

МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«МИРЭА – Российский технологический университет»**

**Институт Искусственного интеллекта**

**Кафедра Проблем управления**

**курсовАЯ работА**по дисциплине

**«Объектно-ориентированное программирование»**

**Тема курсовой работы:**

«Разработка алгоритма для генерации совместимых бинарных последовательностей»

Студент группы КРБО-01-22: **Гизетдинов Марк Рустемович**

Руководитель курсовой работы: **к.т.н., доцент Гарцеев И.Б.**

**Работа представлена к защите:** «\_\_\_» \_\_\_\_\_\_\_\_ 2024 г

*(подпись)*

**«Допущен к защите»:** «\_\_\_» \_\_\_\_\_\_\_\_ 2024 г.

*(подпись)*

**Москва 2024**



МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«МИРЭА – Российский технологический университет»**

**Институт Искусственного интеллекта**

**Кафедра Проблем управления**

**Утверждаю**

Заведующий кафедрой

Романов М.П.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г

**ЗАДАНИЕ**

на выполнение курсовой работы по дисциплине

**«Объектно-ориентированное программирование»**

Студент группы КРБО-01-22: **Гизетдинов Марк**

1. **Исходные данные:** входные данные и их форматы описаны в условии задачи
2. **Тема:** «Разработка алгоритма для генерации совместимых бинарных последовательностей»
3. **Перечень вопросов, подлежащих разработке, и обязательного графического материала:**

- техническое задание на работу

- алгоритм работы программного обеспечения

- исходные коды программного обеспечения

- набор испытательных тестов

- отчет о выполнении курсовой работы

- презентация для защиты курсовой работы

1. **Срок представления к защите курсовой работы:** «\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г

**Задание на курсовую работу выдал:** «\_\_\_\_» \_\_\_\_\_\_\_\_ 2024 г

*(подпись)*

**Задание на курсовую работу получил:** «\_\_\_\_» \_\_\_\_\_\_\_\_ 2024 г

*(подпись)*

# Аннотация

В рамках данной курсовой работы разработано алгоритмическое и программное обеспечение, предназначенное для решения задачи по генерации пары совместимых бинарных последовательностей.

Под парой совместимых бинарных последовательностей подразумевается такая, что содержит последовательности, для которых операция исключающего ИЛИ равна операции сложения.

Алгоритмическое и программное обеспечение разрабатывается в рамках курсовой работы по дисциплине «Объектно-ориентированное программирование».

При выполнении программа использует не более 15 Мб оперативной памяти и работает не более 0.15 секунд.

Программа является консольным приложением. Язык разработки – Python 3.9. Для компиляции может быть использован любой интерпретатор Python, поддерживающий данную версию языка и выше. Успешно проходит проверку системы Timus Online Judgement.

# Общая постановка задачи №1530 «Ones and Zeroes»

**Ограничение времени**: 1 секунда

**Ограничение памяти:** 64 Мб

Consider two binary sequences *A* and *B* of length *n* each. Let us call these sequences *compatible* if *A* XOR *B* = *A + B* where XOR is the element-wise exclusive OR operation.

Given an integer *n* and a pair of binary sequences *p* and *q*, find the pair of compatible binary sequences *a* and *b* which comes first after the pair (*p*, *q*). Pair (*a*, *b*) is said to be lexicographically less that (*c*, *d*) if *a* is lexicographically less than *c*, or *a* and *c* are equal and *b* is lexicographically less than *d*. If there is no pair of compatible binary sequences lexicographically greater than (*p*, *q*), output the lexicographically first compatible pair.

### Исходные данные

There is a single integer *n* (1 ≤ *n* ≤ 100000) on the first line of the input. The sequence *p* is given on the second line, and the sequence *q* is on the third one. There are no spaces or other delimiters inside the sequences; however, there could be trailing whitespace on these two lines.

### Результат

Output the sequences *a* and *b* on the first two lines of the output, correspondingly. Use the same format as the input.

Пример:

|  |  |
| --- | --- |
| Исходные данные | Результат |
| 1  0  0 | 0  1 |
| 2  01  10 | 10  00 |

# Общие сведения

Программа «Ones and Zeroes» является решением задачи №1530. Целью её разработки является автоматическая генерация пары «совместимых» двоичных чисел, что будут лексикографически больше изначально заданной пары двоичных чисел.

Числа будут являться совместимыми в случае, если операция исключающего ИЛИ (XOR) между ними равна операции сложения *a*+*b*.

Примеры совместимых двоичных чисел:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 |

Программа выполняет следующие действия:

* получает со стандартного потока ввода пару любых двоичных чисел и их длину;
* анализирует числа на наличие лексикографически большей совместимой пары;
* если найдена подходящая совместимая пара, выводит на экран данную пару, в ином случае – наименьшую совместимую пару.

Время работы программы на любых корректных входных данных не превышает 0.15 секунд.

Входные данные предполагаются корректными, поэтому никаких проверочных действий не производится. На любом корректном наборе входных данных программа заканчивает работу безаварийно.

Программа написана на языке программирования Python 3.9, гарантированно выполняется с помощью любого Python интерпретатора, поддерживающего данную версию языка.

*Примечание: здесь, как и в продолжении текста, под словом «число» подразумевается и соответствующая ему бинарная последовательность.*

# Ключевые идеи и алгоритм решения

Данная задача решена с помощью метода инкрементации и поэтапной проверки чисел.

Метод состоит в следующем:

1. Поскольку в описании программы указано, что для лексикографически большей пары будет достаточно увеличить лишь число **b**, в первую очередь инкрементируем его. Для инкрементации чисел в программу введена функция **up()**, что, помимо увеличения числа, возвращает *True* или *False* в зависимости от того, переполнено ли число при инкрементации, или нет.
2. Если инкрементированное число **b** не переполнено, проводится проверка на его совместимость с числом **a**. Циклом проходимся справа налево по обоим числам, конвертированным в *List*, в поисках такого разряда дв. числа, в котором у двух чисел будут единицы. Далее есть два варианта:

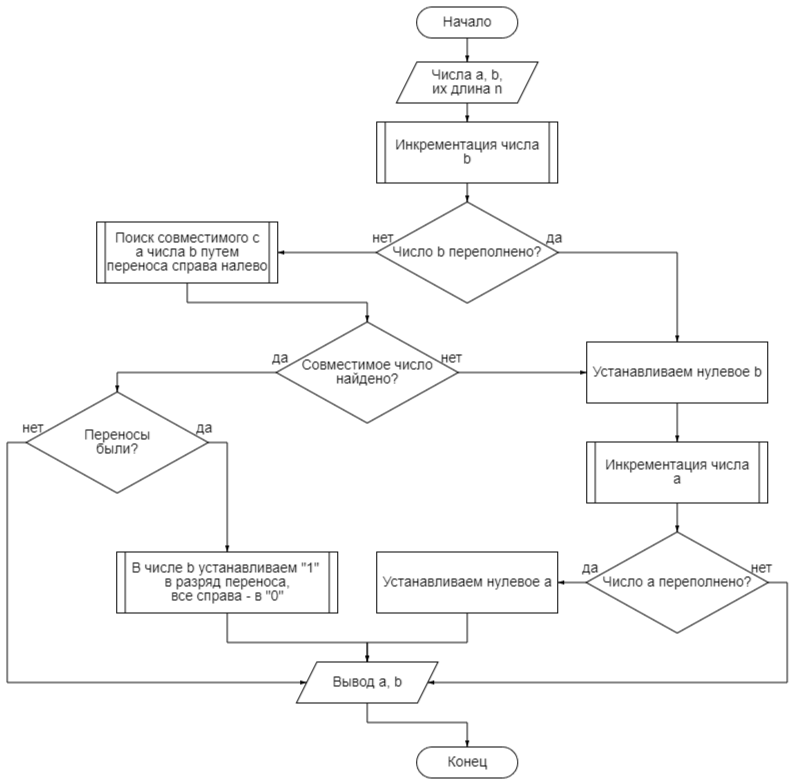
А) В случае, если подобный разряд найден, следующий разряд числа **b** инкрементируется, сравнивается со следующим разрядом числа **a** и так до тех пор, пока не будет найден разряд, в котором у числа **а** – 0, а у увеличенного числа **b** – 1. Для этого введен флаг переноса **p**. Сумма значений разряда + значение **p** (0 или 1) не должна превышать 1.

[**Индекс разряда** + 1] записывается в переменную **p\_index**. В данном случае **индекс разряда** - это номер элемента списка, в который сконвертировано дв. число. Приравниваем **b[p\_index - 1]** к единице, все разряды справа – зануляем, т.к. нам нужно наименьшее число **b**, что будет больше прошлого числа **b**, но совместимым с неизменным числом **a**. Возвращаем числа, получаем ответ.

В случае, если **индекс разряда** равен нулю, это значит, что либо в наибольшем разряде чисел **a** и **b** у обоих стоят единицы, либо, что подходящий разряд с «1» и «0» с учетом переноса при увеличении найден не был. Тогда переходим к п. 3.

Б) Если такой разряд не найден, то числа уже совместимы, возвращаем и получаем ответ.

1. При переполнении числа **b**, либо при невозможности в прошлом пункте найти совместимое число, инкрементируем число **а** и возвращаем нулевое число **b**. Это так же лексикографически большая пара. Возвращаем числа, получаем ответ.
2. Если даже число **a** переполнено, возвращаем лексикографически меньшую совместимую пару, то есть дв. числа из нулей.

 *Рис. 1. Общая схема алгоритма*

# Доказательство правильности алгоритма, его оптимизация

*Примечание: введем определение – «несовместимый разряд». Это разряд двоичных чисел, значение которого и у А, и у В будет равно единице.*

Программа выдает нужную пару чисел, не меньше, не больше. Это происходит потому, что:

1. Поиск совместимого числа B – проход по числу идет справа налево. В случае, если найдены несовместимые разряды, такой подход позволяет легко найти наименьшее совместимое число, большее, чем прошлое B.
2. Учтены неочевидные и краевые случаи ввода. К примеру, при вводе пары

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

Ответ равен:

При таком вводе программе нужно найти число больше (или, в случае отсутствия переносов, равно), чем 1001, которое будет совместимо с числом 0111. Такого числа нет, поэтому программа увеличит на 1 число А, а число В установит нулевым. Если число В будет другим, новая пара чисел будет больше прошлой пары, но не в лексикографическом порядке среди совместимых чисел.

Или же, если ввести

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

Ответ равен:

При таком вводе функция up() дважды вернет True, из-за чего программа выдаст лексикографически наименьшую совместимую пару чисел.

**Далее приведены особенности программы, что делают её более оптимизированной и эффективной:**

1. В алгоритме присутствуют достаточно долгие циклы прохода по спискам, в которые до этого конвертируются вводимые бинарные последовательности. Чтобы таких проходов было меньше, используются они с максимальной пользой. К примеру, функция up() несет в себе сразу две функции – помимо инкрементации поданного ей числа, она возвращает булево значение в зависимости от того, переполнено ли число при увеличении, или нет.
2. Для нахождения совместимого с числом А числа В не используется функция, которая возвращает True или False в зависимости от успешности поиска. Программа работает так, что после цикла for, в котором происходит проход по обоим числам, дальнейшая её работа определяется двумя условиями if, которые зависят от найденного совместимого числа, и, по выполнении которых, программа прекращает свою работу и выводит ответ. Если не выполнено ни одно из условий, это значит, что совместимое число не найдено, программа продолжает свое выполнение. Такое решение позволяет не использовать дерево условий из if, уменьшить количество проверяемых условий.
3. Вместо конвертации list в string используется метод join, что также позволяет немного сократить время выполнения программы.

# Структура программного обеспечения

Ниже представлена структурная схема программного обеспечения (рис. 10).



*Рис. 2. Структурная схема программного обеспечения*

В теле **main()** происходит ввод и вывод данных, активация функции **solve()**.

В функции **solve(n, a, b)** выполняется основной функционал программы: поиск совместимых чисел, вызов функции **up()**, возврат полученной пары последовательностей. Входные параметры:

**n** – длина двоичных последовательностей.

**a** – первая последовательность.

**b** – вторая последовательность.

Функция **up(a)** предназначена для инкрементации числа, а также определения переполнения при его увеличении. Входные параметры:

**a** – число, подлежащее инкрементации.

# Структуры данных

В программе используются следующие основные элементы данных:

**n** – переменная, содержащая длину последовательностей.

**a** – переменная, содержащая первую последовательность.

**b** – переменная, содержащая вторую последовательность.

**z** – переменная, содержащая строку из нулей длиной n, нулевая последовательность.

**a\_list** – список из бинарной последовательности. Переменная a, конвертированная в list.

**b\_list** – список из бинарной последовательности. Переменная b, конвертированная в list.

**p** – переменная-флаг переноса.

**p\_index** – переменная, содержащая индекс разряда, в который происходит перенос.

**result** – список, содержащий итоговую пару чисел.

# Порядок тестирования и приемки

В первую очередь программа была проверена на парах последовательностей, для которых не составит труда самому посчитать ответ на поставленную задачу.

Ввод: Ответ программы:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |

Здесь понятно, что необходимо увеличивать число В до тех пор, пока оно не станет совместимым с числом А. Полученная последовательность – 0100.

Ввод: Ответ программы:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 |

При таком вводе получить нужную пару чисел путем увеличения числа В не получится – в последнем разряде обоих чисел находится «1», поэтому нужно инкрементировать число А, а последовательность В должна состоять из нулей. Таким образом число А – 1100, В – 0000.

Ввод: Ответ программы:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |

Увеличиваем число В на 1, получаем ответ.

Ввод: Ответ программы:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

В данном случае увеличивать числа уже некуда – произойдет переполнение. Ответ по условию задачи – лексикографически наименьшая пара совместимых чисел, то есть последовательности из нулей.

# Библиография

1. <https://acm.timus.ru/forum/?space=1&num=1530> – обсуждение на форуме тимус, поиск тестовых данных.
2. <https://stackoverflow.com/> - поиск возможных способов оптимизации кода, помощь в исправлении ошибок.

# Код программы

def up(a):  
 *'''  
 Увеличиваем вводимое двоичное число на 1, проверяем на переполнение  
  
 :param a: двоичное число в формате list  
 :return: true или false, в зависимости от наличия переполнения  
 '''* p = 1  
 for i in range(len(a) - 1, -1, -1):  
 if a[i] == '0':  
 a[i] = '1'  
 p = 0  
 break  
 else:  
 a[i] = '0'  
 p = 1  
 return bool(p)  
  
def solve(n, a, b):  
 *'''  
 Решение задачи  
  
 :param n: длина двоичных чисел  
 :param a: первое число  
 :param b: второе число  
 :return: два двоичных числа, подходящие под условие задачи  
 '''* z = '0' \* n *# Создаем нулевое дв. число необходимой длины* b\_list = list(b)  
 if not up(b\_list): *# Если нет переполнения. Второе число увеличено на 1* p = 0 *# Перенос* p\_index = -1 *# Индекс переноса, если переносов не было - равен -1* for i in range(n - 1, -1, -1): *# Ищем справа налево разряд дв. числа, в котором в первом числе 0, во втором - 1, с учетом переноса* if int(a[i]) + int(b\_list[i]) + p > 1:  
 p = 1  
 p\_index = i  
 else:  
 p = 0  
 if p\_index == -1: *# Если переносов не было - числа уже совместимы, возвращаем* return a, ''.join(b\_list)  
 if p\_index > 0: *# В ином случае получаем наименьшее второе число, совместимое с первым, на месте нужного разряда - единица, справа - нули* b\_list[p\_index - 1] = '1'  
 for i in range(p\_index, n):  
 b\_list[i] = '0'  
 return a, ''.join(b\_list)  
 *# Если второе переполнено, либо не получилось найти совместимое с числом a число b* a\_list = list(a)  
 if up(a\_list):  
 return z, z *# Если а переполнено, возвращаем 0, 0* return ''.join(a\_list), z *# Если а не переполнено, возвращаем его с b = 0  
  
# Чтение входных данных*n = int(input())  
a = input().strip()  
b = input().strip()  
  
*# Решение*result = solve(n, a, b)  
  
*# Вывод результата*print(result[0])  
print(result[1])

# Рецензия преподавателя