Задание на четвертую неделю.

№0

См. в прошлом задании номер 9.

№1

(i) Из условия очевидно, что:

$$\phi \in L \Leftrightarrow \exists \vec{x} : \forall \vec{y} \to \phi(\vec{x}, \vec{y}) = 1.$$

$\Rightarrow$ Из определения $L \in \Sigma_2$.
(ii) Возьмем следующую задачу: *Язык булевых формул от трех наборов переменных* $\varphi(x_1, \ldots, x_n, y_1 \ldots y_n, z_1 \ldots z_n) = \varphi(\vec{x}, \vec{y}, \vec{z})$ *таких, что при некоторых значениях* $\vec{x}, \vec{z}$ *они справедливы вне зависимости от значений* $y_1, \ldots, y_n$.

$$\phi \in L \Leftrightarrow \exists \vec{x} : \forall \vec{y} \to \exists \vec{z} : \phi(\vec{x}, \vec{y}, \vec{z}) = 1.$$

$\Rightarrow$ Из определения $L \in \Sigma_3$.
(iii)

$$L \in \Sigma_k \Longleftrightarrow (\exists y_1, \ldots y_k, R(x, \vec{y}) : R \in P, \forall i |y_i| = poly(|x|), x \in L \Leftrightarrow$$

$$\Leftrightarrow \exists y_1 \forall y_2 \exists \ldots y_k \to R(x, y_1, \ldots, y_k) = 1) \Longleftrightarrow (\exists y_1, \ldots y_k, y_{k+1}, R(x, \vec{y}) :$$

$$R_1 \in P, \forall i |y_i| = poly(|x|), x \in L \Leftrightarrow$$

$$\Leftrightarrow \forall y_{k+1} \exists y_1 \forall y_2 \exists \ldots y_k \to R_1(x, y_1, \ldots, y_k, y_{k+1}) = R(x, y_1, \ldots, y_k) = 1).$$

$\Rightarrow$ Из определения $L \in \Pi_{k+1}$.
Аналогично,

$$L \in \Sigma_k \Longleftrightarrow (\exists y_1, \ldots y_k, R(x, \vec{y}) : R \in P, \forall i |y_i| = poly(|x|), x \in L \Leftrightarrow$$

$$\Leftrightarrow \exists y_1 \forall y_2 \exists \ldots y_k \to R(x, y_1, \ldots, y_k) = 1) \Longleftrightarrow (\exists y_1, \ldots y_k, y_{k+1}, R(x, \vec{y}) :$$

$$x \in L \Leftrightarrow \exists y_1 \forall y_2 \exists \ldots y_k (\forall ? \exists) y_{k+1} \to R_1(x, y_1, \ldots, y_k, y_{k+1}) = R(x, y_1, \ldots, y_k) =$$

где ? означает, что может быть один из кванторов (это ни на что не влияет, т. к. R_1 не зависит от y_{k+1}). $\Rightarrow$ Из определения $L \in \Sigma_{k+1}$.
(iv) Покажем, что $3SAT \in PSPACE$:
Пусть дана формула из n переменных. Для каждого из 2^n возможных набора значений переменных проверим истинность этой формулы на нем (это делается за $O(n)$ пространства), освобождая после каждой проверки использованное пространство. $\Rightarrow 3SAT \subseteq PSPACE$.
Т. к. 3SAT это $NP-complete$, то значит любую задачу из NP можно решить за PSPACE, сведя полиномиально к 3SAT. $NP \subseteq PSPACE$.
В машине Тьюринга с полиномиальным числом ячеек не более экспоненциального числа всесвозможных конфигураций. Значит МТ справится, перебрав их, за экспоненциальное число шагов. $\Rightarrow PSPACE \subseteq EXPTIME$.

№2

Рассмотрим матрицу $n \times n$, где

$$a_{xy} = \begin{cases} 1, & \text{элементу с номером } i_x \text{ можно занимать позицию } j_y \\ 0, & \text{элементу с номером } i_x \text{ запрещено занимать позицию } j_y \end{cases}$$

Тогда перманент этой матрицы равный

$$Per = \sum_{k=1}^{n!} a_{1y_1(k)} \dots a_{ny_n(k)},$$

где $y_1(k), \dots y_n(k)$ это k-ая перестановка n множества чисел $\{1, , n\}$, и есть искомое в задаче значение: т. к. произведение $a_{1y_1(k)} \dots a_{ny_n(k)}$ отлично от нуля и равно единицы тогда и только тогда, когда k-ая перестановка такая, что соответствующая ей перестановка элементов допустимая (т. е. все элементы $i_{y_x(k)}$ стоят на разрешенных местах j_x), иначе $a_{xy_x(k)}$ равнялось бы нулю. Таким образом, количетво единичных слагаемых в сумме равно кол-ву допустимых перестановок из n элементов и равно перманентну введенной матрицы.

№3

Покажем полиномиальный алгоритм, решающий задачу L выполнимости ДНФ:
Пусть дана формула с n переменными и m конъюнктами, чтобы проверить ее выполнимость будем для каждого конъюкта проверять, есть ли в нем противоположные литералы (т. е. x и $\bar{x}$). Формула выполнима тогда и только тогда, когда нашелся хотя бы один конъюкт, в котором нет противоположных литералов. Проверка конъюкта занимает не более n^2, попарных сравнений элементов в конъюкте. Всего конъюктов m, значит сложность алгоритм $O(n^2m)$, т. е. полиномиальный алгоритм построен. $\Rightarrow L \in P$.
Рассуждение, что *любую КНФ можно преобразовать в эквивалентную ДНФ, поэтому задача выплнимости КНФ сводится к задаче выполнимости ДНФ и лежит в P*, не правильно, т. к. не учитывает сложность сведения КНФ к ДНФ (она может быть экспоненциальна).

№3.5

Как уже раньше было показано проверка выполнимости CNF это $NP - complete$. Из прошлой задачи, проверка выполнимости DNF это P.
Проверить формулу ϕ на тавтологию $\Longleftrightarrow$ проверить $\bar{\phi}$ на выполнимость. С помощью правил Моргана, если ϕ имеет форму CNF или DNF, ее отрицание можно за полиномиальное время преобразовать в DNF или CNF соответственно. Т. к. проверка выполнимости CNF это $NP - complete$, то проверка на тавтологию DNF $co - NP - complete$. Аналогично, проверка на тавтологию CNF это P.

№5

Да, останется. Покажем как свести к этому языку L обычный язык $3SAT$: Пусть переменная x повторяется формуле ϕ k раз. Вместо каждого вхождения x поставим переменные $x_1, \dots, x_k$ и добавим дизъюнкт $(x_1 \vee \bar{x_2}) \wedge (x_2 \vee \bar{x_3}) \dots ((x_{k-1} \vee \bar{x_k}) \wedge (x_k \vee \bar{x_1})$. Эту операцию повторим для всех переменных, входящих в формулу больше двух раз. Это и будет функция преобразования (она очевидно полиномиальна по времени). В полученной формуле каждый литерал содержится не более двух раз. Каждый присоединенный дизъюнкт верен тогда и только тогда, когда новые перемены, заменяющие x, равны между собой. Значит, если ϕ принимало истинные значения на некотором наборе, то преобразованная формула так же будет истинна на наборе значений, где заменяющим переменным присвоено значение переменной из старого набора, которую они заменяют. И наоборот. Таким образом, $3SAT \leq_p L$, а значит L полный язык. Т. е. сведение выполнено.

№6

Пусть функция f такая, что $f(G) = G$, если в графе $< 2k$ вершин или G это вообще не граф, а иначе $f(G) = G'$, где G' граф, полученный из G добавлением $V - 2k$ вершин (V это количество вершин в G), соединенных попарно со всеми остальными вершинами. f - полиномиальная функция.
Если в графе G есть клика размером k и $V \geq 2k$, то в графе G' вершины из этой клики вместе с новодобавленными вершинами образуют клику размером $V - 2k + k$, а это половина вершин в новом графе из $V - 2k + V$ вершин. $\Rightarrow G \in L_k \Longrightarrow f(G) \in L_{0.5}$.
Если в графе G' с $2V - 2k$ вершинами есть клика размером хотя бы $V - k$, то после того как мы уберем добавленные $V - 2k$ вершин (применим f^{-1}, размер клики уменьшится не более чем на $V - 2k$, т. е. в графе G есть клика размером k.
$\Rightarrow G \in L_k \Leftrightarrow f(G) \in L_{0.5}$. Значит, $L_k \leq_p L_{0.5}$

№4

Из матанализа $k^\alpha \leq \int_k^{k+1} x^\alpha dx \leq (k+1)^\alpha$. $\Rightarrow \int_0^k x^\alpha dx \leq \sum_{k=1}^n k^\alpha \leq \int_1^{k+1} x^\alpha dx$. $\Rightarrow k^{\alpha+1} \cdot \frac{1}{\alpha+1} \leq \sum_{k=1}^n k^\alpha \leq (k+1)^{\alpha+1} \cdot \frac{1}{\alpha+1} - \frac{1}{\alpha+1}$.
$\Rightarrow \sum_{k=1}^n k^\alpha = \Theta(k^{\alpha+1})$. В частности:

$$\sum_{k=1}^{n} \sqrt{k} = \Theta(k^{3/2})$$

.

№7 а)

Да, верно, например $f(n) = n^{\log_2 n}$:

- $\frac{n^c}{n^{\log_2 n}} = \frac{1}{n^{c - \log_2 n}} \longrightarrow 0$ при $n \to \infty$. $\Rightarrow f(n) = \omega(n^c)$.
- $\sqrt{\log f(n)} = \sqrt{\log^2 n} = \log_n = O(\sqrt{n}) = O(\sqrt{\log f(2^{nd}}). \Rightarrow f(n) = O(2^{nd})$.

№7 б)

Нет, неверно, т. к. не ясно, можно ли свести полиномиально DNF к 4DNF. Если такой все-таки такой алгоритм есть, то тогда из задачи это верно, т. к. вытекает из задачи 3.