Задания для 5–8 класса

Заключительный этап (приведен один из вариантов заданий)

1. Кодирование информации, информационный объем (1 балл)

[Старый плеер]

Даша нашла на старом плеере три музыкальные композиции и заметила интересную особенность: все композиции имеют одинаковую частоту дискретизации, однако глубина кодирования первой композиции в 2 раза меньше глубины кодирования второй композиции, а глубина кодирования третьей – в 3 раза больше глубины кодирования второй. Тогда Даша придумала задачку и предложила ее решить своей подруге Алисе: определить длительность третьей композиции. Все композиции двухканальные, а также известно, что первая и третья композиции имеют одинаковый информационный объём, а длительность первой композиции равна 120 секундам. Помогите Алисе решить задачку Даши. В ответ запишите одно целое число – длительность третьей композиции в секундах.

Ответ: 20

Решение:

Применим формулу $I = D \cdot t \cdot i \cdot k$. Обозначим за x глубину кодирования первой композиции. Тогда 2x – глубина кодирования второй композиции, а $3 \cdot 2x$ – глубина кодирования третьей композиции.

$$I_1 = D \cdot t \cdot i \cdot k = D \cdot 120 \cdot x \cdot 2$$
$$I_2 = D \cdot t \cdot i \cdot k = D \cdot t_2 \cdot 2x \cdot 2$$
$$I_3 = D \cdot t \cdot i \cdot k = D \cdot t_3 \cdot 3 \cdot 2x \cdot 2$$

Применим то, что 1я и 3я композиции имеют одинаковый информационный объем.

$$I_1 = I_3$$
$$D \cdot 120 \cdot x \cdot 2 = D \cdot t_3 \cdot 3 \cdot 2x \cdot 2$$

Решим уравнение.

$$120 = t_3 \cdot 6$$
$$20 = t_3$$

2. Комбинаторика (2 балла)

[Шахматы 2: Спасти рядового Пешку]

Дана шахматная доска размера 16x16 клеток и фигура, которая может ходить только по диагонали, но со следующими ограничениями:

1) Фигура не может ходить по диагонали вправо-вниз.
2) На ходах с нечётным номером фигура может пойти **только** влево-вверх.
3) На ходах с чётным номером фигура может выбрать **одно из двух** направлений:

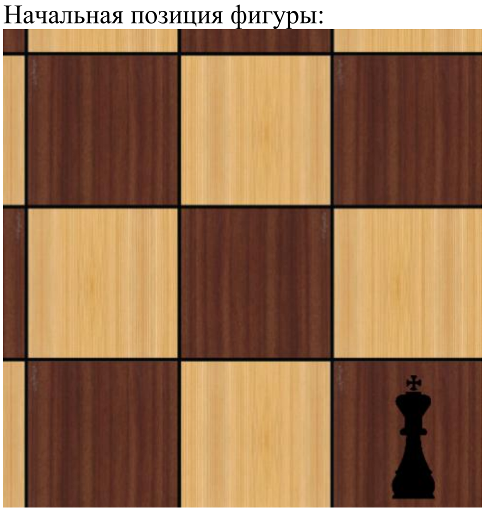
a. Пойти вправо-вверх.

b. Пойти влево-вниз.

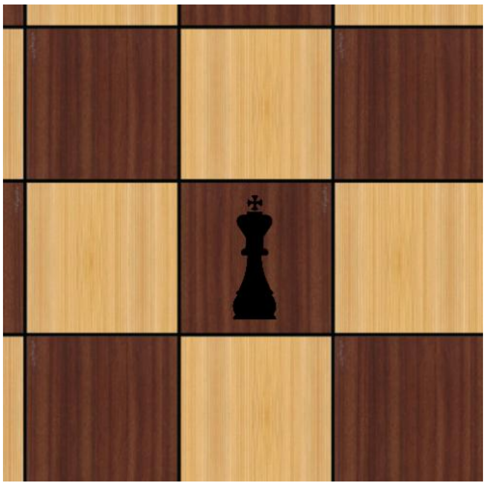
4) Нумерация ходов начинается с единицы.

Пример:

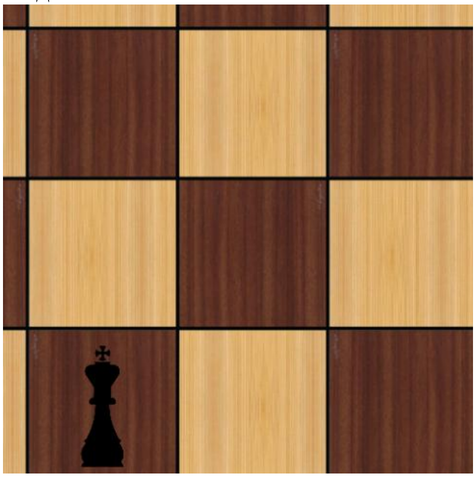
Начальная позиция фигуры:



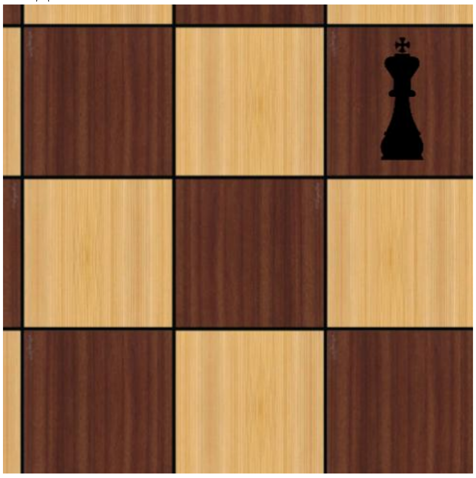
Ход №1:



Ход №2а:

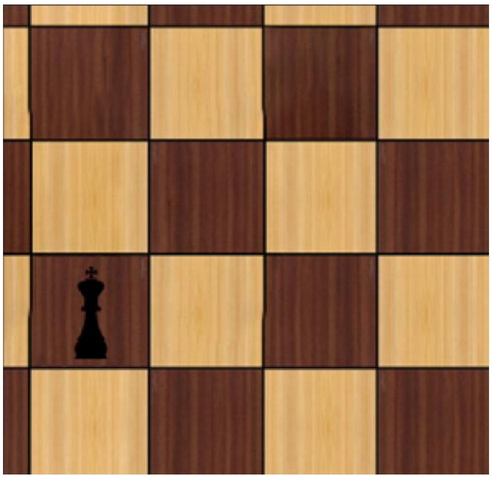


Ход №2б:

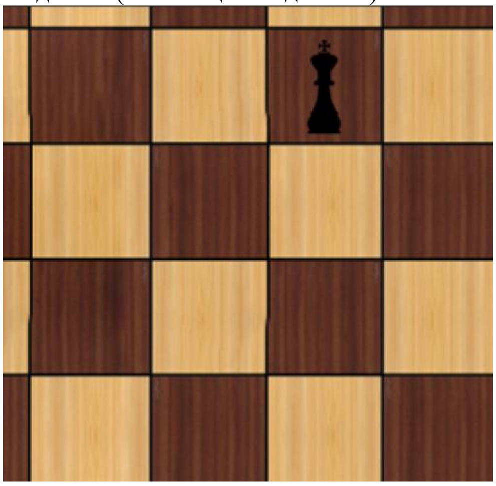


Ходы №2а и №2б являются взаимоисключающими.

Ход №3а (из позиции хода №2а):



Ход №3б (из позиции хода №2б):

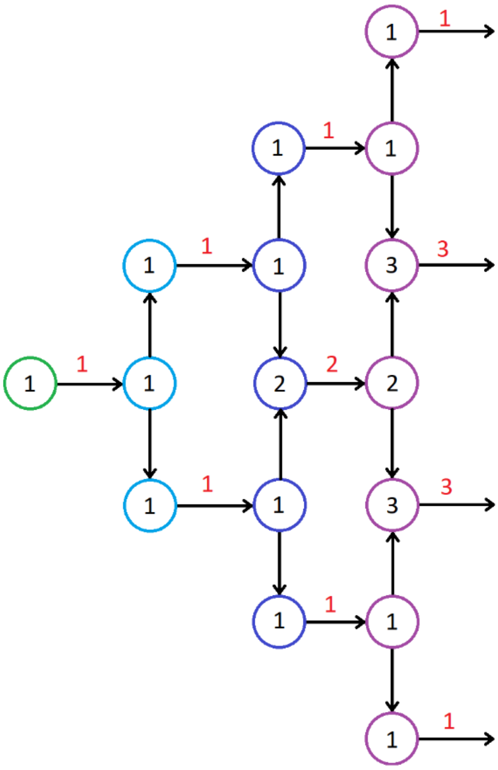


Сколько существует различных траекторий, отличающихся хотя бы одним ходом, для того, чтобы добраться из правого-нижнего угла доски в левый-верхний угол? Число возможных траекторий напишите в ответ.

Ответ: 3432

Решение:

Сначала для удобства представим несколько первых переходов между клетками в виде графа



Каждый цвет представляет первые несколько отдельных диагоналей (параллельные диагонали из левого-нижнего угла в правый-верхний)

Для первых нескольких диагоналей посчитаем количество возможных переходов на следующую диагональ. Заметим, что количество переходов из клеток образуют треугольник Паскаля. Треугольник так и будет расти до момента, когда мы дойдем до самой длинной диагонали в середине доски. Легко понять, что при размере доски 16x16 клеток, длина самой длинной диагонали из левого нижнего края в правый верхний, на которую мы можем встать, будет равна 15 клеткам. Соответственно для того, чтобы дойти из угла до диагонали потребуется 7 переходов между диагоналями из начальной позиции. Значит нам нужна 8 строка в треугольнике Паскаля. Заметим, что после того, как мы дошли до самой длинной диагонали белого цвета, крайние клетки диагонали начинают нам не подходить, так как из них нет путей к следующим диагоналями, и с каждым переходом нужно «отбрасывать» по два крайних числа в строке. И продолжать так делать, пока не останется одно число:

8 строка в треугольник Паскаля:

1 7 21 35 35 21 7 1

Получим следующую строку и сразу отбросим крайние члены

8 28 56 70 56 28 8

Получим следующую строки и вновь отбросим члены

36 84 126 126 84 36

Продолжим применять этот алгоритм:

120 210 252 210 120

330 462 462 330

792 924 792

1716 1716

В итоге складываем последние два числа и получаем **3432**

3. Системы счисления. Теория игр (2 балла)
[Игра в числа]

Прохор и Василиса придумали игру с числами.

В начале игры на доске записывается некоторое стартовое число в десятичной системе счисления. Далее игроки ходят по очереди. Каждый игрок в свой ход стирает число на доске и записывает вместо него запись этого числа в восьмеричной или в девятеричной системе счисления (на свой выбор). Запись на доске следующим игроком читается как новое десятичное число. Выигрывает игрок, записавший в свой ход на доске «300» или большее число. Первым ходит Прохор. Какое минимальное стартовое число должно было быть на доске, чтобы гарантированно выиграл Прохор своим вторым ходом? В ответ запишите искомое число в десятеричной системе счисления.

Пример игры, если бы стартовым числом было 100, а для победы было необходимо записать число, равное или большее 200:

Прохор своим первым ходом переводит 100 в восьмеричную систему счисления и заменяет число на доске на 144. Василиса своим первым ходом переводит 144 в девятеричную систему счисления и заменяет число на доске на 170. Прохор своим вторым ходом переводит 170 в девятеричную систему счисления, заменяет число на доске на 208 и побеждает.

Ответ: 114

Решение:

Заметим, что перевод в 8 систему счисления всегда даст большее число, чем перевод в 9 систему счисления.
Переведем число 300 из 8 системы счисления.

$$300_8 = 192_{10}$$

Таким образом, если после первого хода Василисы на доске останется любое число, равное или большее 192, то Прохор победит, переведя его в восьмеричную систему счисления. Василиса в свой ход может переводить предыдущее число как в восьмеричную, так и в девятеричную системы счисления. Так как в задаче спрашивается минимальное число, необходимое для гарантированной победы Прохора, будем считать, что Василиса будет переводить в свой ход число в 9 систему счисления, но будем следить за тем, чтобы перед ходом Василисы не было выигрышной позиции (192 или больше).
Какое число при переводе в девятеричную систему счисления даст число 192? Никакое, так как цифры 9 в девятеричном числе быть не может. Какое минимальное число, большее 192, может быть переведено из девятеричной системы в десятичную? 200. $200_9 = 162_{10}$. Таким образом, минимальное число, которое можно оставить для Василисы на начало ее хода – 162. Тогда минимальное стартовое число, которое Прохор мог превратить в 162 – это 114 (переводом в восьмеричную систему счисления).
Проверим, что при стартовом числе 114 Прохор действительно выиграет независимо от ходов Василисы:
114 -> (Прохор переводит число в 8 систему счисления) -> 162 -> (Василиса переводит число в 9 систему счисления) -> 200 -> (Прохор переводит число в 8 систему счисления) -> 310, Прохор побеждает.
114 -> (Прохор переводит число в 8 систему счисления) -> 162 -> (Василиса переводит число в 8 систему счисления) -> 242 -> (Прохор переводит число в 8 систему счисления) -> 362, Прохор побеждает.
Проверим также, что при меньшем 114 числе (113) Прохор не сможет выиграть вторым ходом при правильном ходе Василисы.
113 -> (Прохор переводит число в 8 систему счисления) -> 161 -> (Василиса переводит число в 9 систему счисления) -> 188 -> (Прохор переводит число в 8 систему счисления) -> 274, Прохор не побеждает вторым ходом. (Рассматривать варианты ходов Прохора, где он переводит числа в девятеричную систему счисления не имеет смысла, так как это не увеличит итоговое число)

4. Основы логики (3 балла)
[Булева функция]

Петя придумал следующую булеву функцию:

$$\left(p \to (\text{ НЕ}(q) \text{ ИЛИ } r) \text{ ИЛИ НЕ}(r)\right) \text{ И } (p \text{ И НЕ}(q \to r) \text{ ИЛИ } \left(q \to (\text{НЕ}(p) \text{ ИЛИ } r)\right))$$

Также Петя знает о существовании функций, обладающих следующим свойством:

$$f(\text{НЕ}(x_1), \dots, \text{НЕ}(x_n)\ ) = \text{НЕ}(f(x_1, \dots, x_n))$$

Теперь Петя хочет узнать, сколько у него есть способов сделать из своей функции новую, обладающую этим свойством. Новую функцию Петя может сделать следующим образом: записать новую переменную, затем логическую операцию, затем придуманную им исходно функцию в скобках. Причем новая переменная должна быть из следующего множества: {p, q, r, НЕ(p), НЕ(q), НЕ(r)}, а логическая операция из множества: {И, ИЛИ, →}.

Пример: если изначальная функция была ***p ИЛИ q****, то одним из примеров новой функции будет:* ***<u>p И</u> (p ИЛИ q).***

В ответе нужно записать целое число – количество способов составления новой функции, обладающей указанным свойством.

Примечание: → - ИМПЛИКАЦИЯ, логическая операция, таблица истинности которой выглядит следующим образом:

p	q	p→q
0	0	1
0	1	1
1	0	0
1	1	1

Ответ: 6

Решение:

Исходя из описанного свойства, заметим, что функция должна принимать противоположные значения на противоположных значениях ее аргументов (т.е. функция самодвойственна).

Если мы будем смотреть на таблицу истинности такой функции, то значения, полученные при прочтении ее сверху вниз до середины и при прочтении снизу вверх до середины должны быть противоположны друг другу.

Составим таблицу истинности для записанного Петей выражения:

P	q	r	**f**
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

Замечаем, что данное выражение является тождеством. Следовательно, при добавлении в начало логической операции и переменной результат полученного выражения будет во многом зависеть от новой переменной.

Однако, если в качестве добавляемой логической операции мы будем использовать «логическое или» или «импликацию» с любой переменой, то полученное выражение у нас останется тождеством, т.к. (***x*** or 1 = 1) и (***x*** → 1 = 1) – на месте ***x*** может стоять любая из переменных, принадлежащих множеству допустимых переменных из задачи. Тогда остается логическое И.

Если рассматривать «логическое и», то в этом случае результат функции будет совпадать со значением добавленной переменной, если в качестве переменной будет выбрана одна из {p, q, r}, либо будет противоположно значению добавленной переменной, если в качестве новой переменной будет выступать одна из {НЕ(p), НЕ(q), НЕ(r)}. И если посмотреть на составленную таблицу истинности, то можно заметить, что значения переменных в столбцах при прочтении снизу вверх и сверху вниз как раз противоположны друг другу, причем у всех трех переменных. Следовательно новая функция $x \text{ И } ((p \to (\text{ НЕ}(q) \text{ ИЛИ } r) \text{ ИЛИ НЕ}(r)) \text{ И } \left(p \text{ И НЕ}(q \to r) \text{ ИЛИ } \left(q \to (\text{НЕ}(p) \text{ ИЛИ } r)\right)\right))$ будет обладать нужным свойством, где в качестве x стоит одна из допустимых переменных. Итого вариантов составить нужную функцию будет 6.

5. Кодирование информации, структуры данных (3 балла)
[Шифрование на графе]

Бобу нужно было передать пароль двум друзьям. Известно, что пароль состоит только из букв русского алфавита, причем в пароле нет повторяющихся букв.

Боб не хочет, чтобы его пароль узнал кто-то посторонний, поэтому придумал для каждого из друзей различные шифры – «шифр в глубину» и «шифр в ширину».

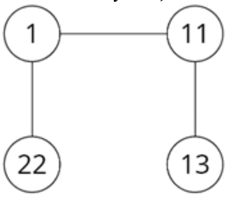
Но для построения шифра нужно сначала построить граф пароля. Строится он следующим образом:

1. Каждая буква в пароле заменяется на число от 1 до 33 в соответствии с ее порядковым номером в русском алфавите.

2. Строится полный граф (т.е. граф, в котором каждая вершина соединена с каждой), в котором в качестве вершин выступают полученные числовые значения букв пароля.

3. Из полученного графа удаляются ребра, которые соединяют между собой те числа, буквенные значения которых НЕ стоят рядом в пароле.

Пример: «ФАЙЛ» - исходный пароль => «22 1 11 13» - численное представление. Тогда граф исходного пароля будет выглядеть следующим образом:

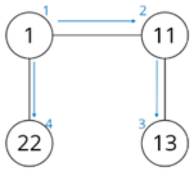


Теперь по этому графу можно построить два шифра. Для построения обоих шифров в качестве начальной вершины выбирается вершина с наименьшим номером. Соседями текущей вершины называются такие вершины, которые соединены с текущей вершиной ребром.

Для построения «шифра в глубину» необходимо:

0. Изначально все вершины графа считаются не посещенными.
1. В качестве первой текущей вершины выбрать вершину с наименьшим числом и записать ее. Считаем эту вершину посещенной.
2. Далее из соседей текущей вершины выбирается вершина с наименьшим номером и записывается ее номер. Переходим в эту вершину (т.е. она становится текущей) и считаем эту вершину посещенной.
3. Повторяем действие 2, пока не встретим вершину, у которой нет не посещённых соседей.
4. Переходим к первой посещенной вершине и повторяем действие 2 для не посещенных соседей.
5. Далее переходим ко второй посещенной вершине и повторяем действие 2 для не посещенных соседей.
6. И так далее идем по посещенным вершинам в порядке их посещения и повторяем действие 2 для не посещенных соседей, пока в графе все вершины не станут посещенными.

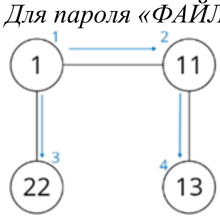
Для пароля «ФАЙЛ» построение и сам «шифр в глубину» будет выглядеть следующим образом: 1 11 13 22



Для построения «шифра в ширину» необходимо:

0. Изначально все вершины графа считаются не посещенными.
1. В качестве первой текущей вершины выбрать вершину с наименьшим числом и записать ее. Считаем эту вершину посещенной.
2. Далее записываем всех ее не посещенных соседей в порядке возрастания чисел, записанных в этих вершинах. В этом же порядке считаем эти вершины посещенными.
3. Далее идем по посещенным вершинам (в порядке их посещения) и повторяем действие 2.
4. Идем так, пока в графе все вершины не станут посещенными

Для пароля «ФАЙЛ» построение и сам «шифр в ширину» будет выглядеть следующим образом: 1 11 22 13



Ева смогла получить оба шифра Боба и узнать какой шифр соответствует какому алгоритму:

Шифр в глубину: 10 14 16 25 20 33

Шифр в ширину: 10 14 20 16 33 25

Также Ева узнала, что пароль начинается с буквы Я.

Помогите Еве разгадать пароль Боба.

В ответ запишите пароль заглавными буквами без пробелов.

Примечание: русский алфавит:

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я			

19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

Ответ: ЯТИМОЧ

Решение:

Заметим, что так как буквы в пароле не повторяются, и в окончательном графе пароля остаются только ребра, соединяющие соседние буквы, то тогда у каждой вершины может быть либо один, либо два соседа (под соседями подразумеваются соседние по ребрам вершины)

Рассмотрим два случая, когда у вершины 10 будет 1 сосед и когда 2 соседа:

1) Если у вершины 10 один сосед.

Тогда, если посмотреть на шифр в ширину, то по его алгоритму соседом вершины 10 будет только вершина 14. И так как у вершины 10 больше нет соседей, то по алгоритму далее должна рассматриваться вершина 14 и получается, что у нее тоже только один сосед – вершина 20 (два соседа у нее не могут быть, так как далее идет число 16, а если бы вершина 16 была соседом вершины 14, то в шифре 16 стояло бы до 20).

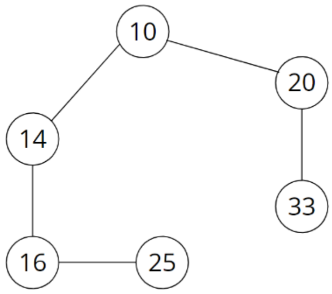
Однако если посмотреть на шифр в глубину, то сначала получается, что у вершины 14 есть сосед 16, что не сходится с первым шифром. Противоречие.

2) Если у вершины 10 два соседа.

Смотря на шифр в ширину, понимаем, что соседи вершины 10 – вершина 14 и вершина 20.

Далее по шифру в глубину заметим, что у вершины 14 соседом является вершина 16. И так как у вершины 14 уже два соседа, то из шифра в ширину получаем, что вершина 33 является соседом вершины 20, и тем самым и у вершины 20 получаются два соседа, тогда оставшаяся вершина 25 является соседом вершины 16, что можно заметить из шифра в глубину.

Тогда получается, что граф численного представления пароля будет выглядеть следующим образом:



В начале мы заметили, что в оставшемся графе ребрами соединены вершины, чьи буквенные эквиваленты стоят рядом в пароле, то тогда можно восстановить пароль последовательно пройдясь по получившейся цепочке. Остается только узнать каким буквам соответствуют данные порядковые номера и с какого конца нужно будет начинать считывание, но так как в задаче сказано, что пароль начинается с буквы Я, порядковый номер которой 33, то тогда с этой вершины и следует начинать дешифрование.

Тогда численный пароль будет: 33 20 10 14 16 25, а буквенный пароль ЯТИМОЧ

6. Теория игр (2 балла)
[Весенняя уборка]

Однажды, Муми-тролль вместе со Сниффом решили убраться в доме после долгой зимней спячки. К своему удивлению, они нашли так много грецких орехов, что сложили из них целую кучку. С этой кучкой они решили сыграть в следующую игру:

Игра состоит из раундов. В каждом раунде сначала ходит первый игрок, а потом второй. Каждый из игроков в свой ход может взять из кучи либо 7 орехов, либо t орехов (или все оставшиеся, если орехов в куче меньше 7 или меньше t). Игрок, после хода которого, в кучке не осталось орехов считается победителем.

В первом раунде t = 2. Но с каждым следующим раундом t = t · 2. То есть во втором раунде t = 4, в третьем t = 8 и так далее.

Определите, какое минимальное число орехов изначально могло быть в кучке, чтобы гарантированно победил 1-ый игрок и при том на четвёртом раунде.

Ответ: 36

Решение

Если игра была закончена первым игроком и при том на четвёртом раунде, то к началу четвёртого раунда в куче должно было быть от 1 и до 16 орехов.

Тогда в третьем раунде (t = 8): Перед ходом второго игрока в кучке должно быть (1 + 8 = 9; 16 + 7 = 23) от 9 и до 23 орехов, а перед ходом первого игрока в третьем раунде в кучке должно быть (9 + 7 = 16; 23 + 8 = 31) от 16 и до 31 орехов.

Во втором раунде (t = 4): Перед ходом второго игрока в куче должно быть (16 + 7 = 23; 31 + 4 = 35) от 23 и до 35 орехов. Перед ходом первого игрока в куче должно быть (23 + 4 = 27; 35 + 7 = 42) от 27 и до 42 орехов.

В первом раунде (t = 2): Перед ходом второго игрока в куче должно быть (27 + 7 = 34; 42 + 2 = 44) от 34 и до 44 орехов. Перед ходом первого игрока в куче должно быть (34 + 2 = 36; 44 + 7 = 51) от 36 и до 51 орехов.

По сути, в этих вычислениях итеративно изменяется числовой интервал возможного числа орехов. Для второго игрока нижняя граница интервала увеличивается на максимальное число из чисел 7 и t, чтобы второй игрок своим ходом не мог попасть ниже нижней границы интервала. Аналогично верхняя граница увеличивается на минимальное число из чисел 7 и t.

А для первого игрока всё наоборот, нижняя граница увеличивается на минимальное из 7 и t, а верхняя на максимальное из 7 и t. В этой задаче верхнюю границу мы могли и не учитывать, чтобы получить верный ответ.

Минимальное изначальное число орехов, при котором победил первый игрок на четвёртом раунде – это 36

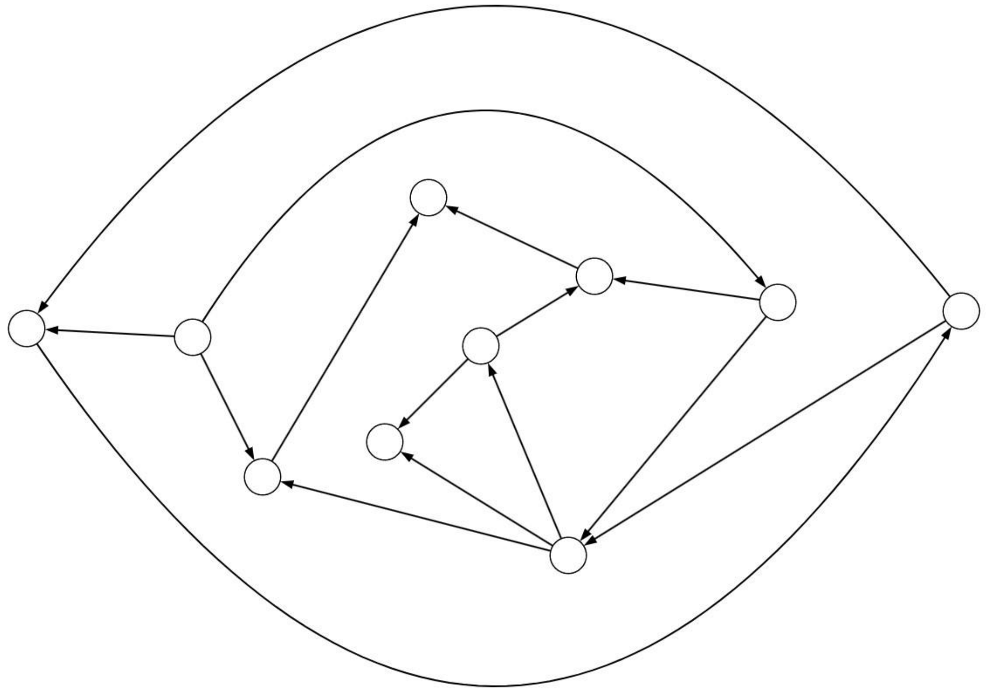
7. Моделирование на графе (1 балл)

[Музейная инновация]

Архитекторы решили оснастить музейные залы самодвижущимися дорогами - траволаторами. По такой дороге возможно двигаться лишь в одном направлении. К сожалению, из-за просчётов в плане, некоторые из дорог развернули не в ту сторону. Вы можете поменять направление каких-то дорог так, чтобы независимо от того, из какой комнаты посетитель попадает на траволатор, он мог добраться по траволатору до любой комнаты.

План музея представляет собой ориентированный граф, где комнаты обозначаются вершинами, а дороги траволатора – рёбрами. При перемещении по музею, по траволатору можно двигаться только в направлении указанному стрелкой.

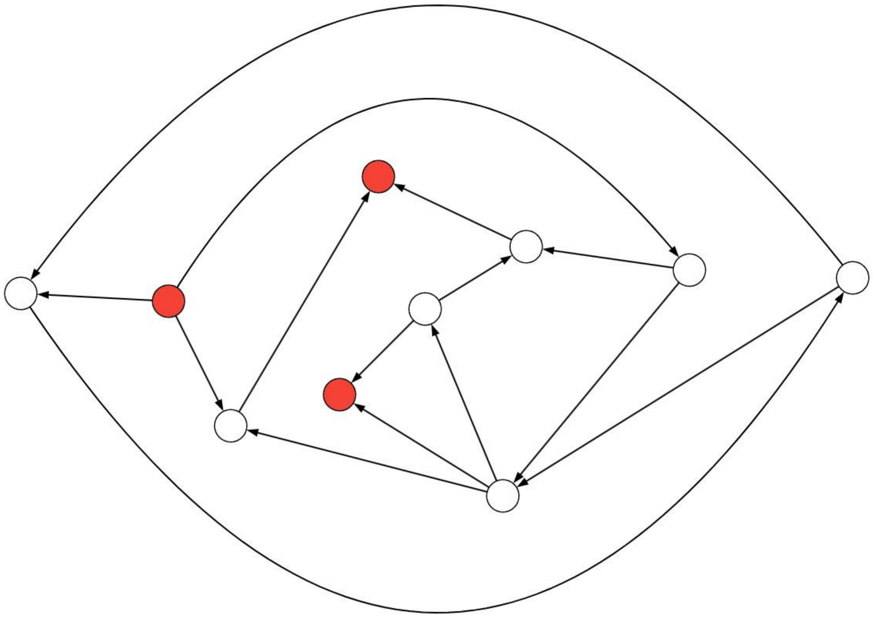
Требуется исправить план музея так, чтобы число перенаправленных дорог было минимально. В ответе укажите число перенаправленных дорог.



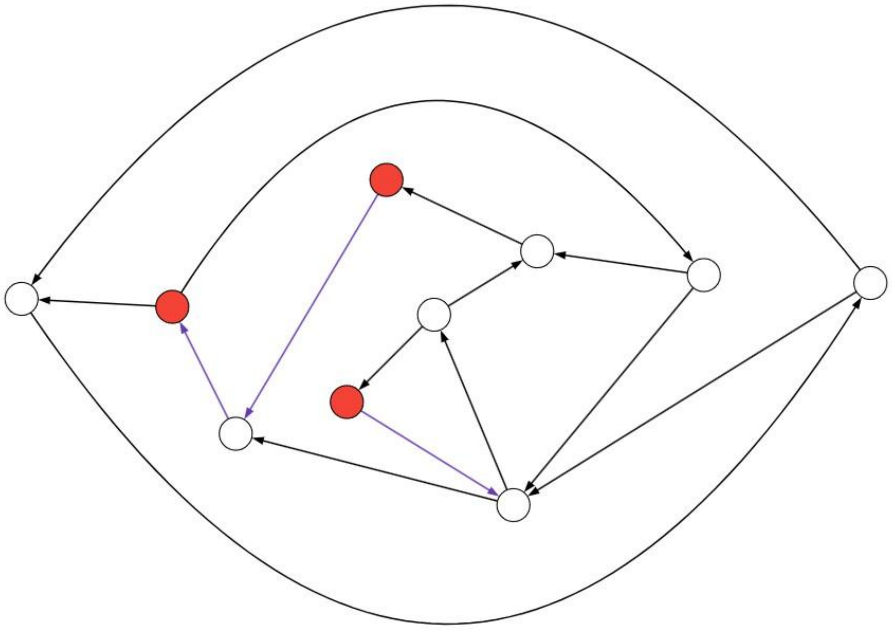
Ответ: 3

Решение:

Для начала, отметим такие вершины, у которых все рёбра имеют одинаковую направленность, т.е. либо все рёбра исходят из вершины, либо все рёбра входят в вершину. Назовём такие вершины неправильными.



Для того, чтобы из любой вершины был путь в любую другую, нужно как минимум для каждой из выбранных вершин изменить направление как минимум одного соседнего с ней ребра так, чтобы такой выбор ребер не создал новых неправильных вершин – так как в неправильные вершины либо невозможно попасть, либо невозможно из них выйти. Возможный выбор ребер:



Заметим, что теперь желаемая цель достигнута, из любой вершины можно попасть в любую другую, следовательно дополнительных изменений не требуется.

8. Моделирование (1 балл)
[Алгоритм на таблице]

Расстоянием между числами является модуль их разности, т.е. разность между большим и меньшим числом.

Ближайшим числом к текущему числу называется такое число, что расстояние между текущим и этим числом минимально.

Дана матрица размера 10x10, заполненная числами от 1 до 2^8:

40	151	91	201	127	188	167	26	47	60
86	201	173	61	33	67	3	178	57	57
116	139	187	174	252	209	184	27	84	152
85	250	103	238	97	103	84	188	50	251
158	243	33	182	90	115	46	164	111	100
45	88	48	244	130	25	220	135	195	190
12	114	69	76	176	52	206	220	117	49
157	91	74	104	154	243	156	113	37	66
166	74	210	188	43	115	41	29	171	129

224	221	165	60	95	200	39	39	21	181

К ней применятся следующий алгоритм:

1. Из отрезка [1, 2^8] выбираются 2^{8-i} чисел, таких что в отсортированном виде все соседние пары чисел имеют одинаковое расстояние. Числа 1 и 2^8 входят в этот список. Выбранные числа не обязательно должны быть целыми.
2. Каждое число в матрице заменяется на ближайшее число из выбранных на первом шаге чисел. Если у числа в матрице два ближайших числа из выбранных на первом шаге, то оно заменяется на меньшее из ближайших.

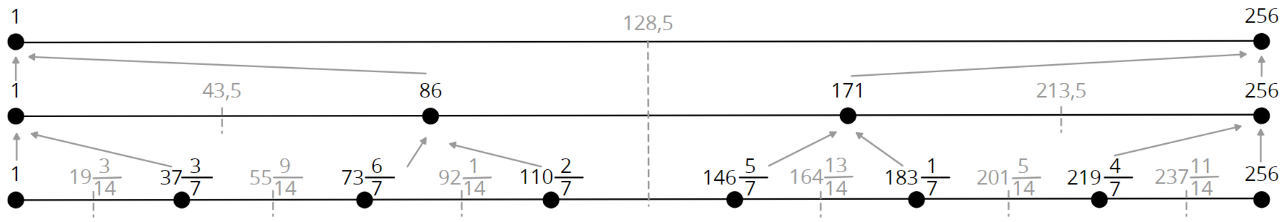
Алгоритм последовательно выполняется для значений i от 0 до 7 включительно.

Требуется узнать, сколько чисел 1 станет в матрице после выполнения алгоритма. В ответ запишите целое число.

Ответ: 55

Решение:

Рассмотрим на числовом отрезке последовательно несколько итераций для какого-либо числа, как оно будет меняться:



Черным отмечены точки, на которые будут заменяться наши числа. Серым отмечены точки – середины всех отрезков, с помощью которых и определяется какое число из двух соседей конкретно нужно выбрать (т.е. сначала смотрим, в какой отрезок попадает число, потом сравниваем это число с серединой данного отрезка, если наше число меньше середины отрезка, то на данной итерации число заменяется на число, равное левому концу данного отрезка, и наоборот).

Заметим, что после первой итерации все числа в матрице будут принадлежать числам из отрезка [1, 2^8], выбираемым первым шагом алгоритма. И так далее с каждой итерацией числа в матрице числа будут принадлежать выбранным на первом шаге алгоритма числам. Так же заметим, что на каждой итерации выбираемых первым шагом алгоритма числе будет в два раза меньше. Тогда можно заметить, что каждому числу из текущей итерации будут соответствовать 2 числа из предыдущей итерации (отмечено на схеме стрелками).

Так же, нетрудно заметить, что после выполнения работы алгоритма, все числа в матрице будут равны либо 1 либо 2^8. Тогда, чтобы узнать кол-во единиц в конце необязательно проделывать все шаги алгоритма, а достаточно узнать, сколько чисел из исходной матрицы будут меньше 128,5 – середины отрезка [1, 2^8]. В нашем случае таких числе будет 55.

9. Алгоритмизация. Анализ блок-схемы (3 балла) [Преобразование массива]

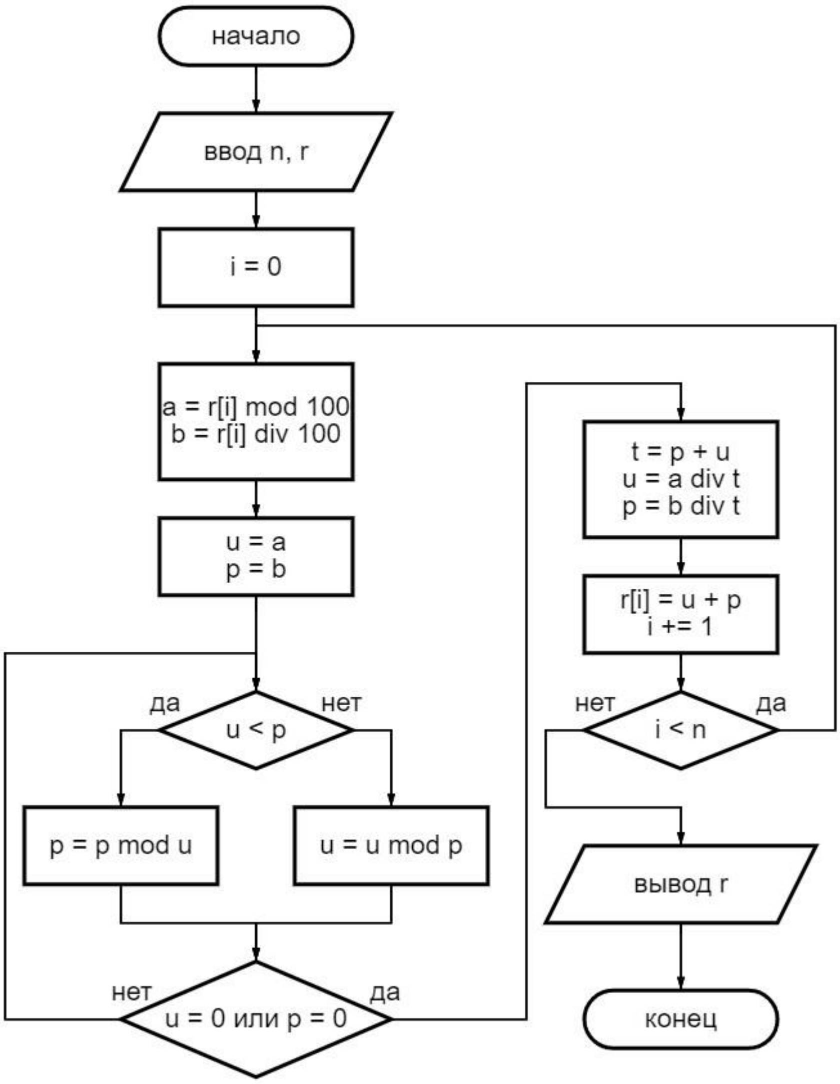
Вам дана блок-схема алгоритма, который получает на вход: число *n* – число элементов в массиве *r* и сам массив *r*. После алгоритм делает преобразования над элементами массива и возвращает изменённый массив *r*. Элементы в массиве нумеруются с нуля.

Алгоритм запустили для следующего массива (n = 5):

r = {7042, 2412, 981, 19590, 199}

В ответ запишите значения элементов массива после преобразований алгоритма подряд без пробелов в порядке их расположения в массиве.

(Допустим, что массив из двух элементов после работы алгоритма выглядит как: {44, 6}, тогда в ответ следует записать: 446)



Примечание:

1. “a += b” – переменной *a* присваивается значение суммы переменных *a* и *b*.
2. “p mod q” – взятие числа *p* по модулю *q* (остаток от целочисленного деления *p* на *q*).
3. “p div q” – целочисленное деление числа *p* на *q*.
4. “r[i]” – обращение к элементу массива под номером *i*.

Ответ: 831019100

Решение:

Проанализируем блок-схему: видим, что алгоритм перебирает все элементы массива в цикле, где переменная *i* хранит номер рассматриваемого элемента на текущей итерации цикла. В переменную *a* записывается число из двух последних цифр рассматриваемого элемента массива, а в переменную *b* – число из оставшихся цифр рассматриваемого элемента массива. Можно увидеть, что затем используется алгоритм Евклида для нахождения наибольшего общего делителя (НОД) чисел *a* и *b*. Рассчитанное значение НОД записывается в переменную *t*, после этого изначальные части, рассматриваемого элемента массива, *a* и *b* делятся на их НОД. Полученные значения суммируются и записываются в массив вместо старого значения.

r = {7042, 2412, 981, 19590, 199}

7042 => a = 42, b = 70 => НОД = 14 => r[0] = 3 + 5 = 8
2412 => a = 12, b = 24 => НОД = 12 => r[1] = 1 + 2 = 3
981 => a = 81, b = 9 => НОД = 9 => r[2] = 9 + 1 = 10
19590 => a = 90, b = 195 => НОД = 15 => r[3] = 6 + 13 = 19
199 => a = 99, b = 1 => НОД = 1 => r[4] = 99 + 1 = 100

Получили, что ответ равен: 831019100

10. Сортировка и фильтрация данных (2 балла)
[Сортировка Шелла]

Оля реализовала новую для себя сортировку. На вход программа получает массив из N элементов и набор значений d, расположенных от больших к меньшим. На каждом шаге алгоритма между собой сравниваются и сортируются все элементы, находящиеся друг от друга на расстоянии d, после чего берётся следующее значение для d и так продолжается до тех пор, пока d не будет равен 1 (см. пример). После этого массив будет гарантированно отсортирован.

Примечание:

Элемент A[j] находится на расстоянии d от элемента A[i], если A[i + d] = A[j] или A[i - d] = A[j].

D – массив из элементов d_i . Алгоритм выполняется для всех d_i из массива D последовательно.

Пример работы сортировки: для массива A = [7, 2, 4, 6] и значений D = [3, 2, 1]

d = 3: [7, 6]; [2]; [4] => [6, 7]; [2]; [4] => [6, 2, 4, 7]

d = 2: [6, 4]; [2; 7] => [4; 6]; [2; 7] => [4, 2, 6, 7]

d = 1: [4, 2, 6, 7] => [2, 4, 6, 7]

Получили [2, 4, 6, 7]

На вход программе подаётся:

Массив A = [72, 26, 114, 15, 21, 39, 6, 40, 115, 13, 7]

И значения D = [7, 3, 1]

Представим, что из-за сбоя алгоритм не закончил сортировку и остановился сразу перед значением d = 1 (выполнив сортировку для d = 7 и d = 3, но не выполнив для d = 1).

Какой элемент будет находиться в массиве по индексу “6” после этих преобразований? Индексы в массиве пронумерованы с нуля. В ответ запишите число.

Ответ: 40

Решение

Для того, чтобы узнать какой элемент окажется на 6-ой позиции в массиве, нам не обязательно выполнять сортировку полностью, а достаточно лишь рассмотреть элементы из которых выбирается значение A[6] на шаге d = 3. Это элементы с индексами: 0, 3, 6, 9.

0	1	2	3	4	5	6	7	8	9	10
72	26	114	15	21	39	6	40	115	13	7

Теперь рассмотрим какие из интересующих нас элементов менялись на шаге алгоритма для d = 7.

0	1	2	3	4	5	6	7	8	9	10
72	26	114	15	21	39	6	40	115	13	7

Интересующие нас группы: [72, 40]; [114, 13]; [15; 7] => [40, 72]; [13; 114]; [7; 15]

d = 7

0	1	2	3	4	5	6	7	8	9	10
40	26	13	7	21	39	6	72	115	114	15

Теперь выбираем элементы с индексами: 0, 3, 6, 9. Это [40; 7; 6; 114] => [6; 7; 40; 114]. Получается, что A[6] = 40