

Информатика.11 класс

Решения и ответы

Вариант 1.

№	Правильный ответ
1.	Число $2 \cdot 3^{51} + 2 \cdot 3^{31} = 20 \dots 020 \dots 0_3$, где нулей в начале 31. Отнимем теперь 11_3 и получим $20 \dots 0122 \dots 2212$. Итого к имеющимся двум двойкам добавляется 30 двоек, полученных из нулей, и отнимается одна двойка. Когда мы умножаем на три в сотой, к троичной записи добавятся только нули. Итого 31 двойка.
2.	До точки у нас должны быть буква c, за ней d, а также после буквы d должна быть c. Для этого могут подойти два сочетания: cdc и dcd. При этом между ними может быть любое число символов. После точки же будет a*e. Включая точку, получаем 6 символов, значит, нужно добавить ещё один символ. Можно вставить до и после точки. Если после точки, то имеем dcd.a?e или cdc.a?e. Вместо вопроса может быть пять символов, итого 10 вариантов. Если до точки вставить символ, тут несколько вариантов. Либо этот символ d или c, тогда возможны сочетания ccdc, cdcc, cddc, dccd, cdcd, dcdd, ddcd, dcdd, 8 вариантов. И если этот символ a, b, e, то его можно ставить внутри cdc или dcd 4 способами, итого ещё $3 \cdot 4 \cdot 2 = 24$ варианта. Всего 42 варианта.
3.	Для начала преобразуем выражение. $((F \text{ xor } X \text{ xor } 1) \rightarrow (F \text{ xor } X)) = F \text{ xor } X$. Дальше можно провести простой анализ, начиная с $Y=1$. В этом случае выражение преобразуется в $1 \text{ xor } F \text{ xor } X$, при $X=1$ $F=1$, $X=0$ $F=0$. Значение Z при этом не имеет значения. При $Y=0$, имеем выражение $\text{not}((\text{not } F \text{ and } \text{not } X) \rightarrow Z) \text{ xor } F \text{ xor } X$. Тут можно провести такой же анализ и получить при $Z = 1$ $X = 1$ $F = 0$, $Z = 1$ $X = 0$ $F = 1$. И наконец, при 010 функция должна быть равна 0, ну и при трёх нулях она по условия должна быть равна нулю. Получив все значения функции, можно придумать формулу, например, $(X \text{ and } Y) \text{ or } (\text{not } X \text{ and } \text{not } Y \text{ and } Z)$.
4.	Прямолинейным способом будет напрямую посчитать число сочетаний по формуле, но это не очень эффективно (к тому же, значения факториала от числа 40, например, не помещаются в long long int, а вот сами C да, хотя это и не критично для языка Python, но всё же). Лучше воспользоваться вычислением рекуррентно, через формулу $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$. Однако и это не самый эффективный способ. Проще всего занести знаменатель в факториал в знаменателе и домножить всё на $n+1$, тогда получим сумму сочетаний без некоторых слагаемых, которая равна 2^{n+1} . Не будет слагаемых $C_{n+1}^0 + C_{n+1}^1$, которые можно отнять и получить простую формулу $(2^{n+1} - n - 2)/(n + 1)$. Тем самым,

	получить решение в несколько строчек. До этой формулы можно было додуматься и не зная ничего о биномиальных коэффициентах, просто вычислив первые несколько значений. При этом не требуется доказательство формулы, достаточно рассуждений и того, что программа будет написана правильно.
5.	Один из примеров кода на языке Python 3.7 <pre> N = int(input()) data = [] data.append([int(x) for x in (input().split())]) a = [[0 for x in range(100)] for x in range(100)] def F(data, beg, end): if (a[beg][end] != 0): return a[beg][end] if beg == end: a[beg][end] = data[0][beg] return a[beg][end] d = 0 for i in range(beg, end + 1): d = d + data[0][i] a[beg][end] = max(d - F(data, beg+1, end), d - F(data, beg, end-1)) return a[beg][end] x = F(data, 0, N-1) y = sum(data[0]) if (x>y-x): print("М") elif (x < y-x): print("Б") else: print("Н") </pre> Задача решается рекурсивно по такой схеме: мы проверяем два варианта хода, с левого края и с правого. В каждом из них выигрыш равен сумме всех элементов минус выигрыш первого игрока на более короткой подпоследовательности. Для неё делаем то же самое, пока не приходим к последовательностям из 1 элемента, где очевидно, кто выигрывает. Теперь же возвращаемся назад и записываем в массив выигрыш первого игрока на каждой подстроке изначального ряда. Решение с помощью обычного жадного алгоритма не проходит даже первый тест. Есть решения, которые заглядывают чуть глубже, например, можно считать разность тех «очков», что мы получим, и максимум того, что мы дадим второму игроку, и сравнивать их. Тем не менее, в этой задаче надо видеть всю игру до конца, и подобные этому решения часто не проходят один из двух этих тестов: <pre> 1 1 6 5 1 2 11 7 5 </pre>

Вариант 2.

№	Правильный ответ
1.	Число $4 \cdot 5^{42} + 4 \cdot 5^{30} = 40 \dots 040 \dots 0_5$, где нулей в начале 30. Отнимем теперь 111_5 и получим $40 \dots 0344 \dots 4334$. Итого к имеющимся двум четвёркам добавляется 28 четвёрок, полученных из нулей, и одна 4 становится 3. Когда мы умножаем на пять в двенадцатой, к пятеричной записи добавятся только нули. Итого 29 четвёрок.
2.	До точки у нас должны быть буква а, за ней е, а также после буквы е должна быть а. Для этого могут подойти два сочетания: аеа и еае. При этом между ними может быть любое число символов. После точки же будет d*bc. Включая точку, получаем 7 символов, значит, нужно добавить ещё один символ. Можно вставить до и после точки. Если после точки, то имеем аеа.d?bc или еае.d?bc. Вместо вопроса может быть шесть символов, итого 12 вариантов. Если до точки вставить символ, тут несколько вариантов. Либо этот символ а или е, тогда возможны сочетания ааеа, аеаа, аееа, еаеа, аеае, еаеа, еееа, еаее, 8 вариантов. И если этот символ с, b, d, f, то его можно ставить внутрь аеа или еае 4 способами, итого ещё $4 \cdot 4 \cdot 2 = 32$ варианта. Всего 52 варианта.
3.	Если подставить значения из условия в формулу, получится выражение $\text{not } F \text{ xor not } F$, которое при любом значении F равно 0. Поэтому в принципе ни одна из функций здесь не подойдёт, и это будет правильным ответом. Однако не снимаются баллы в том числе и тем, кто нашёл функцию, делающую верным выражение в остальных 7 случаях, а в 100 равную нулю, это функция $(x \text{ and } z) \text{ or } (\text{not } x \text{ and not } z)$.
4.	Прямолинейным способом будет напрямую посчитать число сочетаний по формуле, но это не очень эффективно (к тому же, значения факториала от числа 40, например, не помещаются в long long int, а вот сами C да, хотя это и не критично для языка Python, но всё же). Лучше воспользоваться вычислением рекуррентно, через формулу $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$. Однако и это не самый эффективный способ. Проще всего занести знаменатель в факториал в знаменателе и домножить всё на n+1, тогда получим сумму сочетаний без некоторых слагаемых, которая равна 2^{n+1} . Не будет слагаемых $C_{n+1}^0 + C_{n+1}^{n+1}$, которые можно отнять и получить простую формулу $(2^{n+1} - 2)/(n + 1)$. Тем самым, получить решение в несколько строчек. До этой формулы можно было додуматься и не зная ничего о биномиальных коэффициентах, просто вычислив первые несколько значений. При этом не требуется доказательство формулы, достаточно рассуждений и того, что программа будет написана правильно.
5.	Один из примеров кода на языке Python 3.7 <pre>N = int(input()) data = [] data.append([int(x) for x in (input().split())])</pre>

```

a = [[0 for x in range(100)] for x in range(100)]
def F(data, beg, end):
    if (a[beg][end] != 0): return a[beg][end]

    if beg == end:
        a[beg][end] = data[0][beg]
        return a[beg][end]
    d = 0
    for i in range(beg, end + 1): d = d + data[0][i]
    a[beg][end] = max(d - F(data, beg+1, end), d - F(data, beg, end-1))
    return a[beg][end]
x = F(data, 0, N-1)
y = sum(data[0])
if (x>y-x): print("М")
elif (x < y-x): print("Б")
else: print("Н")

```

Задача решается рекурсивно по такой схеме: мы проверяем два варианта хода, с левого края и с правого. В каждом из них выигрыш равен сумме всех элементов минус выигрыш первого игрока на более короткой подпоследовательности. Для неё делаем то же самое, пока не приходим к последовательностям из 1 элемента, где очевидно, кто выигрывает. Теперь же возвращаемся назад и записываем в массив выигрыш первого игрока на каждой подстроке изначального ряда. Решение с помощью обычного жадного алгоритма не проходит даже первый тест. Есть решения, которые заглядывают чуть глубже, например, можно считать разность тех «очков», что мы получим, и максимум того, что мы дадим второму игроку, и сравнивать их. Тем не менее, в этой задаче надо видеть всю игру до конца, и подобные этому решения часто не проходят один из двух этих тестов:

```

1 1 6 5
1 2 11 7 5

```

Вариант 3.

№	Правильный ответ
1.	Переведём число в 16 систему, так как оно преобразует последние 4 цифры в один символ. Получим 135_{16} . Возводя в степень, смотрим только на последнюю цифру. Первый раз у нас получится 9, потом D, потом 1, потом снова 5. То есть цикл длины 4. Мы возводим в число 22, после 20 возведений мы ничего не поменяем, получается вторая степень 5, её цифра равна 9, как мы уже видели.
2.	Дополнительный символ можно поставить лишь в два места, нам нужно распределить три символа между двумя звёздочками, сделать это можно 4 способами $B????AB$, $B???AB?$, $B??AB??$, $B?AB???$. Сначала выберем место для буквы C, потом на каждом из трёх мест выберем одну из 3 букв, получаем $4*3*3*3=108$ вариантов для каждой маски. Однако можно заметить, что у них есть пересечения, а именно первая и третья одновременно удовлетворяют именам вроде $B??ABAB$, этой маске удовлетворяет ещё 6 имён. Для первой и четвёртой, а так же второй и четвёртой рассуждения аналогичны. Сложим, что было в начале и отнимем те маски, которые считаются дважды, получим $108*4 - 18 = 414$.
3.	Чтобы данное выражение становилось верным, нужно, чтобы в X, Y, Z функция равнялась единице, а в not Z, X, Y нулю. Сопоставим каждой паре первого вида пару второго вида <pre> 000 100 001 000 010 101 011 001 100 110 101 010 110 111 111 011 </pre> Теперь давайте зациклим их с помощью этой таблицы $000 \rightarrow 100 \rightarrow 110 \rightarrow 111 \rightarrow 011 \rightarrow 001 \rightarrow 000$, и $010 \rightarrow 101 \rightarrow 010$. В первом цикле 6 элементов, во втором два. Чтобы наше выражение стало правдивым как можно чаще, в цикле функция должна попеременно принимать значения 0 и 1. Для каждого цикла есть только два таких варианта, итого 4 функции. В качестве какой-то одной можно взять $X \wedge (Y \rightarrow Z)$ or $(\text{not } X \wedge \text{not } Y \wedge Z)$.
4.	Прямолинейным способом будет напрямую посчитать число сочетаний по формуле, но это не очень эффективно (к тому же, значения факториала от числа 40, например, не помещаются в long long int, а вот сами C да, хотя это и не критично для языка Python, но всё же). Лучше воспользоваться вычислением рекуррентно, через формулу $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$. Однако и это не самый эффективный способ. Проще всего занести знаменатель в факториал в знаменателе и домножить всё

	на $n+1$, тогда получим сумму сочетаний без некоторых слагаемых, которая равна 2^{n+1}. Не будет слагаемых $C_{n+1}^0 + C_{n+1}^{n+1}$, которые можно отнять и получить простую формулу $(2^{n+1} - 1)/(n + 1)$. Тем самым, получить решение в несколько строчек. До этой формулы можно было додуматься и не зная ничего о биномиальных коэффициентах, просто вычислив первые несколько значений. При этом не требуется доказательство формулы, достаточно рассуждений и того, что программа будет написана правильно.
5.	Один из примеров кода на языке Python 3.7 <pre> N = int(input()) data = [] data.append([int(x) for x in (input().split())]) a = [[0 for x in range(100)] for x in range(100)] def F(data, beg, end): if (a[beg][end] != 0): return a[beg][end] if beg == end: a[beg][end] = data[0][beg] return a[beg][end] d = 0 for i in range(beg, end + 1): d = d + data[0][i] a[beg][end] = max(d - F(data, beg+1, end), d - F(data, beg, end-1)) return a[beg][end] x = F(data, 0, N-1) y = sum(data[0]) if (x > y-x): print("М") elif (x < y-x): print("Б") else: print("Н") </pre> Задача решается рекурсивно по такой схеме: мы проверяем два варианта хода, с левого края и с правого. В каждом из них выигрыш равен сумме всех элементов минус выигрыш первого игрока на более короткой подпоследовательности. Для неё делаем то же самое, пока не приходим к последовательностям из 1 элемента, где очевидно, кто выигрывает. Теперь же возвращаемся назад и записываем в массив выигрыш первого игрока на каждой подстроке изначального ряда. Решение с помощью обычного жадного алгоритма не проходит даже первый тест. Есть решения, которые заглядывают чуть глубже, например, можно считать разность тех «очков», что мы получим, и максимум того, что мы дадим второму игроку, и сравнивать их. Тем не менее, в этой задаче надо видеть всю игру до конца, и подобные этому решения часто не проходят один из двух этих тестов: <pre> 1 1 6 5 1 2 11 7 5 </pre>