1. Задача 2.G

Строка называется бинарной, если она состоит только из символов «0» и «1». Строка v называется *подстрокой* строки w, если она имеет ненулевую длину, и ее можно прочитать, начиная с некоторой позиции, в строке w. Например, у строки «010» есть шесть подстрок: «0», «1», «0», «01», «10», «010». Две подстроки считаются различными, если их позиции вхождения различны. Другими словами, каждую подстроку нужно учитывать столько раз, сколько она встречается.
Дана бинарная строка s. Ваша задача — найти количество ее подстрок, содержащих ровно k единиц.

№	Когда	Кто	Задача	Язык	Вердикт	Время	Память
45919024	2018-11-18 18:27:00	tamarinvs19	G - Еще одна строковая задача	PyPy 3	Полное решение	436 мс	15100 КБ
45918969	2018-11-18 18:24:57	tamarinvs19	G - Еще одна строковая задача	Python 3	Превышено ограничение времени на тесте 39	2000 мс	10600 КБ

```
k = int(input())
ss = input()
m = len(ss)

zeros = []
past = '1'
for s in ss:
    if past == '1' and s == '1':
        zeros.append(0)
    elif past == '1' and s == '0':
        zeros.append(1)
    elif past == '0' and s == '0':
        zeros[-1] += 1
    past = s
if ss[-1] == '1':
    zeros.append(0)
res = 0
if k != 0:
    i = 0
    while i + k < len(zeros):
        res += (zeros[i]+1)*(zeros[i+k]+1)
        i += 1
elif k == 0:
    i = 0
    while i + k < len(zeros):
        res += int(0.5*zeros[i]*(zeros[i+k]+1))
        i += 1
print(res)
```

2. Задача 6.H

Известно, что простыми называются целые положительные числа, у которых ровно два различных положительных делителя. По аналогии назовем целое положительное число t T-простым, если у t ровно три различных положительных делителя.
Вам дан массив, состоящий из n целых положительных чисел. Для каждого из них определите, является ли оно T-простым или нет.

№	Когда	Кто	Задача	Язык	Вердикт	Время	Память
45975633	05:23:37	tamarinvs19	H - T-простые числа	Python 3	Превышено ограничение времени на тесте 33	2000 мс	7600 КБ
45975599	05:22:32	tamarinvs19	H - T-простые числа	PyPy 3	Полное решение	1590 мс	19600 КБ

```
def is_prime(n):
    if int(n**0.5)**2 == n:
        n = int(n**0.5)
    else:
        return False
    a = 2
    res = True
    if n == 1:
        res = False
    else:
        while a**2 <= n:
```

```
                if n % a == 0:
                    res = False
                    break
                a += 1
        return res

    n = int(input())
    ans = {True: 'YES', False:'NO'}
    xs = [ans[is_prime(x)] for x in map(int, input().split())]

    print('\n'.join(map(str, xs)))
```

3. Задача 6.E !!

Задан набор из всех целых чисел от l до r включительно, $l<r$, $(r-l+1) \le 3 \cdot 10^5$ и $(r-l)$ всегда нечетно.

Вы хотите разделить эти числа на ровно $\frac{r-l+1}{2}$ пар таким образом, чтобы в каждой паре (i,j) наибольший общий делитель i и j равен 1 . Каждое число должно встретиться ровно в одной паре.
Выведите полученные пары или сообщите, что решения не существует. Если существует несколько корректных решений, то выведите любое из них.

№	Когда	Кто	Задача	Язык	Вердикт	Время	Память
45976275	05:48:46	tamarinvs19	E - Взаимно простые пары	Python 3	Полное решение	280 мс	28000 КБ
45976237	05:47:11	tamarinvs19	E - Взаимно простые пары	PyPy 3	Полное решение	545 мс	49400 КБ

```
l, r = map(int, input().split())
print('YES')
ans = '{} {}'
pairs = [ans.format(l + 2*i, l + 2*i + 1) for i in range((r - l + 1)//2)]
print('\n'.join(pairs))
```

4. Задача 6.D

По введенным числам *A* и *B* вывести все простые числа в интервале от *A* до *B* включительно.

45967432	01:21:24	tamarinvs19	D - Решето Эратосфена	Python 3	Полное решение	452 мс	4600 КБ
45967383	01:20:23	tamarinvs19	D - Решето Эратосфена	PyPy 3	Полное решение	186 мс	6300 КБ

```
    a, b = map(int, input().split())
    a, b = min(a, b), max(a, b)
    is_comp = [False] * (b - 1)
    for i in range(2, b+1):
        if not is_comp[i-2]:
            j = i ** 2
            while j <= b:
                is_comp[j-2] = True
                j += i
    p = [i+2 for i in range(len(is_comp)) if not is_comp[i] and i+2 >= a]
    print(' '.join(map(str, p)))
```

5. Задача 5.C

Дано *n* отрезков на числовой прямой и m точек на этой же прямой. Для каждой из данных точек определите, скольким отрезкам они принадлежат. Точка x считается принадлежащей отрезку с концами a и b , если выполняется двойное неравенство $min(a,b) \le x \le max(a,b)$.

46070362	2018-11-22 18:59:40	tamarinvs19	D - Дорешивание	Python 3	Полное решение	967 мс	19200 КБ
45477475	2018-11-09 16:08:34	tamarinvs19	C - Точки и отрезки	PyPy 3	Полное решение	1294 мс	22400 КБ

```
from sys import stdin
    def association_segments(events):
        i = 1
        last = -1
        c = 0
        events.sort()
        res = dict()
```

```
    for x, t in events:
        if t == -1:
            c += 1
        elif t == 0:
            res[x] = c
        else:
            c -= 1
    return res

n, m = map(int, input().split())
events = []
for _ in range(n):
    l, r = map(int, stdin.readline().strip().split())
    if l > r:
        l, r = r, l
    events.append((l, -1))
    events.append((r, 1))
points = [(x, 0) for x in map(int, input().split())]
events += points
res = association_segments(events)
for p in points:
    print(res[p[0]], end=' ')
```

6. Задача 5.D

Как известно, после обеда в ЛКШ проходит много интересных мероприятий, но все равно каждый ЛКШонок старается хотя бы ненадолго заглянуть в комповник, чтобы дорешать задачи, не сделанные во время практики.
В этом году погода стоит особо жаркая, поэтому в комповнике очень душно и важно следить за тем, чтобы в комповнике не находилось одновременно очень много школьников. Поэтому завуч записал время прихода и ухода из комповника каждого ЛКШонка.
Теперь завуч хочет узнать, сколько ЛКШат встретил в комповнике каждый ЛКШонок.

46070362	2018-11-22 18:59:40	tamarinvs19	D - Дорешивание	Python 3	Полное решение	967 мс	19200 КБ
45525712	2018-11-10 18:34:21	tamarinvs19	D - Дорешивание	PyPy 3	Полное решение	1606 мс	13400 КБ

```
n = int(input())
xs = []
for i in range(1, n+1):
    y = list(map(int, input().split()))
    xs.append((y[0], -i))
    xs.append((y[1], i))

xs.sort()
count_open = 0
count_close = 0
ys = [0]*(n+1)
for x, t in xs:
    if t < 0:
        ys[abs(t)] -= count_close
        count_open += 1
    else:
        ys[abs(t)] += count_open
        count_close += 1
for y in ys[1:]:
    print(y-1)
```

7. Задача 7.C

Определите, какое наименьшее количество операций требуется, чтобы получить из числа 1 число N.
Имеется калькулятор, который выполняет следующие операции:

i. умножить число X на 2;
ii. умножить число X на 3;
iii. прибавить к числу X единицу.

№	Когда	Кто	Задача	Язык	Вердикт	Время	Память
46284220	26:05:00	tamarinvs19	C - Калькулятор	Python 3	Превышено ограничение времени на тесте 3	2000 мс	31900 КБ
46284198	26:04:19	tamarinvs19	C - Калькулятор	PyPy 3	Полное решение	451 мс	26100 КБ

```
class f:
    def __init__ (self, n):
        self.n = n
    def apply(self, x):
        if self.n == 1:
            return True, x - 1
        elif x % self.n == 0:
            return True, x//self.n
        else:
            return False, ''


n = int(input())
d = [(0, 1)]*(n+1)
d[1] = (0, 1)

for i in range(2, n+1):
    m = min((d[f[1]], f[1]) for f in [f(j+1).apply(i) for j in range(3)] if f[0])
    d[i] = (1 + m[0][0], m[1])
i = n
steps = [n]
while i != 1:
    i = d[i][1]
    steps.append(i)

print(d[-1][0])
print(' '.join(map(str, steps[::-1])))
```

8. Задача 7.H

Найдите максимальный вес золота, который можно унести в рюкзаке вместительностью S, если есть n золотых слитков с заданными весами.

№	Когда	Кто	Задача	Язык	Вердикт	Время	Память
46378199	3 дня	tamarinvs19	H - Рюкзак без стоимости	Python 3	Полное решение	717 мс	5700 КБ
46378184	3 дня	tamarinvs19	H - Рюкзак без стоимости	PyPy 3	Полное решение	140 мс	6700 КБ

```
s, n = map(int, input().split())
ws = list(map(int, input().split()))

d = [[False for _ in range(s+1)] for _ in range(n+1)]
d[0][0] = True
for i in range(1, n+1):
    for j in range(s+1):
        if j - ws[i-1] >= 0:
            d[i][j] = d[i-1][j - ws[i-1]] or d[i-1][j]
        else:
            d[i][j] = d[i-1][j]
print(max(j for j in range(s+1) if d[n][j]))
```

9. Задача 7.I

Дано n предметов массой $m_1,\ldots,m_n$ и стоимостью $c_1,\ldots,c_n$ соответственно.

Ими наполняют рюкзак, который выдерживает вес не более m. Определите набор предметов, который можно унести в рюкзаке, имеющий наибольшую стоимость.

46481315	6 дней	tamarinvs19	I - Рюкзак	Python 3	Ошибка исполнения на тесте 59	998 мс	7200 КБ
46481248	6 дней	tamarinvs19	I - Рюкзак	PyPy 3	Полное решение	155 мс	5200 КБ

```
n, m = map(int, input().split())
ws = list(map(int, input().split()))
cs = list(map(int, input().split()))
ws.insert(0, 0)
cs.insert(0, 0)
d = [[0 for _ in range(m+1)] for _ in range(n+1)]
```

```
for i in range(1, n+1):
    for j in range(m+1):
        d[i][j] = d[i-1][j]
        if j >= ws[i] and d[i-1][j-ws[i]] + cs[i] > d[i][j]:
            d[i][j] =  d[i-1][j-ws[i]] + cs[i]

ans = []
def answer(i, j):
    if d[i][j] == 0:
        return
    elif d[i-1][j] == d[i][j]:
        answer(i-1, j)
    else:
        answer(i-1, j - ws[i])
        ans.append(i)
answer(n, m)
print(len(ans))
print(' '.join(map(str, ans)))
```

10. Задача 8.C

Дана текстовая строка. С ней можно выполнять следующие операции:
1. Заменить один символ строки на другой символ.
2. Удалить один произвольный символ.
3. Вставить произвольный символ в произвольное место строки.
Например, при помощи первой операции из строки «СОК» можно получить строку «СУК», при помощи второй операции - строку «ОК», при помощи третьей операции - строку «СТОК».
Минимальное количество таких операций, при помощи которых можно из одной строки получить другую, называется стоимостью редактирования или расстоянием Левенштейна.
Определите расстояние Левенштейна для двух данных строк.

Мои посылки

№	Когда	Кто	Задача	Язык	Вердикт	Время	Память
46774513	5 дней	tamarinvs19	B - Расстояние по Левенштейну	Python 3	Полное решение	1387 мс	18900 КБ
46774454	5 дней	tamarinvs19	B - Расстояние по Левенштейну	PyPy 3	Полное решение	187 мс	5900 КБ

```
a = input()
b = input()
n, m = len(a), len(b)

d = [[0] * (m+1) for _ in range(n+1)]
for i in range(1, n+1):
    d[i][0] = i
for j in range(1, m+1):
    d[0][j] = j

for i in range(1, n+1):
    for j in range(1, m+1):
        d[i][j] = min(d[i-1][j] + 1, d[i][j-1] + 1, d[i-1][j-1] + int(a[i-1] != b[j-1]))

print(d[-1][-1])
```

11. Олимпиада Ломоносов.B

Возьмем некоторое составное число в N-ричной системе счисления и разложим его на простые множители. Если сумма цифр числа и сумма цифр всех его простых множителей совпала, назовем его числом Смита. Если два натуральных числа идут друг за другом и оба являются числами Смита, они называются близнецами Смита. Для заданного N найдите вторых близнецов Смита.

TL = 1 секунда

Python3.3

N	Результат	Время (с)	Баллы
1	OK	0.192	0 (0)
2	OK	0.032	6 (6)
3	OK	0.084	6 (6)
4	Превышено максимальное время работы	>1.000	0 (6)
5	OK	0.019	6 (6)
6	OK	0.249	6 (6)
7	OK	0.039	7 (7)
8	OK	0.952	7 (7)
9	OK	0.196	7 (7)
10	OK	0.035	7 (7)
11	OK	0.030	7 (7)
12	Превышено максимальное время работы	>1.000	0 (7)
13	OK	0.020	7 (7)
14	OK	0.185	7 (7)
15	OK	0.537	7 (7)
16	OK	0.068	7 (7)

PyPy3.2

N	Результат	Время (с)	Баллы
1	OK	0.271	0 (0)
2	OK	0.217	6 (6)
3	OK	0.263	6 (6)
4	OK	0.352	6 (6)
5	OK	0.201	6 (6)
6	OK	0.287	6 (6)
7	OK	0.226	7 (7)
8	OK	0.363	7 (7)
9	OK	0.281	7 (7)
10	OK	0.220	7 (7)
11	OK	0.207	7 (7)
12	OK	0.383	7 (7)
13	OK	0.198	7 (7)
14	OK	0.286	7 (7)
15	OK	0.312	7 (7)
16	OK	0.251	7 (7)

```
def to_digits(n, b):
    digits = []
    while n > 0:
        digits.append(n % b)
        n //= b
    return digits[::-1]

def to_primes(n):
    d = 2
    res = []
    while n > 1:
        if n % d == 0:
            res.append(d)
            n //= d
        else:
            d += 1
    return res

def main():
    N = int(input())
    n = 2
    m = 0
    x = 0
    while m != 2:
        if len(to_primes(n)) > 1 and \
                    sum(to_digits(n, N)) == sum(sum(to_digits(p, N)) \
                    for p in to_primes(n)):
            x += 1
            if x >= 2:
                m += 1
        else:
            x = 0
        n += 1
    print(n-2)

if __name__ == '__main__':
    main()
```

12. Задача 10.a

Дан ориентированный взвешенный ациклический граф. Требуется найти в нем кратчайший путь из вершины *s* в вершину *t*.

№	Когда	Кто	Задача	Язык	Вердикт	Время	Память
47302684	3 дня	tamarinvs19	A - Кратчайший путь	PyPy 3	Полное решение	1918 мс	20200 КБ
47302555	3 дня	tamarinvs19	A - Кратчайший путь	Python 3	Превышено ограничение времени на тесте 23	2000 мс	18500 КБ

```
from math import inf
def main():
    n, m, s, t = map(int, input().split())
    s, t = s-1, t-1
    parents = [list() for _ in range(n)]
    fs = ['empty']*n
    for _ in range(m):
        a, b, w = map(int, input().split())
        parents[b-1].append((a-1, w))

    queue = [t]
    while len(queue) > 0:
        v = queue[-1]
        if v == s:
            fs[v] = 0
            del queue[-1]
        elif len(parents[v]) == 0:
            fs[v] = inf
            del queue[-1]
        else:
            ans = inf
            new_queue = []
            for i, w in parents[v]:
                if fs[i] == 'empty':
                    new_queue.append(i)
                else:
                    ans = min(ans, fs[i] + w)
            if len(new_queue) == 0:
                fs[v] = ans
                del queue[-1]
            else:
                queue += new_queue
    result = fs[t]
    if result == inf:
        result = 'Unreachable'
    print(result)
if __name__ == '__main__':
    main()
```

13. Задача 10.G

Требуется найти количество ребер в конденсации ориентированного графа.
Примечание: конденсация графа не содержит кратных ребер.
* *Дополнительная информация в папке 10_g*

№	Когда	Кто	Задача	Язык	Вердикт	Время	Память
47894585	2019-01-04 12:25:02	tamarinvs19	G - Конденсация графа	Python 3	Превышено ограничение времени на тесте 29	2000 мс	14600 КБ
47894561	2019-01-04 12:24:13	tamarinvs19	G - Конденсация графа	PyPy 3	С Новым годом!	1169 мс	14200 КБ

```
def my_input():
    n, m = map(int, input().split())
    g = [list() for _ in range(n)]
    h = [list() for _ in range(n)]
    for _ in range(m):
        a, b = map(int, input().split())
        g[a-1].append(b-1)
        h[b-1].append(a-1)
    return n, m, g, h
```

```
def dfs1():
    global tout, used, stack, stack_parents
    while stack != []:
        v = stack[-1]
        if v == stack_parents[-1]:
            del stack_parents[-1]
            del stack[-1]
            tout.append(v)
        elif not used[v]:
            stack_parents.append(v)
            used[v] = True
            for u in g[v]:
                if not used[u]:
                    stack.append(u)
        else:
            del stack[-1]

def dfs2():
    global component, col, es, stack_parents
    while stack != []:
        v = stack.pop()
        component[v] = col
        for u in h[v]:
            if component[u] == 0:
                stack.append(u)
            elif component[u] != col:
                es.append((col, component[u]))

def main():
    global col, component, stack, stack_parents, g, h, es, tout, used
    n, m, g, h = my_input()
    used = [0]*n
    tout = []
    component = [0]*n
    col = 1
    es = []
    i = 0
    stack = []
    stack_parents = []
    for v in range(n):
        if not used[v]:
            stack = [v]
            stack_parents = [-1]
            dfs1()
    for v in tout[::-1]:
        if component[v] == 0:
            stack = [v]
            dfs2()
            col += 1
    return len(set(es))
if __name__ == '__main__':
    print(main())
```

14.