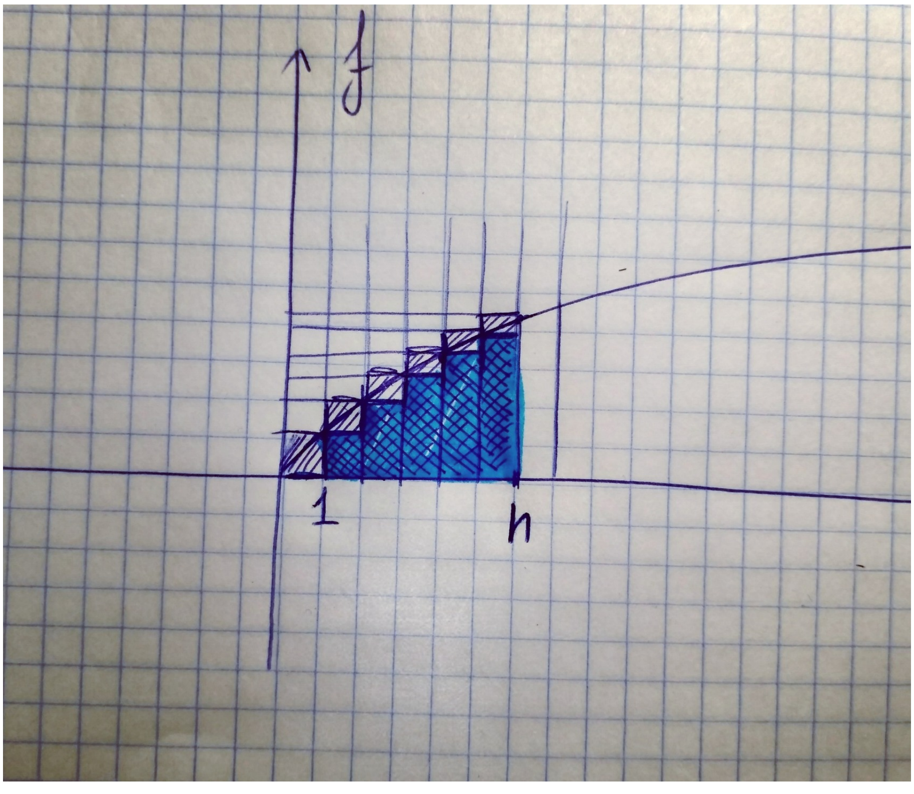
Филиппова Анастасия

Домашнее задание 4

0.
$3911 - 1 = 2 \cdot 5 \cdot 17 \cdot 23$, найдем первообразный корень по модулю 3911 : проверим 13 : $13^2 = 169, 13^4 = 1184 \; mod \; 3911, 13^8 = 1718 \; mod \; 3911, 13^{16} = 2630 \; mod \; 3911, 13^{32} = 2252 \; mod \; 3911, 13^{64} = 2848 \; mod \; 3911, 13^{128} = 3601 \; mod \; 3911, 13^{256} = 2236 \; mod \; 3911, 13^{512} = 1438 \; mod \; 3911, 13^{1024} = 2836 \; mod \; 3911$,
$13^{2 \cdot 23 \cdot 17} = 13^{782} = 13^{512+256+8+4+2} \; mod \; 3911 = 3349 \equiv 1 \; mod \; 3911$,
$13^{2 \cdot 5 \cdot 17} = 13^{170} = 13^{128+32+8+2} = 1790 \; mod \; 3911$,
$13^{2 \cdot 5 \cdot 23} = 13^{230} = 13^{128+64+32+4+2} = 1346 \; mod \; 3911$,
$13^{5 \cdot 17 \cdot 23} = 13^{1955} = 13^{1024+512+256+128+32+2+1} = 3910 \; mod \; 3911$, следовательно, 13 — первообразный по модулю 3911.
Действуем по алгоритму и перебираем простые делители: 1) $2, 5$ — простые.
2) 17: первообразный по модулю 17 3, проверка: $3^2 = 9 \; mod \; 17, 3^4 = 81 = -4 \; mod \; 17, 13^8 = 16 = -1 \; mod \; 17, 3^{16} = (-1)^2 = 1 \; mod \; 17$, при этом $16 = 2^4$, 2 — простое. Следовательно, 17 простое.
3) 27 : проверим, яыляется ли 5 первообразным корнем по модулю 23: $22 = 11 \cdot 2$, $5^2 = 25 = 2 \; mod \; 23, 5^{11} = 5^{5 \cdot 2+1} = 32 \cdot 5 = 160 = -1 \; mod \; 23$, проверяем 11 : $2^{1024} = 1 \; mod \; 11, 10 = 2 \cdot 5, 2^2 = 4 \; mod \; 11, 2^5 = 32 = -1 \; mod \; 11 \Rightarrow 11$ простое. Таким образом построили сертификат простоты для $p = 3911$.
4. Как показано на рисунке снизу, сумма (синий цвет) $\sum_{k=1}^{n} k^{1/2} < \int_1^n x^{1/2} dx = (n^{3/2} - 1)/2 = O(n^{3/2}) \Rightarrow \sum_{k=1}^{n} k^{1/2} = O(n^{3/2}$. С другой стороны (на рисунке заштриховано ручкой) $\sum_{k=1}^{n} k^{1/2} + 1 = \sum_{k=1}^{n} (k+1)^{1/2} = \sum_{p=2}^{n+1} p^{1/2} = \sum_{p=1}^{n} p^{1/2} - 1 + (n+1)^{1/2} > \int_1^n x^{1/2} dx, \Rightarrow \sum_{p=1}^{n} p^{1/2} > (n^{3/2} - 1)/2 - (n+1)^{1/2} \Rightarrow \sum_{p=1}^{n} p^{1/2} = \Omega(n^{3/2})$
Получаем, что $\sum_{p=1}^{n} p^{1/2} = \Theta(n^{3/2})$.
Проведя аналогичные рассуждения для $\sum_{p=1}^{n} p^{\alpha}$ получаем, что $\sum_{p=1}^{n} p^{\alpha} = \Theta(n^{\alpha+1})$.



1.(i) $\Sigma_1 = NP$ и $\Sigma_1 \in \Sigma_2$, $SAT \in \Sigma_1$. Пусть $\chi(x1, ..., x_n) \in SAT$. Преобразуем КНФ следующим образом: каждый дизъюнкт исходной КНФ $_i$ преобразуем к $\beta_i = \alpha_i \vee y_1 \vee ... \vee y_n$. Полученный язык булевых формул, очевидно, выполним при любом наборе $y_1 \vee ... \vee y_n$ и лежит в Σ_2.
(ii) Пример задачи из Σ_3: $x \in L \iff \exists y_1 \forall y_2 \exists y_3 \hookrightarrow R(x, y_1, y_2, y_3) = 1$. На вход подается граф G и 2

числа k_1, k_2. Верно ли, что в G k_1 — размер максимальной клики, k_2 — размер второй максимальной клики (при этом множество вершин второй клики не является подмножеством вершин первой клики)?
(iii) Докажем, что $\Sigma_k \subset \Sigma_{k+1} \cap \Pi_{k+1}$.
$\Sigma_{\subset\Sigma_{k+1}}$, действительно, пусть $L \in \Sigma_k$, тогда $\forall x \in L \exists\ y_1 \forall y_2 ... \hookrightarrow R(x, y_1, ..., y_k) = 1$. Тогда справедливо: $\forall x \in L\ _1 : \forall y_2, ..., \forall y_{k+1}|_{k+1} \hookrightarrow R(x, y_1, ..., y_k) = 1$. (просто оставляем предикат, не учитывающий y_{k+1}). Теперь докажем, что $\Sigma_k \subset \Pi_{k+1}$. Пусть $x \in L \in \Sigma_k$, тогда $\forall x \in L\ \exists\ y_1 \forall y_2 ... \hookrightarrow R(x, y_1, ..., y_k) = 1$. Добавим какую-то переменную y_{k+1}, от которой не будет зависить предикат R, получаем: $\forall x \in L\ \forall y_{k+1} \exists\ y_1 \forall y_2 ... \hookrightarrow R(x, y_1, ..., y_k) = 1$. Следовательно, $\forall L \in \Sigma_k \hookrightarrow L \in \Pi_{k+1}$. Доказано. (iv) Докажем, что $\mathcal{NP} \subset \mathcal{PSPACE} \subset \mathcal{EXPTIME}$.
$\mathcal{PSPACE}$ — задачи, разрешимые на машине Тьюринга, с полиномиальным ограничением пространства (то есть которая использует полиномиальное число ячеек). Пусть МТ переходит от одной ячейки к другой за время a, тогда время работы не больше $a^{p(n)}$. Тогда получаем, что если язык разрешим на МТ, использующей полиномиальное количество памяти, то он разрешим за экспаненциальное время от длины входа. Пусть $L \in \mathcal{NP}$, тогда пронумеруем все возможные сертификаты для x, их полиномиальное число от длины входа (так как длина сертификата равна $O(poly|x|)$), следовательно, можем пронумеровать сертификаты. Будем Подавая на ленту x и сертификат по возрастанию номера, проверяем каждый сертификат. Очевидно, что используем для проверки каждого сертификата полиномиальную память. Следовательно, $\mathcal{NP} \subset \mathcal{PSPACE} \subset \mathcal{EXPTIME}$. **3.** Язык выполнимых ДНФ лежит в P.
Действительно, чтобы доказать, что ДНФ выполнима, нужно просто нужно пройтись по каждой скобочке и убедиться, что хотя бы в одной нет пары $x \wedge \bar{x}$. Если хотя бы в одном из конъюнктов нет такой пары, то сможем подобрать набор, на котором этот конъюнкт будет выполним. Следовательно, будет выполнима ДНФ. Проверить каждую скобку можно за квадрат от количества литералов в скобке. Очевидно, что алгоритм работает за полином.
Не любая КНФ преобразовывается к ДНФ таким образом, чтобы длина полученной КНФ — полином от длины ДНФ. Но можем поступить другим образом: навесим отрицание над исходной ДНФ, тогда м можем преобразовать КНФ в ДНФ за полиномиальное время, но не в эквивалентную, а всего лишь в ту, которая будет не выполнима на наборе, на котором будет выполнима исходная ДНФ. Пусть есть ДНФ. Тогда построим КНФ, навесив отрицание (делаем это за полином). Пусть на наборе $x_1, ..., x_n$ ДНФ оказалась выполнимой, тогда КНФ, построенная по данной ДНФ окажется на этом наборе не выполнимой.
Языки тавтологичных ДНФ и КНФ обозначу $TDNF$ и $TKNF$ соответственно. Получается, что $TDNF =$ НЕВЫПОЛНИМЫЕ КНФ $\in$ ЯЗЫК ВЫПОЛНИМЫХ ДНФ. Это очевидно, ведь для любой КНФ найдется ДНФ, получаемая навешиванием отрицания. Тогда язык $TDNF$ сводится к языку не выполнимых КНФ. Следовательно, $TDNF \in co - \mathcal{NP} - complete$. Язык $TKNF$ сводится к языку не выполнимых ДНФ, а так как выполнимые ДНФ — язык из P, то дополнение тоже принадлежит P. Получаем следующую таблицу:
$3\frac{1}{2}$.

	Выполнимость	Тавтологичность
КНФ	$\mathcal{NP} - complete$	$\mathcal{P}$
ДНФ	$\mathcal{P}$	$co - \mathcal{NP} - complete$

7. a)Существует функция для каждой пары констант (то есть для каждого c подбираем свою функцию). По определению o-малого: $\lim\limits_{n\to\infty} \frac{n^c}{n^{a\cdot c}} = 0$, это справедливо при любом $a > 1$.
$\lim\limits_{n\to\infty} \frac{n^{a\cdot c}}{2^{n\cdot d}} = 0$ по правилу Лопиталя (ac раз взяли производную). Получаем, что можно определелить функцию для конкретных констант. Но такой функции, при которой оба предела обращаются в ноль не существует. Действительно, пусть $f(n) = o(2^{nd})$, правилу Лопиталя: $\lim\limits_{n\to\infty} \frac{f(n)}{2^{n\cdot d}} = \lim\limits_{n\to\infty} \frac{f(n)^{(a)}}{2^{n\cdot d}\cdot(\ln_2)^a\cdot d^a} = 0$, тогда $\exists a \in N : f(n)^{(a)} = 0$. Тогда: $\lim\limits_{n\to\infty} \frac{n^c}{f(n)} = \lim\limits_{n\to\infty} \frac{(n^c)^{(a)}}{f(n)^{(a)}}$ и при $c > a$ предел не равен 0. Следовательно,такой функции не существует.
b) Да, верно. Как было доказано в задаче 3, что тавтологичность ДНФ лежит в $co - \mathcal{NP}$ и так как

по устловию любой МТ требуется $\Omega(n \log_2^{\log_2 n} n)$ тактов для того, чтобы проверять тавтологичность формул, заданных в формате 4-ДНФ, имеем, что существуют формулы в языке тавтологичных ДНФ, для которых проверка принадлежности языку происходит не за полином от длины входа. Следовательно, данный язык лежит в $co-\mathcal{NP}$, но не лежит в $\mathcal{P}$. Таким образом, доказано, что $\mathcal{P}$ не совпадает с $co-\mathcal{NP}$.

5. Описанный в условии язык, очевидно, лежит в $\mathcal{NP}$. Докажем, что он является $\mathcal{NP}-hard$. Сведем язык $3-SAT$ к языку выполнимых формул, в которых каждая переменная входит не более 3 раз, а каждый литерал — не более 2 раз. Для этого в КНФ оставим только одно вхождение переменной, а остальные заменим на переменные новые вида $x_{n+1}, x_{n+2}, \ldots$. К исходной КНФ припишем скобки вида $(x_i \equiv x_{n+1}) \wedge (x_i \equiv x_{n+2}) \wedge \ldots$. Формула $(x_i \vee \bar{x}_{n+1}) \wedge (x_{n+1} \vee \bar{x}_{n+2}) \wedge (x_{n+2} \vee \bar{x}_i)$ равно выполнима с преписанными эквивалетностями (эта идея заимствована у Алима Бухараева, который в свою очередь заимствовал ее у Михаила Сысака), поэтому заменим приписанные скобки на данную формулу. Очевидно, что исходная КНФ будет равновыполнима с полученной, таким образом свели язык $3-SAT$ к языку выполнимых формул, в которых каждая переменная входит не более 3 раз, а каждый литерал — не более 2 раз, следовательно, останется $\mathcal{NP}-complete$.

6. Построим полиномиальную сводимость языка (G, k) графов, в которых есть k-клика к языку графов, в которых есть клика хотя бы на половине вершин.

Пусть есть граф G на n вершинах. Добавим в граф еще $(n-2k)$ вершин, соединив их со всеми остальными вершинами графа и между собой. Получим граф на $2(n-k)$ вершинах. Докажем, что в полученном графе на $2(n-k)$ вершинах есть клика хотя бы на половине вершин тогда и только тогда, когда в исходном графе есть клика размера k.

Пусть в исходном графе есть клика размера k, тогда добавив к графу $(n-2k)$ вершин, смежных со всеми остальными просто увеличим размер клики до $(n-k)$ и получим граф, в котором есть клика хотя бы на половине вершин.

Пусть в исходном графе не было клике на k вершинах (заметим, что если есть клика размера $k' > k$, то, очевидно, есть клика и на k вершинах), а в построенном графе оказалась клика на $(n-k)$ вершинах. Но мы добавили в граф $n-2k$ вершин, смежных со всеми, следовательно, если в исходном графе не было клики размера k, то в новом графе никак бы не получилась клика размера $n-k$. То есть,таким образом, получили противоречие.

$n-2k$ вершин к исходному графу добавляем за полиномиальное время. Построили полиномиальную сводимость.