# Первый курс, осенний семестр 2018/19
# Практика по алгоритмам #14

## Введение в сложность

## 8 декабря

Собрано 15 декабря 2018 г. в 09:02

---

## Содержание

# Введение в сложность

1. **Лежит ли задача в** NP**? А в** coNP **или** P**?**
   a) SUBSET-SUM (KNAPSACK без стоимостей).
   b) KNAPSACK со стоимостями.
   c) GI – проверка изоморфизма двух графов.
   d) Проверка *отсутствия* пути $a \rightsquigarrow b$ в графе.
   e) $\text{FACTOR}_d$ – наличие делителя не более $d$.
   f) Поиск кратчайшей эквивалентной записи булевой формулы
      1. Формула задана логическим выражением с `OR`, `AND`, `NOT`.
      2. Формула задана таблицей истинности.

2. **Постройте сведение (полиномиальное или по Куку)**
   a) HAMPATH $\leftrightarrow$ HAMCYCLE
   b) 3-COLORING $\rightarrow$ 3-SAT
   c) 3-SAT $\leftrightarrow$ NAE-3-SAT
   NAE = not all equal значит, что в каждом клозе хотя бы 1 ноль и хотя бы 1 единица.
   d) NAE-3-SAT $\rightarrow$ MAX-CUT (дан взвешенный граф, разделить множество вершин на две непустых доли, максимизировать суммарный вес рёбер между долями).
   e) SAT-MAX-ONE (максимизировать число единиц) $\rightarrow$ CIRCUIT-SAT

3. **Доказать** NP**-полноту**
   a) $(a, b)$-CSP (consrtaint satisfaction problem) – есть $n$ переменных, у каждой $a$ возможных значений, есть $m$ ограничений (условий), каждое на $b$ переменных. Например, 3-SAT – частный случай $(2, 3)$-CSP.
   b) SAT (через уже доказанный CIRCUIT-SAT)
   c) MAX-3-SAT (выполнить, воможно, не все, но максимальное число клозов)
   d) VERTEX-COVER (выбрать минимальное число вершин, чтобы задеть все ребра)

4. **Предъявите для задачи decision-версию и сведите к ней search-версию**
   a) MAX-SAT (максимизировать число выполненных клозов), найти сам набор переменных.
   b) HAMPATH
   c) 3-COLORING

5. NP $\subseteq$ EXP

6. Решить 3-SAT за $\mathcal{O}^*(1.5^{\mathbf{m}})$.

7. Решить 3-SAT за $\mathcal{O}^*((2-\varepsilon)^n)$.

8. (*) **Полиномиальные нижние оценки!**
   OV = Orthogonal Vectors: даны вектора $a_1, \ldots, a_n, b_1, \ldots, b_n$, есть ли пара $\langle a_i, b_j \rangle = 0$?
   Покажите SETH $\Rightarrow$ OV нельзя решить за $\mathcal{O}(n^{2-\varepsilon})$.

† 9. (*) HAM-CYCLE $\in$ NPc

10. (*) 3-SAT $\rightarrow$ 3-COLORING

11. (*) 3-COLORING $\to$ EXACT-SET-COVER
    EXACT-SET-COVER: даны $U$ и $\{B_i \subseteq U\}$, есть ли такое $I\colon \cup_{i\in I} B_i = U \,\wedge\, \forall\, i, j\; B_i \cap B_j = \varnothing$.
12. (*) DOMINATING-SET $\in$ NPc
    DOMINATING-SET: найти min $A \subseteq V\colon A \cup N(A) = V$.
13. (*) STEINER-TREE $\in$ NPc
    STEINER-TREE: даны взвешенный граф и $S \subseteq V$, найти поддерево min веса, содержащее все вершины $S$.

# Разбор задач практики

1. **Лежит ли задача в** NP? **А в** coNP **или** P?
   a) SUBSET-SUM. Подсказка для NP: набор предметов.
   b) KNAPSACK со стоимостями. Decision-версия: можно ли набрать цену больше $C$.
      Подсказка для NP: набор предметов.
      Решать за полином не умеют (динамика за $\mathcal{O}(nS)$ работает за экспоненту от длины $S$).
   c) GI. Подсказка для NP: перестановка, переводящая вершины первого графа в вершины второго. Умеют решать за $2^{\text{poly}(\log n)}$. Полагают, что не P и не NPc.
   d) Отсутствие пути в P (dfs) $\Rightarrow$ в NP и в coNP (задачи из P решаются с пустой подсказкой).
   e) FACTOR$_d$. Подсказка для NP и coNP: разложение на простые.
      Чтобы проверить подсказку для coNP, нужно уметь проверять на простоту за полином от длины числа. Это умеют за $n^6$ ($n = \log N$ – длина числа).
      FACTOR умеют за $2^{Cn^{1/3}}$. Полагают, что не P и не NPc.
   f) Кратчайшая запись булевой формулы.
      Короткая запись – не подсказка: непонятно, как проверить ее эквивалентность исходной формуле. Не знают, лежит ли в NP.
      Если формула задана таблицей истинности (длина входа $2^n$), то проверим эквивалентность этой таблицы $T$ и короткой записи $\varphi$ перебором всех подстановок переменных. Время $2^n \cdot |\varphi| = \mathcal{O}(\text{poly}(2^n)) = \mathcal{O}(|T|) \Rightarrow$ в NP.
      Если длина формулы $\varphi$ больше $2^n n$, то она сразу не кратчайшая, ДНФ короче.

2. **Постройте сведение**
   a) HAMPATH $\leftrightarrow$ HAMCYCLE.
      HAMPATH $\leqslant_p$ HAMCYCLE: добавим вершину $a$, соединим ее со всеми.
      Был путь в старом графе $\Rightarrow$ цикл в новом графе: (путь в старом $v - \ldots - u$) $+(u - a - v)$.
      Есть цикл в новом графе $\Rightarrow$ выкинем $a$, получим путь в старом графе.
      Работает и для ор, и для неор. В ор надо соединять $a$ со всеми в обе стороны.

      HAMCYCLE $\leqslant_p$ HAMPATH: раздвоим вершину 1 на две – `in` и `out`, из копии `out` только исходящие ребра, в `in` только входящие. Путь в новом графе обязан начаться в `out` (у нее нет входящих) и кончиться в `in` (нет исходящих).
      Цикл в старом графе = путь в новом со склеенными `in` и `out`.
      Для неорграфа надо проводить из `in` и `out` все ребра, но подвесить к `in` и `out` по листу.
   b) 3-COLORING $\to$ 3-SAT.
      Переменные $x_{vc}$ – покрашена ли $v$ в $c$.
      Ребро $(u, v)$ для каждого цвета $c$ дает клоз $(\neg x_{uc} \vee \neg x_{vc})$. Получили корректную покраску.
      Для каждой $v$ добавим клоз $(x_{v1} \vee x_{v2} \vee x_{v3})$. Теперь $v$ покрашена хотя бы в один цвет.
      Покраска и удовлетворяющий набор переменных легко получаются друг из друга.
   c) 3-SAT $\leftrightarrow$ NAE-3-SAT.

      NAE-3-SAT $\to$ 3-SAT: запишем для каждого клоза $(l_1 \vee l_2 \vee l_3)$ парный $(\neg l_1 \vee \neg l_2 \vee \neg l_3)$.
      Решим SAT из $2m$ клозов.

      3-SAT $\to$ NAE-4-SAT: добавим в каждый клоз переменную $w$ (одну и ту же).

Было решение 3-SAT $\Rightarrow$ берем его и делаем $w = 0$.
Есть решение $(y_1, y_2, \ldots, y_n, w)$ для NAE-4-SAT $\Rightarrow$ ответ для 3-SAT это $x_i = y_i \,\hat{}\, w$.

NAE-4-SAT $\to$ NAE-3-SAT: $i$-й клоз из 4 литералов $(l_1 \vee l_2 \vee l_3 \vee l_4)$ переводим в клозы $(l_1 \vee l_2 \vee w_i)$, $(\neg w_i \vee l_3 \vee l_4)$, $w_i$ – новая переменная (так же, как в 4-SAT $\to$ 3-SAT).
Чтобы перевести решение NAE-4-SAT в решение NAE-3-SAT и обратно, разбор 5 случаев: $l_1 = l_2 = 0/1$, $l_3 = l_4 = 0/1$, иначе.

d) NAE-3-SAT $\to$ MAX-CUT
Для каждой переменной $x_i$ заводим две вершины: $x_i$, $\overline{x_i}$, соединяем их ребром.
Каждый клоз преобразуем в три ребра: $\triangle$ на литералах.
Получили граф из $2n$ вершин и $3m + n$ рёбер.
Проверим существование разреза размера хотя бы $n + 2m$, больше точно не бывает.
Если такой есть, все $x_i$ и $\overline{x_i}$ лежат по разные стороны разреза, и в каждом треугольнике ровно два ребра в разрезе (для этого и нужно NAE).

e) SAT-MAX-ONE $\to$ CIRCUIT-SAT
Преобразуем сперва SAT к CIRCUIT-SAT, обозначим выходной гейт $v_{\texttt{out}}$.
Создадим гейты $f_{i,j}$ – среди первых $i$ переменных хотя бы $j$ единиц.
$f_{i,j} = f_{i-1,j} \vee (f_{i-1,j-1} \wedge x_i)$. Ответ в $v_{\texttt{out}} \wedge f_{n,k}$.

3. Чтобы доказать NP-полноту, нужно свести какую-либо NP-полную задачу к данной.
   a) $(a, b)$-CSP. 3-SAT – частный случай, свелось.
   b) SAT. Сведём к нему CIRCUIT-SAT: каждому узлу соответствует формула $v_i = f_i(v_j, v_k)$. Это булева формула от трех переменных, запишем ее в КНФ. Соединим все КНФ вместе.
   c) MAX-3-SAT. Сводим к ней 3-SAT: $\varphi \to \langle \varphi, m \rangle$, выполнено хотя бы $m$ клозов, то есть все.
   d) VERTEX-COVER. Сведём к нему INDEPENDENT-SET: $\langle G, k \rangle \to \langle G, n - k \rangle$. Множество – вершинное покрытие $\Leftrightarrow$ его дополнение – независимое множество.

4. **Decision $\to$ search**
   a) MAX-SAT. Бинпоиском ищем, сколько клозов можно удовлетворить, нашли $k$.
   Берем переменную $x_1$, подставляем ее равной 1 в формулу $\varphi$, получили $\varphi'$.
   Если верен MAX-SAT от $\langle \varphi', k \rangle$, то $x_1 = 1$, иначе $x_1 = 0$.
   Подставляем $x_1$ и ищем остальные переменные.
   b) HAMPATH. Надо найти начало пути. Переберём $v$ и добавим вершину $w$ с одним исходящим ребром в $v$. Если теперь есть гамильтонов путь, то раньше он начинался с $v$.
   Удаляем $v$ и так же ищем остальные вершины пути, только дальше надо перебирать не все вершины, а только соседей предыдущей.
   c) 3-COLORING. Пока в графе больше трёх вершин, какие-то две должны быть покрашены в один цвет. Переберём, какие. Чтобы проверить, что их можно покрасить в один цвет, стянем их в одну вершину.

5. NP $\subseteq$ EXP: переберем подсказку.

6. Решить 3-SAT за $\mathcal{O}^*(1.5^m)$.

   Так же, как в прошлом семестре решали 3-LIST-COLORING.
   Выкинем из каждого клоза случайный литерал $\Rightarrow$ с вероятностью $\geqslant (\frac{2}{3})^m$ получим решающийся 2-SAT. Повторим $1.5^m$ раз, получим вероятность ошибки $e^{-1}$.

7. Решить 3-SAT за $\mathcal{O}^*((2-\varepsilon)^n)$.
   Возьмём любой клоз с тремя литералами, есть 8 вариантов присвоить переменным значения, но только 7 из них обратят клоз в истину. Получили $T(n) = 7T(n-3) = 7^{n/3} \approx 1.91^n$.
   Также можно решать и $k$-SAT за $\mathcal{O}((2^k-1)^{n/k})$.

8. (*) **Полиномиальные нижние оценки!**
   SAT $\to$ OV. Пусть дана формула с клозами $C_1, \ldots, C_m$.
   Рассмотрим переменные $x_1, \ldots, x_{n/2}$. Рассмотрим все $2^{n/2}$ подстановок этих переменных.
   Подстановке $s$ сопоставим вектор $a_s = \langle a_{s1}, \ldots, a_{sm} \rangle$: $a_{si} = \neg(C_i$ выполнен подстановкой $s)$.
   Аналогично строим набор из $2^{n/2}$ векторов $b_t$ по второй половине переменных.
   $\langle a_s, b_t \rangle = 0 \Leftrightarrow \forall i\,(a_{si} = 0 \vee b_{ti} = 0) \Leftrightarrow$ подстановкой $st$ удовлетворены все клозы.
   Решаем OV за $n^{2-\varepsilon} \Rightarrow$ решаем $\forall\, k$-SAT за $\mathcal{O}((2^{n/2})^{2-\varepsilon}) = \mathcal{O}(2^{(1-\frac{\varepsilon}{2})n})$. Противоречие с SETH.
   Note: размерность векторов мала относительно их количества: $m = \mathcal{O}(n^k) = \text{poly}(\log(2^{n/2}))$.

9. (*) HAM-CYCLE $\in$ NPc. Сведём 3-SAT. Пара картинок.

10. (*) 3-SAT $\to$ 3-COLORING
    Заведем вершины $T, F, N$ (`True`, `False`, `Neutral`) и попарно соединим рёбрами.
    Теперь они должны быть разных цветов, которые так и назовем: $T, F, N$.
    $\forall i$ заведем вершины $x_i$, $\neg x_i$ и рёбра $(x_i, \neg x_i), (x_i, N), (\neg x_i, N)$.
    Теперь одна из $x_i$, $\neg x_i$ должна быть $T$, другая $F$.

    Научимся для вершин $a, b$ строить некое подобие `OR`.
    Добавим вершины $u$, $v$, $z$, рёбра $(a, u), (b, v)$ и треугольник $(u, v, z)$.
    Если $a = b = F$, то $z$ тоже обязательно красить в $F$. Иначе **существует** корректная покраска $u$ и $v$, что $z \neq F$.
    Если повторить эту конструкцию два раза $((a \vee b) \vee c)$, то при $a = b = c = F$ выход обязательно $F$, иначе **существует** корректная покраска, в которой выход – $T$.
    Осталось соединить выход с $N$ и $F$. Добавляем такую конструкцию для каждого клоза.

    Подробнее и с картинками в pdf.

11. (*) 3-COLORING $\to$ EXACT-SET-COVER
    Универсум: все вершины, все ребра и все тройки $\langle v, e, c \rangle$, где $v$ – конец $e$, $c$ – один из цветов (итого $n + m + 6m$ элементов). Множества:
    a) $\forall$ вершины $v$, цвета $c$: $\{v\} \cup \{\langle v, e, c \rangle \mid e = (v, u)\}$.
       Если взято, то красим $v$ в цвет $c$. Каждая вершина покрасится, причем в один цвет.
    b) $\forall$ ребра $e = (v, u)$, пары цветов $c_1 \neq c_2$: $\{e\} \cup \{\langle v, e, c \rangle \mid c \neq c_1\} \cup \{\langle u, e, c \rangle \mid c \neq c_2\}$.
       Раз каждое ребро чем-то покрыто, то оно «забирает с собой» все цвета у своих концов, кроме двух неравных.

12. (*) DOMINATING-SET $\in$ NPc: VERTEX-COVER $\to$ DOMINATING-SET.
    Из графа $G = (V, E)$ делаем новый $G' = (V', E')$.
    $V' = V \cup E$. $E' = \{(v, u) \mid v, u \in V\} \cup \{(v, e) \mid e = (v, u) \in E\}$.
    Чтобы задоминировать все вершины, которые соответствуют старым ребрам, нужно как раз набрать VERTEX-COVER. Полный граф на старых вершинах нужен, чтобы они сами собой задоминировались.

13. (*) STEINER-TREE $\in$ NPc: SET-COVER $\to$ STEINER-TREE.
    Вершины графа – элементы и множества.

Ребра веса 0 между множествами и лежащими в них элементами.
Фиктивная вершина $v_0$, соединенная ребрами веса 1 со всеми множествами.
$S$ – множество элементов. Вес min дерева = (количеству множеств в min SET-COVER $-1$).