Задачи и решения очного тура Олимпиады
по Дискретной Математике и Теоретической Информатике-2019

1. Последовательности 1 типа. Логическая схема.

Постройте логическую схему с шестью входами и одним выходом.

На вход подаётся последовательность нулей и единиц (сверху вниз). Схема должна выдавать значение "1" ("ИСТИНА") на тех (и только на тех) последовательностях, в которых для любого начального отрезка количество нулей не превосходит количество единиц.

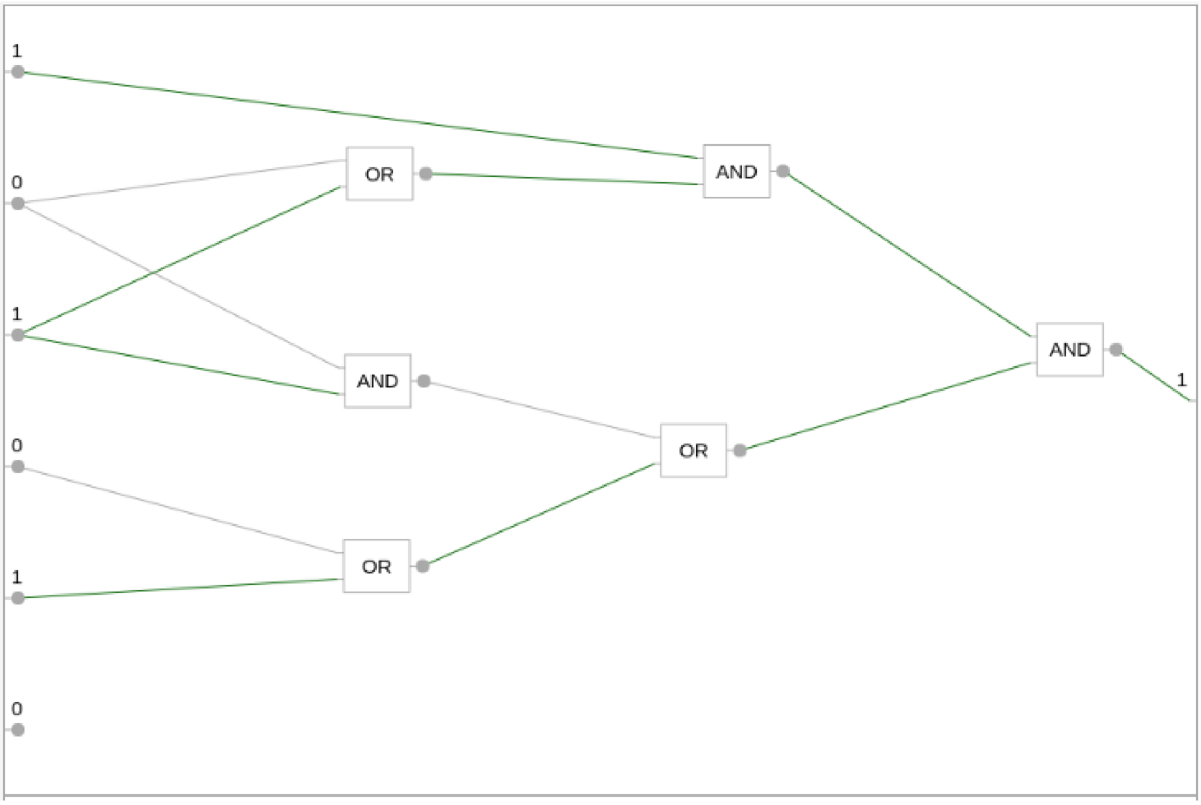
Так, **101010** — подходящая последовательность, а **110001 —** нет, так как среди первых пяти элементов три нуля и только две единицы.

Решение:
Во-первых, заметим, что проверку условия достаточно выполнить только для начальных отрезков нечётной длины, так как выполнение условия для отрезка длины $2n$-1 автоматически означает и выполнение условия для отрезка длины $2n$.

1) Первый элемент обязан быть единицей.
2) При условии того, что первый элемент единица, среди второго и третьего также должна быть хотя бы одна единица. Проверке этого условия соответствует левый верхний элемент **OR** на схеме.
3) Выполнение условия для отрезка длины 5 можно обеспечить двумя способами:
3.1) Добавление ещё одной единицы на четвёртой или пятой позиции (левый нижний элемент **OR**).
3.2) Наличие трёх единиц в начале последовательности (элемент **AND** слева посередине.)

Остальные элементы на схеме служат для того, чтобы связать условия:
(Условие 1) AND (Условие 2) AND ((Условие 3.1) ИЛИ (Условие 3.2))

Итоговая схема:



На рисунке показано, как эта схема работает на последовательности **101010.**

2. Последовательности 1 типа. Логическая схема-2.

На вход подаётся последовательность нулей и единиц (сверху вниз). Схема должна выдавать значение "1" ("ИСТИНА") на тех (и только на тех) последовательностях, в которых для любого начального отрезка количество нулей не превосходит количество единиц.

Так, **101010 —** подходящая последовательность, а **110001 —** нет, так как среди первых пяти элементов три нуля и только две единицы.

Докажите, что можно построить такую логическую схему для n>100 входов, использовав не более $((n+2)^2)/4$ логических элементов.

Решение:
Заметим, что достаточно решить задачу только для нечётных чисел: в случае чётного n последний вход никак не влияет на выполнение условия, то есть достаточно того же количества элементов, что и при нечётном числе входов на единицу меньше. Кроме того, при данном нечётном n=2m+1 достаточно проверить выполнение условия только для всех начальных отрезков нечётной длины.

Построим схему следующим образом. Во-первых, для каждой пары соседних входов с номерами 2k и 2k+1, кроме последней, соединим соответствующие входы при помощи как логического элемента **AND**, так и элемента **OR**. Последние два входа соединим только при помощи элемента **OR**. Это даст нам n=2m-1 логический элемент. Обозначим эти элементы **AND[*k*]** и **OR[*k*]** соответственно.

Для каждого начального отрезка длины 2k+1 устраивающее нас количество единиц может принимать значение от k+1 до 2k+1. Будем пытаться отслеживать все возможные значения этого количества при помощи логических элементов **AND**, и **OR**. Часть схемы, проверяющую то, что количество единиц в первых 2k+1 входах хотя бы s обозначим **A[*k*,*s*]**

Первый вход должен быть равен единице. Проверку этого сделаем в конце, а пока будем считать, что это условие выполняется: подсхему **A[0,1]** при этом фактически образует первый вход логической схемы. Для её построения вообще не нужно логических элементов. При построении остальных подсхем будем считать, что первый вход равен единице. То есть, фактически, подсхема **A[*k*,*s*]** будет давать верный результат только при этом условии.

Для пары чисел 2 и 3 истинность элемента **AND** будет означать, что в первых 3 входах ровно 3 единицы; истинность элемента **OR** — что хотя бы две. То есть, эти элементы и есть подсхемы **A[1,2]** и **A[1,3]**

Пусть для числа 2k-1 мы уже построили части схемы, отслеживающие то, что количество единиц в первых 2k-1 входах хотя бы s, хотя бы s-1 и хотя бы s-2 (подсхемы **A[*k*-1,*s*]**, **A[*k*-1,*s*-1]**, **A[*k*-1,*s*-2]** в наших обозначениях.

Построим тогда подсхему **A[*k*,*s*]**, проверяющую, что что количество единиц в первых 2k+1 входах хотя бы s. Этого можно добиться 3 способами 1) количество единиц в первых 2k-1 входах хотя уже бы s; 2) количество единиц в первых 2k-1 входах хотя бы s-1 и среди входов с номерами 2k и 2k+1 хотя бы одна единица; 3) количество единиц в первых 2k-1 входах хотя бы s-2 и среди входов с номерами 2k и 2k+1обе единицы.

Проверка третьего варианта осуществляется путём соединения выхода подсхемы **A[*k*-1,*s*-2]** и элемента **AND[*k*]** при помощи элемента **AND**. Проверка второго варианта осуществляется путём соединения выхода подсхемы **A[*k*-1,*s*-1]** и элемента и **OR[*k*]** при помощи элемента **AND**. Первому варианту соответствует просто подсхема **A[*k*-1,*s*]**. Далее мы должны соединить выходы построенных части схемы при помощи двух элементов **OR**. Таким образом, подсхема **A[*k*,*s*]** построена при помощи 4 дополнительных элементов.

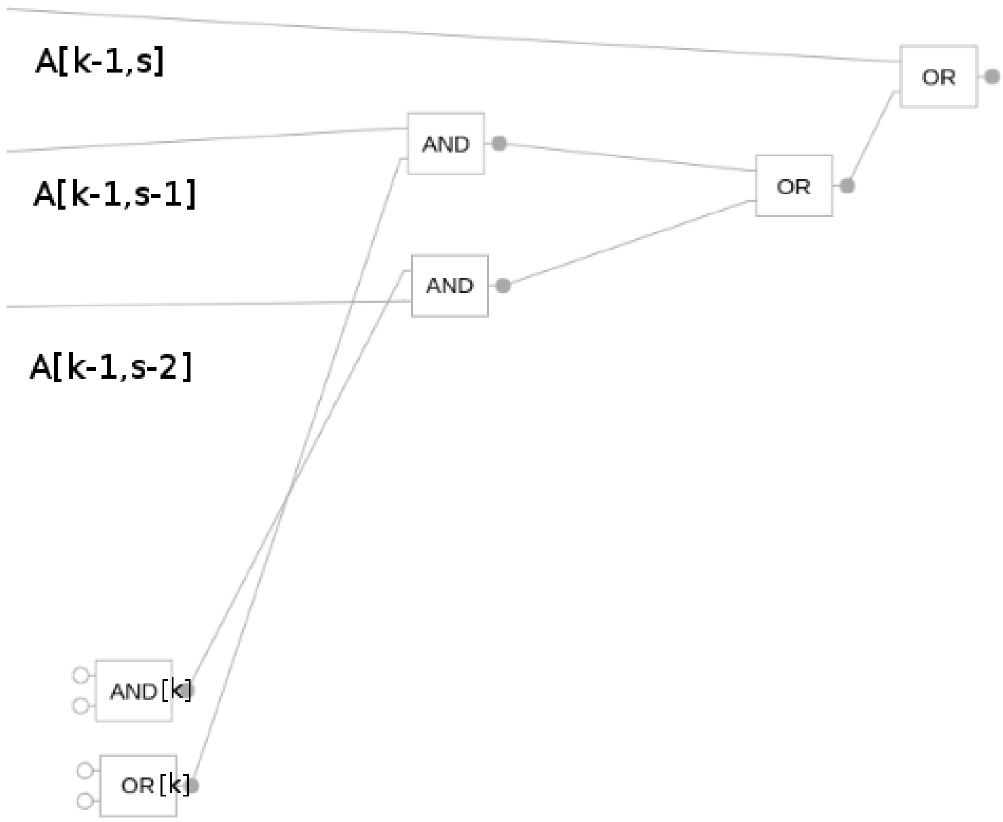


*Схема **A[k.s]** получается из предыдущих схем **A[k-1,s]**, **A[k-1,s-1]**, **A[k-1,s-2].***

Заметим, что для подсхемы **A[*k*,2*k*+1]** может реализоваться только третий из трёх описанных выше вариантов, поэтому для её построения достаточно одного нового элемента **AND**. Для подсхемы **A[*k*,2*k*]** могут реализоваться только второй и третий варианты, поэтому для её построения достаточно трёх новых элементов. Наконец, для подсхемы **A[*k*,*k*+1]** подсхемы **A[*k*-1,*k*-1]** не существует, поэтому могут реализоваться только первый и второй варианты. Кроме того, истинность значения подсхемы **A[*k*-1,*k*]** подразумевается выполнением условия для *k*-1, а значит её можно здесь не проверять. Таким обазом, для построения этой подсхемы достаточно соединить элемент **OR[*k*]** и посдхему **A[*k*-1,*k*+1]** при помощи элемента **OR** и для её построения нужен один новый элемент..

Кроме того, если в какой-то момент количество единиц достигло $m+1$, значит, условие задачи выполнено для всех последующих отрезков, поэтому подсхемы **A[*k*,*s*]** при $s>2m+1$ нам не нужны.

Подсчитаем общее количество элементов, необходимое для постреначения данных подсхем.
Подсхемы **A[*k*,2*k*+1]** строятся для $k>1$ и $2k+1 \le m+1$, то есть для $1 < k \le m/2$.
То есть их количество не превосходит $m/2-1$ и для их построения требуется не более $m/2-1$ дополнительных элементов.
Подсхемы **A[*k*,2*k*]** строятся для k>1 и 2k <= m+1, то есть для 1<k<=(m+1)/2.
То есть их количество не превосходит (m-1)/2 и для их построения требуется не более $3(m-1)/2$ дополнительных элементов.
Подсхемы **A[*k*,*k*+1]** строятся для всех k>1. То есть их количество равно m-1 и для их построения требуется не более $(m-1)$ дополнительных элементов..
Для каждого $2 < k \le (m+1)/2$ строятся также все подсхемы **A[*k*,*s*]**, где $k+1 < s < 2k$, таких подсхем $k - 2$ для каждого такого k. Всего их количество равно сумме чисел от 1 до целой части $(m-3)/2$, то

есть не более (m-3)(m-1)/8, и для их построения требуется не более (m-3)(m-1)/2 дополнительных элементов.

Наконец, для каждого (m+2)/2 ≤ k ≤ m-1 строятся также все подсхемы **A[*k*,*s*]**, где k+1<s<=m+1, таких подсхем m-k-1 для каждого такого k. Всего их количество равно сумме чисел от 1 до целой части (m-4)/2, то есть не более (m-4)(m-2)/8, и для их построения требуется не более (m-4)(m-2)/2 дополнительных элементов..

Наконец, осталось соединить выходы всех подсхем **A[*k*,*k*+1]** при 0 ≤ k ≤ m элементами **AND**, что как раз соответствует проверке утверждения задачи. Для этого потребуется m элементов.

Таким образом, всего нам потребовалось не более

$$(2m-1)+(m/2-1)+3(m-1)/2+(m-1)+(m-3)(m-1)/2+(m-4)(m-2)/2+m=m^2+m+1,$$

что меньше требуемого числа.

Утверждение задачи доказано.

На самом деле, с одной стороны, эту схему можно ещё несколько уменьшить, а также улучшить оценку за счёт более аккуратного подсчёта (в полученной формуле не все слагаемые одновременно могут быть целыми). С другой стороны, формула, которую требовалось доказать, позволяла и менее аккуратный подсчёт.

3. Последовательности 1 типа. Количество последовательностей (Текстовая задача)

Найдите количество таких последовательностей длины n из нулей и единиц, в которых для любого начального отрезка количество нулей не превосходит количество единиц.

Так, **101010 —** подходящая последовательность, а **110001 —** нет, так как среди первых пяти элементов три нуля и только две единицы.

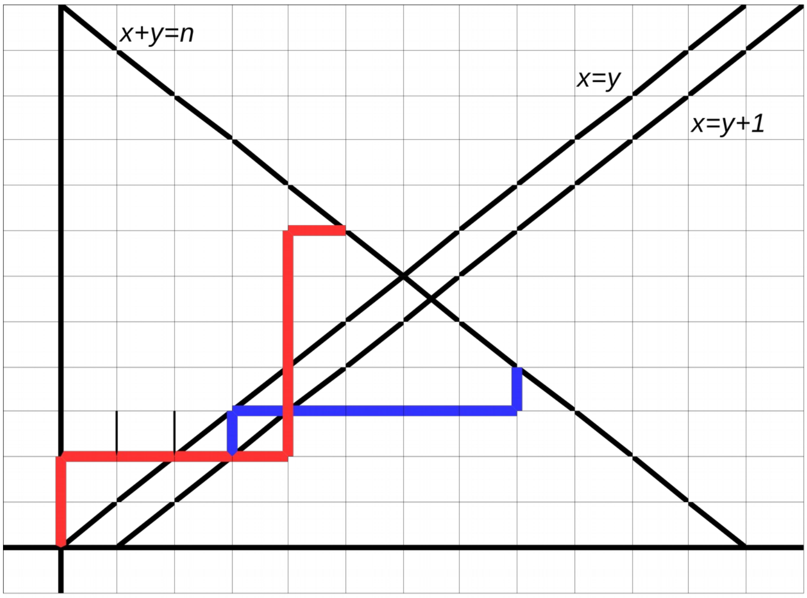
Решение:
Сопоставим каждой такой последовательности длины n путь из n единичных отрезков, начинающихся в начале координат, построенный следующим образом: единице соответствует перемещение вверх по параллельно оси ординат, а нулю перемещение вправо параллельно оси абсцисс.

Поскольку путь содержит ровно n отрезков, его конец находится на прямой $x + y = n$. При этом выполнение требуемого условия для последовательности соответствует тому, что данный путь никогда не опускается ниже прямой $x = y$.

Для этого посчитаем количество всех путей из начала координат, конец которых находится на прямой $x + y = n$ выше прямой $x = y$ или на ней, и вычтем из этого количество тех из этих путей, которые имеют участок ниже прямой $x = y$.

Возьмём какой-нибудь путь, часть которого проходит ниже прямой $x = y$ (красный путь на рисунке). Рассмотрим момент, когда этот путь оказывается ниже прямой $x = y$ первый раз (считая от начала координат). Это происходит в тот момент, когда путь «делает шаг направо» и пересекает прямую ниже $x = y + 1$.

Отразим оставшуюся часть этого пути относительно прямой $x = y + 1$ (синий путь на рисунке) и рассмотрим новый путь, состоящий из старого начала, и нового окончания. Он также состоит из отрезков, идущих вправо или вверх, и содержит n таких отрезков, то есть заканчивается на прямой $x + y = n$. Поскольку конец исходного пути находится строго выше прямой $x = y + 1$, конец нового пути окажется строго ниже него. Любой путь такого вида можно также отразить относительно прямой $x = y + 1$ после первого пересечения с ней, в результате чего получится интересующий нас путь, заканчивающийся не ниже диагонали.



Таким образом, для получения ответа в задаче, надо из числа путей длины n, заканчивающихся не ниже диагонали, вычесть число путей длины n, заканчивающихся ниже прямой $x = y + 1$.

Далее вычисления получаются немного разным для чётного и нечётного числа n. Для чётного $n = 2k$ мы имеем всего 2^{2k} путей, среди них C_{2k}^{k} путей заканчиваются на прямой прямой $x = y$, остальные делятся поровну между путями заканчивающимися выше этой прямой и ниже неё, причём все пути, заканчивающиеся ниже прямой $x = y$, заканчиваются также и ниже прямой $x = y + 1$, поскольку эта прямая пересекается с прямой прямой $x + y = n$ в нецелой точке. При вычитании эти одинаковые количества сокращаются, и остаётся только $C_{2k}^{k} = C_{n}^{n/2}$.

В случае нечётного $n = 2k + 1$ мы имеем ровно половину путей заканчивающихся выше прямой $x = y$ и половину путей, заканчивающихся ниже неё. Среди путей, оканчивающихся ниже этой прямой, C_{2k+1}^{k} заканчиваются в точке пересечения прямых $x = y + 1$ и $x + y = n$; это и есть разность между количеством путей, заканчивающихся не ниже диагонали, и числом путей, заканчивающихся ниже прямой $x = y + 1$.

Общую формулу для ответа можно записать как $C_{n}^{[n/2]}$, где квадратные скобки обозначают целую часть.

4. Последовательности 2 типа. Описание последовательности.

Опишите множество всех последовательностей из букв **a** и **b** таких, что в любом наборе подряд идущих символов (длины больше 1) букв **a** не меньше, чем букв **b**

Обратите внимание! В отличие от предыдущей задачи, рассматриваются не только начальные отрезки последовательности, а вообще любые!

Для описания используйте формулы, которые называются регулярными выражениями.
Так для повторения блока из нескольких букв используйте операцию "звездочка" (итерация), например, **(abb)*** задает множество слов {пустое слово, **abb**, **abbabb**, **abbabbabb**, ...}
Умножение множеств (эту операцию, как обычно в алгебре, изображают точкой или вообще

опускают, что мы и будем делать), описывает склейку всех слов первого множества со словами второго (третьего и т.д.), например **a*cb*** обозначает множество слов: {**c**, **ac**, **cb, acb, aac,..., aaa**...**acb**...**b**, ...}. Обратите внимание что слова, в которых нет букв **a** или **b**, получаются за счет того, что результат итерации может не содержать символов, то есть быть пустым словом.
Последней операцией, которая используется в формулах, является сложение. Сложение соответствует объединению множеств. Так обозначение **(a+b)*c+d(ac*+)** описывает множество всех последовательностей из букв **a** и **b** (обозначается **(a+b)***), к концу которых присоединена буква **c**, объединенного с множеством слов, начинающихся с буквы **d**, за которой следует буква **a**, а за ней любое число букв **c** и ещё одним однобуквенным словом (**d** умножить на пустое слово — это **d**).
Слева приведены примеры слов, которые удовлетворяют нашему условию, справа примеры слов, которые не удовлетворяют ему. Благодаря подсветке вы можете видеть, какие из этих примеров и контрпримеров удовлетворяют построенному вами выражению, а какие — нет.

Решение:
Обратим внимание, что слово, состоящее из одного символа **b,** удовлетворяет условию.

Также поймём, что для последовательностей длины хотя бы 3, нам достаточно проверить выполнение условия только для наборов длины 3. Действительно, из выполнения условия для набора длины 3 (что, в случае нечётного числа означает, что количество букв **a** строго больше) автоматически следует выполнение условия для наборов длины 2, а набор любой другой длины можно составить из наборов длины два и три.

Выполнение условия на наборах длины 2 и 3 в свою очередь равносильно тому, что между любыми двумя буквами **b** есть хотя бы две буквы **a**.

Решением задачи является формула: **a*+a*b(+(aaa*b)*)a*.**

Она построена следующим образом: слагаемое **a*** отвечает за слова, в которых нет буквы **b**.
Если буква **b** есть, перед ней идёт произвольное количество букв **a.** Дальше для каждой следующей буквы **b** должно быть ещё хотя бы две буквы **a**, откуда мы получаем **aaa*b,** повторённое несколько раз. Заканчивается всё ещё несколькими буквами **a**.

5. Последовательности 2 типа. Распознающая схема

Опишите множество всех последовательностей из букв **a** и **b** таких, что в любом наборе подряд идущих символов (длины больше 1) букв **a** не меньше, чем букв **b**

Данная схема состоит из вершин (называемых состояниями) и стрелок, символизирующих правила, по которым работает эта схема. Схема начинает работу в начальном состоянии **S0**, выделенном оранжевым. Поступающее на ход слово анализируется посимвольно. При рассмотрении каждого символа мы переходим из текущего состояния по стрелочке, над которой написан этот символ.

После того, как всё слово проанализировано, мы заканчиваем работу в одном из состояний. Некоторые состояния необходимо пометить как конечные (жирная каёмка). Это те состояния, в которых мы оказываемся, проанализировав нужные слова.

Решение:
Как и в предыдущей задаче, для выполнения условия необходимо и достаточно, чтобы между любыми двумя буквами **b** шли хотя бы две буквы **a**.

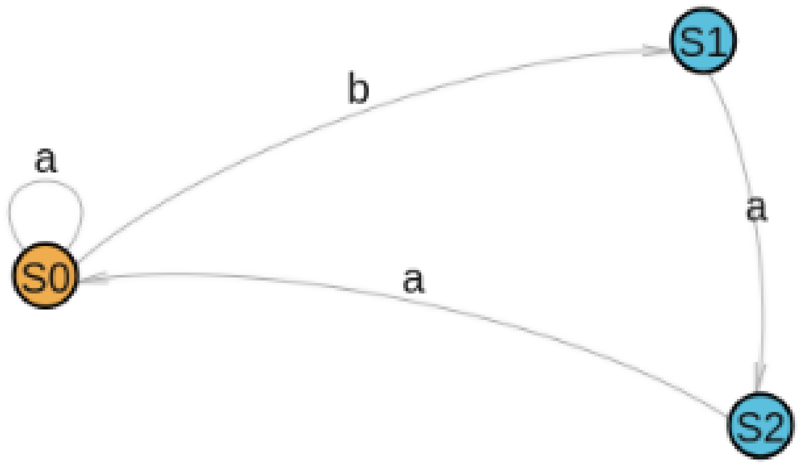
Нам понадобятся всего три состояния:

Состояние **S0** — начальное. В него автомат возвращается, считав хотя бы две буквы **a**. Это единственное состояние, в котором может быть корректно считана буква **b.**

Состояние **S1 —** автомат оказывается в нём, когда считывает букву **b**. Из этого состояния он может корректно выйти только по букве **a**.

Переходит он в состояние **S2**, в котором снова не может быть считана буква **b**. Отсюда автомат по букве **a** возвращается в состояние **S0**, так как после двух букв **a** подряд буква **b** уже возможна.

Все состояния будет конечными, потому что не существует недопустимых последовательностей символов, из которых могут получиться допустимые добавлением символов.



6. Машина Тьюринга. Преобразование последовательности

Постройте машину Тьюринга, которая в любую последовательность из букв **a** и **b** добавляет буквы **a** так, чтобы между любыми буквами **b** было бы хотя бы две буквы **a**.

Например, последовательность **babba** должна превратиться в **baabaaba**

Начальное состояние **s**, конечное **f**

Решение:

s[a] → s[a]R **s[b] → q0[b]R** **q0[a] → q1[a]R** **q1[a] → s[a]R**	Считывающая головка движется направо, пока не встретится символ **b.** Дальше с помощью состояний **q0** и **q1** мы подсчитываем количество символов **a**. После второго символа **a** мы возвращаемся в состояние **s**, так как это означает, что в данный промежуток буквы **a** добавлять не нужно.
q0[b] → q2[b]L **q1[b] → q2[b]L**	Встретив символ **b** в состоянии **q0** или **q1**, мы понимаем, что необходимо двигаться налево, чтобы добавить слева от данной буквы **b** как минимум одну букву **a**.
q2[b] → q3[a]L **q2[a] → q2[a]L** **q3[a] → q2[b]L** **q3[b] → q3[b]L**	Дальше считывающая движется налево. При этом первый символ, на который она попадает, заменяется на **a**, а потом каждый символ заменяется на тот, который стоял от него слева — мы добавили в слово букву **a**, для этого нам пришлось сдвинуть часть слова. Состояния **q2** и **q3** при этом используются для того, чтобы запомнить временно стёртый символ: состояние **q2** означает, что это был символ **a**, а состояние **q3** **—** что **b.**
q3[*] → s[b]N **q2[*] → s[a]N**	Встретив в состоянии **q2** или **q3** символ «звёздочка», мы понимаем, что считывающая головка дошла до левого края слова. На место звёздочки мы вписываем последнюю запомненную букву. После этого мы возвращаемся в начальное состояние, и начинаем работу уже с видоизменённым словом самого начала.
q0[*] → f[*]N **q1[*] → f[*]N** **s[*] → f[*]N**	Когда мы добираемся до правого края слова, мы должны завершить работу.

7. Последовательности 3 типа. (Текстовая задача)

В последовательности из букв **a** и **b** в любом наборе подряд идущих символов длины больше 3 количество букв **a** не меньше, чем количество букв **b**.

Докажите, что это условие эквивалентно тому, что в любом наборе из 5 или 7 подряд идущих символов количество букв **a** больше, чем количество букв **b**.

Замечение: эта формулировка содержит неточность. Указанная эквивалентность имеет место для последовательностей хотя бы из 5 символов.

Решение:
Пусть в последовательности хотя бы 5 символов.
Заметим, что второе условие очевидно следует из первого, поэтому нам нужно только доказать, что первое также следует из второго.

Любой набор из четырёх символов можно дополнить одним символом справа или слева, так чтобы получился набор из пяти символов, в котором количество букв **a** больше, чем количество букв **b**. Аналогично любой набор из шести символов превращается в набор из пяти символов откидыванием одного из крайних символов. Поэтому в этих наборах из 4 или 6 символов количество букв **a** не меньше, чем количество букв **b**.

Таким образом мы уже получили выполнение требуемого неравенства для любых наборов длины от 4 до 7 символов. Любой набор из большего числа символов можно представить в виде **4n + k,** где **k** — одно из чисел 4, 5, 6, 7, то есть он разбивается на наборы из 4, 5, 6 и 7 символов, в которых выполняется нужное неравенство, значит, во всём наборе неравенство также выполняется.

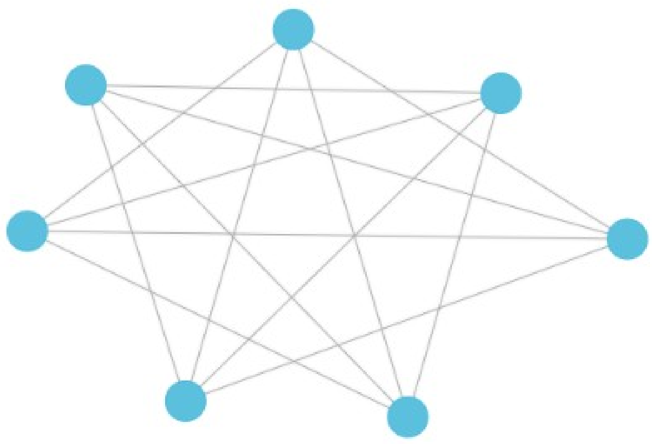
Значит, требуемое неравенство выполняется для любого набора хотя бы из 4 символов, что доказывает нужное утверждение.

8. Графы

Постройте граф, удовлетворяющий следующим условиям:
1) Из каждой вершины выходит ровно 4 ребра
2) В графе нет четырёх вершин, каждые две из которых были бы соединены между собой
3) Граф нельзя покрасить в три цвета "правильно", то есть так, чтобы любые две вершины одного цвета были не соединены.

3 балла ставится за пример, ещё 1 балл за доказательство того, что граф удовлетворяет второму и третьему условиям
Ещё 2 балла Вы получите, если Ваш пример минимальный по количеству вершин и Вы это докажете.

Решение:



Данный граф представляет из себя дополнение цикла на семи вершинах.
В цикле на семи вершинах среди любых четырёх вершин есть две соседних. значит, дополняющий его граф удовлетворяет условию 2).

При покраске такого графа в три цвета какие-то три вершины обязательно окажутся одного цвета. Цикл на семи вершинах не содержит треугольников, значит, в этом цикле какие-то две из этих трёх вершин не будут соединены. Следовательно, они будут соединены в дополняющем графе, а значит, никакая раскраска не является "правильной".

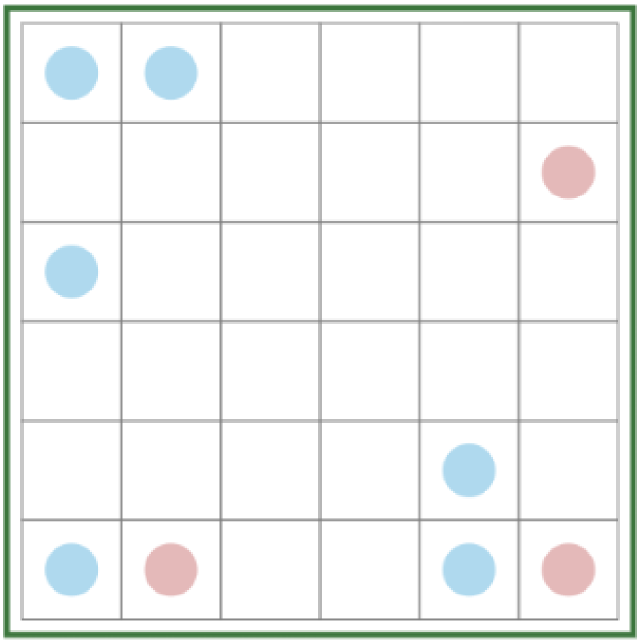
Пример минимальный, потому что граф на 5 вершинах, которые все имеют степень 4, это полный граф и он не удовлетворяет условию 2).
Граф на 6 вершинах, которые имеют степень 4, единственен. Для каждой вершины есть ровно одна, с которой та не соединена. Каждую пару таких вершин мы красим в один цвет, и получаем правильную раскраску в три цвета, следовательно, этот граф не удовлетворяет условию 3.

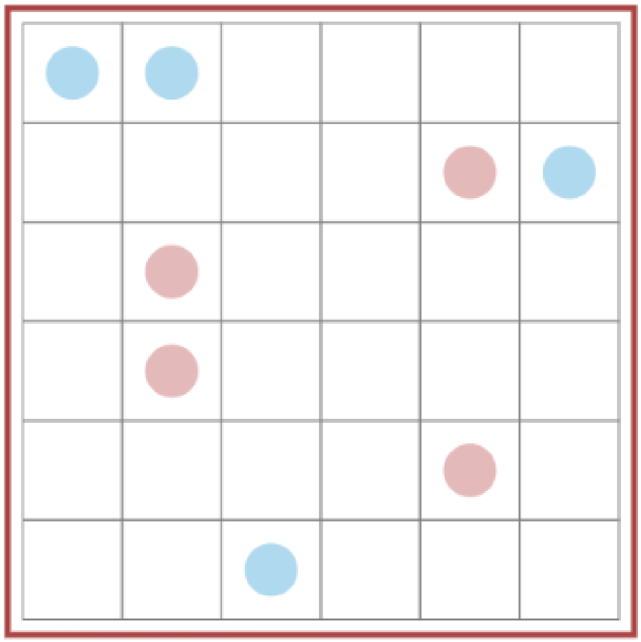
9. Мир Тарского

Напишите логическую формулу, описывающую свойство, которым обладает комбинация фишек на левой картинке, но не обладает комбинация на правой.

В формуле используются следующие обозначения:

Фишки обозначаются переменными x, y, z. Простые свойства описываются такими выражениями как "x синяя", "y красная", "x сосед y" (последнее означает, что фишки стоят на различных клетках, у которых есть общая сторона или угол). Для записи более сложных свойств используются логические связки, которые соединяют простые свойства:
И, ИЛИ, НЕ ВЕРНО ЧТО, СЛЕДОВАТЕЛЬНО, ТОГДА И ТОЛЬКО ТОГДА, ДЛЯ ВСЕХ x, СУЩЕСТВУЕТ x ТАКОЙ, ЧТО (вместо x можно использовать y или z).
Для упорядочения связок используются круглые скобки.





Решение:

Нетрудно заметить, что на левой картинке среди любых двух соседних фишек хотя бы одна синяя.

Соответственно, нас устраивают формулы вида "Если две фишки соседние, то одна из них синяя" или "Нет двух соседних красных фишек".

```
∀x  ∀y  ( СОСЕД(x,y) => ( СИНЯЯ(x) ИЛИ СИНЯЯ(y) ) )
```