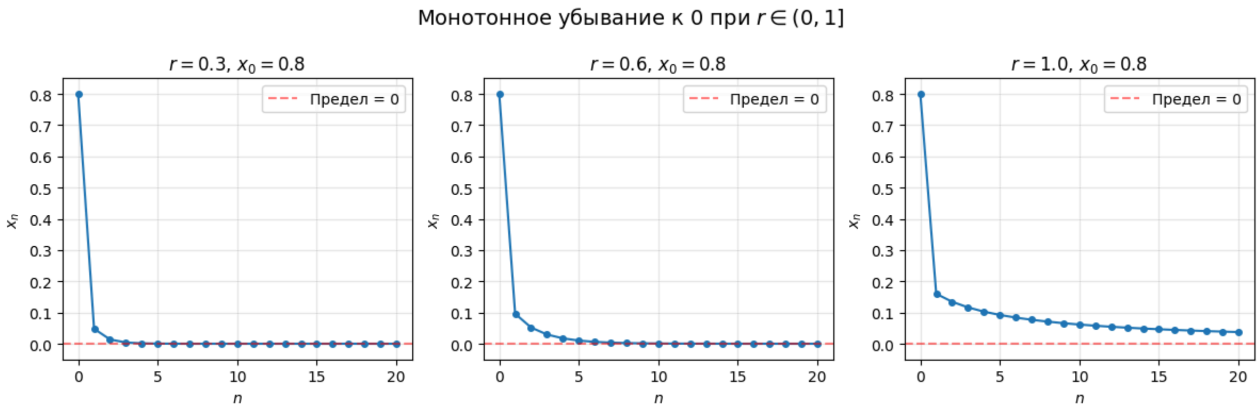
1. $L_1 = 0$
2. $L_2 = 1 - \frac{1}{r}$

При $r \in (0, 1]$:

- Для $r < 1$: $L_2 < 0$, не подходит (предел ≥ 0)
- Для $r = 1$: $L_2 = 0$

Итог: $\lim_{n\to\infty} x_n = 0$ при $r \in (0, 1]$

```
import numpy as np
import matplotlib.pyplot as plt

def logistic_sequence(x0, r, n_iter):
    """Генерирует последовательность логистического отображения"""
    x = [x0]
    for i in range(n_iter):
        x_next = r * x[-1] * (1 - x[-1])
        x.append(x_next)
    return x

# Параметры
r_values = [0.3, 0.6, 1.0]  # r ∈ (0,1]
x0 = 0.8
n_iter = 20

plt.figure(figsize=(12, 4))
for idx, r in enumerate(r_values, 1):
    seq = logistic_sequence(x0, r, n_iter)

    plt.subplot(1, 3, idx)
    plt.plot(range(n_iter+1), seq, 'o-', linewidth=1.5, markersize=4)
    plt.axhline(y=0, color='r', linestyle='--', alpha=0.5, label='Предел = 0')
    plt.title(f'$r = {r}$, $x_0 = {x0}$')
    plt.xlabel('$n$')
    plt.ylabel('$x_n$')
    plt.legend()
    plt.grid(alpha=0.3)
    plt.ylim(-0.05, 0.85)

plt.suptitle('Монотонное убывание к 0 при $r \\in (0,1]$', fontsize=14)
plt.tight_layout()
plt.show()
```



Normal: Задание 3 - Подпоследовательности при $r \in (2, 3)$

Дано: $r \in (2, 3)$, $x_{2n} > x^*$, $x_{2n+1} < x^*$, где $x^* = 1 - \frac{1}{r}$

Анализ: Производная в x^*:

$$f(x) = rx(1-x)$$
$$f'(x) = r(1-2x)$$
$$f'(x^*) = r\left(1 - 2\left(1 - \frac{1}{r}\right)\right) = 2 - r$$

При $r \in (2, 3)$: $-1 < f'(x^*) < 0$, значит f убывает в окрестности x^*.

Монотонность подпоследовательностей:

Рассмотрим $f^{(2)}(x) = f(f(x))$.

Если $x_{2n} > x^*$, то:

1. $x_{2n+1} = f(x_{2n}) < f(x^*) = x^*$ (f убывает)
2. $x_{2n+2} = f(x_{2n+1}) > f(x^*) = x^*$ (f возрастает при $x < x^*$)

Для $x > x^*$ функция $f^{(2)}(x)$ убывает, поэтому:

$$x_{2n+2} = f^{(2)}(x_{2n}) < x_{2n}$$

Для $x < x^*$ функция $f^{(2)}(x)$ возрастает, поэтому:

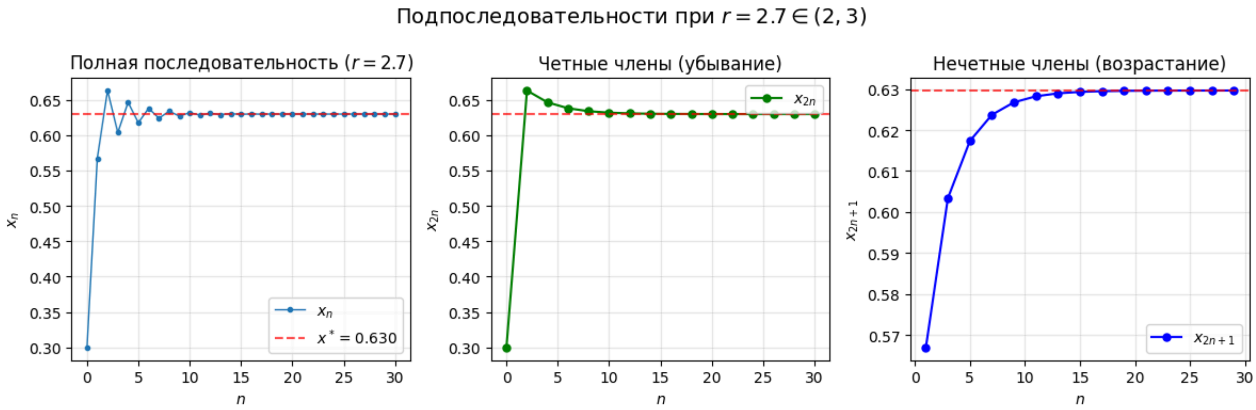
$$x_{2n+3} = f^{(2)}(x_{2n+1}) > x_{2n+1}$$

Вывод:

- $\{x_{2n}\}$ — монотонно убывает
- $\{x_{2n+1}\}$ — монотонно возрастает

```
def generate_and_analyze(r, x0, n_iter=30):
    """Генерирует последовательность и анализирует подпоследовательности"""
    # Генерируем последовательность
    seq = [x0]
    for i in range(n_iter):
        seq.append(r * seq[-1] * (1 - seq[-1]))

    # Неподвижная точка
    x_star = 1 - 1/r

    # Разделяем на четные и нечетные
    even_seq = seq[0::2]  # x0, x2, x4, ...
    odd_seq = seq[1::2]   # x1, x3, x5, ...

    return seq, x_star, even_seq, odd_seq

# Параметры
r = 2.7  # ∈ (2,3)
x0 = 0.3
n_iter = 30

seq, x_star, even_seq, odd_seq = generate_and_analyze(r, x0, n_iter)

# Построение графиков
plt.figure(figsize=(12, 4))

# 1. Полная последовательность
plt.subplot(1, 3, 1)
plt.plot(range(len(seq)), seq, 'o-', linewidth=1, markersize=3, label='$x_n$')
plt.axhline(y=x_star, color='r', linestyle='--', alpha=0.7, label=f'$x^* = {x_star:.3f}$')
plt.title(f'Полная последовательность ($r = {r}$)')
plt.xlabel('$n$')
plt.ylabel('$x_n$')
plt.legend()
plt.grid(alpha=0.3)

# 2. Четные члены
plt.subplot(1, 3, 2)
even_indices = list(range(0, len(seq), 2))
plt.plot(even_indices, even_seq, 'go-', linewidth=1.5, markersize=5, label='$x_{2n}$')
plt.axhline(y=x_star, color='r', linestyle='--', alpha=0.7)
plt.title('Четные члены (убывание)')
plt.xlabel('$n$')
plt.ylabel('$x_{2n}$')
plt.legend()
plt.grid(alpha=0.3)

# 3. Нечетные члены
plt.subplot(1, 3, 3)
odd_indices = list(range(1, len(seq), 2))
plt.plot(odd_indices, odd_seq, 'bo-', linewidth=1.5, markersize=5, label='$x_{2n+1}$')
plt.axhline(y=x_star, color='r', linestyle='--', alpha=0.7)
plt.title('Нечетные члены (возрастание)')
plt.xlabel('$n$')
plt.ylabel('$x_{2n+1}$')
plt.legend()
plt.grid(alpha=0.3)

plt.suptitle(f'Подпоследовательности при $r = {r} \\in (2,3)$', fontsize=14)
plt.tight_layout()
```

```
    plt.show()
```



Normal: Задание 4 - Анализ варианта N=3

Вариант: $x_{n+1} = rx_n(1-x_n)(3-x_n)$, $r \in \left[0; \frac{27}{2(7\sqrt{7}-10)}\right]$

1. Неподвижные точки: Решаем $x = rx(1-x)(3-x)$:

$$x[1 - r(1-x)(3-x)] = 0$$

Решение 1: $x_1^* = 0$ (всегда)

Из $1 = r(1-x)(3-x)$:

$$r(3 - 4x + x^2) = 1$$
$$rx^2 - 4rx + (3r-1) = 0$$

Дискриминант:

$$D = 16r^2 - 4r(3r-1) = 4r^2 + 4r = 4r(r+1)$$

Корни:

$$x_{2,3}^* = \frac{4r \pm 2\sqrt{r(r+1)}}{2r} = 2 \pm \sqrt{1 + \frac{1}{r}}$$

2. Сходимость к нулю: Производная:

$$g'(x) = r[(1-x)(3-x) - x(3-x) - x(1-x)] = r(3 - 8x + 3x^2)$$

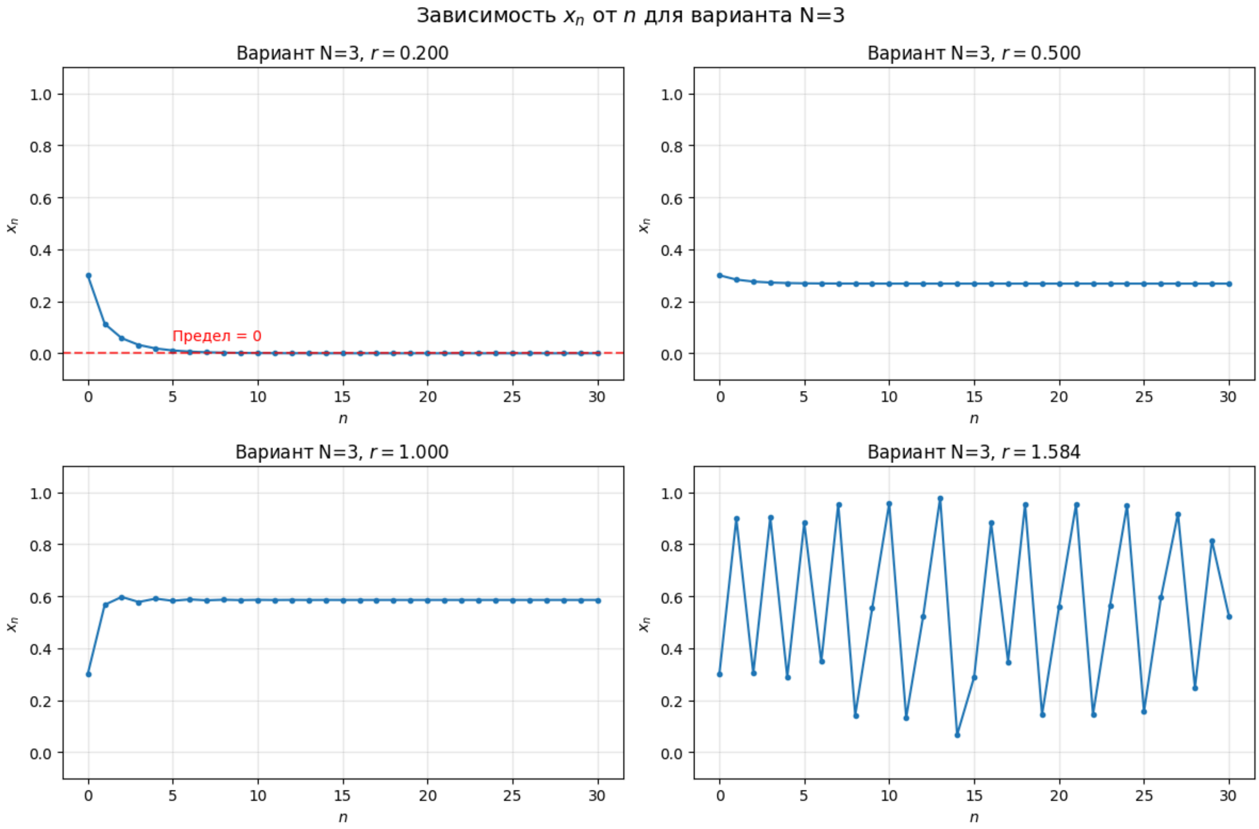
В точке $x = 0$: $g'(0) = 3r$

Условие устойчивости $x = 0$: $|g'(0)| < 1 \Rightarrow 3r < 1 \Rightarrow r < \frac{1}{3}$

Вывод: При $r < \frac{1}{3}$ последовательность монотонно сходится к 0.

```
import matplotlib.pyplot as plt
import math

def variant_sequence(x0, r, n_iter):
    seq = [x0]
    for i in range(n_iter):
        x_next = r * seq[-1] * (1 - seq[-1]) * (3 - seq[-1])
        seq.append(x_next)
    return seq

# Максимальное r
r_max = 27 / (2 * (7 * math.sqrt(7) - 10))

r_values_variant = [0.2, 0.5, 1.0, r_max]
x0 = 0.3
n_iter = 30

plt.figure(figsize=(12, 8))
for idx, r in enumerate(r_values_variant, 1):
    seq = variant_sequence(x0, r, n_iter)

    plt.subplot(2, 2, idx)
    plt.plot(range(n_iter+1), seq, 'o-', linewidth=1.5, markersize=3)

    if r < 1/3:
```

```
        plt.axhline(y=0, color='r', linestyle='--', alpha=0.7)
        plt.text(5, 0.05, 'Предел = 0', color='r', fontsize=10)

    plt.title(f'Вариант N=3, $r = {r:.3f}$')
    plt.xlabel('$n$')
    plt.ylabel('$x_n$')
    plt.grid(alpha=0.3)
    plt.ylim(-0.1, 1.1)

plt.suptitle('Зависимость $x_n$ от $n$ для варианта N=3', fontsize=14)
plt.tight_layout()
plt.show()
```



Hard: Задание 1 - Циклы и бифуркации

r∞ ≈ 3.5699456 - начало хаоса (точка Фейгенбаума)

Изменение длины цикла при r ∈ (3; r∞): Происходит **каскад удвоения периода** (периодические окна):

1. $r \approx 3.0$: возникает цикл периода 2
2. $r \approx 3.449$: период 4
3. $r \approx 3.544$: период 8
4. $r \approx 3.564$: период 16
5. И так далее до r_∞

Ограничения на m (длину цикла): В интервале $r \in (3; r_\infty)$ длина цикла m может принимать только значения:

$$m = 2^k, \quad k = 1, 2, 3, \dots$$

То есть **только степени двойки**.

Причина: Это свойство универсально для одномерных отображений с квадратичным максимумом (теорема Фейгенбаума).

Hard: Задание 2 - Лестница Ламерея

Текст задания:

1. Напишите функцию, которая для заданного параметра r строит лестницу Ламерея.

2. Сделайте выводы: как выглядят циклы различных порядков на графике?

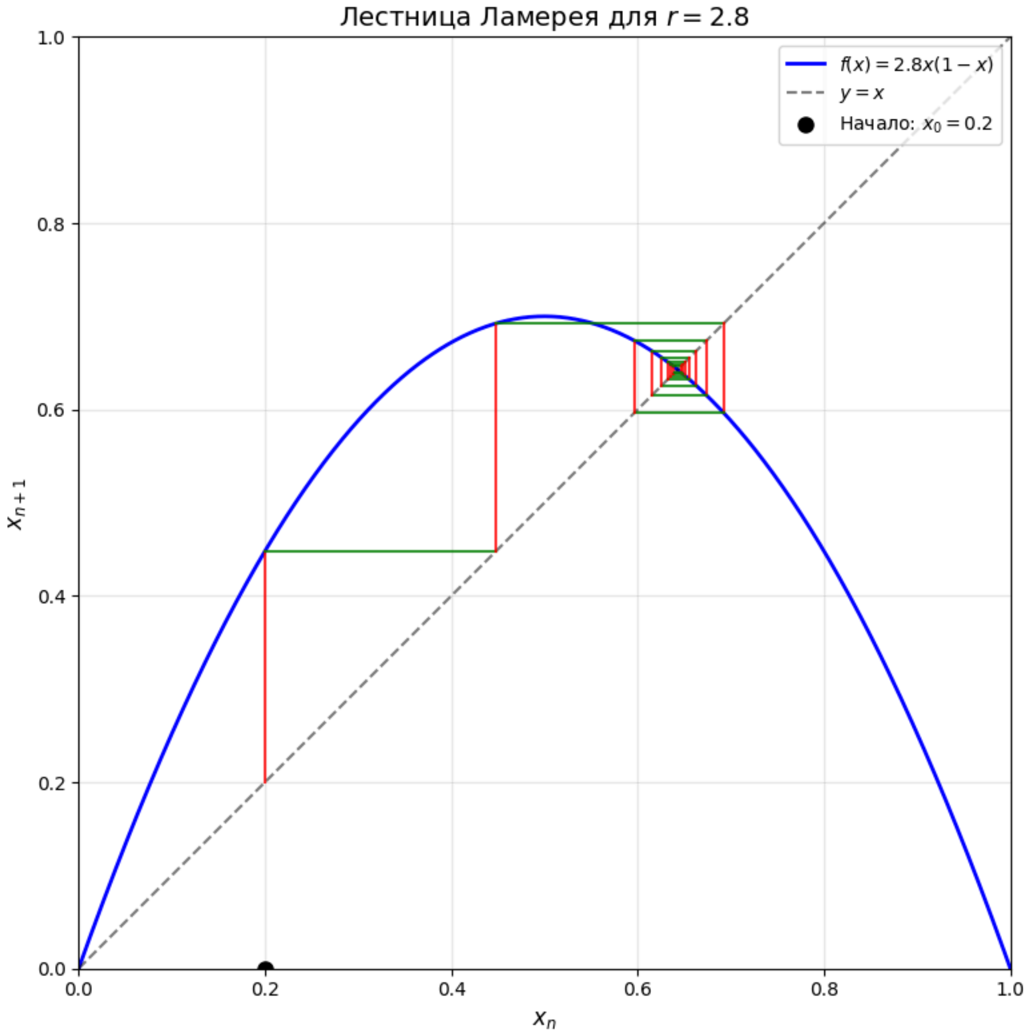
Лестница Ламерея — графический метод анализа сходимости к неподвижной точке или циклу.

Алгоритм построения:

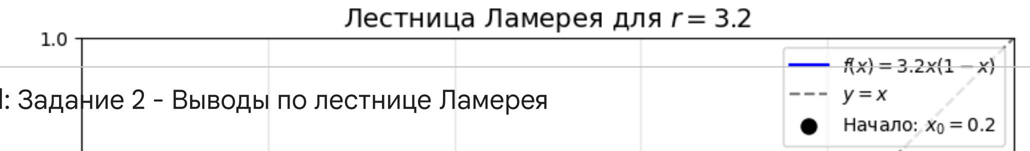
1. Из точки (x_n, x_n) вертикально до кривой отображения: (x_n, x_{n+1})
2. Горизонтально до биссектрисы: $(x_n, x_{n+1}) \to (x_{n+1}, x_{n+1})$
3. Повторять шаги 1-2

```
# Ячейка Code: Лестница Ламерея (полностью автономная)
import numpy as np
import matplotlib.pyplot as plt

def lamerey_diagram(r, x0=0.3, n_iter=20):
    """Автономная функция построения лестницы Ламерея"""
    # Генерируем последовательность
    x = [x0]
    for i in range(n_iter):
        x.append(r * x[-1] * (1 - x[-1]))

    # Создаем график
    fig, ax = plt.subplots(figsize=(8, 8))

    # Кривая отображения
    x_vals = np.linspace(0, 1, 1000)
    y_vals = r * x_vals * (1 - x_vals)
    ax.plot(x_vals, y_vals, 'b-', linewidth=2, label=f'$f(x) = {r}x(1-x)$')

    # Биссектриса
    ax.plot([0, 1], [0, 1], 'k--', alpha=0.5, label='$y = x$')

    # Построение лестницы
    for i in range(n_iter):
        # Вертикальная линия
        ax.plot([x[i], x[i]], [x[i], x[i+1]], 'r-', alpha=0.8, linewidth=1.5)
        # Горизонтальная линия
        ax.plot([x[i], x[i+1]], [x[i+1], x[i+1]], 'g-', alpha=0.8, linewidth=1.5)

    # Начальная точка
    ax.plot(x[0], 0, 'ko', markersize=8, label=f'Начало: $x_0 = {x0}$')

    ax.set_title(f'Лестница Ламерея для $r = {r}$', fontsize=14)
    ax.set_xlabel('$x_n$', fontsize=12)
    ax.set_ylabel('$x_{n+1}$', fontsize=12)
    ax.legend(loc='upper right')
    ax.grid(alpha=0.3)
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.set_aspect('equal')

    plt.tight_layout()
    plt.show()

    return x

# Пример 1: сходимость
print("\n1. r = 2.8 (сходимость к неподвижной точке):")
sequence = lamerey_diagram(2.8, x0=0.2, n_iter=15)

# Пример 2: цикл периода 2
print("\n2. r = 3.2 (цикл периода 2):")
sequence = lamerey_diagram(3.2, x0=0.2, n_iter=25)

# Пример 3: более сложное поведение
print("\n3. r = 3.5 (сложное поведение):")
sequence = lamerey_diagram(3.5, x0=0.2, n_iter=30)
```

```
1. r = 2.8 (сходимость к неподвижной точке):
```




```
2. r = 3.2 (цикл периода 2):
```



Hard: Задание 2 - Выводы по лестнице Ламерея

Как выглядят циклы различных порядков на графике:

Тип цикла	Значение r	Внешний вид на лестнице Ламерея
Период 1 (неподвижная точка)	r < 3	Лестница "закручивается" по спирали к точке пересечения кривой с биссектрисой
Период 2	r ≈ 3.2	Прямоугольник из 4 точек
Период 4	r ≈ 3.5	Более сложная фигура из 8 точек
Период 8	r ≈ 3.56	Ещё более сложная замкнутая фигура
Хаос	r > r∞	Нерегулярная структура, точки разбросаны случайно

Наблюдения:

1. Каждый цикл периода m визуализируется как замкнутая фигура из 2m точек
2. При приближении к точке бифуркации фигура становится более сложной
3. Лестница Ламерея наглядно показывает:
    - Устойчивость/неустойчивость неподвижных точек
    - Возникновение циклов при потере устойчивости
    - Переход к хаотическому поведению

Практическое применение: Лестница Ламерея позволяет визуально определить тип динамики системы без сложных вычислений.

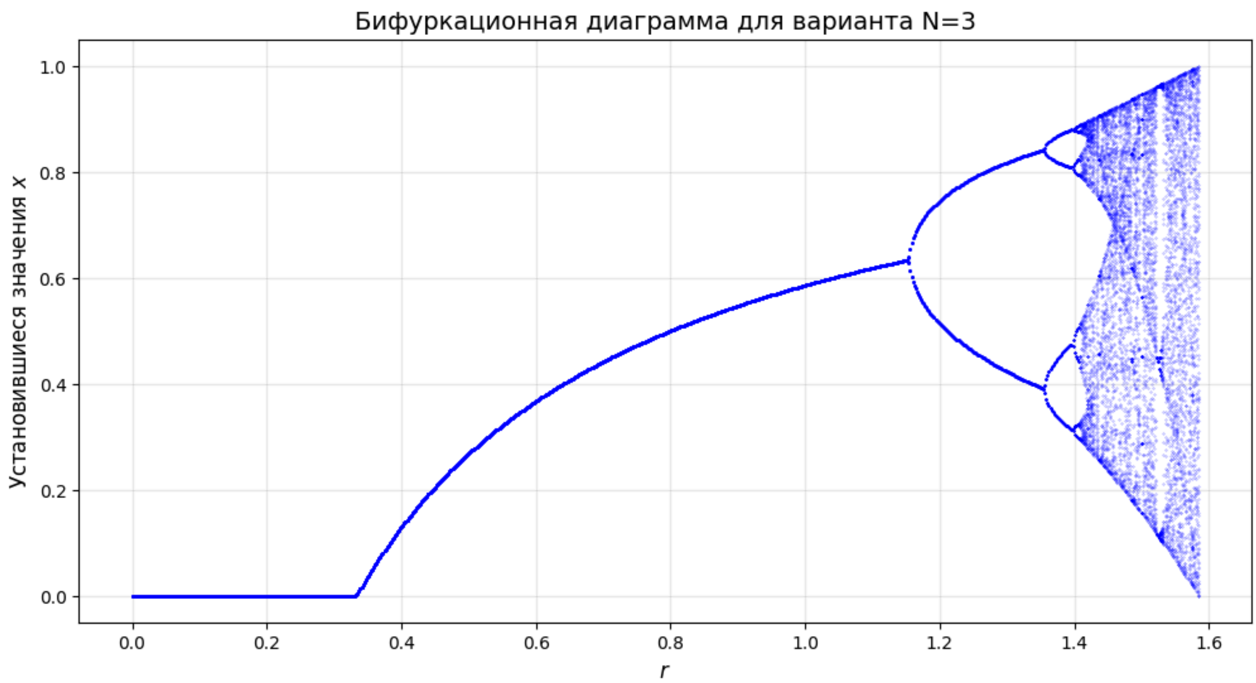
Hard: Задание 3 - Бифуркационная диаграмма для варианта N=3

Построим бифуркационную диаграмму для отображения:

$$x_{n+1} = rx_n(1-x_n)(3-x_n)$$

где $r \in \left[0; \frac{27}{2(7\sqrt{7}-10)}\right]$

```
import numpy as np
import matplotlib.pyplot as plt
import math

# Функция варианта N=3
def variant_map(x, r):
    return r * x * (1 - x) * (3 - x)

# Максимальное r
r_max = 27 / (2 * (7 * math.sqrt(7) - 10))

# БИФУРКАЦИОННАЯ ДИАГРАММА
n_r = 800
n_iter = 800
n_last = 100
r_values = np.linspace(0, r_max, n_r)

plt.figure(figsize=(12, 6))

for r in r_values:
    x = 0.5
    # Выгорание
    for _ in range(n_iter):
        x = variant_map(x, r)

    # Собираем установившиеся значения
    for _ in range(n_last):
        x = variant_map(x, r)
        plt.plot(r, x, 'b.', markersize=0.5, alpha=0.6)

plt.title('Бифуркационная диаграмма для варианта N=3', fontsize=14)
plt.xlabel('$r$', fontsize=12)
plt.ylabel('Установившиеся значения $x$', fontsize=12)
plt.grid(alpha=0.3)
plt.show()
```



Анализ бифуркационной диаграммы и сравнение с логистическим отображением

Наблюдения по диаграмме:

1. **r < 0.33:** Единственная ветвь (сходимость к $x = 0$)
2. **r ≈ 0.33:** Первая бифуркация - рождение ненулевой неподвижной точки
3. **0.33 < r < 1.2:** Одна устойчивая ветвь
4. **r ≈ 1.2:** Начало каскада бифуркаций
5. **r ≈ 1.5:** Переход к хаотическому режиму
6. **r_max ≈ 1.585:** Конец допустимого диапазона

Сравнение с логистическим отображением:

Параметр	Логистическое	Вариант N=3
Первая бифуркация	$r = 1.0$	$r \approx 0.33$
Начало хаоса	$r_\infty \approx 3.57$	$r \approx 1.5$
Диапазон r	$[0, 4]$	$[0, 1.585]$
Структура диаграммы	Классическая	Качественно похожа

Сходства:

1. Наличие каскада удвоения периода
2. Переход от порядка к хаосу через последовательность бифуркаций
3. Универсальная структура бифуркационной диаграммы

Различия:

1. Разные значения точек бифуркаций
2. Разный диапазон параметра r
3. Разная "скорость" перехода к хаосу

Вывод: Вариант N=3 демонстрирует качественно аналогичное поведение логистическому отображению, что подтверждает универсальность сценария перехода к хаосу через каскад удвоения периода. Количественные отличия обусловлены кубической нелинейностью и другим диапазоном параметра.