Задание №15, Алгебра логики

Теория

> *Базовую теорию по алгебре логики см. в методичке по 2 заданию*

Основные преобразования и аксиомы алгебры логики:

Аксиомы:

1. $\bar{\bar{x}} = x,$
2. $x \vee \bar{x} = 1$
3. $x \vee 1 = 1$
4. $x \vee x = x$
5. $x \vee 0 = x$
6. $x \wedge \bar{x} = 0$
7. $x \wedge x = x$
8. $x \wedge 0 = 0$
9. $x \wedge 1 = x$

Свойства:

$a \vee (b \vee c) = (a \vee b) \vee c$	$a \wedge (b \wedge c) = (a \wedge b) \wedge c$	ассоциативность
$a \vee b = b \vee a$	$a \wedge b = b \wedge a$	коммутативность
$a \vee (a \wedge b) = a$	$a \wedge (a \vee b) = a$	законы поглощения
$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$	$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$	дистрибутивность

Законы де Моргана:

$$\overline{x \wedge y} = \bar{x} \vee \bar{y}$$

$$\overline{x \vee y} = \bar{x} \wedge \bar{y}$$

Преобразование импликации:

$$A \to B = \neg A \vee B$$

💡 В задании необходимо найти такие значения некоторого параметра (чаще всего обозначается **A**), при которых заданная логическая функция будет возвращать только истину или только ложь при любых значениях других переменных.

Есть 2 основных типа, которые имеют принципиальную разницу в решении:

- Параметр **A**, который требуется найти, является **числом**. Для ответа требуется лишь указать наибольшее или наименьшее значение. В 90% случаев достаточно решения кодом и увеличением **range** перебора **A** и переменных для подстраховки.
- Параметр **A**, который требуется найти, является **отрезком** - непрерывным конечным набором значений. Часто в условии имеется несколько других отрезков и выражения принадлежности переменных к ним (x ∈ B**).** Такие задачи как правило легче решаются руками, чем первый тип, а решение только кодом может легко запутать и привести к неправильному ответу, поэтому важно уметь себя перепроверять обоими способами.

План выполнения 15 задания:

Кодом:

1. Если в задаче присутствуют функции (ДЕЛ, ТРЕУГ и тд.), то пишем их через def:

```
# Делится ли a на b без остатка
def Del(a,b):
    return a % b == 0
# Может ли существовать треугольник со сторонами a, b, c
def Triangle(a,b,c):
    return a < b + c and b < a + c and c < a + b
```

2. Если в задаче присутствуют отрезки, то обозначаем их:

```
X = list(range(a, b + 1)) # +1 т.к. range не включает конец
```

3. Пишем внешний цикл для параметра A, в 90% случаев достаточно 1000 чисел, но в зависимости от условия могут требоваться как значения больше, так и значения отрицательные:

```
for A in range(1000):
```

4. Через генератор списка составляем список значений заданной в условии функции при заданном переборе x и (если есть) y, учитывая то, какие значения они могут принимать по условию (часто x и y не просто так указаны в условии как натуральные). Через **all** проверяем, что все значения истинные (или **not any**, что ложные), выводим A, если это так:

```
  if all(not(Triangle(x,y+3,A) for x in range(1,1000) for y in range(1,1000)):
      print(A)
```

5. Получаем значения A, при которых функция тождественно (при любых переменных) равна 1. Убедитесь в правильности ответа (особенно, если требуется максимум и полученное значение близко к границе первого перебора), изменив range для A или для переменных (но тогда считаться будет дольше). Перепроверьте решение руками, если остаётся время.

💡 Примечание: задания на отрезки важно уметь решать руками. Программирование может показать, при каких значениях переменных логическое выражение принимает истину или ложь вне зависимости от параметра A, но очень важно учитывать включительность концов полученного набора, ведь если ответом будет полуинтервал (5; 10] длины 5 (=10-5), то код начёт выводить значения только с 6. Можете как минимум это перепроверять тем, является ли 6 началом или концом хотя бы одного из отрезков. Если нет - явный знак, что на самом деле ответом является интервал или полуинтервал.

Руками:

1. Преобразуем выражения до наиболее простого вида, используя свойства выше. Должно получиться достаточно простое выражение из дизъюнкций, конъюнкций, отрицаний и, если они были изначально, эквивалентностей. Как правило достаточно этих действий:
    a. привести импликации к дизъюнкциям
    b. внести отрицание в выражения по закону де Моргана
    c. избавиться от лишних скобок и повторяющихся выражений
2. Если это задание на отрезки, то рисуем прямую с заданными на ней отрезками, на ней помечаем области истинности для тех или иных выражений. Обязательно помечаем крайние точки отрезка невключительными, если это отрицание принадлежности. Области, которые не удовлетворяют требованию условия (функция должна быть всегда истинна или ложна) - должно перекрывать выражение с **A**.
3. Если это задание на уравнения (y > 2 * x и проч.), то, как правило, они легко решаются графически или алгебраически.
4. Если это задание на побитовую конъюнкцию, то нужно перевести все заданные числа в двоичный вид (можно через функцию bin(n) в Python) и для каждого разряда применять конъюнкцию, чтобы понять, какой разряд

должен быть 1 или 0 у переменной x, чтобы у логического выражения получалась истинна или ложь.

5. Если это задание на делимость чисел, то, как правило, нужно сначала найти те значения переменных, которые не удовлетворяют условию тождественности функции, а потом перебирать делители заданных в условии чисел для нахождения общих множителей/уникальных делителей/других закономерностей, которые можно увидеть у неудовлетворяющих значений x.

Алгоритм решения

Примечание: при решении кодом показано 3 разных способа в следующем порядке: с for-else, с флагом (идентичен предыдущему) и с генерацией списка (медленнее предыдущих, но короче в написании). Для решения каждой задачи можно использовать любой из них.

Функция

Обозначим через ДЕЛ(n, m) утверждение «натуральное число n делится без остатка на натуральное число m». Для какого наименьшего натурального числа A формула

ДЕЛ(x, A) → (ДЕЛ(x, 21) ∨ ДЕЛ(x, 35))

тождественно истинна (то есть принимает значение 1 при любом натуральном значении переменной x)?

1. Преобразуем формулу, заменив импликацию на дизъюнкцию:

$$\overline{\text{ДЕЛ}(x, A)} \vee (\text{ДЕЛ}(x, 21) \vee \text{ДЕЛ}(x, 35))$$

2. Не обращая пока внимания на выражение с А, находим *натуральные* значения x, при которых оставшаяся часть формулы (ровно как и вся формула, т.к. между ними дизъюнкция) равна 1:

$$\overline{\text{ДЕЛ}(x, A)} \vee \underbrace{(\text{ДЕЛ}(x, 21) \vee \text{ДЕЛ}(x, 35))}_{= 1 \text{ при } x \vdots 21 \text{ или } x \vdots 35} \quad 21, 35, 42, 63, 70 \ldots$$

[21*n, n - натуральное, 35*k, k - натуральное]

3. Т.к. у нас выражение, что *x не должен делиться на A*, то очевидно, что нужно подобрать такое значение, что *not ДЕЛ(x,A)* = 0 перекрывалось бы *ДЕЛ(x,21) or ДЕЛ(x,35)* = 1.
4. Из этого следует, что A принадлежит множеству чисел, кратных 21 или 35, т.е. имеет вид 21*n или 35*n, где n - натуральное число Наименьшее такое число: **21**
5. Перепроверим себя кодом:

```
for A in range(1,1000):
    for x in range(1,1000):
        F = Del(x, A) <= (Del(x,21) or Del(x,35))
        if not F: # Если F == False хотя бы раз, то дальше нет смысла перебир
                        # т.к. параметр не подошёл
            break # Выходим из цикла x
    else: # Если в цикле x не был вызван break, т.е. F всегда было True
        print(A)
        break # Нужно только наименьшее = первое подходящее. Выходим и
 # Output: 21
```

Уравнение

Укажите наименьшее целое значение А, при котором выражение
(y + 3x < A) ∨ (2y +x > 50) ∨ (4y – x < 40)
истинно для любых целых положительных значений x и y.

1. Решение алгебраически:

y + 3*x < A
или 2*y + x > 50
или 4*y - x < 40

⇨

A > y + 3*x
x > 50 - 2*y
4*y - 50 + 2*y < 40

⇨

A > y + 3*x
x > 50 - 2*15
y < 15

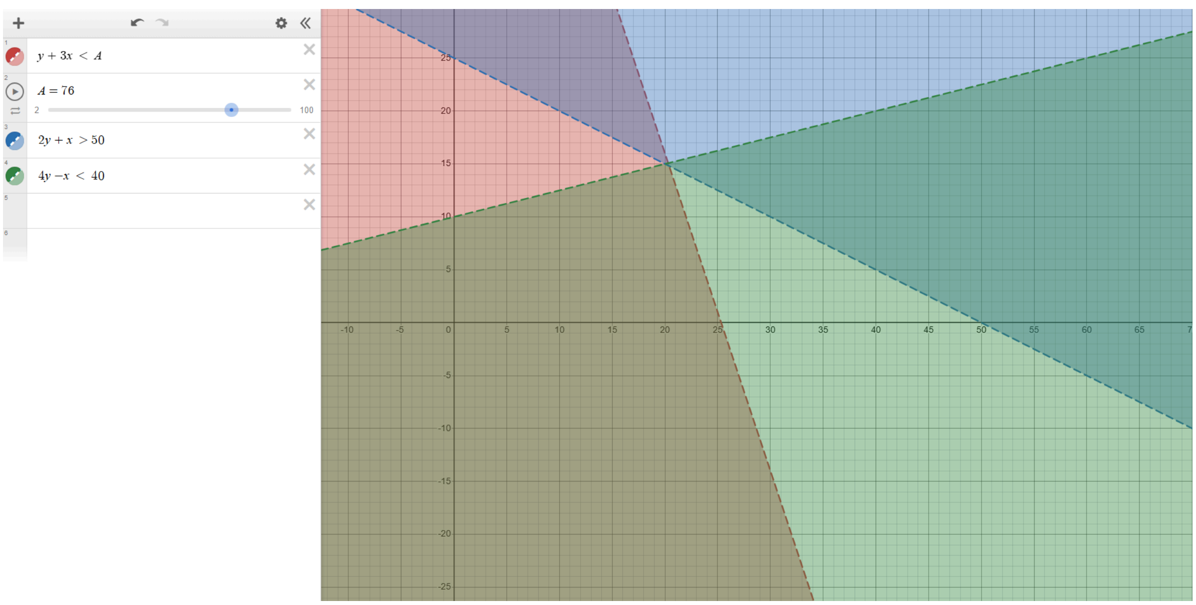
⇨

A > 15 + 3*20
x > 20
y < 15

⇨

A > 75 => A = 76

2. Решение графически (белая область ⇒ False, нужно перекрыть красной):



3. Решение кодом:

```
for A in range(1, 1000):
    flag = True # Флаг того, что F всегда была True при текущем A
    for x in range(1, 1000):
        for y in range(1, 1000):
            if not ((y + 3*x < A) or (2*y + x > 50) or (4*y - x < 40)):
                flag = False # Формула = False ⇒ меняем A на следующее
                break # Выход из цикла y
        if not flag:
```

```
        break # Выход из цикла x
    if flag: # Если все прошло хорошо и флаг не поменялся
        print(A)
        break # Выход из цикла A, т.к. нужно только минимальное
```

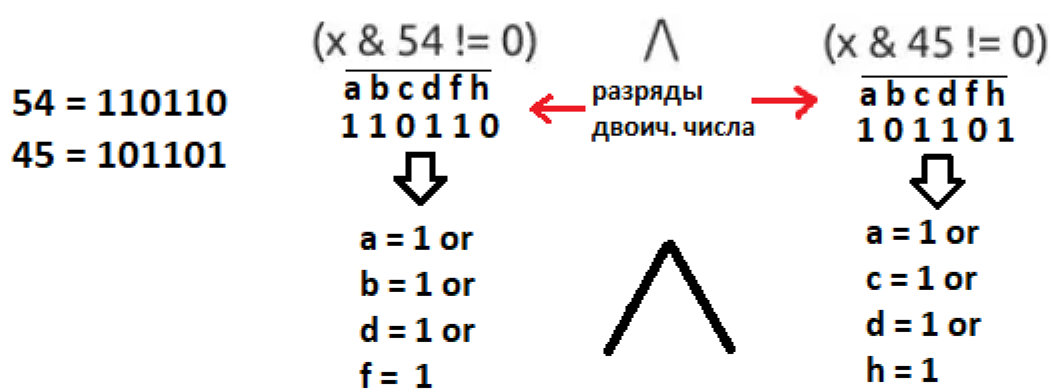
Побитовая конъюнкция

Определите, сколько всего существует натуральных чисел R таких, что выражение
(x & 54 != 0) ⋀ (x & 45 != 0) ⋁ (x & A = 0) ⋁ (x & R = 0)
тождественно истинно при любом натуральном A (то есть принимает значение 1 при любом натуральном значении переменной x)?

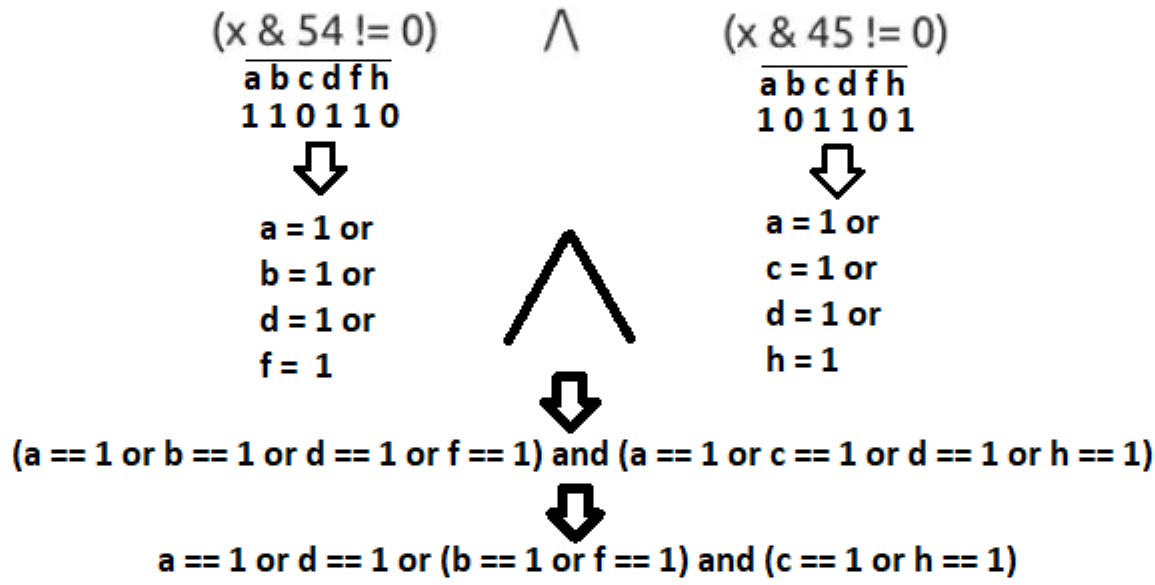
1. Находим представление 54 и 45 в бинарном виде. Можно через Python и функцию *bin(n)*

    ```
    print(bin(54), bin(45))
    #    0b110110 0b101101
    ```

2. Преобразуем выражения побитовой конъюнкции в выражения для разрядов двоичного представления чисел:



3. Из преобразованных выражений для разрядов выводим выражение, при котором общая формула будет равняться истине вне зависимости от A и R:



4. Применив отрицание к полученной формуле и раскрыв её через закон де Моргана, находим ситуации, при которых формула будет ложной:

a == 0 and d == 0 and (b == 0 and f == 0 or c == 0 and h == 0)

5. Исходя из формулы выше, получаем вид чисел при которых общая формула равняется False (бордовый - 0 должен быть всегда; красный - 0 должен быть в паре с другим 0: **c** с **h** или **b** с **f;** черный - может быть и 1 и 0) :

```
   a b c d f h
...0 ? 0 0 ? 0
...0 0 ? 0 0 ?
```

6. На самом деле выражение x & A == 0 в исходной формуле никак не влияет на ответ, т.к. A может быть абсолютно любым числом. Тогда x & R == 0 должно быть положительным для чисел, которые мы получили на прошлом шаге. Для этого R должно удовлетворять следующим условиям (выводится логически): a == 1 or d == 1. Это будут числа **100100 = 36, 100000 = 32** и **000100 = 4.** Получается таких чисел существует всего **3.**

7. Перепроверяем себя кодом:

```
cnt = 0
for R in range(1, 1000):
   if all( (x & 54 != 0) and (x & 45 != 0) or (x & A == 0) or (x & R == 0) \
    for x in range(1, 1000) for A in range(1, 1000)): # \ - перенос строки
      cnt += 1
      print(R)
print(f'Количество: {cnt}')
# output:
# 4
# 32
# 36
# Количество: 3
```

Отрезок

Заданы отрезки P = [13, 37] и Q = [22, 51]. Обозначим через ДЕЛ(n, m) утверждение «натуральное число n делится без остатка на натуральное число m». Найдите наименьшую возможную длину отрезка A, при котором формула

$\neg((x \in Q) \rightarrow (ДЕЛ(x, 18) \vee (x \in P))) \wedge (x \notin A)$

тождественно ложна, то есть принимает значение 0 при любом значении переменной x?

1. Преобразуем выражение:
    a. Заменим импликацию на дизъюнкцию:

$$\neg(\overline{(x \in Q)} \vee (ДЕЛ(x, 18) \vee (x \in P))) \wedge (x \notin A)$$

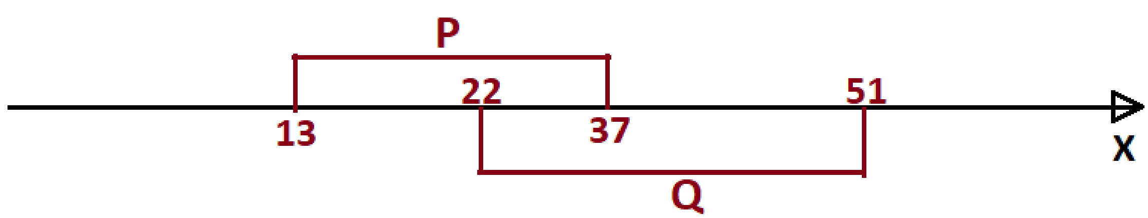
    b. Уберём бесполезные скобки, выделим скобки, к которым относится отрицание:

$$\neg(\overline{(x \in Q)} \vee ДЕЛ(x, 18) \vee (x \in P)) \wedge (x \notin A)$$

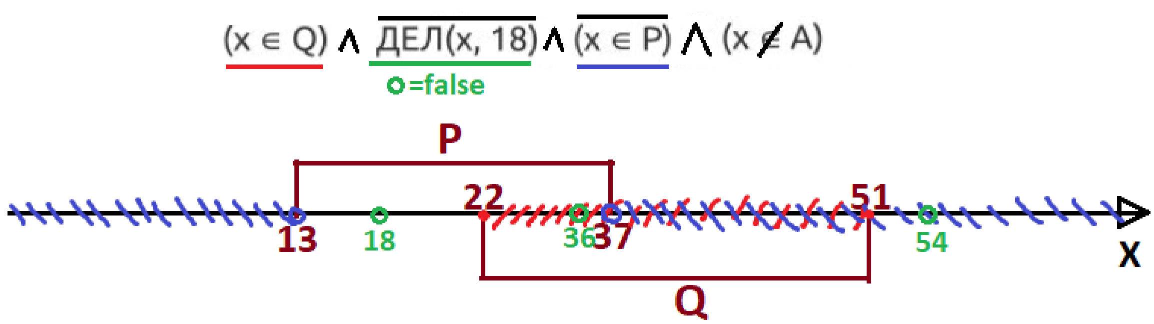
    c. Преобразуем отрицание выражения в скобках по закону де Моргана, уберём лишние скобки:

$$(x \in Q) \wedge \overline{ДЕЛ(x, 18)} \wedge \overline{(x \in P)} \wedge (x \notin A)$$

2. Построим прямую и отметим на ней данные по условию отрезки P и Q:



3. Обозначим цветом отдельные логические выражения и отметим области истинности на прямой (в случае *отриц. ДЕЛ(x,18)* - она всегда положительная кроме выколотых точек 18, 36, 54 и тд.) (важно заметить, что красная область - включительная, а синяя - нет, т.к. концы отрезка входят в отрезок):



4. Т.к. между логическими выражениями стоит конъюнкция (т.е. =1 только когда все выражения =1), то вся функция истинна только на полуинтервале (37, 51]. Т.к. нам нужно, чтобы функция =0 при любых x, то A должно равняться (37;51] (при пустом A выражение *x не принадлежит A = 1* для всех x, а в этом случае она будет равна 0 как раз на найденном полуинтервале). Получаем длину 51 - 37 = **14**.

5. Перепроверим ответ кодом. Сначала убедимся, что мы правильно нашли промежуток на котором формула из условия равна 1:

```
P = list(range(13,37+1)) # +1, т.к. конец в range не включительный
Q = list(range(22,51+1))
A = list()
x_true = list()
for x in range(1000):
    F = ( not ((x in Q) <= ((x % 18 == 0) or (x in P))) and (x not in A) )
    if F:
        x_true.append(x)
print(x_true)
```

```
[38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51]
```

6. Получаем, что формула истинна на отрезке от 38 до 51. Так как ни P, ни Q не имеют на границах 38, то даже без решения руками можно догадаться, что пропущено значение 37 как не включительное.
7. Подставляем полученный отрезок в *A = list(range(37,51+1))*. Запускаем ещё раз код и убеждаемся в том, что x_true теперь пустой и задача решена правильно.