I. Задания заключительного этапа олимпиады 2021-22 года

Заключительный этап для 11 класса (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (3 балла)

[Двоичный ряд]

Петя строит ряд двоичных чисел следующего вида:

$0.(01)_2$, $0.0(01)_2$, $0.00(01)_2$, …

Каждый член ряда представляет собой бесконечную периодическую дробь. Для получения каждого следующего числа Петя берет предыдущее число и вставляет один ноль после разделителя целой и дробной части.

Петя взял первые 30 чисел этого ряда и сложил их, записав результат в шестнадцатеричной системе счисления. Посчитайте сумму цифр в дробной части получившегося числа. В ответе укажите целое число в десятичной системе счисления. Например, если запись суммы чисел в шестнадцатеричной системе счисления будет равна $1,ABC_{16}$, то в ответ нужно записать 33.

Ответ: 78

Решение:

Рассмотрим первый член ряда и запишем равенство:

A: $0.(01)_2 = X_{10}$

Домножим обе части равенства на $100_2=4_{10}$

B: $1.(01)_2=(4*X)_{10}$

Вычтем из равенства B равенство A.

B-A: $1.(01)_2 – 0.(01)_2 =(4*X)_{10} – X_{10}$

$1_2=1_{10}=(3*X)_{10}$

X=1/3

Обратим внимание, что вставка одного нуля после разделителя целой и дробной части эквивалентна делению числа на основание системы счисления. То есть, каждый следующий член ряда получается умножением предыдущего на 1/2. Следовательно, мы имеем геометрическую прогрессию:

1/3, 1/6, 1/12, …

Сумма её первых 30 членов может быть получена с использованием известной формулы:

$(1-q^{n+1})/(1-q)$

Подставив q=1/2 и проведя несложные алгебраические преобразования, получим:

$(2^n-1)/(3*2^{n-1})$

Теперь нам нужно вычислить значение при n=30 и представить в двоичной системе счисления, из которой можно будет уже перевести в шестнадцатеричную. Это можно сделать различными способами. Один из вариантов, проанализировать значения числителя и знаменателя для нескольких значений n. Например, с помощью электронной таблицы:

n	числитель	знаменатель
1	1	3
2	3	6
3	7	12
4	15	24
5	31	48
6	63	96
7	127	192
8	255	384
9	511	768
10	1023	1536

Обратим внимание, что при четных значения n (а 30 – четное число), числитель всегда кратен 3 (это можно доказать, например, через формулу для $a^{2n}-b^{2n}$), а значит для таких значений можно поделить числитель и знаменатель на 3. Построим новые значения числителя и знаменателя, поделенные на 3 для первых нескольких четных значений n и запишем их в двоичной системе счисления:

n	числитель	знаменатель	числитель в дв. с.с.	знаменатель дв. с.с.
2	1	2	1	10
4	5	8	101	1000
6	21	32	10101	100000
8	85	128	1010101	10000000
10	341	512	101010101	1000000000

Заметим, что тогда значения вычисляемой нами суммы для этих значений n в двоичной и шестнадцатеричной системах счисления будут следующими:

n	числитель в дв. с.с.	знаменатель дв. с.с.	сумма членов в дв. с.с.	дробная часть суммц членов в шестнадц. с.с.
2	1	10	0.1	8
4	101	1000	0.101	A
6	10101	100000	0.10101	A8
8	1010101	10000000	0.1010101	AA
10	101010101	1000000000	0.101010101	AA8

Как видно из таблицы, запись искомой суммы членов ряда в шестнадцатеричной системе счисления для значений n кратных 4 будет состоять из n/4 цифр A после разделителя целой и дробной части, а для n не кратных 4 – из (n-2)/4 цифр A и одной цифры 8 после разделителя целой и дробной части. То есть, для n=30 это будет 0,AAAAAAA8 и сумма цифр будет (30-2)/4*10+8=78.

Заметим, что можно вычислить и непосредственно значения числителя и знаменателя для дроби $(2^{30}-1)/(3*2^{30-1})$, поделить их на 3 и перевести в шестнадцатеричную систему, но это потребует достаточно трудоемких вычислений.

Альтернативное решение после того, как определили, что необходимо посчитать сумму членов геометрической прогрессии:

1/3+1/6+1/12 + …1/ (3*2^29) = (1/3) *(1+1/2+1/ (2^2) +1/(2^3)+….+ 1/(2^29))

Сгруппируем по парам

= (1/3) *((1+1/2) + (1/ (2^2)+1/(2^3))+…+(1/(2^28)+1/(2^29)) =

= (1/3) *(3/2+3/ (2^3) +….3/(2^29))

выносим 3 и сокращаем

= 1/2 + 1/(2^3)+1/(2^5)+…+1/(2^29)

Получается сумма отрицательных нечетных степеней 2.

В двоичной системе это равно 0.10101010..1

Всего 29 знаков в дробной части. Переводим в 16-ую систему – по 4 знака от запятой (29=4*7+1)

= 0.AAAAAAA8

Следовательно сумма цифр равна 10*7+8=78

2. Кодирование информации. Объем информации (2 балла) [RLE]

Петя сохраняет в память текст. Известно, что алфавит текста составляет 32 символа. Петя использует равномерное кодирование и сохраняет в память коды символов, используя минимально возможное одинаковое для кодов всех символов количество бит.

Вася и Таня изучают кодирование длин серий run-length encoding (RLE) и пытаются его применить к тексту Пети для того, чтобы он занимал меньше места в памяти.

В случае RLE кодирования весь текст представляется как расположенные друг за другом непересекающиеся последовательности из одинаковых символов так, что символы в двух соседних последовательностях отличаются. При этом в памяти сохраняется для каждой последовательности два значения: длина последовательности и код повторяющегося в ней символа. Для записи кода символа используется минимальное возможное одинаковое для кодов всех символов количество бит. Для записи длины последовательности используется минимально возможное одинаковое для всех возможных значений этого параметра количество бит.

Вася обнаружил, что максимальная длина последовательности из одинаковых символов, которая встречается в тексте, равна M и решил, что в строке могут встречаться все возможные длины последовательностей, не превышающие M. Исходя из этого предположения, он определил количество бит, необходимое для записи длины последовательности и, сохранив в память текст Пети с помощью кодирования RLE, обнаружил, что он занимает ровно 208 байт памяти.

Также Вася обнаружил, что максимальная длина последовательности из одинаковых символов ровно в X раз меньше общего количества символов в тексте Пети.

Таня проанализировала текст более внимательно и обнаружила, что количество различных длин последовательностей, которые встречаются в тексте ровно в 16 раз меньше, чем длина максимальной последовательности. Исходя из этого, она определила другое количество бит, необходимое для записи длины последовательности и, сохранив в память текст Пети с помощью кодирования RLE, обнаружила, что он занимает ровно 144 байта памяти.

Определите значение X, если известно, что текст, сохраненный Петей, занимает на 12272 байта больше памяти, чем текст, сохраненный Васей. Если таких значений несколько, определите минимальное из них. В ответе укажите целое число.

Ответ: 78

Решение:

Поскольку текст, сохраненный Васей, занимает 208 байт и это на 12272 байта меньше, чем текст, сохраненный Петей, объем памяти, который занимает текст, сохраненный Петей, составляет 12272+208=12480 байт.

Для сохранения кода одного символа при равномерном кодировании для алфавита из 32 символов потребуется log2(32)=5 бит.

Следовательно, количество символов в тексте равно 12480*8/5=19968. И это значение представимо как произведение искомой нами величины X на пока неизвестное значение M. То есть X=19968/M.

Найдем значение M. Для этого составим уравнения исходя из известным нам значений объемов памяти, занимаемых при кодировании RLE Васей и Таней.

Обратим внимание, что поскольку по условию требуется найти минимальное значение X, нам требуется найти максимальное значение M.

Пусть N – количество последовательностей, на которое разбивается текст Пети, тогда:

Вася: $N*(\log_2(M)+\log_2(32))=208*8$

Таня: $N*(\log_2*(M/16)+\log_2(32))=144*8$

Поскольку мы ищем максимальное значение M, мы можем считать, что в уравнениях стоят именно значения log2(M), а не их округления в большую сторону до ближайшего целого числа.

Вычтем из первого уравнения второе:

$N*(\log_2(M)+\log_2(32)) - N*(\log_2(M/16)+\log_2(32)) = (208-144)*8$

$N*((\log_2(M)+\log_2(32)) - (\log_2(M/16)+\log_2(32))) = 512$

$N*(\log_2(M)+\log_2(32) - \log_2(M/16) - \log_2(32)) = 512$

$N*(\log_2(M) - \log_2(M/16)) = 512$

$N*(\log_2(M / (M/16))) = 512$

$N*(\log_2(16))=512$

$N*4=512$

$N=128$

Подставим найденное значение N в первое уравнение:

$128*(\log_2(M)+\log_2(32))=208*8$

$128*(\log_2(M)+5)=1664$

$\log_2(M)+5=1664/128$

$\log_2(M)=8$

$M=256$

Таким образом, X=19968/256=78

3. Основы логики (2 балла) [Много чисел]

Логический преобразователь работает следующим образом:

1. На вход преобразователю подаётся целое неотрицательное число N, меньшее 2^{64}_{10}, записанное в шестнадцатеричной системе счисления. Запись числа переводится в двоичную систему счисления и разбивается на 16 групп по 4 разряда (при необходимости в начале записи дописываются незначащие нули).
2. Двигаясь справа налево по очереди берется каждая группа двоичных разрядов и представляется как последовательность из четырех логических переменных $X_0X_1X_2X_3$, так, что единичное значение двоичного разряда соответствует значению "истина", а нулевое - значению "ложь". Например, в числе 0…010111 первая такая последовательность будет X_0 = «ложь», X_1 = «истина», X_2 = «истина», X_3 = «истина».
3. Полученные последовательности логических переменных по очереди подставляются в логическое выражение:
$$(X_0 \vee X_1 \wedge X_2) \wedge (X_1 \vee X_2 \wedge X_3) \wedge (X_2 \vee X_3 \wedge X_0)$$
и вычисляется его значение.
4. Вычисленные значения преобразуются в двоичные разряды (также «истина» = 1, а «ложь» = 0) и образуют двоичную последовательность, заполняемую от младшего к старшему разряду.
5. Результатом работы преобразователя является шестнадцатиразрядное двоичное число (запись может содержать незначащие нули).

На вход преобразователю подали запись числа, равного $FEDCBA9876543210_{16}$ и получили некоторое число M. Сколько всего существует таких чисел, которые можно подать на вход преобразователю и получить в результате такое же число M? Поскольку число может получиться достаточно большое, посчитайте и укажите в ответе сумму его цифр при записи в десятичной системе счисления.

Ответ: 27

Решение:

Построим таблицу истинности для заданного в условии выражения. Это можно сделать как вручную, так и написав программу. Например, так:

```
N=int('FEDCBA9876543210',16)
tab=[]
for i in range(16):
    num=N%16
    st=[]
    for j in range(4):
        st.append(num%2)
        num//=2
    F=1 if (st[3]==1 or st[2]==1 and st[1]==1) and (st[2]==1 or st[1]==1 and st[0]==1) and (st[1]==1 or st[0]==1 and st[3]==1) else 0
    tab.append([st[::-1],F])
    N//=16
for i in tab:
    print(i)
```

X_0	X_1	X_2	X_3	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Заметим, что в таблице 10 строк имеют ложное значение и 6 строк имеют истинное значение.

Обратим внимание, что преобразователь, разбивая двоичную запись на 16 групп по 4 разряда получает двоичные записи отдельных цифр шестнадцатеричного числа (при необходимости, дополненного незначащими нулями). Эти записи можно интерпретировать как значения логических переменных в строках таблицы истинности Приведенное в условии число $FEDCBA9876543210_{16}$ содержит все возможные шестнадцатеричные цифры, расположенные по убыванию, следовательно, если перевести это число в двоичную систему счисления и начать рассматривать группы по 4 разряда с конца к началу записи, как указано в условии, мы получим все комбинации значений логических переменных, то есть все строки построенной выше таблицы истинности. Тогда число M получается из последнего столбца как $1110100011000000_2 = 59584_{10}$.

Значение выражения для каждой строки таблицы истинности получается независимо и может быть истинно или ложно. Тогда, получается, что каждая цифра шестнадцатеричного числа даёт либо 0 либо 1 в очередном по порядку следования двоичном разряде числа M. А из таблицы истинности мы знаем какая цифра какое значение даёт и в каком разряде должно быть какое значение. Следовательно, любая цифра, которая даст значение 1 может быть на любой позиции, на которой должно получится значение 1, а любая цифра, которая даст значение 0 может быть на любой позиции, на которой должно получится значение 0. Теперь вспомним наблюдение, которое мы записали после таблицы истинности и выведем выражение для вычисления количества подходящих чисел: $10^{10}*6^6 = 466560000000000$. Сумма цифр в этом числе равна 27, что и является правильным ответом.

4. Алгоритмизация и программирование. Формальный исполнитель (1 балл) [Лабиринт]

Дан лабиринт, размером 10 на 10 клеток:

```
*.********
*........*
*.*.**.*.*
*.*....*.*
*.*.**.*.*
*.*.**.*.*
*.*....*.*
*.******.*
*........*
**********
```

Каждая клетка содержит один из двух символов: «*» означает непреодолимое препятствие, а «.» свободная клетка. В одну из свободных клеток помещают шарик. Если ниже этой клетки есть еще свободные клетки, шарик падает, пока не наталкивается на непреодолимое препятствие. Далее лабиринт поворачивают следующим образом:

1. На 90 градусов по часовой стрелке
2. На 90 градусов по часовой стрелке
3. На 90 градусов по часовой стрелке
4. На 90 градусов против часовой стрелки.

После каждого поворота, если ниже той клетки, в которой находится шарик, есть еще свободные клетки, шарик падает, пока не наталкивается на непреодолимое препятствие. После этого можно осуществлять следующий поворот.

Сколько существует клеток, в которые можно исходно поместить шарик так, что в результате выполнения указанных поворотов (возможно до их завершения), он выкатится из лабиринта? В ответе укажите целое число.

Ответ: 43

Решение:

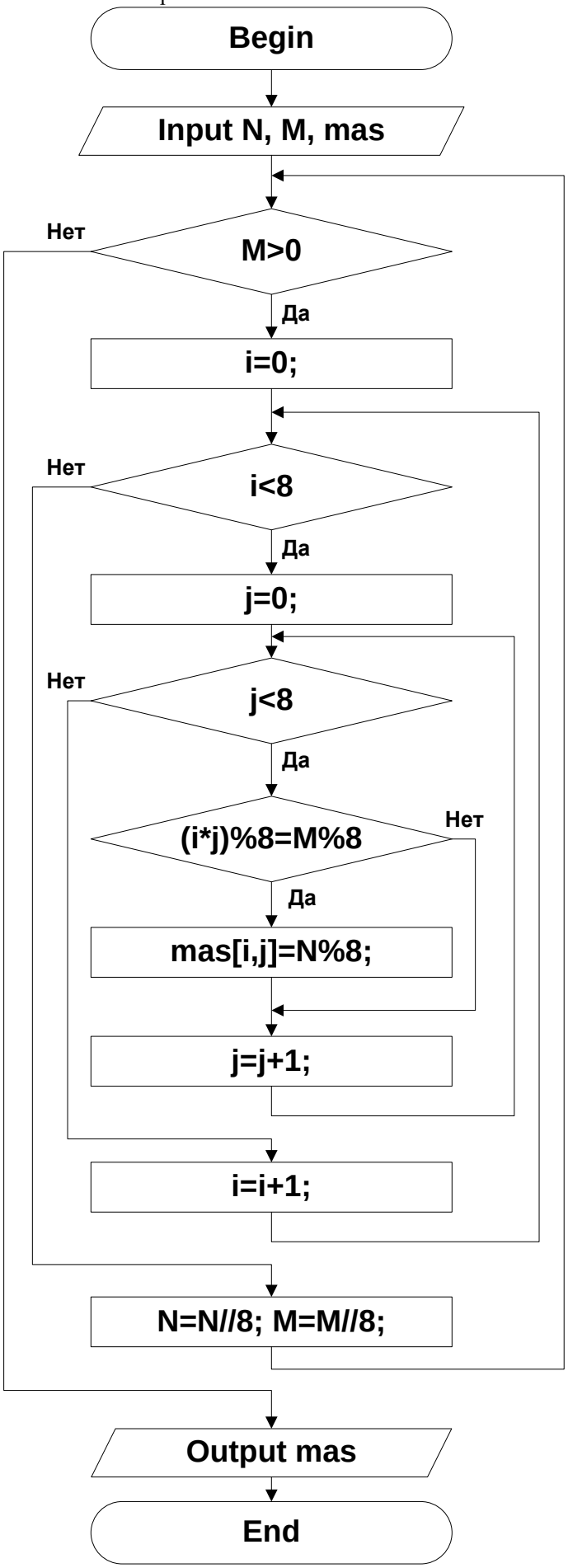
Можно перебрать точки вручную, но в этом случае легко допустить ошибку по невнимательности. Можно реализовать алгоритм в виде программы. Ниже приведен пример на языке Python. Программа избыточна, поскольку демонстрирует также траектории для всех найденных точек, что может быть удобно для анализа решения задания. Для нахождения только ответа на задание её можно упростить.

```
Path='RRRL'
ans=0
for x in range(10):
    for y in range(10):
        A=[]
        A.append(list("*.********"))
        A.append(list("*........*"))
        A.append(list("*.*.**.*.*"))
        A.append(list("*.*....*.*"))
        A.append(list("*.*.**.*.*"))
        A.append(list("*.*.**.*.*"))
        A.append(list("*.*....*.*"))
        A.append(list("*.******.*"))
        A.append(list("*........*"))
        A.append(list("**********"))
        if A[x][y]!='*':
            C=[x,y]
            A[C[0]][C[1]]='O'
            while A[C[0]+1][C[1]]!='*':
                C[0]+=1
                A[C[0]][C[1]]='o'
            D=0 #Направление: 0 - вниз, 1 - вправо, 2 - вверх, 3 - влево
            out=False
            for i in range(len(Path)):
                if not out:
                    D=(D+1)%4 if Path[i]=='R' else (D+3)%4
                    if D==0:
                        while A[C[0]+1][C[1]]!='*':
                            C[0]+=1
                            A[C[0]][C[1]]='o'
                    elif D==1:
                        while A[C[0]][C[1]+1]!='*':
                            C[1]+=1
                            A[C[0]][C[1]]='o'
                    elif D==2:
                        while A[C[0]-1][C[1]]!='*':
                            C[0]-=1
                            A[C[0]][C[1]]='o'
                            if C==[0,1]:
                                print('Finish from', x, y)
                                for i in A:
                                    print("".join(i))
                                out=True
                                ans+=1
                                break
                    else:
                        while A[C[0]][C[1]-1]!='*':
                            C[1]-=1
                            A[C[0]][C[1]]='o'
print(ans)
```

5. Алгоритмизация и программирование. Анализ алгоритма, заданного в виде блок-схемы (2 балла)

[Странная симметрия]

Дана блок-схема алгоритма.



На вход алгоритму подаются два целых положительных числа N и M и двумерный массив mas размером 8x8 элементов инициализированный нулями.

Известно, что на вход было подано число M=8143989. Определите минимальное значение числа N, такое, что на выходе будет массив mas со следующими значениями:

$$\begin{bmatrix} 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 2 & 3 & 7 & 4 & 0 & 1 & 6 \\ 5 & 3 & 4 & 1 & 5 & 3 & 4 & 1 \\ 5 & 7 & 1 & 2 & 4 & 6 & 3 & 0 \\ 5 & 4 & 5 & 4 & 5 & 4 & 5 & 4 \\ 5 & 0 & 3 & 6 & 4 & 2 & 1 & 7 \\ 5 & 1 & 4 & 3 & 5 & 1 & 4 & 3 \\ 5 & 6 & 1 & 0 & 4 & 7 & 3 & 2 \end{bmatrix}$$

В ответе укажите целое число. Если такого числа не существует, запишите в ответ NULL.

Примечания:

1. При обращении к элементам массива первый индекс означает номер строки, а второй – номер столбца, нумерация строк и столбцов начинается с нуля.
2. Операция A%B означает вычисление остатка при целочисленном делении A на B, а операция A//B означает вычисление частного при целочисленном делении A на B.

Ответ: 16434824

Решение:

Проанализируем алгоритм. Во внешнем цикле число M разбирается на цифры его записи в восьмеричной системе счисления от младшего разряда к старшему. Обратим внимание, что с числом N поступают аналогично. Тогда на каждом шаге внешнего цикла значения M%8 и N%8 – это очередные цифры восьмеричных записей чисел M и N, соответственно. Для каждой пары цифр чисел M и N двумя вложенными циклами перебираются все элементы двумерного массива. При этом, если для некоторого элемента выполняется условие, что остаток от деления произведения индексов (i*j) этого элемента равен текущей цифре восьмеричной записи числа M, в этот элемент записывается текущая цифра восьмеричной записи числа N. Цикл продолжается, пока не будут перебраны все цифры в числе M.

Переведем число M в восьмеричную систему счисления и получим, что $8143989_{10}=37042165_8$. Обратим внимание, что в записи числа содержатся все возможные восьмеричные цифры и ровно по одному разу каждая.

Теперь построим таблицу значений остатков (i*j)%8. Это можно сделать вручную, но удобнее воспользоваться программированием или электронными таблицами. Например, используя формулу =ОСТАТ($A2*B$1;8) в ячейках диапазона B2:I9:

	A	B	C	D	E	F	G	H	I	J
1		0	1	2	3	4	5	6	7	
2	0	0	0	0	0	0	0	0	0	
3	1	0	1	2	3	4	5	6	7	
4	2	0	2	4	6	0	2	4	6	
5	3	0	3	6	1	4	7	2	5	
6	4	0	4	0	4	0	4	0	4	
7	5	0	5	2	7	4	1	6	3	
8	6	0	6	4	2	0	6	4	2	
9	7	0	7	6	5	4	3	2	1	
10										

Обратим внимание, что там, где в построенной таблице встречается значение 5, в двумерном массиве из условия стоит младшая цифра восьмеричной записи числа N, то есть она равна 0. Аналогично определим остальные цифры. Обратим внимание, что для этого достаточно проанализировать вторую строку или второй столбец. В результате получим восьмеричное число 76543210_8. Переведем его в десятичную систему счисления и получим ответ 16434824.

6. Телекоммуникационные технологии (2 балла). [Wildcard Mask]

В IPv4 сетевой адрес представляет собой 4-байтное число, записанное в десятичной форме, где отдельные байты разделены точкой. В IP-адресе выделяются две части – адрес сети и адрес узла. Деление происходит с помощью маски – 4-х байтного числа, которое поставлено в соответствие IP-адресу. Маска содержит двоичные 1 в тех разрядах IP-адреса, которые определяют адрес сети и двоичные 0 в тех разрядах IP адреса, которые определяют адрес узла.

Кроме обычной маски существует еще wildcard mask. Эти маски часто применяются для написания правил фильтрации в межсетевых экранах (firewall) для того, чтобы определить применяется ли правило к ip адресу из проверяемого пакета или нет. Wildcard mask – 4-байтное число, которое содержит двоичные единицы в тех разрядах IP адреса, которые могут в ip адресе содержать разные значения, и нули в тех разрядах адреса, которые должны совпадать с заданными в правиле фильтрации. Wildcard mask не обязана содержать подряд идущие единицы или нули.

Межсетевой экран — программный или программно-аппаратный элемент компьютерной сети, осуществляющий контроль и фильтрацию проходящего через него сетевого трафика в соответствии с заданными правилами. В этой задаче мы

будем использовать синтаксис команд для оборудования cisco. Реализована отдельная процедура составления правил. Проверка правил идет сверху вниз по списку до первого срабатывания.

Рассмотрим пример, в котором используются обычные mask и wildcard mask. Будем считать, что для определения типа маски используется её старший бит: 1 – обычная маска, а 0 – wildcard mask. В примере опущены команды, необходимые для переключения контекстов управления и активации списка правил.

```
# правило 1. создать список контроля с именем 123 и добавить в него правило, пропускающее пакеты с одного адреса 192.168.0.11 на любые адреса. То, что правило работает для одного адреса, видно из использованной маски, состоящей целиком из 32-х единиц.
access-list 123 permit ip 192.168.0.11 255.255.255.255 any
# правило 2. добавить в список 123 правило, запрещающее передачу пакетов со всех нечетных адресов сети 192.168.0.0 255.255.255.0 по любым направлениям
access-list 123 deny ip 192.168.0.1 0.0.0.254 any
```

Рассмотрим подробнее правило с wildcard mask. Предположим, на порт поступает пакет с адреса 192.168.0.127. И надо проверить, применять ли правило 2 к этому адресу. Представим в двоичном формате проверяемый адрес, адрес-образец из правила и wildcard mask.

192.168.0.127 – проверяемый адрес	11000000.10101000.00000000.01111111
192.168.0.1 – адрес образец	11000000.10101000.00000000.00000001
0.0.0.254 – wildcard mask	00000000.00000000.00000000.11111110

Желтым отмечены разряды, которые должны совпадать в адресе-образце из правила и проверяемом адресе. Номера этих разрядов определяются по wildcard mask.

Зеленым цветом выделены разряды, которые могут отличаться у проверяемого адреса и адреса-образца из правила (в этих разрядах wildcard mask содержит 1).

Очевидно, что для адреса 192.168.0.127 правило применится, а, например, для адреса 192.168.0.128 это правило не применится (не совпадет последний бит).

Также отметим, что для адреса 192.168.0.11 это правило не будет применено, потому что для этого адреса будет использовано правило 1, находящееся раньше по списку.

Задание

Ниже приведен список команд, составляющий исходный набор правил. Требуется сократить количество правил, а, следовательно, и время проверки, с использованием **только** wildcard mask. Есть набор возможных вариантов команд, которые могут быть включены в новый набор правил. Выберите из этого набора нужные команды и составьте из них набор правил минимально возможной длины, такой, что он будет полностью соответствовать исходному набору правил. Приведите в ответе через пробел номера из набора возможных вариантов команд. Номера указывайте в нужном порядке, там, где это необходимо. Если подходящих вариантов порядка несколько, приведите тот, в котором номера идут по возрастанию. Если есть равнозначный выбор между несколькими правилами с одинаковой wildcard mask, выбирается правило с минимальным адресом-образцом.

Межсетевой экран непосредственно подключен к сети с адресом 172.21.25.144 и маской 255.255.255.240.

Список правил составляется **только** для реальных адресов указанной сети.

Исходный набор правил:

```
access-list 123 permit ip 172.21.25.145 255.255.255.255 any
access-list 123 permit ip 172.21.25.146 255.255.255.255 any
access-list 123 permit ip 172.21.25.147 255.255.255.255 any
access-list 123 permit ip 172.21.25.148 255.255.255.255 any
access-list 123 permit ip 172.21.25.149 255.255.255.255 any
access-list 123 permit ip 172.21.25.150 255.255.255.255 any
access-list 123 permit ip 172.21.25.151 255.255.255.255 any
access-list 123 deny ip 172.21.25.152 255.255.255.255 any
access-list 123 deny ip 172.21.25.153 255.255.255.255 any
access-list 123 deny ip 172.21.25.156 255.255.255.255 any
access-list 123 deny ip 172.21.25.157 255.255.255.255 any
access-list 123 permit ip 172.21.25.154 255.255.255.255 any
access-list 123 permit ip 172.21.25.155 255.255.255.255 any
access-list 123 permit ip 172.21.25.158 255.255.255.255 any
```

Возможные варианты команд:

```
1.  access-list 123 permit ip 172.21.25.155 0.15.0.1         any
2.  access-list 123 deny   ip 172.21.25.152 0.0.0.110        any
3.  access-list 123 permit ip 172.21.25.145 0.0.0.34         any
4.  access-list 123 permit ip 172.21.25.154 0.0.1.250        any
5.  access-list 123 permit ip 172.21.25.170 0.0.0.8          any
6.  access-list 123 permit ip 172.21.25.144 0.0.0.7          any
7.  access-list 123 permit ip 172.21.25.0   0.0.7.27         any
8.  access-list 123 permit ip 172.21.25.0   255.255.255.240  any
9.  access-list 123 deny   ip 172.21.25.165 0.0.0.13         any
10. access-list 123 deny   ip 172.21.25.152 0.0.0.5          any
11. access-list 123 deny   ip 172.21.25.154 0.0.0.7          any
12. access-list 123 permit ip 172.21.25.154 0.0.0.5          any
```

Ответ: 6 10 12

Решение:

В условии указано, что межсетевой экран непосредственно подключен к сети с адресом 172.21.25.144 и маской 255.255.255.240 и список правил составляется **только** для реальных адресов указанной сети. Переведем этот адрес и маску в двоичную систему счисления:

Адрес сети: 10101100.00010101.00011001.10010000

Маска: 11111111.11111111.11111111.11110000

Поскольку маска содержит только 4 нуля, в указанной сети может быть только 2^4-2=14 реальных узлов (помним, что еще два адреса зарезервированы для служебного адреса сети и адреса ограниченного широковещания и не могут быть назначены реальным узлам). Тогда все адреса реальных узлов указанной сети будут иметь первые три октета 172.21.25, а в последнем октете входить в диапазон от 145 до 158, то есть в двоичном виде отличаться только значениями младших четырех бит.

Заметим, что в исходном наборе правил также ровно 14 правил и для каждого адреса из определенного выше диапазона индивидуально определено значение «разрешен» (permit) или «запрещен» (deny).

Возьмем исходный набор правил и отсортируем его так, чтобы сначала шли все разрешающие правила, а потом запрещающие и чтобы в каждой группе правила были отсортированы по возрастанию значения в последнем октете адреса. Также посчитаем значения младших 4-х бит этих адресов, поскольку, как было отмечено выше, только они могут различаться для указанной сети.

	3	2	1	0	Действие
145	0	0	0	1	permit
146	0	0	1	0	permit
147	0	0	1	1	permit
148	0	1	0	0	permit
149	0	1	0	1	permit
150	0	1	1	0	permit
151	0	1	1	1	permit
154	1	0	1	0	permit
155	1	0	1	1	permit
158	1	1	1	0	permit
152	1	0	0	0	deny
153	1	0	0	1	deny
156	1	1	0	0	deny
157	1	1	0	1	deny

Теперь построим для предлагаемых вариантов правил следующую таблицу: младшие 4 бита последнего октета адреса-образца, последние 4 бита wildcard mask и в последних 4-х столбцах отметим значения тех битов адреса-образца, которые фиксируются с помощью wildcard mask. Правило 8 уберем сразу, поскольку оно не содержит wildcard mask.

№ правила	Последний октет адреса-образца и его младшие 4 бита					Действие	Последний октет wildcard mask и его младшие 4 бита						Фиксированные значения младших 4-х бит			
		3	2	1	0			3	2	1	0					
1	155	1	0	1	1	permit	1	0	0	0	1		1	0	1	
2	152	1	0	0	0	deny	110	1	1	1	0					0
3	145	0	0	0	1	permit	34	0	0	1	0		0	0		1
4	154	1	0	1	0	permit	250	1	0	1	0			0		0
5	170	1	0	1	0	permit	8	1	0	0	0			0	1	0
6	144	0	0	0	0	permit	7	0	1	1	1		0			
7	0	0	0	0	0	permit	27	1	0	1	1			0		
9	165	0	1	0	1	deny	13	1	1	0	1				0	
10	152	1	0	0	0	deny	5	0	1	0	1		1		0	
11	154	1	0	1	0	deny	7	0	1	1	1		1			
12	154	1	0	1	0	permit	5	0	1	0	1		1		1	

Заметим, что в исходном наборе правил есть и разрешающие и запрещающие правила. Значит, минимально возможное количество правил в новом наборе равно двум. Для запрещающих правил из таблицы видно, что они охватывают все комбинации из 4-х бит, у которых 1-й бит (если считать слева направо от 1) равен 1, а 3-й бит равен 0. По второй таблице определим, что этому соответствует правило 10. Разрешающие правила можно разбить на две группы. Для первой группы 1-й бит равен 0, и в исходных правилах есть все возможные комбинации, кроме (0,0,0,0), но эта комбинация является служебным адресом данной сети и не может быть назначена в адресе реального узла. По второй таблице определим, что для этой группы подойдет правило номер 6. Осталось найти правило для второй группы разрешающих правил. Заметим, что в эту группу входят все комбинации, в которых 1-й и 3-й биты равны 1 (кроме (1,1,1,1), которая будет соответствовать адресу ограниченного

широковещания, и не может быть назначена реальному узлу). Такие комбинации соответствую правилу 12 из второй таблицы. Поскольку все три правила не имеют пересечений по адресам, их можно указывать в любом порядке, а, значит, в соответствии с условием их нужно указать в порядке возрастания. Тогда ответ будет 6 10 12.

7. Технологии обработки информации в электронных таблицах, сортировки и фильтрации данных (1 балл)

[Остатки]

Дан фрагмент электронной таблицы в режиме отображения формул:

	A	B	C	D	E	F	G	H	I	J
1		2	3	5	6	10	15	30		
2		=СЧЁТЕСЛИ(B3:B102;"0")								
3		=ОСТАТ($A3;B$1)							=СЧЁТЕСЛИ(B3:H3;"0")	
4										

Ячейку B2 скопировали во все ячейки диапазона C2:I2. Ячейку B3 скопировали во все ячейки диапазона B3:H102. Ячейку I3 скопировали во все ячейки диапазона I4:I102. Диапазон A3:A102 заполнили некоторым набором целых положительных чисел.

Известны значения некоторых ячеек, получившиеся после этого:

	A	B	C	D	E	F	G	H	I	J
1		2	3	5	6	10	15	30		
2		50	33	20	17	10	7	4		
3										
4										
5										

Определите значение в ячейке I2. В ответе укажите целое число.

Ответ: 27

Решение:

Обратим внимание на числа в диапазоне B1:H1. Сначала идут три простых числа (2, 3, 5), затем три пары их возможных произведений (2*3=6, 2*5=10, 3*5=15) и, наконец, произведение всех трех чисел (2*3*5=30).

Диапазон B3:B102 заполнен формулой =ОСТАТ($A3;B$1). То есть, в столбце B этого диапазона будут остатки от деления чисел из диапазона A3:A102 на 2, в столбце C – остатки от деления на 3, в столбце D – остатки от деления на 5 и т.д.

Тогда в ячейках диапазона B2:H2 будут числа, означающие, сколько чисел из диапазона A3:A102 делятся на 2, сколько на 3, сколько на 5, сколько на 6, сколько на 10, сколько на 15 и сколько на 30. Обратим внимание, что эти числа можно интерпретировать как мощности трех пересекающихся множеств: множество чисел из диапазона A3:A102, которые делятся нацело на 2, множество числе этого же диапазона, которые делятся нацело на 3 и множество чисел из этого диапазона, которые делятся нацело на 5. Формулы из диапазона I3:I102 подсчитывают для каждого числа количество делителей из набора {2, 3, 5}. Значение 0 в этом столбце будет только у тех чисел, которые не делятся ни на 2, ни на 3, ни на 5. Тогда в ячейке I2, значение которой нам нужно найти, будет количество таких чисел из диапазона A3:A102, которые не делятся ни на 2, ни на 3, ни на 5.

Посчитаем мощность объединения множеств, чисел из диапазона A3:A102, которые делятся на 2, 3 или 5. Поскольку мы знаем про все пересечения отдельных множеств, воспользуемся формулой включений-исключений и получим M=50+33+20-17-10-7+4=73. Всего в диапазоне A3:A102 ровно 100 чисел. Следовательно, 100-73=27 из них не делятся ни на 2, ни на 3, ни на 5. Значит, именно это число в ячейке I2.

8. Технологии программирования (2 балла)

[Биржа]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

В данной задаче вам требуется реализовать подсчет прибыли при торговле акциями.

Торги проходят в течении *n* дней, всего на рынке представлены акции m*m* компаний, цены акций компании фиксированы в течение одного дня. Сделки бывают двух типов:

Купить *x* акций компании *comp*

Продать все акции компании *comp*

За каждую сделку надо заплатить 1% комиссии. Например, если купить 10 акций по 300 рублей, то суммарно заплатить придется 3030 рублей. Если же продавать 10 акций стоимостью 300 рублей каждая, то за них можно получить 2970 рублей.

Прибылью с продажи будем считать разность полученных при продаже денег и суммарно потраченных денег при покупках. Например, если 10 акций были куплены по 300 рублей, а затем еще 5 акций были куплены по 400 рублей, то в случае продажи по стоимости 500 прибыль составит: $15 \cdot 500 \cdot 0.99 - (10 \cdot 300 \cdot 1.01 + 5 \cdot 400 \cdot 1.01) = 7425 - (3030 + 2020) = 2375$ рублей. При

этом акции могут быть проданы в убыток (за меньшую стоимость, чем были куплены), тогда прибыль с продажи будем считать отрицательной.

Суммарной прибылью в момент времени будем считать сумму всех прибылей с продаж, которые произошли до этого момента. Обратите внимание, что деньги, потраченные к этому моменту на покупку акций, которые еще не были проданы, не учитываются. Также будем считать, что у нас неограниченный бюджет для покупки акций, но рассматривать будем только прибыль. Число акций для покупки/продажи для каждой компании на рынке также не ограничено.

Вам даны k событий покупки/продажи. Необходимо найти минимальную суммарную прибыль среди всех моментов времени.

Формат входных данных

В первой строке входных данных содержится одно целое число t — число тестовых наборов ($1 \le t \le 30$).

Затем следуют t тестовых наборов. Каждый тестовый набор описывается следующим образом:

В первой строке тестового набора содержатся три целых числа n, m и k — число дней, в которые проходят торги, число компаний на рынке и число событий, соответственно ($1 \le n, m \le 100$, $1 \le k \le 1000$).

В следующих m строках записаны названия компаний и n чисел — стоимости акций компании в рублях в каждый из дней торгов. Названия компаний состоят из не более чем 10 строчных букв латинского алфавита и попарно различны. Стоимости акций — целые числа в диапазоне от 1 до 10^5 включительно.

В следующих k строках заданы события покупки/продажи в хронологическом порядке. Событие покупки задается в формате **<день> buy <число акций> <название компании>**, а событие продажи задается в формате **<день> sell <название компании>**. При этом <день> — целое число от 1 до n, а <число акций> — целое число от 1 до 1000. Гарантируется, что все события следуют в порядке неубывания дней и корректны, а именно нет продаж некупленных акций и покупок акций, которых нет на рынке.

Формат выходных данных

Для каждого тестового набора выведите в отдельной строке минимальную прибыль среди всех моментов времени, с относительной или абсолютной погрешностью не более 10^{-4}.

Пример

Стандартный ввод	**Стандартный вывод**
4 3 1 3 comp 300 400 500 1 buy 10 comp 2 buy 5 comp 3 sell comp 3 2 4 gazp 100 111 300 yndx 1000 1100 1111 1 buy 10 gazp 2 buy 1 yndx 3 sell yndx 3 sell gazp 3 1 3 comp 300 400 200 1 buy 10 comp 2 buy 5 comp 3 sell comp 2 2 3 bdn 100 100 nik 1 100 1 buy 300 bdn 1 buy 10 nik 2 sell nik	0 -11.11 -2080.0 0

Замечание

В первом тестовом наборе изначально до продаж суммарная прибыль равна 0, после первой продаже суммарная прибыль становится 2375 (случай разобран в примере).

Во втором тестовом наборе промежуточные прибыли равны 0, −11.11 (акция продана дороже, но комиссия больше разницы) и 1948.89.

В третьем тестовом наборе промежуточные прибыли равны 0 и −2080.

В четвертом тестовом наборе промежуточные прибыли равны 0 и 979.9, деньги, потраченные на непроданные акции, не учитываются.

Решение:

Для решения этой задачи необходимо написать ровно то, что просят в условии. Наиболее удобно это можно сделать, если воспользоваться ассоциативным массивом, он же std::map в C++ или dict в Python.

Тогда, обрабатывая события, будем для каждой компании поддерживать сколько денег мы потратили на покупку акций и сколько у нас акций этой компании есть в текущий момент. Также надо не забыть домножить на 1.01, чтобы учесть комиссию.

При обработке события продажи будем умножать количество имеющихся акций на текущую цену и на 0.99, потому что комиссия будет идти уже в минус. После чего добавить к текущей прибыли, разность полученных денег потраченных денег, которые мы как раз храним в ассоциативном массиве.

Соответственно далее остается только найти минимум/максимум «промежуточной» прибыли.

С примерами авторских решений можно ознакомиться в папке solutions в архиве задачи.

9. Технологии программирования (4 балла) [Трансформация массива]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

Вам дано t пар массивов a_i и b_i равной длины.

За одну операцию модификации можно:

Поменять местами любые два элемента массива a_i, но каждый элемент массива может участвовать не более чем в одном обмене.

Прибавить к любому элементу массива a_i единицу. Данную операцию можно применять неограниченное число раз к любому элементу массива.

Для каждой пары массивов найдите минимальное число операций, которые необходимо применить к массиву a_i, чтобы получить массив b_i, или определите, что это невозможно.

Формат входных данных

В первой строке дано число t — число пар массивов ($1 \le t \le 40$).

В следующих $3t$ строках содержатся описания пар массивов. Каждая пара описывается тремя строками.

В первой из них дано число n_i — количество элементов в каждом массиве i-й пары ($1 \le n_i \le 10$). Во второй строке заданы n_i чисел $a_{i,j}$ — элементы массива a_i ($1 \le a_{i,j} \le 1000$). В третьей строке заданы n_i чисел $b_{i,j}$ — элементы массива b_i ($1 \le b_{i,j} \le 1000$).

Гарантируется, что сумма n_i по всем тестовым наборам не превосходит 150.

Формат выходных данных

Для каждого пары массивов выведите одно число — минимальное число операций, которые необходимо применить к массиву a_i, чтобы получить массив $b_{i,j}$, или -1, если для данной пары это невозможно.

Пример

Стандартный ввод	Стандартный вывод
6 1 2 1 4 1 2 3 4 2 4 1 2 1 1 2 5 1 2 3 4 5 1 5 1 1 1 5 1 2 3 4 5 5 1 5 5 5 3 1 2 3 3 1 2	-1 -1 1 -1 7 -1

Решение

Данную задачу можно решать двумя способами: динамическим программированием или жадным подходом.

Для начала заметим, что если элемент массива a_i не участвует в обмене, то мы однозначно знаем сколько нужно операций, чтобы получить соответствующий элемент b_i (или что это невозможно). Также заметим, что после конкретного обмена мы также знаем сколько нужно операций изменения элемента.

Тогда, если решать динамическим программированием, можно сказать dp[mask] — минимальное число операций, чтобы все элементы массива a_i с индексами соответствующими битовой маске mask, стали соответствующими элементами массива b_i. Решение очень похоже на поиск паросочетания минимального веса, в целом, тут и правда спрятан граф.

Для жадного решения нужно было доказать/поверить в то, что обмены довольно дешевые и их всегда выгодно делать, а затем перебирая элементы массива a_i в порядке возрастания находить оптимальную пару для обмена.

С примерами авторских решений можно ознакомиться в папке solutions в архиве задачи.