I. Задания заключительного этапа олимпиады 2020-21 года

Заключительный этап 11 класса (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (2 балла) [Периодическая система]

Дана система уравнений, неизвестными в которой являются основания двух позиционных систем счисления X и Y:

$$\begin{cases} 0,(34)_X + 0,(13)_Y = 1 \\ 2 * 101_X - 102_Y = 1 \end{cases}$$

где левая часть первого уравнения представляет собой сумму двух периодических дробей.

Найдите и запишите в ответ через пробел сначала значение X, а затем значение Y, удовлетворяющие этой системе уравнений. Если таких пар несколько, запишите пару с наименьшим значением X. Если такой пары не существует, запишите в ответ NULL.

Ответ: 5 7

Решение:

Определим область допустимых значений: $x \geq 5; y \geq 4$

Перепишем первое уравнение, используя развернутую запись чисел в системах счисления с основаниями X и Y:

$$\frac{3x+4}{x^2-1} - \frac{y+3}{y^2-1} = 1$$

Обратим внимание на знаменатели.

Перепишем второе уравнение, используя развернутую запись чисел в системах счисления с основаниями X и Y:

$$2(x^2+1) - (y^2+2) = 1$$

Преобразуем уравнение так, чтобы была возможность заменой удобно привести первое уравнение к общему знаменателю:

$$2x^2 + 2 - y^2 - 2 = 1$$
$$2x^2 - 1 = y^2$$
$$2(x^2-1) = y^2 - 1$$

Подставляем в первое уравнение, приводим к общему знаменателю:

$$(3x+4) \cdot 2 + \sqrt{2x^2-1} + 3 = 2 \cdot (x^2-1)$$

Решим уравнение, понимая, что нас интересует только целый положительный корень, удовлетворяющий ОДЗ. Он единственный: $x = 5$. Соответственно, находим $y = 7$ и получаем ответ «5 7».

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

Петя решил построить генератор случайных последовательностей символов. Генератор состоит из трех источников символов, общего буфера и таймера. Таймер отсчитывает секунды, начиная с 1. Источник 1 каждую секунду записывает в буфер один символ, по кругу выбирая символы из набора {A, B, C}. Источник 2 каждую секунду, кратную 2, записывает в буфер один символ A. Источник 3 каждую секунду, кратную 3, записывает в буфер один символ, по кругу выбирая символы из набора {A, B}. Если в некоторую секунду несколько источников должны записать в буфер свои символы, они всегда делают это по порядку возрастания их номеров.

Вот результаты работы генератора на первых 6 секундах:

1. A
2. ABA
3. ABACA
4. ABACAAA
5. ABACAAAB
6. ABACAAABCAB

Петя настолько уверен в надежности своего генератора, что, сформировав в буфере строку из N символов, решил записать её в память как последовательность двоичных кодов отдельных символов, используя для кода каждого символа одинаковое минимально возможное количество бит. Вася проанализировал работу генератора Пети и убедился, что он не такой уж надежный. Вася увидел, что длина строки Пети N кратна трем, и разбил строку в буфере на тройки идущих подряд символов, начиная с начала строки. Он посчитал, сколько всего различных троек символов встречается в строке, и решил записать в память строку в виде последовательности двоичных кодов троек символов, используя для кода каждой тройки одинаковое, минимально возможное количество бит. Вася обнаружил, что ему потребовалось для записи в память строки на 40 бит меньше, чем Пете. Определите и запишите в ответ длину строки N.

Ответ:60

Решение:

Проанализируем построение строк на первых 6 секундах работы генератора и построим строку, которая получится на 7-ой секунде. В скобках указано, какие источники и какие символы добавляют на соответствующем шаге.

1. A (Источник 1, A)
2. ABA (Источник 1, B; Источник 2, A)
3. ABACA (Источник 1, C; Источник 3, A)
4. ABACAAA (Источник 1, A; Источник 2, A)
5. ABACAAAB (Источник 1, B)
6. ABACAAABCAB (Источник 1, C; Источник 2, A; Источник 3, B)

7. ABACAAABCABA (Источник 1, A)

Обратим внимание, что на секунде 7 повторилось состояние всех источников: источник 1 добавляет символ A, источник 2 готов на следующей (четной) секунде добавить символ A и источник 3 готов через секунду (на секунде 9) добавить символ A. То есть, начинается такой же цикл из 6 секунд, который прибавит в конец строки, полученной на 6-ой секунде её копию еще через 6 секунд.

Обратим внимание, что длина строки на 6-ой секунде была 11 символов. То, есть, если мы повторим цикл из 6 секунд 3 раза, на 18-ой секунде получится строка из трех одинаковых идущих друг за другом подстрок, длиной 33 символа (кратно трем). А затем все будет повторяться. Следовательно, все возможные тройки символов содержатся в этой строке из 33 символов. Получим её, склеив три копии строки, полученной на 6 секунде, и выделим тройки:

ABACAAABCABABACAAABCABABACAAABCAB

Обратим внимание, что тройка ABA повторяется два раза, следовательно, всего в этом фрагменте строки, а следовательно и в любой строке большей длины (по условию кратной трем), могут встретиться только 10 различных троек.

Теперь посчитаем затраты памяти. В строке могут встречаться только 3 различных символа, следовательно, при равномерном кодировании, которое требуется по условию задачи, Петя будет тратить на код каждого символа 2 бита и, следовательно, его строка будет занимать в памяти 2*N бит. Поскольку в строке может быть только 10 различных троек, на код одной тройки Вася будет тратить 4 бита. Тогда, строка Васи будет занимать в памяти 4*(N/3) бит, то есть, 4/3*N бит. Легко составить уравнение:

2*N – 4/3*N = 40 и получить N=60, что и будет правильным ответом.

3. Основы логики (1 балл) [Эквиваленции]

Известно следующее равенство:

$$\left((\bar{B} \to \bar{A}) \to \bar{B}\right) \vee (\bar{A} \to \bar{C}) \wedge \bar{C} \vee \overline{A \to D} = \text{ложь}$$

Тогда для каких из перечисленных логических выражений можно однозначно определить их логическое значение (истинность или ложность)? Отметьте **все** подходящие выражения.

1. $A \wedge B \leftrightarrow C \wedge D$
2. $(A \to B) \leftrightarrow (C \to D)$
3. $A \leftrightarrow B \wedge C \wedge D$
4. $A \wedge C \leftrightarrow B \wedge D$
5. $(A \to C) \leftrightarrow (B \to D)$
6. $B \leftrightarrow A \wedge C \wedge D$
7. $B \wedge C \leftrightarrow A \wedge D$
8. $C \leftrightarrow A \wedge B \wedge D$
9. $D \leftrightarrow A \wedge B \wedge C$
10. $(D \to C) \leftrightarrow (A \to B)$

Ответ: 6, 7, 8, 10

Решение:

Упростим выражение в левой части равенства и получим:

$$\boldsymbol{\bar{B} \vee \bar{C} \vee A \wedge D} = \text{ложь}$$

Следовательно, B может принимать значение только «истина», C – также может принимать значение только «истина», а для A и D существует 3 возможных комбинации: A = «ложь», D = «ложь»; A = «ложь», D = «истина»; A = «истина», D = «ложь».

Для удобства можно составить фрагмент таблицы истинности, включающий те комбинации значений переменных A, B, C и D, при которых равенство будет выполняться:

A	B	C	D
0	1	1	0
0	1	1	1
1	1	1	0

Рассмотрим каждую из предложенных эквиваленций:

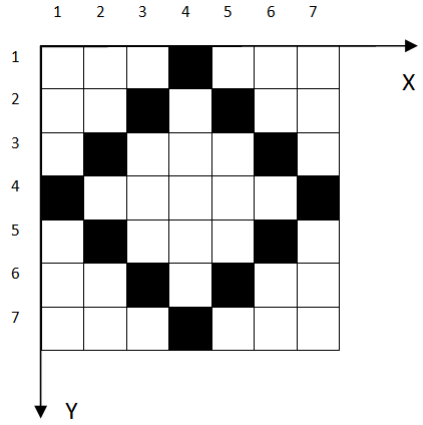
1. Для первой эквиваленции в случае первой комбинации (0110) и левая и правая части будут ложными, следовательно, эквиваленци будет истинна. Для второй строки левая часть эквиваленции будет ложна, а правая истинна, то есть эквиваленция будет ложна. Следовательно, для этого варианта невозможно однозначно установить истинность или ложность.
2. Аналогично вторая эквиваленция будет принимать ложное значение для первой комбинации и истинное для второй.
3. Третья эквиваленция будет принимать истинное значение для первой комбинации и ложное для второй комбинации.
4. Четвертая эквиваленция будет принимать истинное значение для первой комбинации и ложное для второй.
5. Пятая эквиваленция будет принимать ложное значение для первой комбинации и истинное для второй.

6. Для шестой эквиваленции левая часть всегда будет истинна, а правая всегда ложна. Значит мы можем однозначно определить логическое значение шестого выражения – оно будет ложно при всех комбинациях значений переменных, которые мы вывели из исходного логического равенства.
7. Аналогично в седьмой эквиваленции левая часть всегда будет истина, а правая ложна. Следовательно логическое значение этого выражения мы также можем однозначно определить.
8. В восьмой эквиваленции левая часть всегда истина, а правая всегда ложна. Значит и для этого выражения мы можем однозначно определить значение.
9. Девятая эквиваленция будет истинна для первой комбинации и истинна для второй. Не подходит.
10. Последняя, десятая эквиваленция. Левая часть для всех трех комбинаций будет истинна, поскольку в импликации справа истинное значение. Аналогично правая часть также во всех трех случаях будет истинной. Следовательно мы можем однозначно определить значение этого выражения – оно будет истинным.

Таким образом, мы определили, что только для 6, 7 8 и 10 выражений мы можем однозначно определить их истинность исходя из исходного равенства.

4. Алгоритмизация и программирование. Формальные исполнители (3 балла) [Рекурсивное закрашивание]

Петя очень давно участвует в олимпиадах и знает, что есть алгоритм рекурсивного закрашивания растрового изображения. У Пети есть множество черно-белых (bitmap) изображений, размером N на N пикселей, где N – нечетное число. На каждом изображении присутствует контур, имеющий форму ромба, центр которого совпадает с центром изображения, а вершины расположены в центрах первой и последней строки, и первого и последнего столбца изображения, соответственно. Линии имеют толщину в один пиксель. Вот пример такого изображения, размером, 7 на 7 пикселей:



Традиционно для компьютерной графики, система координат имеет начало в верхнем левому углу (начальный пиксель имеет координаты (1,1)) и направления осей как показано на рисунке.

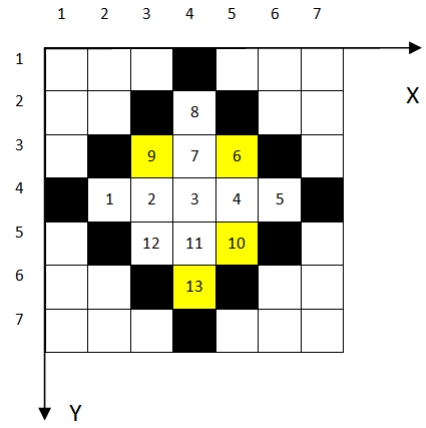
Алгоритм рекурсивного закрашивания заключается в рекурсивном вызове процедуры «Закрасить», которой передаются два параметра – координаты X и Y пикселя.

Процедура *Закрасить(X,Y)*, может быть описана следующим образом:

1. Если цвет пикселя с координатами (X,Y) белый, то:
    a. Изменить цвет пикселя с этими координатами на черный;
    b. Вызвать процедуру *Закрасить(X+1,Y);*
    c. Вызвать процедуру *Закрасить(X,Y-1);*
    d. Вызвать процедуру *Закрасить(X-1,Y);*
    e. Вызвать процедуру *Закрасить(X,Y+1);*
2. Иначе завершить процедуру.

Будем называть прыжком ситуацию, когда следующий закрашиваемый пиксель не находится в четырехсвязной окрестности предыдущего пикселя, то есть не является пикселем, прилегающим слева, справа, сверху или снизу к нему.

Петя решил закрашивать своё изображение, осуществив исходный вызов процедуры Закрасить(2,4) и проанализировал порядок, в котором закрашивались пиксели, выделив пиксели, закрашивание которых произошло в результате прыжков:



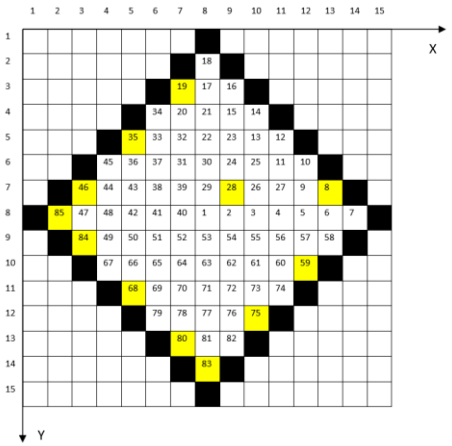
Пете стало интересно, и он решил узнать, сколько пикселей закрасится в результате прыжков, если взять изображение, размером 15 на 15 пикселей и осуществить исходный вызов процедуры закрашивания в пикселе, являющимся центром изображения. Определите их количество и запишите в ответ целое число.

Ответ: 12

Решение:

Для того, чтобы определить количество «прыжков», необходимо аналогично примеру проанализировать порядок закрашивания пикселей. Это можно сделать вручную или написав несложную реализацию определенного в условии рекурсивного алгоритма. Например, так:

```
K=1 #Счетчик закрашиваемых пикселей
def plot(x,y):
    if S[x][y]==0:
        global K
        S[x][y]=K
        K+=1
        plot(x,y+1)
        plot(x-1,y)
        plot(x,y-1)
        plot(x+1,y)
N=15
S=[]
for i in range(N):
    S.append([])
    for j in range(N):
        S[i].append(0)
for i in range(N//2+1): #Цикл отрисовки границ закрашиваемой области
    S[N//2+i][i]=-1
    S[N//2-i][i]=-1
    S[i][N//2+i]=-1
    S[N-i-1][N//2+i]=-1
plot(N//2,N//2)
```

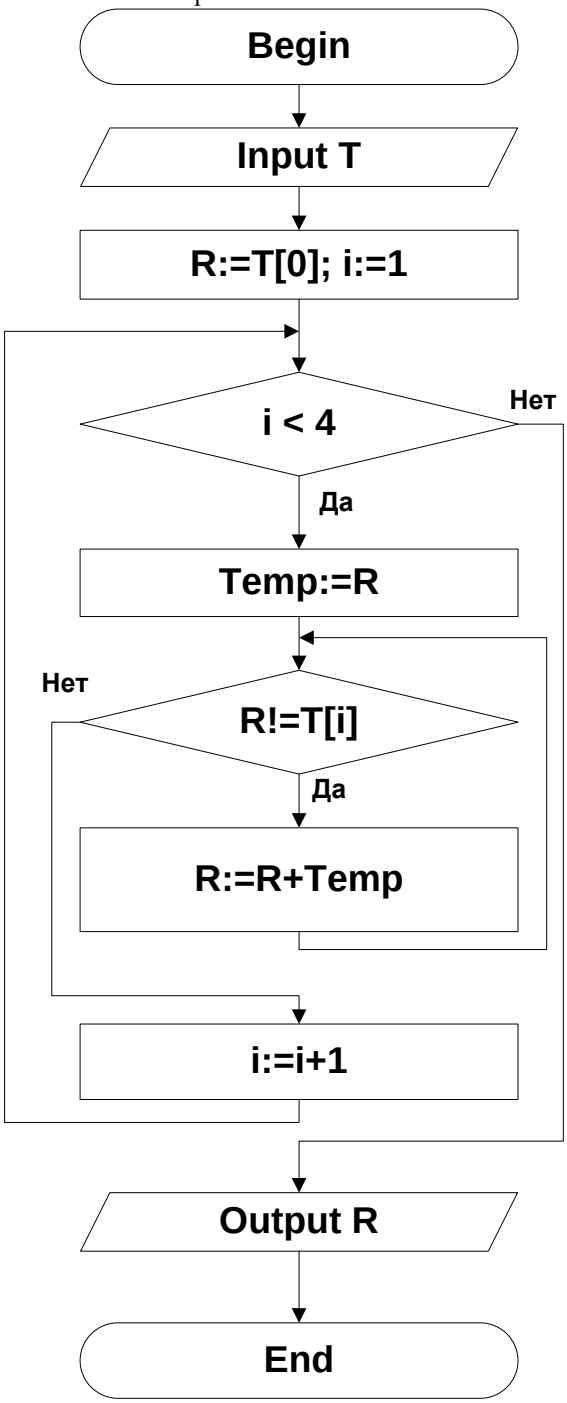


На приведенном рисунке аналогично примеру показан порядок закрашивания пикселей и выделены пиксели, которые были закрашены в результате прыжка. Легко убедиться, что их ровно 12.

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

[Пороги]

Дана блок-схема алгоритма:



На вход подается массив из четырех целых положительных чисел T=[7, X, Y, Z]. Все элементы массива попарно различны. Известно, что в результате выполнения алгоритма было выведено число 23023. Определите неизвестные значения элементов входного массива и укажите в ответе через пробел значения первых двух из них: сначала значение X и затем значение Y. Если существует несколько подходящих вариантов, укажите такой вариант, в котором сумма X и Y будет минимальна. Если такого массива не существует, укажите в ответе NULL.

Примечание: нумерация элементов массива идет с нуля, оператор «!=» означает «неравно».

Ответ: 77 1001

Решение:

Проанализируем алгоритм.

На первом шаге внешнего цикла значение переменной Temp приравнивается к значению переменной R, которая ранее была приравнена к первому элементу массива T, то есть равна 7. Во вложенном цикле это значение будет последовательно суммироваться с переменной R, пока её значение не станет равно первому элементу искомого массива. Таким образом, первый элемент массива равен произведению 7 на некоторое целое число. Обозначим его за A. То есть, первый элемент массива равен 7*A. Затем, на втором шаге внешнего цикла, значение переменной Temp приравнивается к текущему значению переменной R, то есть также становится равно 7*A, и это значение суммируется с переменной R до тех пор, пока результат не станет равен второму элементу массива. Следовательно, второй элемент массива равен произведению 7*A на некоторое целое число.

Обозначим его за B. Таким образом, второй элемент массива равен 7*A*B. После этого, на третьем шаге внешнего цикла, значение переменной Temp вновь приравнивается к текущему значению переменной R, то есть также становится равно 7*A*B, и это значение суммируется с переменной R до тех пор, пока результат не станет равен третьему элементу массива. Следовательно, третий элемент массива равен произведению 7*A*B на некоторое целое число. Обозначим его за C. Таким образом, третий элемент массива равен 7*A*B*C. Обратим внимание, что третий элемент массива равен значению, которое будет выведено в результате исполнения алгоритма, поскольку его значение является условием выхода из цикла.

Следовательно, выведенное в результате число 23023=7*A*B*C. Для того, чтобы найти эти значения, разложим число 23023 на простые сомножители. Число 23023=7*11*13*23 является произведением четырех простых чисел. Следовательно, исходя из требования, что все элементы массива T должны быть различными, а, следовательно, значения A, B и C не могут быть равны 1, значения A, B и C соответствуют значениям 11, 13 и 23 с точностью до перестановки. Обратим внимание, что для того, чтобы сумма первых элементов массива была минимальной, значения A, B и C нужно взять такими, чтобы они были расположены по возрастанию. Следовательно, A=11, B=13, C=23. Тогда T[0]=7*11=77, T[1]=7*11*13=1001. Заметим, что тогда T[2]=7*11*13*23=23023. Таким образом, ответ будет «77 1001».

6. Телекоммуникационные технологии (2 балла). [EUI-64]

В протоколе IPv6 адрес представляет собой 128 битную двоичную последовательность, записываемую в шестнадцатеричном формате, группами по 4 цифры. Группы разделяются двоеточиями.

В адресе IPv6 есть две логические части:

1) первая часть, называемая префиксом, содержит адрес сети. По умолчанию она начинается с первого (начального) бита адреса и заканчивается на 64-м бите. Однако длина префикса может меняться. Длину префикса в битах указывают вместе с адресом, в виде десятичного числа, записанного через символ “/”
2) вторая часть, начинающаяся сразу после префикса – адрес узла (например, сетевой карты компьютера).

Для удобства IPv6 адрес можно записывать сокращенно для чего:

1) не пишутся ведущие нули в каждой группе
2) не указываются группы, содержащие только «0». Но это делается только для одной последовательности нулевых групп с конца адреса.

Пример:
для адреса
2001:0000:**00**AF:ABCD:**0000:0000:0000**:1234 / 64
сокращенная запись будет:
2001:0000:AF:ABCD::1234 / 64
В примере число бит «/64» показывает, что граница префикса проходит по середине адреса, а жирным выделены сокращенные при записи участки.

Часть IPv6 адреса, соответствующая адресу узла редко назначается вручную. Стандартным механизмом формирования адреса служит алгоритм EUI-64 при котором узлу сообщается префикс адреса, а вторую часть адреса узел генерирует сам на основании MAC адреса.

MAC адрес – это аппаратный адрес сетевого устройства, состоящий из 48 бит, записываемых в шестнадцатеричной форме. Адрес имеет свою структуру: первые 24 бита идентифицируют производителя, последние 24 бита назначаются производителем для конкретного устройства. MAC адреса не сокращаются.

MAC адрес, как правило может быть изменен пользователем, но, по стандарту, 7-й с начала адреса бит не может быть выбран произвольно. Действует правило:

1) Если адрес назначен на заводе, то 7-й бит равен 0,
2) если адрес установлен вручную, то 7-й бит равен 1.

Алгоритм EUI-64 сводится к следующим шагам:

1) MAC-адрес делится на две равные части;
2) Между частями вставляется комбинация "FFFE" (так что бы получилось 64-битное значение);
3) Инвертируется 7-й с начала бит получившейся комбинации.

Пусть в консоли маршрутизатора CISCO были набрана команда формирования IPv6 адреса с указанным префиксом по алгоритму EUI-64:

```
INT1(config-if)#ipv6 address 2001:1234:5678:abcd::/64 eui-64
```

После завершения работы команды была выполнена команда получения адресной информации по протоколу IPv6 и получен консольный вывод.

```
INT1#show ipv6 interface fa0/0
FastEthernet0/0 is up, line protocol is up
Global unicast address(es):
2001:1234:5678:ABCD:C000:FF:FE5C:1, subnet is
2001:1234:5678:ABCD::/64 [EUI]
```

Определите исходный MAC адрес конфигурируемого интерфейса и то, как был установлен адрес: установлен вручную или оставлен по умолчанию заводской адрес.

В ответ укажите MAC адрес в шестнадцатеричном виде и, через пробел букву L – в случае, если адрес назначен вручную и G, если адрес оставлен заводской адрес. Например, 001122334455 G.

Ответ: C200005C0001 L

Решение:

Из консольного вывода видно, что интерфейс получил адрес 2001:1234:5678:ABCD:C000:FF:FE5C:1.

Поскольку длина префикса сети указана как /64, собственно адресом узла являются младшие 64 бита, то есть C000:FF:FE5C:1.

Дополним ведущие нули, чтобы получить полную запись адреса: C000:00FF:FE5C:0001.

Поскольку в соответствии с алгоритмом EUI-64 в середину MAC адреса вставляется последовательность FFFE, удалим её из адреса узла и получим C000:005C:0001.

Поскольку MAC адрес записывается без разделителей, уберем двоеточия:

C000005C0001

Третьим шагом алгоритма EUI-64 была инверсия 7-ого бита. Следовательно, нужно провести еще раз инверсию этого бита, чтобы вернуться к его исходному значению. 7-ой бит будет соответствовать 3-ему биту во втором шестнадцатеричном разряде. Таким образом мы заменим первый 0 в адресе на 2 и получим окончательный адрес:

C200005C0001.

Теперь остался последний шаг – определить, были ли этот адрес изменен вручную. Поскольку 7-ой бит равен 1, адрес были изменен вручную. Следовательно, в ответ нужно записать «C200005C0001 L».

7. Технологии обработки информации в электронных таблицах, сортировка и фильтрация данных (1 балл)

[5 чисел]

Дан фрагмент электронной таблицы в режиме отображения формул:

	A	B	C	D
1		=СРЗНАЧ(A1:A3)	=МАКС(A1:A3)	
2				
3				
4				
5				
6				

В ячейках диапазона A1:A5 находятся пять неповторяющихся целых положительных чисел. Если выделить диапазон A1:A5 и отсортировать ячейки по убыванию, выяснится, что ячейка B1=18, а ячейка C1=25. Если же отсортировать ячейки этого диапазона по возрастанию, то ячейка B1=10, а ячейка C1=13. Известно, что в одной из ячеек содержится число 9. Определите значения всех ячеек диапазона A1:A5 и запишите в ответ в порядке убывания.

Ответ: 25 16 13 9 8

Решение:

Обозначим неизвестные значения, отсортированные в порядке убывания как x_1, x_2, x_3, x_4 и x_5 соответственно. Рассмотрим результат сортировки ячеек по убыванию. Поскольку ячейка C1 содержит максимальное значение, после сортировки по убыванию, в ней окажется значение x_1. Так как ячейка B1 в этом случае равна 18, получается, что $\frac{25+x_2+x_3}{3} = 18$, то есть $x_2 + x_3 = 29$.

Теперь рассмотрим сортировку по возрастанию. Поскольку все числа разные, максимальное значение будет соответствовать x_3, то есть $x_3 = 13$. Тогда из предыдущего равенства $x_2 = 29 - 13 = 16$. Значение B1 после сортировки по возрастанию равно 10, следовательно $\frac{13+x_4+x_5}{3} = 10$, а значит $x_4 + x_5 = 17$. По условию одно из чисел равно 9, значит это x_4, поскольку x_5 будет 17-9=8. Теперь остается записать найденные значения в указанном порядке: 25 16 13 9 8

8. Технологии программирования (2 балла)

[Экспресс тест]

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

В данной задаче вам предлагается автоматизировать оценку результата экспресс теста.

Вам дана двухцветная картинка размером 10×20. Для обозначения цветов используются символы «#» и «.». Тест считается отрицательным, если на картинке изображена одна вертикальная полоска и положительным, если три. В любом другом случае тест считается испорченным.

Полоской будем считать область картинки 10×k, состоящую из символов «#», где k может быть произвольным. При этом все соседние клетки с этой областью должны быть «.». Полоска может находиться на границе картинки.

Формат входных данных

В первой строке входных данных задано число t - число тестов (1≤t≤30). В следующих t·10+(t−1) строках заданы картинки тестов. Соседние картинки разделены пустыми строками. После последней картинки, пустой строки нет.

Каждая картинка состоит из10 строк по20 символов, каждый из которых либо «#», либо «.».

Формат выходных данных

Для каждой картинки выведите результат теста в отдельной строке:
Negative - если тест отрицательный;
Positive - если тест положительный;
Incorrect - если тест испорчен.

Пример

Стандартный ввод	Стандартный вывод
4 .....##..... .....##..... .....##..... .....##..... .....##..... .....##..... .....##..... .....##..... .....##..... .....##..... .....##..... .....##..... .....#..... .....##..... .....##..... .....##..... .....##..... .....##..... .....##..... ..#.....#....###. ..#.....#....###. ..#.....#....###. ..#.....#....###. ..#.....#....###. ..#.....#....###. ..#.....#....###. ..#.....#....###. ..#.....#....###. ..#.....#....###. ..#.....#....###. ..#.....#....###. ..##.....#....###. ..##.....#....###. ..##.....#....###. ..##.....#....###. ..#.....#....###. ..#.....#....###. ..##.....#....###. ..#.....#....###.	Negative Incorrect Positive Incorrect

Решение:

Для решения данной задачи в варианте с вертикальными полосками в каждом тесте достаточно рассмотреть первую строчку. Чтобы тест был корректным должны быть выполнены следующие условия: • Если в первой строке в i-м столбце стоит символ «#», то в остальных строках i-го столбца тоже должны стоять символы «#» • Если в первой строке в i-м столбце стоит символ «.», то в остальных строках i-го столбца тоже должны стоять символы «.» В случае, если вышеуказанные условия выполнены, можно утверждать, что на картинке изображены корректные вертикальные полоски и, возможно, тест не испорчен. Теперь нужно посчитать сколько их. Число полосок на картинке равно числу их левых границ. Левая граница полоски находится в столбце i, если в i-м столбце первой строки находится символ «#» и либо i = 1, либо в i − 1-м столбце

первой строки находится символ «.». Таким образом, посчитав число левых границ полосок можно посчитать число полосок. Согласно условию «Тест считается отрицательным, если на картинке изображена одна вертикальная полоска и положительным, если три. В любом другом случае тест считается испорченным». Соответственно, остается только рассмотреть случаи и вывести ответ.

9. Технологии программирования (4 балла) [Убрать циклы]

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

Мальчик Коля любит графы, но не любит циклы. У Коли есть несколько ориентированных не обязательно связных графов. Графы могут содержать параллельные ребра, но не содержат петли. К сожалению, некоторые из них могут содержать циклы.

Коля хочет развернуть некоторые ребра в каждом графе, чтобы избавиться от циклов. Разворотом ребра будем считать замену ребра из вершины a в вершину b на противоположное ребро, которое будет направлено из вершины b в вершину a. При этом Коля хочет развернуть наименьшее число ребер в каждом графе, чтобы получившиеся графы не содержали циклов.

Помогите Коле узнать, какое наименьшее число ребер ему нужно развернуть в каждом графе.

Формат входных данных

В первой строке входных данных содержится число t - число графов ($1 \le t \le 8$).

В первой строке описания каждого графа дано два числа n и m - число вершин и число ребер соответственно ($2 \le n \le 10$, $1 \le m \le 1000$).

В следующих m строках содержатся пары чисел a_i, b_i - описание ребер графа ($1 \le a_i, b_i \le n$, $a_i \ne b_i$).

Для удобства, описания графов разделены пустыми строками. Пустой строки после последнего графа нет.

Гарантируется, что суммарное число вершин во всех графах не превосходит 50.

Формат выходных данных

Для каждого графа выведите одно число в отдельной строке - наименьшее число ребер, которые нужно развернуть, чтобы полученный граф не содержал циклов.

Пример

Стандартный ввод	Стандартный вывод
4 2 2 1 2 2 1 4 2 1 4 4 3 3 3 1 2 2 3 3 1 3 5 1 2 2 3 3 1 2 3 1 3	1 0 1 1

Замечание

В первом графе нужно развернуть одно из ребер, чтобы убрать цикл. Во втором графе циклов изначально нет, так что ничего разворачивать не надо. В третьем графе можно развернуть любое ребро, чтобы убрать цикл. В четвертом графе нужно развернуть ребро (3,1) или ребро (1,2), чтобы новый граф не содержал циклов.

Решение:

В данной задаче нужно было сделать граф ациклическим путем разворота некоторых ребер. Если граф ациклический, то у него есть топологическая сортировка. Значит, если бы позволяли ограничения, можно было бы перебрать все перестановки вершин графа в качестве топологических сортировок и найти ту, которая получается минимальным числом разворотов ребер. Для того, чтобы узнать для фиксированной топологической сортировки, сколько ребер нужно перевернуть, можно пройти двумя циклами по всем парам вершин и если первая вершина идет в топологической сортировке раньше, чем вторая, то нам необходимо развернуть все ребра из второй вершины в первую, чтобы топологическая сортировка была корректна. Соответственно, сумма таких ребер «справа налево» будет ответом для данной топологической сортировки. Таким образом, минимальным числом ребер, которые нужно развернуть, будет минимум из полученных ответов для всех топологических сортировок.

К сожалению, данное решение работает за $O(n! \cdot n^2)$, что не укладывается ограничения этой задачи, поэтому нужно как-то соптимизировать данное решение. Для оптимизации используем метод динамического программирования по подмножествам.

• dp[mask] — минимальное число ребер, которые нужно развернуть, чтобы граф состоящий из вершин в маске mask стал ациклическим

• В пустом графе ничего разворачивать не нужно, значит dp[0] = 0

• $dp[mask] = \min(dp[mask - 2^i] + cnt(i, mask - 2^i))$ для всех i, которые есть в маске. Здесь $cnt(i, mask - 2^i)$ — число ребер, которые направлены из вершины i в вершины, которые есть в маске $mask - 2^i$. Суть данного перехода в том, что мы уже знаем ответ для масок $mask - 2^i$, значит для них существует некоторая топологическая сортировка, к этой топологической сортировке, мы как-будто дописываем в конец вершину i. Данное решение будет работать за $O(2^n \cdot n^2)$, что в свою очередь уложится в необходимые ограничения.