|  |  |  |
| --- | --- | --- |
| **Дисциплина** | **Лабораторная** | **ФИО** |
| Математические основы защиты информации и информационной безопасности | №7 | Александра Миличевич |

##Цель лабораторной работы

Цель лабораторной работы №7 заключается в ознакомлении студентов с алгоритмом Полларда для дискретного логарифмирования. Студенты должны изучить и реализовать алгоритм, который позволяет находить дискретные логарифмы в конечных полях, а также понять его применение в криптографии и теории чисел.

#### Как работает:

1. **Базовый случай:** Если b равно 0, возвращается кортеж (a, 1, 0).
2. **Рекурсия:** Рекурсивно вызывается extended\_euclidean(b, a % b), результат разворачивается, и вычисляются новые коэффициенты x и y.

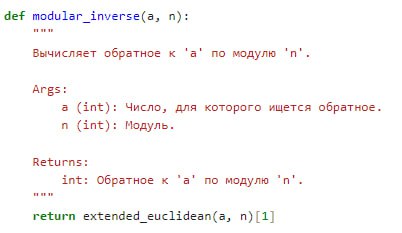
### 1. Функция modular\_inverse(a, n)

Эта функция вычисляет обратное к a по модулю n.

#### Описание:

* **Вход:**
  + a (int): Число, для которого ищется обратное.
  + n (int): Модуль.
* **Выход:**
  + Обратное к a по модулю n.

#### Как работает:

1. Используется функция extended\_euclidean(a, n) для получения коэффициентов Безу.
2. Возвращается коэффициент x (второй элемент в кортеже), который является обратным к a по модулю n. 

### 2. Функция pollard\_step(x, a, b, params)

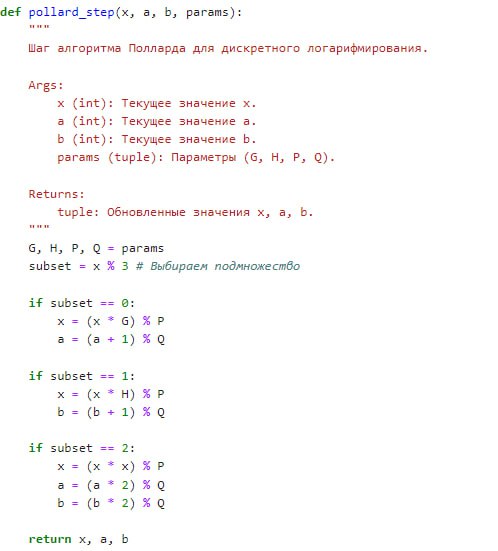
Эта функция реализует один шаг алгоритма Полларда для дискретного логарифмирования.

#### Описание:

* **Вход:**
  + x (int): Текущее значение x.
  + a (int): Текущее значение a.
  + b (int): Текущее значение b.
  + params (tuple): Параметры (G, H, P, Q).
* **Выход:**
  + Кортеж обновленных значений (x, a, b).

#### Как работает:

1. **Разделение на подмножества:** Использует x % 3 для определения подмножества.
2. **Обновление значений в зависимости от подмножества:**
   * Если x % 3 == 0: x умножается на G по модулю P, a увеличивается на 1 по модулю Q.
   * Если x % 3 == 1: x умножается на H по модулю P, b увеличивается на 1 по модулю Q.
   * Если x % 3 == 2: x возводится в квадрат по модулю P, a и b умножаются на 2 по модулю Q.



pollard step

### 3. Функция pollard\_rho\_discrete\_log(generator, value, prime)

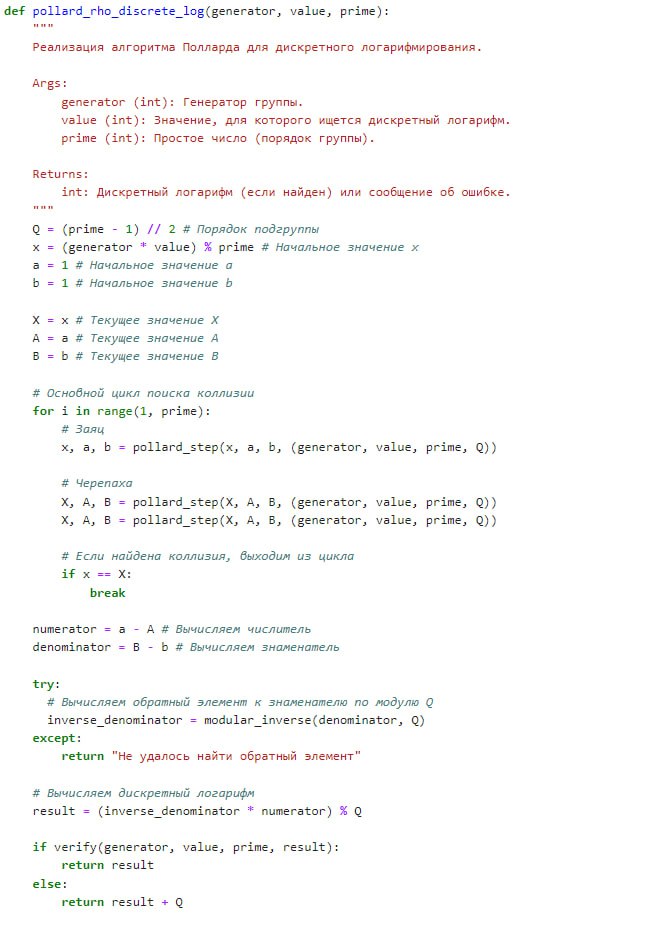
Эта функция реализует алгоритм Полларда для дискретного логарифмирования.

#### Описание:

* **Вход:**
  + generator (int): Генератор группы.
  + value (int): Значение, для которого ищется дискретный логарифм.
  + prime (int): Простое число (порядок группы).
* **Выход:**
  + Дискретный логарифм (если найден) или сообщение об ошибке.

#### Как работает:

1. **Инициализация:** Устанавливаются начальные значения Q, x, a, b, X, A, B.
2. **Основной цикл:**
   * Используются “заяц” и “черепаха” для поиска коллизии, где заяц делает два шага за итерацию, а черепаха - один.
   * Функция pollard\_step применяется для каждого шага.
   * Цикл выполняется до тех пор, пока не будет найдена коллизия (x == X).
3. **Вычисление дискретного логарифма:**
   * Вычисляется числитель a - A и знаменатель B - b.
   * Вычисляется обратный элемент знаменателя по модулю Q с помощью функции modular\_inverse.
   * Вычисляется дискретный логарифм: (inverse\_denominator \* numerator) % Q.
4. **Верификация:** Вызывается функция verify для проверки правильности найденного логарифма.
   * Если логарифм верный, то он возвращается.
   * Если логарифм не верный (при pow(generator, result, prime) != value), то result увеличивается на Q и возвращается.



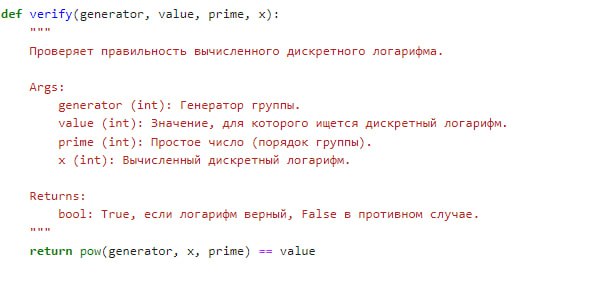
pollard descrete log

### 4. Функция verify(generator, value, prime, x)

Эта функция проверяет правильность вычисленного дискретного логарифма.

#### Описание:

* **Вход:**
  + generator (int): Генератор группы.
  + value (int): Значение, для которого ищется дискретный логарифм.
  + prime (int): Простое число (порядок группы).
  + x (int): Вычисленный дискретный логарифм.
* **Выход:**
  + True, если логарифм верный, False в противном случае.



verify

#### Как работает:

1. Вычисляет generator^x mod prime.
2. Сравнивает с value. Возвращает True, если равны, иначе False.