Лекция 8

Элементы теории вычислительной сложности.

NP-полные задачи.

Детерминированные и недетерминированные автоматы

- Алгоритмические задачи разбивают на классы алгоритмов по времени выполнения или требованиям к расходуемым ресурсам на каком-то определенном или абстрактном устройстве.
- Наиболее распространенными видами абстрактных машин являются детерминированные или недетерминированные автоматы. В качестве автомата используем модель вычислителя – машину Тьюринга МТ.

$$\text{МТ}=\{A, Q, \varphi(q,a), \psi(q,a), H(q,a)\}$$

A- алфавит входных и выходных символов;

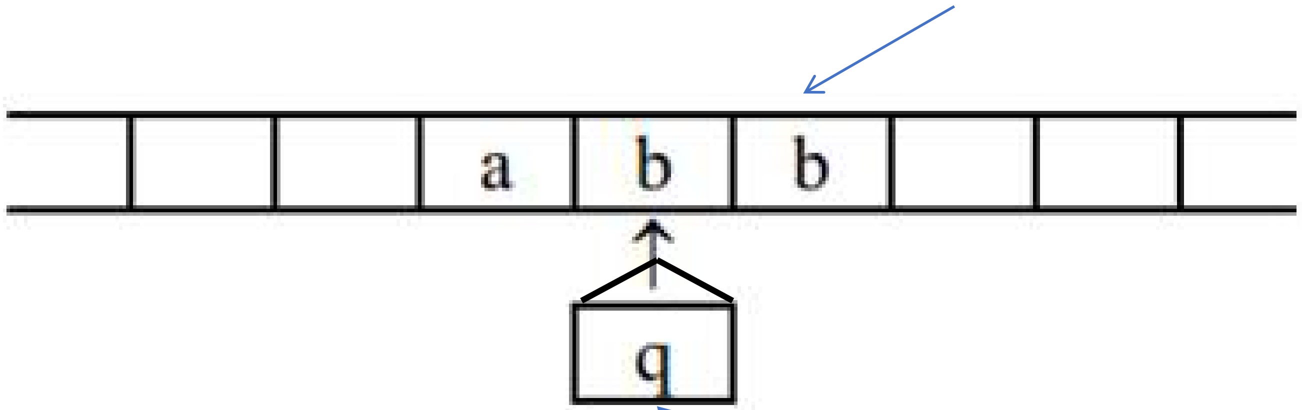
Q- алфавит состояний;

$\varphi(q,a)$ - функция переходов ($\varphi: Q \times A \to Q$);

$\psi(q,a)$ - функция выходов ($\psi: Q \times A \to A$);

$H(q,a)$ - функция сдвига ($H: Q \times A \to \{R, L, S\}$)

Бесконечная лента, разбитая на ячейки



Считывающее-записывающее устройство

Машина Тьюринга является расширением понятия конечного автомата

Команда $qb \rightarrow q'b'h$

Функция, правильно вычислимая по Тьюрингу и тезис Тьюринга

Рассмотрим алгоритмы, имеющие дело только с натуральными числами:

$$f: \mathbb{N}^n \to \mathbb{N}$$

- f – правильно вычислима по Тьюрингу, если для нее можно записать программу из команд МТ, которая начинает работать из q_1 и заканчивает в q_z (принимающем состоянии), печатает ответ на ленте или работает бесконечно долго, если f не определена на $x_1, x_{2,\dots,} x_n$.
- Например, $\mathrm{A} = \{\lambda, 1\}$

  $f(x) = x + 1$

$$q_1 1 \to q_1 1R$$
$$q_1 \lambda \to q_z 1S$$

Программа

Согласно тезису Тьюринга, если для данной задачи существует алгоритм, его можно реализовать на МТ.

Использование МТ для оценки пространственной или емкостной сложности алгоритма

- Если алгоритм полиномиален $(O(n^p))$, то он сохранит это свойство на любом вычислительном устройстве $\left(O\left(n^k\right)\right)$, также, если он экспоненциален $\left(O\left(a^{P_1(n)}\right)\right)$, то он останется экспоненциальным на любом устройстве$\left(O\left(a^{P_2(n)}\right)\right)$. МТ используется для того, чтобы не зависеть от конкретного устройства.
- МТ, как и автомат может быть <u>детерминированной</u> или <u>недетерминированной</u>.

В ДМТ для каждой команды $qb \rightarrow q'b'h$ правая часть однозначно зависит от левой части, поэтому программа выполняется строго детерминировано.

В НМТ $qb \begin{cases} \nearrow q'b'h_1 \\ \searrow q''ch_2 \end{cases}$

В НМТ в начале работы (недетерминированный этап) срабатывает модуль угадывания, который предлагает сертификат (слово-решение).

Затем МТ работает в детерминированном режиме и проверяет, правильно ли это решение. Если процесс заканчивается в состоянии q_y, то проверка завершается успешно, если в q_N или вообще не заканчивается – вычисление не принимается.

В теории сложности обычно рассматриваются задачи поиска, оптимизации и **распознавания** (задачи принятия решения – ответ «да» или «нет»).

Задача коммивояжера (TSP)

Дан список городов, соединенных дорогами (граф с *n* вершинами- городами и ребрами – Дорогами различной длины). Коммивояжер должен обойти все города и прийти в исходный город, причем длина пройденного пути должна быть минимальна. Это оптимизационная задача. Переделаем ее в задачу распознавания: есть ли в графе путь, проходящий через все вершины и имеющий длину не больше B?

Задачи оптимизации/поиска сводятся к задачам распознавания, т.е. задачи распознавания не сложнее задач оптимизации/поиска.

Для задач распознавания используют теорию формальных языков.

X – алфавит, X^* - входной словарь, язык $L \subseteq X^*$

- Над языками можно производить операции объединения, конкатенации, итерации. Алгоритм A принимает слово $x \subseteq X^*$, если выход алгоритма «1» (true) – принимающее состояние q_y; иначе – отвергает.
- Язык $L = \{x \subseteq X^*\colon\ M_A(x) = 1(q_y)\}$
- Пример: Выполнима ли 2-КНФ $(x_1 \vee \overline{x}_2)(\overline{x}_1 \vee x_3)(\overline{x}_2 \vee \overline{x}_3)$?

–да. Существует алгоритм, который за полиномиальное время ответит на этот вопрос ($x = 101 \in L$)

- Пример: Выполнима ли 3-КНФ $(x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_2 \vee x_4)(\overline{x}_1 \vee \overline{x}_4 \vee \overline{x}_3)$? Не существует алгоритма, который бы за полиномиальное время ответил на этот вопрос. Но проверить за полином можно (1 1 0 1).

Распознавание языков

- МТ распознает язык L за время $f(n)$, если она принимает все слова, лежащие в L и отвергает все слова, не лежащие в L, и на каждом слове x делает не больше $f(|x|)$ шагов.
- Классификации языков (временная и емкостная).

I. Временная сложность.

1) **DTIME** – класс языков, распознаваемых на ДМТ за $f(n)$ (n –длина входа)

- *P* ($f(n) = O(n^p)$)
- EXPTIME ($f(n) = O(2^{P(n)})$)

2) **NTIME** – класс языков, распознаваемых на НМТ за $f(n)$

- *NP* ($f(n) = O(n^p)$) - на угадывание и проверку сертификата
- NEXPTIME ($f(n) = O(2^{P(n)})$) - на угадывание и проверку сертификата

II. Емкостная (пространственная) сложность.

1) **DSPACE** – класс языков, распознаваемых на ДМТ, при этом количество ячеек, используемых ДМТ, не больше $\mathrm{T}(n)$ (n –длина входа)

- PSPACE ($T(n) = O(n^p)$)
- EXSPACE ($T(n) = O\left(2^{P(n)}\right)$)
- L ($T(n) = O(\log n)$)

2) **NSPACE** – класс языков, распознаваемых на НМТ, при этом может потребоваться не больше $\mathrm{T}(n)$ памяти

- NSPACE ($T(n) = O(n^p)$)
- NEXSPACE ($T(n) = O\left(2^{P(n)}\right)$)

Некоторые соотношения между классами:

$$P \subseteq NP \; (\mathrm{DTIME}\,(f(n)) \subset \mathrm{NTIME}\,(f(n))$$

PSPACE=NSPACE, EXSPACE=NEXSPACE и т.д.

Основной вопрос: ***P=NP*** ?

Большая часть ученых считает, что $P \subset NP$.

Если задачу можно быстро проверить (NP), можно ли ее быстро решить (P).

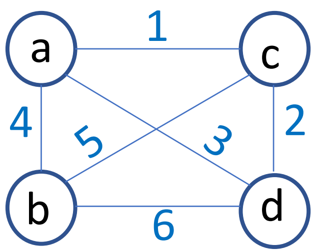
Примеры:

(1, 2)

1) MST - задача о минимальном остовном дереве (алгоритм Прима или Крускала) $O((|V| + |E|) \log|V|)$ или $O(|V| \cdot |E|)$ - эффективный алгоритм (лежит в P).

2) TSP – задача о коммивояжере – переборная задача, эффективный алгоритм не найден. Но задачу можно быстро проверить за полиномиальное время.

Например: B=13



сертификат y={ab, bd, dc, ca}

проверка 4+6+2+1≤13. Ответ «1» (да)

Имеем экземпляр задачи, соответствующий NP-трудному языку.

Определение: $L \subset NP \Leftrightarrow \exists$ алгоритм $A\colon L = \{x \in X^*\colon$
$\exists$ серификат y длины $|y| = O(|x|^c)$, где $M_A(x, y) = 1\}$

- (3, 4)

 3) поиск самого короткого пути в ориентированном графе $O(|V| \cdot |E|)$ (Беллмана-Форда);

 4) поиск самого длинного пути в ориентированном графе полиномиальный алгоритм не известен.

- (5, 6) Задача о выполнимости (SAT). Дана КНФ. Существует ли выполняющий набор?

 5) 2-КНФ полиномиальный алгоритм есть (поиск компонент сильной связности);

 6) 3-КНФ не известен полиномиальный алгоритм.

- (7-8) Существует ли в графе

 7) Эйлеров цикл? (полиномиальная задача)

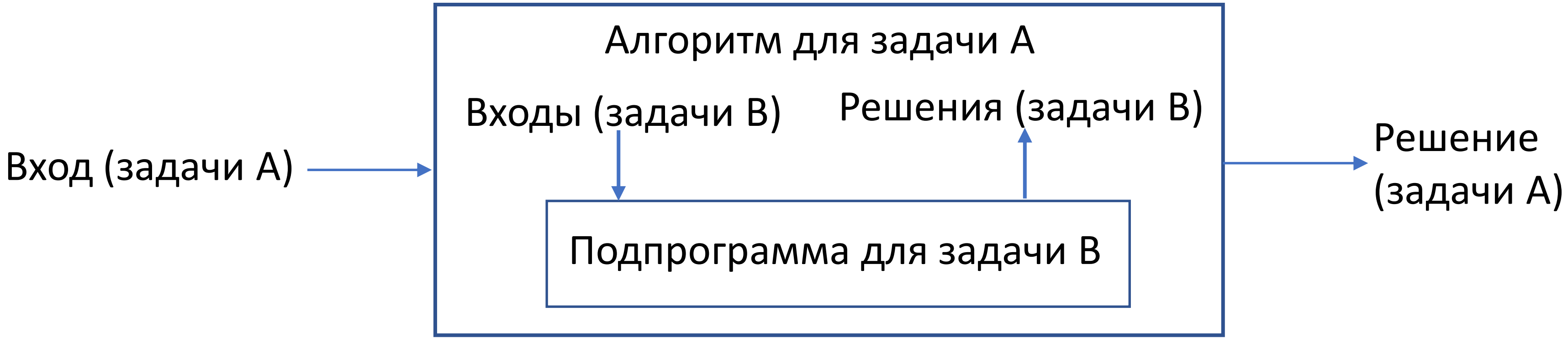
 8) Гамильтонов цикл? - Не известен полиномиальный алгоритм.

Полиномиальная сводимость

Язык L_A полиномиально сводится к языку L_B ($L_A \leqslant_P L_A$), если существует такая полиномиальная функция f, что $\forall x\ (x \in L_A \Leftrightarrow f(x) \in L_B)$.

В терминах задач:

Задача А полиномиально сводится к задаче В, если алгоритм, решающий задачу В, может быть легко переведен в алгоритм, решающий задачу А.

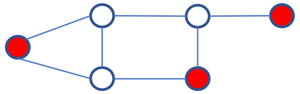


Полиномиальное число вызовов В и полиномиальный объем дополнительной работы

Пример сводимости: 3SAT$\leqslant_p$INDSET

Задача INDSET – независимое множество размера k:

В графе $G = (V, E)$ $\exists S \subset V$: $|S| = k$ и $\forall u, v \in S$ $(u, v) \notin E$,

например, $k = 3$, INDSET 

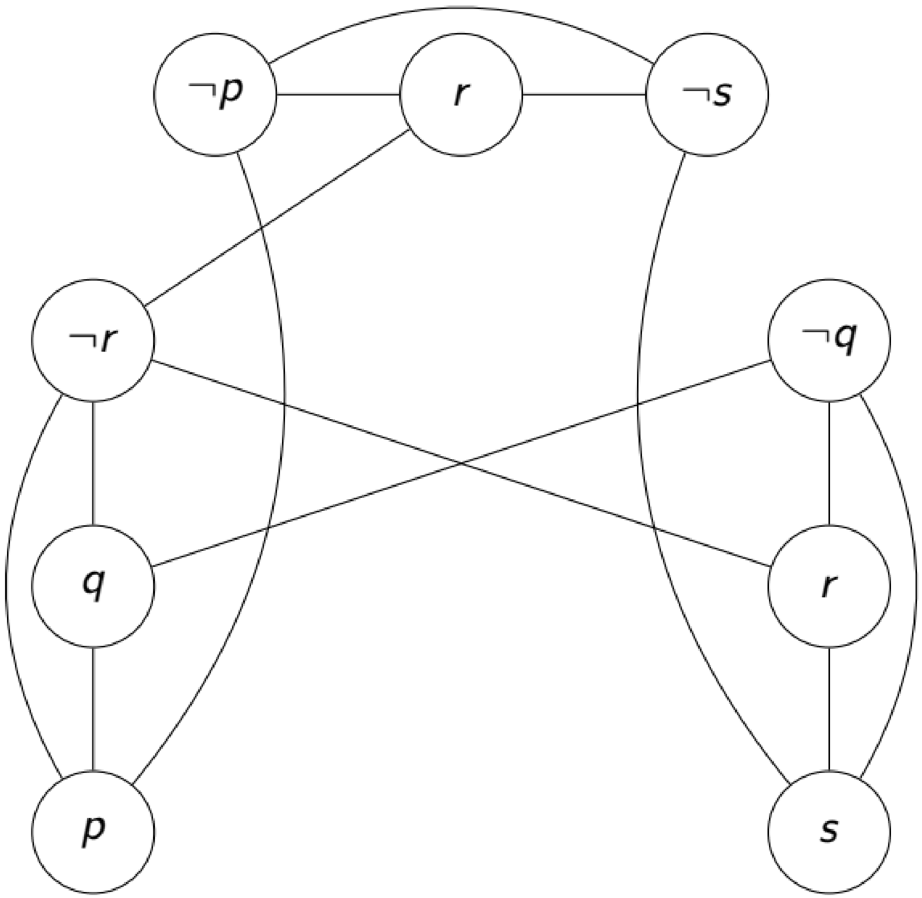
Рассмотрим 3SAT. Каждый литерал –> вершина графа. Если k дизъюнктов, то имеем $3k$ вершин (3SAT). Вершины, соответствующие литералам одного дизъюнкта, соединим ребрами. Соединим противоположные литералы (p и $\overline{p}$).

k – размер независимого множества.

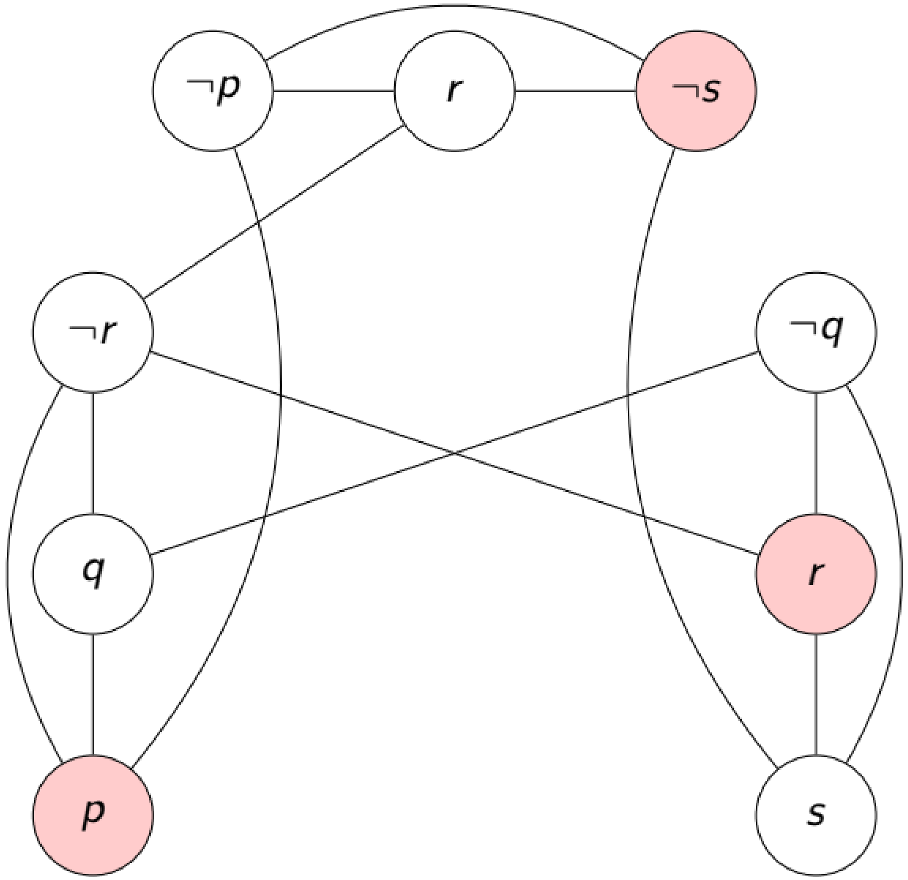
Пример:3SAT

$$(p \vee q \vee \neg r) \wedge (\neg p \vee r \vee \neg s) \wedge (\neg q \vee r \vee s)$$

$(p \vee q \vee \neg r) \wedge (\neg p \vee r \vee \neg s) \wedge (\neg q \vee r \vee s)$



В каждом дизъюнкте хотя-бы один независимый литерал



Если есть выполняющий набор, то есть независимое множество размера k и наоборот.

Свойства сводимости

- Основной принцип: композиция полиномиально вычислимых функций полиномиально вычислима
- Свойство 1: если $A \leqslant_p B$ и $B \leqslant_p C$, то $A \leqslant_p C$
- Доказательство: $x \in A \Leftrightarrow f(x) \in B \Leftrightarrow g(f(x)) \in C$
- Свойство 2: если $A \leqslant_p B$ и $B \in \mathrm{P}$, то $A \in \mathrm{P}$
- Доказательство: $x \in A \Leftrightarrow f(x) \in B \Leftrightarrow M(f(x)) = 1$
- Свойство 3: если $A \leqslant_p B$ и $B \in \mathrm{NP}$, то $A \in \mathrm{NP}$
- Доказательство: $x \in A \Leftrightarrow f(x) \in B \Leftrightarrow \exists y \; V(f(x), y) = 1$

NP-трудность

Определение. Язык B называется NP-трудным, если $\forall A \in NP \ \ A \leqslant_P B$

Утверждение:

$\boxed{\text{Если какой-то NP-трудный язык B} \in P\text{, то } P = NP}$

NP-полнота

Определение. Язык B называется NP-полным (NPC), если он NP-трудный и B$\in NP$

Следствие: $\boxed{\text{если какой-то NP-полный язык B} \in P\text{, то } P = NP}$

Получение NP-трудности и NP-полноты

- Если B является NP-трудным и B $\leqslant_P$ C , то C – NP-трудный.

- Если B является NP-полным и B $\leqslant_P$ C , то C – NP-полный.

Для доказательства:

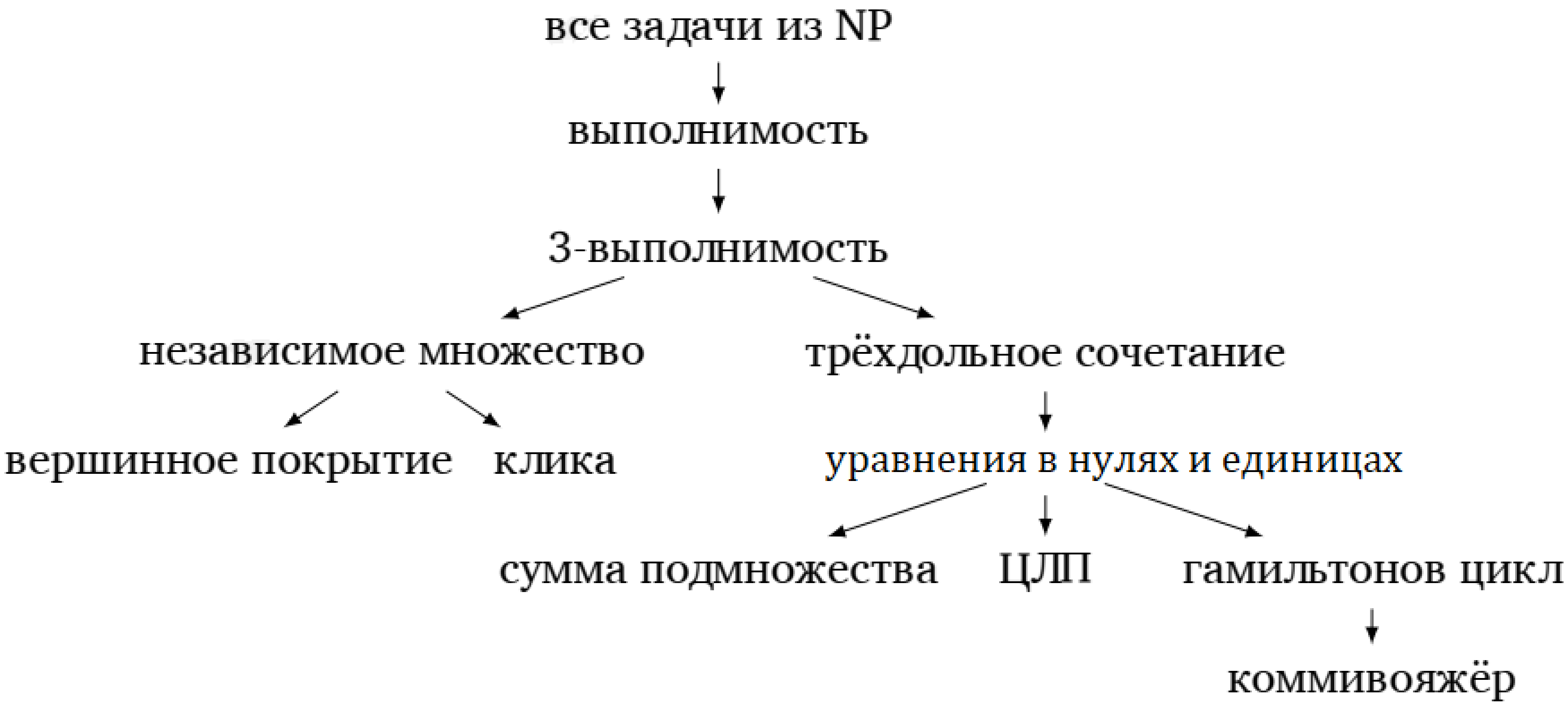
- доказать, что C ∈ NP
- свести к нему какой-либо известный NPC-язык

Примеры NPC-задач

- Первая задача, для которой была доказана NP-полнота – задача о выполнимости КНФ (SAT является NP-полной)

- Можно доказать, что SAT $\leqslant_P$ 3SAT (см. семинар)
- Поэтому задача о независимом множестве INDSET является NPC

Схема сведе́ний основных NPC-задач



Как решать NP-трудные задачи?

NP-трудность исключает алгоритмы, имеющие одновременно 3 свойства:

1. Универсальность.
2. Правильность
3. Быстрота.

Одним из трех пунктов необходимо пожертвовать.