Домашнее задание № 3

Гунаев Руслан, 776 группа

7 марта 2019 г.

1.

Пусть на вход первой программе подается множество

$$X = \{x_1, x_2, \ldots, x_n\}$$

, тогда она должна вывести последовательность номеров

$$P = \{p_1, p_2, \ldots, p_n\}$$

, таких что множество $X_P = \{x_{p_1}, x_{p_2}, \ldots, x_{p_n}\}$ является строго возрастающим.

Построим функцию f, которая на вход будет принимать множество X, а на выходе мы будем получать новое множество точек

$$X_Y = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

Определим, когда $|B(X_Y)| = n$, где B – алгоритм, решающий вторую задачу. Заметим, что если множество X_Y образует выпуклый n-угольник, с вершинами в точках из этого множества, то не существует набора из k точек из множества, образующих выпуклый k-угольник($k < n$), содержащий в себе или на своей границе все точки множества.

Допустим, что такое k нашлось, тогда обязательно найдется хотя бы одна точка, лежащая внутри или на границе k-угольника. В обоих случаях очевидно получим противоречие с условием того, что мы также можем построить выпуклый n-угольник.

Опишем работу функции. Пусть на вход подается множество X. Пробежимся по множеству и для каждого элемента выполним:

1. возьмем x_i.
2. создадим точку (x_i, x_i^2). Операция возведения в квадрат считается за константу.
3. положим эту точку в X_Y.

Очевидно, что данная функция выпоняет всю работу за линейное время.

В силу того, что функция $y = x^2$ выпуклая, точки, принадлежащие множеству X_Y, будут являться вершинами выпуклого n-угольника.

Покажем, что сводимость верна.

На вход первой программе подается X. На выходе мы получаем множество P.

На вход второй программе подается $X_Y = f(X)$. Как было описано ранее, точки из данного множества образуют выпуклый n-угольник. Так как все координаты x у точек разные, то сортировать точки мы должны только по этой координате, а значит вторая программа выведет ровно то же множество P.

Получили, что

$$A(X) = P = B(X_Y) = B(f(X))$$

. Сводимость построена.

2.

Покажем, что если $L \in \mathbf{NP} - \mathbf{complete}$, то $\overline{L} \in \mathbf{co} - \mathbf{NP} - \mathbf{complete}$.

1. Так как язык принадлежит $\mathbf{NP} - \mathbf{complete}$, то он принадлежит и самому $\mathbf{NP}$, а значит $\overline{L} \in \mathbf{co} - \mathbf{NP}$
2. возьмем произвольный язык $A \in \mathbf{NP}$, так как $L \in \mathbf{NP} - \mathbf{complete}$, то $A \leqslant_m L$.
3. $\exists f : x \in A \iff f(x) \in L$. В силу того, что если $x \in A \iff x \notin \overline{A}$, то в качестве сводимости $\overline{A}$ к $\overline{L}$ можем взять ту же функцию f.
4. Получили, что если $A \in \mathbf{NP}$ то $\overline{A} \leqslant_m \overline{L}$. Если $B \in \mathbf{co} - \mathbf{NP}$, то $\overline{B} \in \mathbf{NP} \Rightarrow \overline{B} \leqslant_m L \Rightarrow B \leqslant_m \overline{L}$.
5. Получили, что $\overline{L} \in \mathbf{co} - \mathbf{NP} - \mathbf{hard}, \overline{L} \in \mathbf{co} - \mathbf{NP} \Rightarrow \overline{L} \in \mathbf{co} - \mathbf{NP} - \mathbf{complete}$.

В обратную сторону доказывается аналогично, в силу того, что

$$\mathbf{co} - \mathbf{NP} = \{L | \overline{L} \in \mathbf{NP}\}$$

Получим

$$L \in \mathbf{NP} - \mathbf{complete} \iff \overline{L} \in \mathbf{co} - \mathbf{NP} - \mathbf{complete}$$

3.

Пусть граф $G(V, E)$ задан матрицей смежности. Построим функцию, которая будет идти по этой матрице и строить граф, дополняющий исходный граф до полного. Понятно, что функция будет полиномиальной.

Очевидно, что если в графе присутствует полный подграф(клика), то в дополнении графа вершины, образующие клику, будут попарно не соединены ребром. Верно и в обратную сторону. Это означает, что представленная полиномиальная функция действительно годится для сводимости как в одну сторону, так и в другую.

Таким образом,

$$CLIQUE \leqslant_m INDEP, INDEP \leqslant_m CLIQUE.$$

4.

a)

Пусть функция f принимает граф $G(V, E)$, добавляет новую вершину и делает ребра от новой вершины до всех остальных. Таким образом если в графе есть Гамильтонов путь, то в новом графе найдется и цикл, если пути нет, то в новом графе не будет цикла, потому что если он там есть, то в изначальном мы сможем найти путь, ведь отбрасывая любую вершину из цикла, останется путь без этой вершины.

Очевидно, что функция f обрабатывает граф за полиномиальное время, а значит

$$HAMPATH \leqslant_m HAMCYC$$

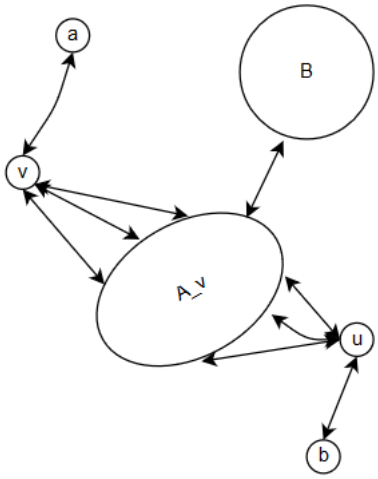
.

b)

Пусть функция получает на вход граф G. Предположим, что в графе нет цикла, но есть путь, тогда после работы функции должен остаться граф, не имеющий пути. Возьмем любую вершину v в графе и добавим новую u так, что она соединяется со всеми смежными с v вершинами. Добавим две вершины: одну вершину a соединим с v, другую b с u. Получили граф G'. Если в графе был цикл, то после добавления этих вершин точно останется путь.

$$a \to v \to \text{ вершины, образующие цикл } \to u \to b$$

Пусть в графе не было цикла. Обозначим множество A_v как множество смежных с v и соответсвтенно с u вершин. За множество B возьмем оставшиеся вершины.



Допустим, что при таком построении найдется Гамильтонов путь. Тогда данный путь должен начинаться и заканчиваться в вершинах a, b, ведь если эти вершины встречаются не в начале и не в конце, то их степень должна быть как минимум 2. Пусть начали в a, закончили в b. Единственная вершина в которую мы можем попасть из a – это v. Соответсвтенно, чтобы закончить путь, мы должны оказаться в вершине u, ведь только из нее можно попасть в b. Но тогда в G обязательно должен быть цикл, ведь если мы рассмотрим наш путь и уберем начальную и конечную вершину, то получим путь начинающийся с v и заканчивающийся в u. Но эти точки полностью эквивалентны по построению. Это означает, что можно в исходном графе можно предложить путь, начинающийся с v и заканчивающийся в v, а это цикл. Противоречие.

Таким образом, если в графе был цикл, то после работы функции там будет и путь. Если цикла не была, то не будет и пути. Очевидно, что время работы такой функции полиномиально, ведь она всего лишь добавляет 4 вершины и дублирует некоторые ребра.

Получим, что

$$HAMCYC \leqslant_m HAMPATH$$

5.

Построим таблицу. По данной таблице построим КНФ и ДНФ.

1. КНФ.

   $$f(y,a,b) = (y \vee a \vee \overline{b}) \wedge (y \vee \overline{a} \vee b) \wedge (y \vee \overline{a} \vee \overline{b}) \wedge (\overline{y} \vee a \vee b)$$

   Заметим, что если мы производим данную замену, то появляется 4 новых конъюнкта, всего в записи можжет быть максимум n замен, получим, что алгоритм, производящий следующую замену является полиномиальным.

2. ДНФ.

$$f(y,a,b) = (\overline{y} \wedge \overline{a} \wedge \overline{b}) \vee (y \wedge \overline{a} \wedge b) \vee (y \wedge a \wedge \overline{b}) \vee (y \wedge a \wedge b)$$

Аналогично первому пункту видим, что функция полиномиальна.

y	a	b	$f(y,a,b)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

А значит, что сводимость тоже полиномиальна.

6.

Задача / Запись	Выполнимость	Тавтологичность
КНФ	**NP – complete**	**P**
ДНФ	**P**	**co – NP – complete**

КНФ. Выполнимость.

По теореме Кука-Левина задача SAT является **NP – complete**.

КНФ. Тавтологичность.

Рассмотри язык КНФ, таких, что существует набор, на котором формула равна 0. Для этого достаточно, чтобы хотя бы один дизъюнкт был равен нулю. Приведем алгоритм аналогичный алгоритму из следующего пункта, который является полиномиальным.

Заметим, что данный язык является дополнением языка тавтологичных КНФ, а значит , в силу того что класс полиномиальных языков замкнут относительно дополнения, и этот язык лежит в классе P.

ДНФ. Выполнимость.

Пусть у нас есть ДНФ. Для выполнимости формулы достаточно, чтобы хотя один конъюнкт был выполним. Пробежимся по всей формуле, если в каждом конъюнкте присутствует переменная и ее отрицание, значит формула не выполнима, для этого затратим $O(n^2)$ времени. Если нашелся конъюнкт, в котором каждая переменная встречается либо один раз, либо нет переменной и ее отрицания, то формула выполнима. Если в этом дизъюнкте есть переменная x_i, то для этой перемеменной возьмем значение 1, если встречается $\overline{x_j}$, то для нее возьмем 0, для переменных, которые не встретились в этом конъюнкте положим произвольные значения, таким образом, проверка на выполнимость ДНФ является полиномиальной.

ДНФ. Тавтологичность.

Рассмотрим язык ДНФ, для которых существует набор, на котором функция принимает значение 0. Назовем его L.

Он является **NP**, так как в качестве сертификата можем привести набор, на котором принимается значение 0.

Построим сводимость $SAT \leqslant_m L$. Заметим, что отрицание КНФ есть ДНФ. Таким образом, функция просто будет бежать по КНФ и строить ее отрицание(вместо конъюнкции ставить дизъюнкцию и наоборот, также переменную менять на ее отрицание). Получим, что если КНФ на каком-то наборе принимает значение 1, то ее отрицание на этом наборе будет равно 0. Если КНФ на всех наборах принимает значение 0, то отрицание на всех наборах будет равно 1. Построили полиномиальную сводимость, таким образом, $L \in \mathbf{NP - complete}$.

Заметим, что язык тавтологичных ДНФ ялвяется дополнением языка L. В силу второй задачи язык тавтологичных ДНФ ялвяется $\mathbf{co - NP - complete}$.

7.

Пусть есть КНФ, в каждом дизъюнкте которого не больше 3 переменных.

Будем бежать по записи функции. Если видим дизъюнкт из 1 элемента делаем слудующее. Не нарушая общности, пусть мы встретили x_1.

$$x_1 = ((x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3 = x_1))$$

$$(x_1 \vee x_2 \vee x_3 = x_1) = ((x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}))$$

Мы добавили две переменные и 4 дизъюнкта длиной 3. Таким образом, если есть дизъюнты из одного элемента мы за полиномиальное время можем сделать дизъюнкты из 3 элементов.

Пусть встретили дизъюнкты длиной 2. Сделаем аналогично

$$x_1 \vee x_2 = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x_3})$$

Добавили 1 переменную и два дизъюнкта длиной 3.

Если среди дизъюнктов нашелся дизъюнкт, в котором встречается x_i и $\overline{x_i}$, то данный дизъюнкт всегда дает 1, значит его из записи можем убрать.

Таким образом задачу $3 - SAT$ мы свели к $EXACTLY - 3 - SAT$.

8.

Мы показали, что $3 - SAT \leqslant_m EXACTLY - 3 - SAT$. Поэтому будем сводить $EXACTLY - 3 - SAT$ к $3 - COLOR$.

Обоснуем корректность алгоритма. Пусть исходная формула выполнима. Тогда в каждый дизъюнкт обязан входить хотя бы один истинный литерал, а значит в каждом построенном подграфе вершина, выходящая на палитру, будет попадать на черный цвет. Это правда, так как в таком случае мы сможем именно так раскрасить граф.

Пусть теперь мы можем покрасить построенный граф. В силу построения каждый подграф может выходить только на черную вершину, а значит этот подграф не может быть построен только на белых вершинах, следовательно, если покраска существует, то если мы выберем все вершины черного цвета и перекодируем так, что если $\chi(x_i) = B, \chi(x)$-функция, показывающая цвет вершины, то в качестве x_1 возьмем 1,соответственно для $\chi(\overline{x_i}) = 0$

9.

Проверим, можно ли покрасить граф, если нет, выводим, что покраски не существует, если есть продолжаем алгоритм. Будем добавлять ребра в граф до тех пор, пока алгоритм проверки на раскраску не скажет нет. Если он сказал нет, то можем красить вершины, являющиеся концами последнего добавленного ребра, в 1 цвет, поэтому объединим их в одну. Уберем все

ранее добавленные ребра. Далее сделаем аналогичное для графа с $n-1$ вершиной. В конце алгоритма получим три множества вершин, которые покрасим произвольно. На каждом шаге мы можем добавить $O(n^2)$ ребер, поэтому сложность алгоритма $O(n^{3+c})$, где c – это сложность проверки на раскраску.

Допустим, что была какая-то раскраска графа, мы добавили ребро, и алгоритм сказал нам, что раскраска все равно возможна. Это значит, что если изначально была раскраска, то после добавления будет два варианта: либо раскраска останется такой же, либо поменяется. Если раскраска осталась, какая и была, то это значит, что вершины, которые соединены новым ребром разного цвета. Если раскраска поменялась, то даже если мы уберем это ребро, данная раскраска все равно будет подходить изначальному графу. Получили,что добавление нового ребра таким образом на любом этапе никак не влияет на раскраску графа в случае положительного ответа алгоритма. Это значит, что алгоритм корректен.

10.

a)

Заметим, что $INDSET \leqslant_m VERTEX-COVER, VERTEX-COVER \leqslant_m INDSET$. Если вершинное покрытие задело все рёбра, то между оставшимися вершинами рёбер быть не может, иначе эти рёбра не были бы задеты. И наоборот, если между оставшимися рёбер нет, то все рёбра покрыты взятыми. А значит дополнение к любому вершинному покрытию из k вершин есть независимое множество из $n-k$.

Покажем, что $INDSET \in NP$. В качестве сертификата возьмем набор из k вершин. За время $O(k^2)$ можно проверить, является ли данное множество вершин независимым.

Построим сводимость $EXACTLY-3-SAT \leqslant_m INDSET$. Пусть имеется функция 3-КНФ. Для каждого вхождения литерала будем строить вершину. Всего получим $3m$ вершин, если всего имеется m дизъюнктов. Все литералы из одного дизъюнкта соединим ребрами, также соединим все противоположные литералы.

Если формула выполнима, то в силу того, что это КНФ, в каждом дизъюнкте обязан быть хотя бы один истинный литерал.

Заметим, что количество вершин, образующих независимое множество не больше k, ведь тогда хотя бы в одном подграфе, соответствующему какому-то дизъюнкту, есть две вершины из этого множества, но они попарно соединены.

Покажем, что количество вершин не меньше k. Пусть меньше. Тогда найдется подграф такой, что каждая его вершина соединена с какой-то вершиной из другого подграфа,которая входит в множество независимых верших, а значит в нем нет истинных литералов.

Если, наоборот, есть независимое множество из k вершин, то эти вершины обязаны соответствовать различным дизъюнктам. А поскольку среди этих вершин нет одновременно литералов вида x и $\overline{x}$, то можно взять такие значения переменных, при которых все задействованные литералы истинны. Эти значения и будут выполняющим набором.

Очевидно, что сводимость полиномиальна. Учитывая предыдущие задачи, получили, что

$$INDSET, VERTEX-COVER \in \mathbf{NP-complete}.$$

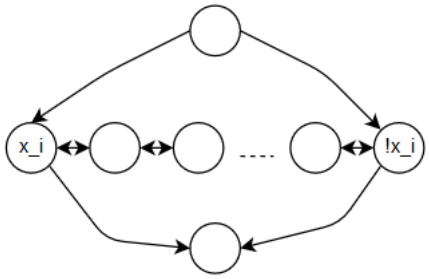
b)

Построим сводимость $3-SAT \leqslant_m HAMP$. В качестве сертификата для задачи $HAMP$ можно взять вершины этого графа в порядке обхода, это будет проверяться за полиномиальное время, нам нужно просто пройтись по матрице смежности и посмотреть существуют ли соответственные ребра.

По 3−КНФ построим граф. Пусть есть n переменных и m дизъюнктов. Для каждой переменной построим по 2 вершины, она для литерала, другая для его отрицания. Так же между

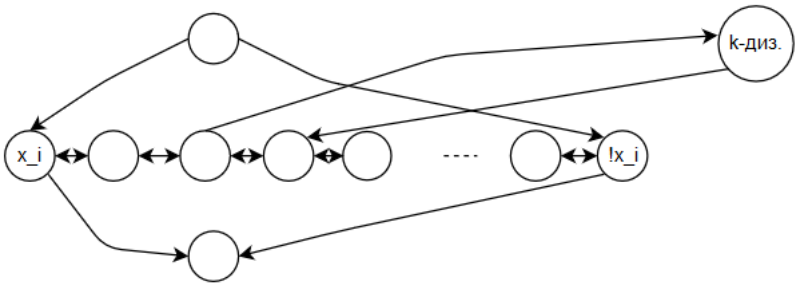
этими вершинами построим еще $3m + 1$ вершину. Помимо этого построим m вершин, соответствующие дизъюнктам.

На рисунке подграф для каждой вершины.



Нам нужно, чтобы каждый дизъюнкт был истинен, этому будет соответсвовать переход из подграфа какой-то переменной в соответствующую вершину дизъюнкта. Построим ребра следующим образом. Если какая-то переменная входит в некоторый дизъюнкт без отрацания, то проводим из этого подграфа два ребра: одно из промежуточной вершины с номером $3k - 1$ в k-ую вершину дизъюкта, и из него проведем ребро в $3k$ промежуточную вершину.

Пример:



Если переменная входит с отрицанием, то делаем наоборот из $3k$ в $3k - 1$.

Подграфы соединим ребром из нижней вершины i-ого подграфа в верхнюю вершину $i + 1$-ого подграфа. Обход будем начинать в самой верхней вершине итогового графа, закончим в самой нижней.

Корректность.

Пусть формула выполнима, значит в каждом дизъюнкте присутствует хотя бы один истинный терминал. Таким образом, зная набор, можно показать весь путь. Если переменная вошла в набор как ложь обходит подграф , спускаясь с верхней вершину в крайнюю правую, далее идет к крайней левом, а потом вниз. Если вошла как истина, то делаем аналогично, только в противоположные стороны. Если переменная сделала какой-то дизъюнкт истинным, то во время обхода подграфа мы должны сделать переход в соответствующем месте в соответствующий дизъюнкт, а потому сразу же вернуться обратно в тот же подграф. Заметим, что так как формула выполнима, то песещена каждая вершина–дизъюнкт, так как посещение ее эквивалентно тому, что это истина. Также при таком обходе мы точно проходим каждую вершину подграфа. Значит все вершины пройдены ровно один раз.

Пусть теперь в графе существует Гамильтонов путь. Существает два варианта. Либо, попадая в какую-то вершину-дизъюнкт, мы перещли в другой подграф, либо в тот же. Рассмотрим первый случай. Допустим, что находясь каком-то подграфе мы попали в вершину-дизъюнкт, а пото пришли в другой. Заметим, что в каждом подграфе существуют вершины, которые не ведут никуда вне этого подграфа, также нельзя попасть в эту вершину, не находясь в подграфе, а именно промежуточные вершины с номерами $3t$. Пусть мы попали в подграф, нам нужно

также посетить верхние и нижние вершины, но находясь в промежуточных, мы в силу построения за раз сможем посетить только одну из них, поэтому нам еще раз придется возвращаться в этот подграф, но теперь мы прошли часть промежуточных вершин. Пусть мы снова попали в этот подграф, мы не можем двигаться в одну из сторон(влево или вправо), так как какие-то вершины уже посещены, но так же заметим, что если мы попали туда из вершины дизъюнкта, а здесь возможно только это, то остается две вершины с номерами кратными 3(их может быть больше, но нас интересуют только те две, что находятся непосредственно слева и справа), но посетить мы можем только одну, так как иначе нам придется заходить в уже посещенные вершины. Значит такой вариант быть не может.

Пусть теперь, посещая каждую вершину-дизъюнкт, мы попадаем в тот же подграф. Проделая путь, мы можем с легкостью восстановить набор. Если подграф обходится справа, то берем соотствующую переменную в набор и ставим ей значени 0. Если слева – 1. Такой набор будет точно делать каждый дизъюнкт истинным по построению графа, значит формула выполнима.

В силу этого и предоставленного сертификата следует, что $HAMP \in \mathbf{NP-complete}$.

c)

Построим сводимость $3-SAT \leqslant_m SUBSETSUM$. В качестве сертификата приведем элементы из подмножества. Проверка правильности займет линию, просто сложим эти числа и сравним с нужным, проверка принадлжености предлагаемых чисел множеству займет $O(n^2)$.

Пусть всего n переменных и m дизъюнктов. Надо зашифровать каждую переменную и каждый дьзъюнкт в десятичные числа. Если набор, приводящий к выполнению функции, существует, то существует последовательность из нулей и единиц длины n. Каждой переменной сопоставим два числа. v_i, u_i оба будут состоять из $m+n$ цифр. Для первого чисел в разрядах $m+i$ будут стоять 1, в остальных старших – 0. Если x_i встречается в записи j-го дизъюнкта, то поставит в разряд $j-1$, в остальных – нули. Если $\overline{x_i}$ встречается в $k-$ом дизъюнтке, то поставит на соответственный разряд единицу, в остальные нули. Для каждого дизъюнкта также создадим 2 числа такой же длины. e_i, t_i. У первого в разряде, соответствующему номеру дизъюнкта будет стоять единица, у второго двойка. Заметим, что если функция выполнима, и для каждой переменной в зависимости от ее значения в истинном наборе, а именно если переменная имеет значение 1, то берем v_i, иначе u_i, то просуммировав взятые числа получим число вида

$$s' = \underbrace{111\ldots1}_{n}\underbrace{t_{i_1}t_{i_2}\ldots t_{i_k}}_{k}, t_j \in \{1,2,3\}$$

t_j не может равняться нулю, ведь это будет означать, что данный дизъюнкт не принимает значение 1 на таком наборе. Ведь если x_i есть в этом дизъюнкте, то у числа v_i на соответственном порядке должна стоять 1, а раз в итогой сумме там стоит 0, то в наборе, на котором выполняется данная функция, для переменной x_i взято значение 0, а значит она не дает истинность данному дизъюнкту и так для каждой переменной входящей в дизъюнкт с отрицанием или без.

Также заметим, что t_j не может превосходить 3. Ведь всего в дизъюнкте три переменные, а значит и чисел(v_i или u_i), у которых на данном месте стоит единица, может быть тоже максимум 3.

Заметим, что если мы возьмем $s = \underbrace{111\ldots1}_{n}\underbrace{44\ldots4}_{k}$, то мы всегда число s' можем добить до числа s, просто взяв в нужном количестве числа e_i, t_i.

Пусть теперь из такого из набора всех этих чисел мы можем взять какой-то набор M чисел , в сумме дающих s. Тогда очевидно,что $\forall i \in \{1,\ldots,n\} \hookrightarrow v_i$ или $u_i \in M$. Если таких чисел меньше, то мы не получим всех единиц на высших n разрядах, если больше, то найдется i такой, что для соответсвенной переменной мы взяли оба числа, а значит в сумме точно будет двойка в одном из старших разрядов. Значит раз можно выбрать набор, то оставшиеся числа – это числа, соответсвующие дизънктам. По такому набору чисел мы легко можем восстановить нужный набор переменных, на которых выполняется функция.(очевидно по построению.)

Понятно, что такая сводимость полиномиальна.

В силу этого и предоставленного сертификата следует, что

$$SUBSETSUM \in \mathbf{NP} - \mathbf{complete}.$$

11.

Пусть в графе есть цикл нечетной длины. Не нарушая общности, пусть начальная и соответсвтенно конечная вершина этого цикла покрашена в белый цвет. Значит все вершины с номерами $1 + 2n$ будут также покрашены в белый цвет. Пусть длина цикла равна $1 + 2k$, а значит, что цвета последней(она же первая) и предпоследней одинаковы, а значит мы не можем покрасить граф в два цвета.

Пусть в графе нет цикла нечетной длины. Построим остовное дерево по алгоритму Прима за полином. Так как это дерево, то в нем нет циклов, значит между любыми двумя вершинами существует только один путь. Покрасим корень дерева в красный цвет. Если из корня до некоторой вершины путь имеет четную длину, то также покрасим ее в красный, иначе в синий. Допустим, в исходном графе не существует 2-покраски. Это значит, что найдется ребро соединяющее две вершины одинакового цвета. Пусть это вершины a, b. Рассмотрим маршрут $a \to v \to b \to a$. Длина маршрута равняется $t+p+1 = 2m+1$, так как t, p одной четности, а значит выделим цикл нечетной длины. Это правда, так как даже если какой-то участок этого маршрута был пройден дважды, то мы можем рассмотреть другой замкнутый маршрут,исключающий повторяющийся участок, без повторений, длина которого также будет нечетна.

Предложим алгоритм поиска цикла нечетной длины.

Построим остовное дерево с помощью алгоритма Прима. Далее покрасим вершины аналогично предыдущему пункту. Далее будем добавлять к нашему дереву недостающие вершины исходного графа, каждый раз проверяя, не совпадают ли цвета вершин, которые соединены этим ребром. Пусть это вершины a, b. Это займет линейное время. Если такая вершина нашлась, проделаем следующее: пройдемся по дереву из корня в обе вершины и выпишем эти пути. Найдем вершину u, до которой эти пути совпадают, такое место обязательно найдется, хотя бы потому, что оба пути начинаются в корне. В качестве ответа выведем $b, a, \{$вершины из первого пути от a до u$\}, \{$вершины из второго пути от u до b$\}$

Данный алгоритм полиномиальный.

12.

Сначала положим $x_1 = 0$. После такого допущения, часть скобок либо сократится, либо останутся дизъюнкты из 1 переменной. Эти дизъюнкты должны равняться 1, поэтому дальше делаем аналогичные замены. Если на каком-то этапе мы встретили два дизъюнкта $x_i \wedge \overline{x_i}$, то должны вернуться в начало и положить $x_1 = 1$, если и здесь получим противоречие, то формула невыполнима. Пусть после замены первой переменной и последующих вынужденных заменах переменных, которые представляли из себя одиночный дизъюнкт, в формуле остались лишь двойные дизъюнкты, то мы получили новую формулу, выполнимость которой эквивалентна выполнимости изначальной формулы.

Докажем это. Пусть новая формула $f(x_k, x_{k+1}, \ldots, x_n)$выполнима, то есть существует набор, на котором она равна 1. Заметим, что эту формулу мы получили, подставляя конкретные значения для первых k переменных, а значит если мы соединим набор оставшихся переменных и набор первых переменных, то на нем будет выполнима и исходная формула. Пусть теперь новая формула невыполнима, то есть на любом наборе она принимает значение 0. Так как эта формула представляет из себя конъюнкцию двойных дизъюнктов, то это значит, что в ходе работы программы мы уже проверили всевозможные наборы первых переменных, на которых изначальная функция могла выполняться. Рассмотрим набор σ, который привел нас к этой

формуле. Заметим, что все остальные наборы вели нас к противоречию. Набор σ единственный кандитат, на котором наша формула может выполняться, но в тоже время подставляя эти значения переменных в исходную формулу, мы получим невыполнимую формулу, а значит исходная формулу и правда невыполнима. Следовательно, данный алгоритм корректен.

Оценим асимптотику.

После подстановки первой переменной, мы могли дойти до формулы из одной переменной и получить противоречие, таким образом, совершив $2n$ подстановок и проверок. То есть на каждом шаге нашей программы мы совершаем максимум $O(n^2)$ операций. Получим, что

$$T(n) \leqslant T(n-1) + O(n^2) \Rightarrow T(n) = O(n^3)$$

13.

Пусть функция выполнима. Запишем условие из вершины a есть путь в $\overline{a}$ и наоборот.

$$(a \to \overline{a}) \wedge (\overline{a} \to a)$$

Но эта функция невыполнима, а значит это невыполнимо для любой вершины. Но так как исходная функция выполнима, получаем противоречие.

Пусть функция невыполнима. Аналогично предыдущему заданию, будем также брать первую переменную и подставлять значение, так как функция невыполнима, то на любом таком наборе мы будем получать противоречие. То есть в какой-то момент мы точно получим что-то вида $a \wedge \overline{a}$, но

$$a \wedge \overline{a} = \overline{a \to \overline{a}}$$

Последняя запись эквивалента отсутсвию хотя бы одного из путей $a \to \overline{a}, \overline{a} \to a$.

А это значит, что если функция невыполнима, то найдется вершина, для которой не существует хотя бы одного из путей.

Оценим асимптотику. Сначала строится граф, что занимает линейное от длины формулы время. Далее запускаем алгоритм поиска в ширину для каждой вершины, если для любой a нет пути в $\overline{a}$ и наоборот, то формула выполнима. Получим асмиптотику

$$T(|f|) = O(|f|^2).$$