Заключительный этап для 10 класса (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (2 балла)

[Много нулей]

Дано арифметическое выражение:

$$11_{16} \cdot (20_{16}^{N} + 1)^2 = X_2$$

Найдите минимальное целое положительное значение N, при котором в двоичной записи числа X будет более 1000 значащих нулей. Постройте двоичную запись числа X для найденного значения N и найдите в ней последовательность из идущих подряд нулей максимальной длины. Укажите в ответе длину найденной последовательности в виде числа в десятичной системе счисления.

Ответ: 501

Решение:

Обозначим за $\{0\}_K$ последовательность из K идущих подряд нулей. Тогда $20_{16}^N + 1 = 100000_2^N + 1$ можно записать как $1\{0\}_{(5N-1)}1$. Возведение этого числа в квадрат равносильно умножению его само на себя: $1\{0\}_{(5N-1)}1$ * $1\{0\}_{(5N-1)}1$ = $1\{0\}_{(5N-1)}1$ * $1\{0\}_{(5N)}$ + $1\{0\}_{(5N-1)}1$ = $1\{0\}_{(5N-1)}1\{0\}_{(5N)}$ + $1\{0\}_{(5N-1)}1$ = $1\{0\}_{(5N-2)}1\{0\}_{(5N)}1$. Для получения значения X нужно еще умножить на $11_{16}=10001_2$. $1\{0\}_{(5N-2)}1\{0\}_{(5N)}1$ * 10001 = $1\{0\}_{(5N-2)}1\{0\}_{(5N)}10000$ + $1\{0\}_{(5N-2)}1\{0\}_{(5N)}1$. Для любого N>1 в результате получится: $10001\{0\}_{(5N-6)}10001\{0\}_{(5N-4)}1001$. Обратим внимание, что общее количество значащих нулей в записи будет 3+(5N-6)+3+(5N-4)+3 = 10N-1, а максимальное количество идущих подряд нулей будет 5N-4. Тогда минимальным N, при котором общее число нулей в записи числа X будет более 1000 является N=101. Следовательно, длина максимальной последовательности идущих подряд нулей будет равна 5*101-4=501

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

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

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 (3 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.

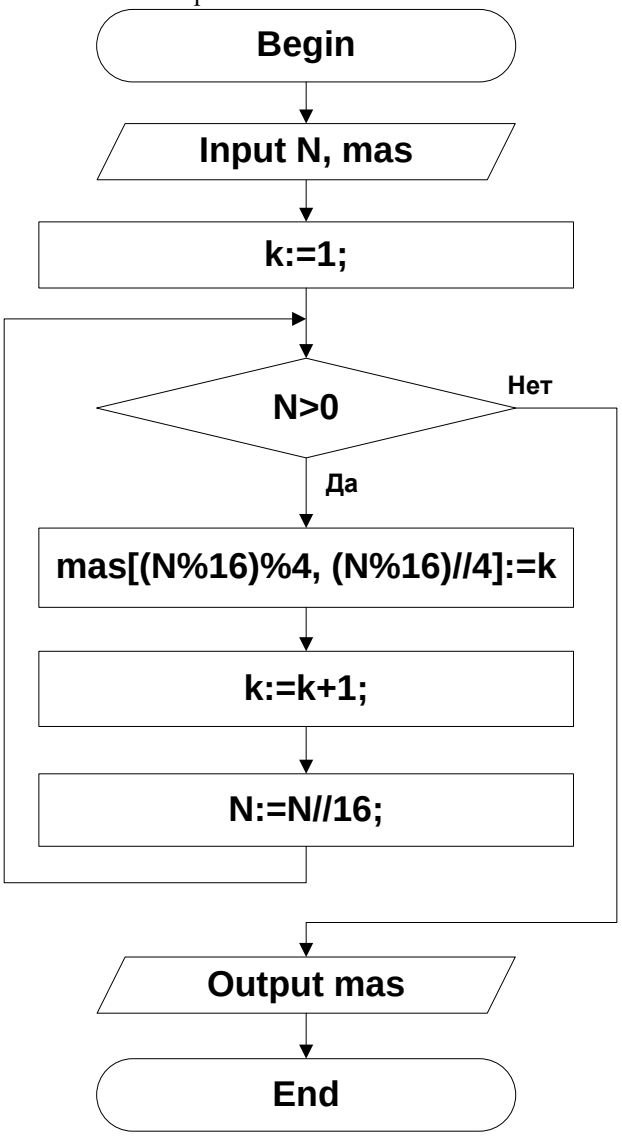
```
Path='RRRL'
ans=0
for x in range(10):
    for y in range(10):
        A=[]
        A.append(list("*.********"))
        A.append(list("*........*"))
        A.append(list("*.*.**.*.*"))
        A.append(list("*.*....*.*"))
        A.append(list("*.*.**.*.*"))
        A.append(list("*.*.**.*.*"))
        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)

[4*4=16]

Дана блок схема алгоритма:



На вход алгоритма подали целое положительное число N и целочисленную матрицу mas, размером 4 на 4 элемента, заполненную нулями. На выходе получили следующую матрицу mas:

$$\begin{bmatrix} 14 & 9 & 4 & 7 \\ 13 & 5 & 2 & 1 \\ 6 & 10 & 8 & 16 \\ 15 & 12 & 3 & 11 \end{bmatrix}$$

Определите минимальное число, N для которого это возможно. Представьте запись числа N в двоичной системе счисления, сложите те цифры этой записи, номер которых (нумеруя цифры с 0 от младшего разряда к старшему) делится нацело на 3 и запишите результат в десятичной системе счисления. Например, если $N=10101001_2$, ответом будет 2.

Примечания. Операция A%B вычисляет остаток от целочисленного деления A на B; операция A//B вычисляет частное от целочисленного деления A на B. При обращении к элементам матрицы первый индекс – номер строки, а второй – номер столбца, нумерация с (0,0).

Ответ: 12

Решение:

Заметим, что на каждом шаге цикла происходит обращение к значению N%16, а потом деление N//16. Это не что иное как разбор на отдельные цифры шестнадцатеричной записи числа N. То есть, на каждом шаге цикла берется цифра с номером k (считая справа налево от 1) в записи числа в шестнадцатеричной системе счисления. Вспомним, что любая шестнадцатеричная цифра может быть представлена как пара цифр для записи в четверичной системе счисления, например $D_{16}=31_4$. Тогда (N%16)%4 – это младшая (вторая) цифра в такой паре, а (N%16)//4 – старшая (первая) цифра в такой паре. Тогда операция mas[(N%16)%4, (N%16)//4]:=k означает, что в элемент матрицы, номер строки которого соответствует младшей цифре такой пары, а номер столбца – старшей цифры записывается номер соответствующей шестнадцатеричной цифры. Значит, мы можем брать последовательно элементы матрицы со значениями от 1 до 16, формировать для них пары (номер столбца, номер строки) и записывать такие пары справа налево (заметим, что нумерация идет от нуля):

32030001133312102230021120232131

Пользуясь тем, что любая четверичная цифра может быть представлена независимо как две двоичных, переведем в двоичную систему счисления и выделим разряды, номера которых (при нумерации с 0 от младшего к старшему разряду) делятся нацело на 3:

1110001100000001011111101100100101011000010010110001011100111101

Посчитаем количество выделенных единиц – их будет ровно 12, что и является ответом на задание.

6. Телекоммуникационные технологии (2 балла). [Три порта]

Петя моделирует работу прототипа коммутатора. У этого прототипа есть три порта. Два порта принимают данные из двух сетей со скоростью 5 КБайт в секунду каждый. Третий порт отдает данные в третью сеть со скоростью 10 КБайт в секунду. Данные из первой сети приходят пакетами по 40 КБайт, причем первый пакет начинает приниматься в начальный момент времени и далее новый пакет начинает приниматься по истечении каждых 10 секунд. Данные из второй сети приходят пакетами по 30 КБайт, причем первый пакет начинает приниматься по истечении 4 секунд от начального момента времени и далее новый пакет начинает приниматься по истечении каждых 8 секунд.

Прототип коммутатора работает следующим образом. Поступающий пакет по мере передачи данных записывается в буферную память. Если одновременно принимаются пакеты в оба порта их запись в буферную память происходит одновременно и независимо. Как только некоторый пакет целиком принят, он становится готов к передаче через выходной порт. Передача данных через выходной порт происходит исходно полученными пакетами по 40 и по 30 КБайт соответственно, причем, если в некоторый момент времени есть возможность начать передать пакеты обоих размеров, сначала передаётся пакет большего размера. В один момент времени через выходной порт может передаваться только один пакет и пакет всегда передается целиком. Передача пакета через выходной порт начинается сразу, как только появляется готовый пакет, если в этот момент не передается другой пакет или сразу после завершения передачи предыдущего пакета. Как только очередной пакет передан, он удаляется из буфера. Если в один момент времени нужно удалить переданный пакет и записать очередную порцию данных принимаемого пакета, сначала происходит удаление. Если вся память в буфере занята, но происходит попытка принять данные, происходит переполнение буфера и аварийная остановка коммутатора. Определите минимальный размер буфера в КБайтах, при котором в процессе работы данной модели никогда не произойдет аварийная остановка. В ответе укажите целое число.

Ответ: 90

Решение:

Вручную или с помощью электронной таблицы построим модель работы прототипа коммутатора. Одна строка означает состояние модели по истечение очередной секунды. В столбцах «Порт 1» и «Порт 2» отображается объем данных в КБайт, полученных через этот порт к очередному моменту времени. В столбце «Выход» показаны в виде отрицательных чисел данные, удаленные из буфера к очередному моменту времени. В столбце «Буфер» считается объем данных, хранимых в буфере в указанный момент. Желтые клетки означают секунду, на которой заканчивается приём очередного пакета. Со следующей секунды возможна его передача. Зеленые клетки означают секунды, на которых передаётся в третью сеть пакет, размером, 40 КБайт, а голубые – размером 30 КБайт.

Секунда	Порт 1	Порт 2	Выход	Буфер
1	5			5
2	10			10
3	15			15
4	20			20
5	25	5		30
6	30	10		40
7	35	15		50
8	40	20		60
9	40	25		65
10	40	30		70
11	45	30		75
12	50	30		80
13	55	35	-40	50
14	60	40	-40	60
15	65	45	-40	70
16	70	50	-70	50
17	75	55	-70	60
18	80	60	-70	70
19	80	60	-70	70
20	80	60	-70	70
21	85	65	-70	80

22	90	70	-70	90
23	95	75	-110	60
24	100	80	-110	70
25	105	85	-110	80
26	110	90	-140	60
27	115	90	-140	65
28	120	90	-140	70
29	120	95	-140	75

Обратим внимание, что по истечении 22-ой секунды от начального момента времени в буфере занято 90 КБайт. При этом, уже в начале следующей секунды из буфера будет удалено 40 КБайт, а поступит 10, причем по условию удаление будет предшествовать заполнению. Легко заметить, что достижение занятости в буфере 90 КБайт происходит в худшем случае, когда одновременно завершен на 18-й секунде прием сразу двух пакетов. В этом случае будет происходить накопление данных в буфере во время передачи большего пакета при наличии в памяти полного пакета меньшего размера. В принципе, можно продолжить таблицу и убедиться, что вторая такая же ситуация с одновременным завершением приёма двух пакетов произойдет по истечению 56-й секунды и, следовательно, занятость буфера в 90 КБайт следующий раз будет достигнута на 60-й секунде. Следовательно, процесс изменения памяти буфера будет цикличен и значение 90 КБайт никогда не будет превышена. Значит это и есть максимальный объем буфера, при котором будет неограниченная во времени безаварийная работа.

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 mm 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.