Адрианов Н.М.
Иванов А.Б.

Алгоритмы и структуры данных

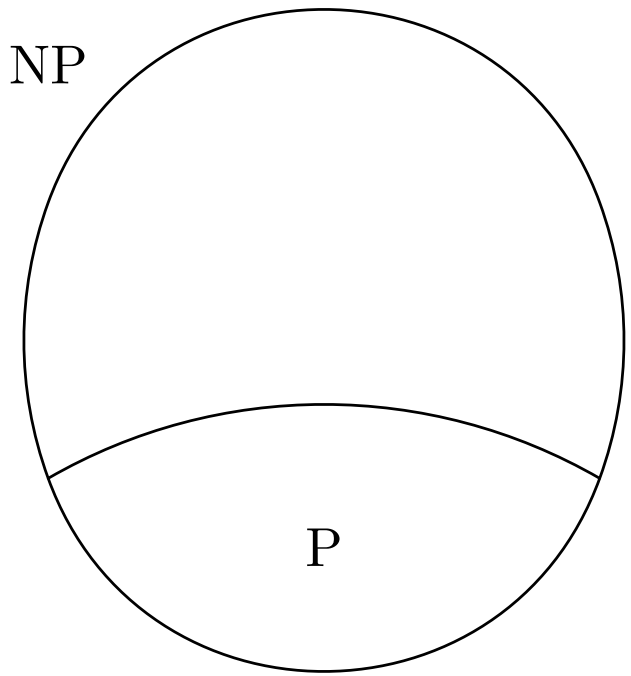
NP-полные задачи

С. Дасгупта, Х. Пападимитриу, У. Вазирани
Алгоритмы
2014

Б. Корте, Й. Фиген
Комбинаторная оптимизация. Теория и алгоритмы
МЦНМО, 2015

P и NP

P = класс задач, разрешимых за полиномиальное время
NP = (пока что не понятно, что за класс)



Определение класса NP

NP = Non-deterministic Polynomial (исторически определение класса NP было дано с использованием недетерминированных алгоритмов)

Задача поиска A называется NP-задачей, если существует *полиномиальный* алгоритм $\mathcal{C}$, который для любых входных данных I и кандидата на решение S выполняет проверку:

$$\mathcal{C}(I, S) = \begin{cases} 1, & S - \text{решение для входных данных } I \\ 0, & \text{иначе} \end{cases}$$

Примеры:

- SAT
- гамильтонов цикл

Задача коммивояжёра

Travelling salesman problem (TSP)

Коммивояжёр находится в одном из n городов, хочет их все объехать кратчайшим путем, заезжая в каждый город ровно 1 раз.

Дано: полный неориентированный граф с неотрицательными стоимостями ребер.

Требуется: найти гамильтонов цикл минимальной стоимости.

Задача коммивояжёра – задача оптимизации.

Задача о гамильтоновом цикле – задача поиска.

Задача коммивояжёра как задача поиска

Дано:

- полный неориентированный граф с неотрицательными длинами ребер;
- бюджет b.

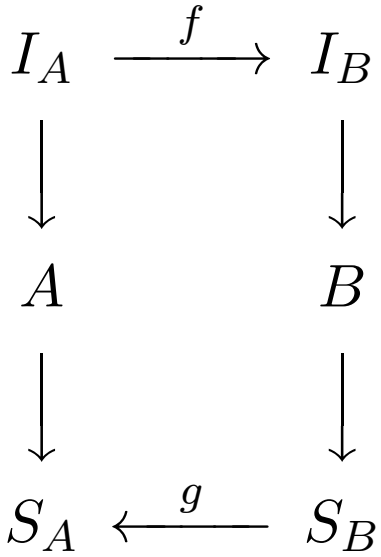
Требуется: найти гамильтонов цикл стоимости $\leq b$.

Если мы умеем решать задачу коммивояжёра с бюджетом (за полиномиальное время), то мы можем решить оптимизационную задачу коммивояжера (за полиномиальное время) с помощью бинарного поиска.

Сведение задачи A к задаче B

Существуют *полиномиальные* алгоритмы f и g такие, что

- для любых I_A – входных данных для задачи A;
- $I_B = f(I_A)$ – входные данные для B;
- S_B – решение задачи B со входными данными I_B;
- $S_A = g(S_B)$ – решение задачи A с входными данными I_A.



NP-полнота

Задача из NP называется NP-полной, если любая другая задача из NP может быть сведена к ней.

Stephen Cook, 1971:
- понятие NP-полной задачи;
- доказательство, что SAT NP-полна.

Независимо результат получен Леонидом Левиным, опубликован в только 1973, но докладывал его за несколько лет до этого. Теорема о NP-полноте SAT иногда называется теоремой Кука-Левина.

Richard Karp, 1972:
- список – 21 NP-полная задача.

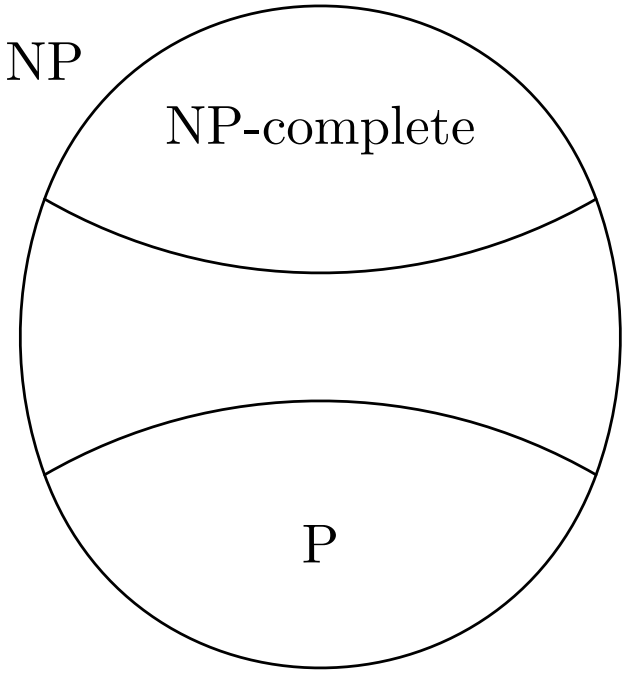
Cook: премия Тьюринга, 1982
Karp: премия Тьюринга, 1985
Левин: премия Кнута, 2012

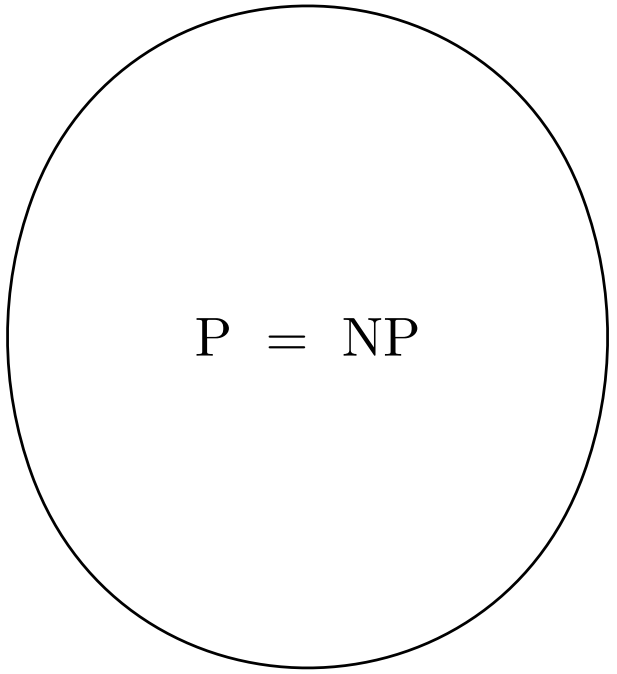
(Некоторые) NP-полные задачи

- 3-SAT
- гамильтонов путь
- задача коммивояжёра
- рюкзак (сумма подмножества)
- 3-дольное сочетание
- самый длинный путь в графе
- максимальное независимое множество в графе
- минимальное вершинное покрытие
- целочисленное линейное программирование
- уравнения в нулях и единицах

Задача о разложении числа на простые множители (лежит в основе RSA) – для нее неизвестно ни полиномиального алгоритма, ни доказательства NP-полноты.

P = NP?

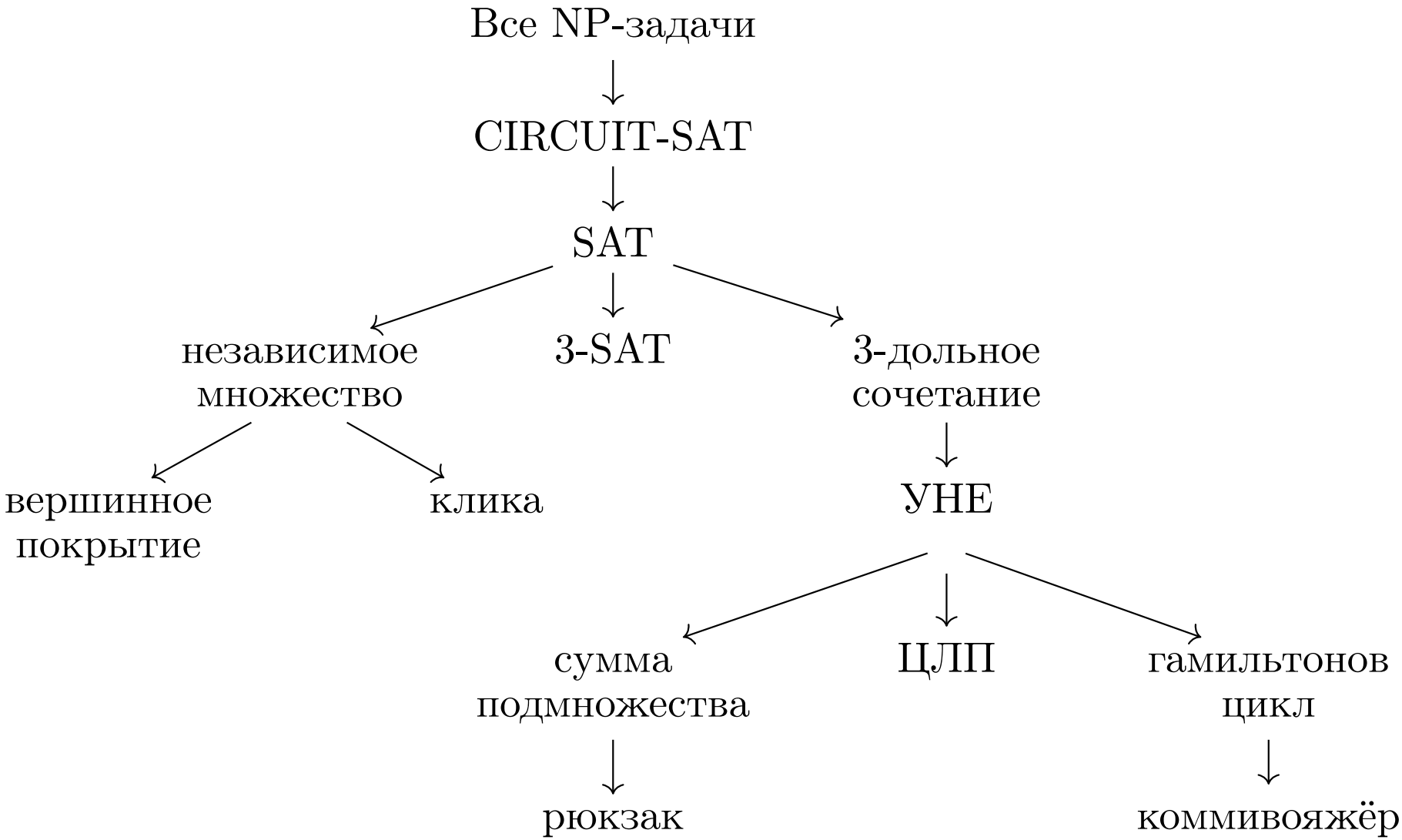




От простого до сложного – один шаг

NP-полные	**P**
3-SAT	2-SAT
3-дольное сочетание	2-дольное сочетание
гамильтонов путь	эйлеров путь
коммивояжёр	минимальное остовное дерево
максимальный путь	кратчайший путь
независимое множество	независимое множество в дереве
целочисленное линейное программирование	линейное программирование

Схема доказательства NP-полноты



Независимое множество $\to$ ...

В заданном графе $G = (V, E)$ требуется найти множество $S \subset V$

1. (минимальное) **вершинное покрытие**:

$$\forall (u, v) \in E \quad u \in S \text{ или } v \in S$$

$$|S| \to \min$$

2. (максимальное) **независимое множество**:

$$\forall u, v \in S \ (u, v) \notin E$$

$$|S| \to \max$$

3. (максимальная) **клика**:

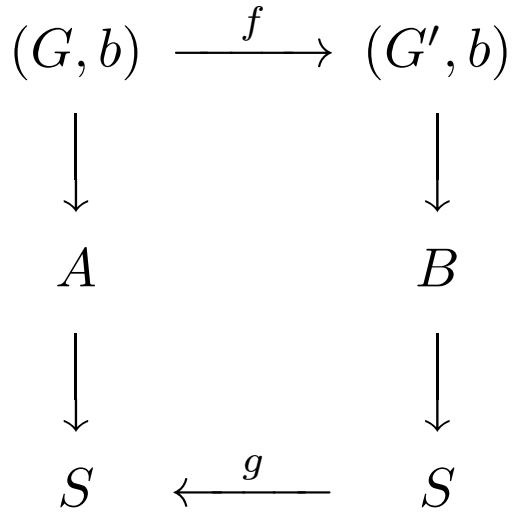
$$\forall u, v \in S \ (u, v) \in S$$

$$|S| \to \max$$

Независимое мн-во $\to$ клика

Задача A: независимое множество

Задача B: клика



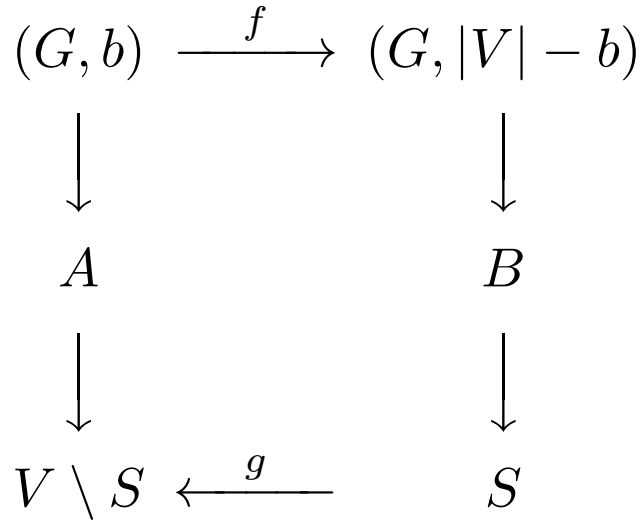
$G' = (V, E')$ – дополняющий граф для $G = (V, E)$:

$$(u, v) \in E \quad \Leftrightarrow \quad (u, v) \notin E'.$$

Независимое мн-во $\to$ вершинное покрытие

Задача A: независимое множество

Задача B: вершинное покрытие



SAT $\rightarrow$ НЕЗАВИСИМОЕ МНОЖЕСТВО

- КНФ: конъюнкция некоторого количества дизъюнктов (клауз);
- Клауза: дизъюнкция некоторого количества термов;
- Терм: переменная или ее отрицание.

Строим граф G: для каждой клаузы добавляем клику на s вершинах (s = количество термов в клаузе). Вершины клики помечаем термами. Все пары вершин, помеченные x_i и $\bar{x}_i$, соединяем ребрами.

Задача: найти в графе G независимое множество из m вершин (m = количество клауз).

Задача CIRCUIT-SAT

Схема из функциональных элементов: ациклический ориентированный граф, каждая вершина имеет не более 2 входящих ребер

- вершины входящей степени 0: `true` / `false`
- вершины входящей степени 1: `NOT`
- вершины входящей степени 2: `AND` / `OR`

Вершины степени 0 называем входными.

Задача CIRCUIT-SAT: дана схема из функциональных элементов, одна вершина помечена как выход. Некоторые входные вершины - неизвестные, можно менять их значения. Можно ли подобрать значения входных вершин так, чтобы значение выходной вершины стало равным `true`?

CIRCUIT-SAT $\to$ SAT

Введем булевскую переменную для каждой вершины схемы из функциональных элементов.

Для каждой вершины g (кроме неизвестных) запишем КНФ:

- `true`:
$$(g)$$
- `false`:
$$(\bar{g})$$
- `NOT`, входит ребро из вершины h:
$$(g \vee h) \wedge (\bar{g} \vee \bar{h})$$
- `OR`, входят ребра из вершин h_1, h_2:
$$(g \vee \bar{h}_1) \wedge (g \vee \bar{h}_2) \wedge (\bar{g} \vee h_1 \vee h_2)$$
- `AND`, входят ребра из вершин h_1, h_2:
$$(\bar{g} \vee h_1) \wedge (\bar{g} \vee h_2) \wedge (g \vee \bar{h}_1 \vee \bar{h}_2)$$

Любая NP-задача $\to$ CIRCUIT-SAT

Утверждение. Любой полиномиальный алгоритм можно смоделировать схемой из функциональных элементов полиномиального размера.

Абстрактный компьютер: память содержит

- данные
- инструкции программы
- счетчик инструкций

Представим каждый бит памяти вершиной `true` / `false`.

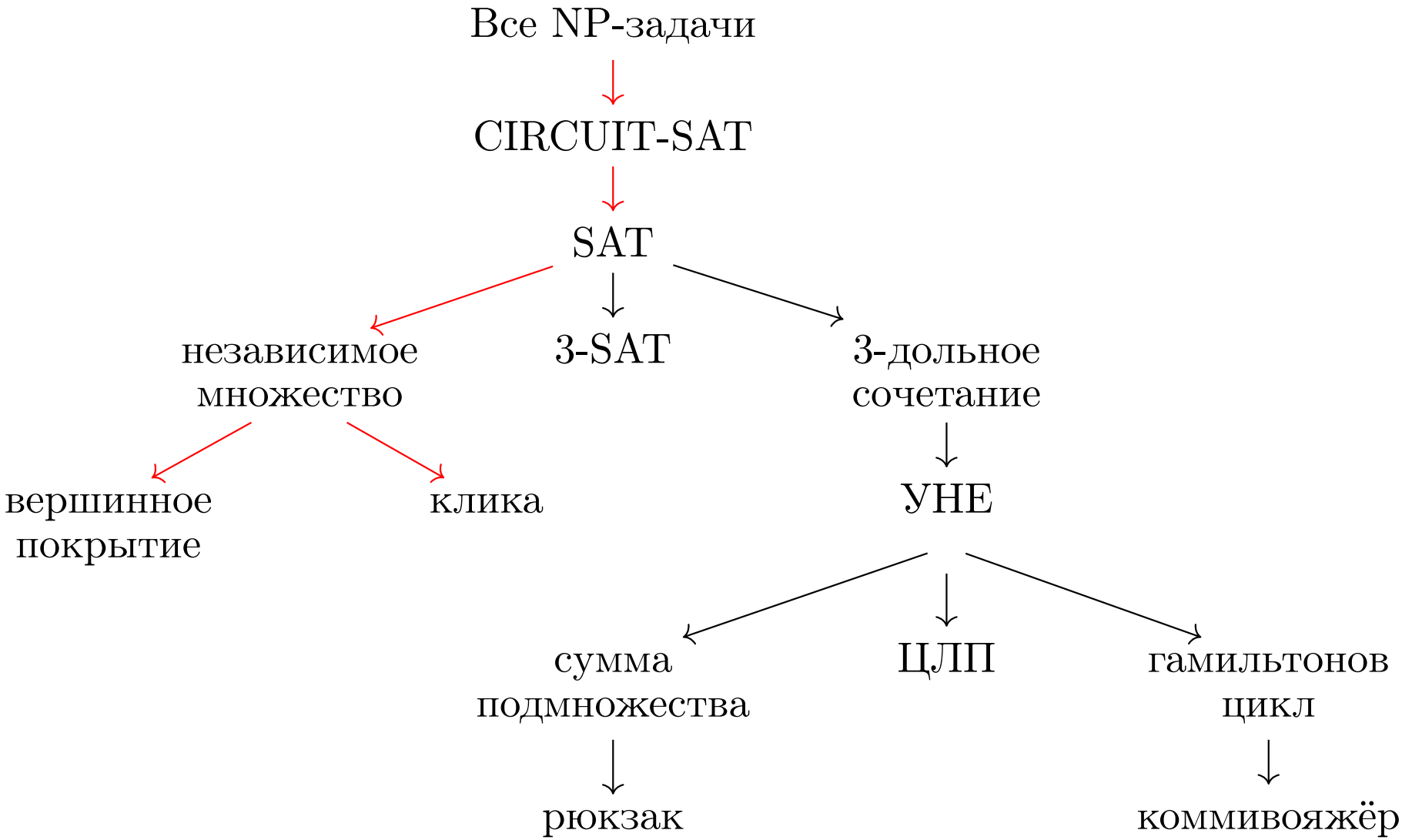
Исполнение одной инструкции можно реализовать конечной схемой из функциональных элементов. Если время работы программы ограничено полиномом, то она моделируется схемой полиномиального размера.

Любая NP-задача $\rightarrow$ CIRCUIT-SAT

Для произвольной NP-задачи существует полиномиальный проверяющий алгоритм $\mathcal{C}(I, S)$.

Представим его схемой из функциональных элементов. Биты, соответствующие кандидату на решение S, будут неизвестными. Решение NP-задачи сводится к подбору неизвестных входных решений так, чтобы на выходе получить `true`.

Схема доказательства NP-полноты



SAT $\to$ 3-SAT

Пусть имеется дизъюнкт

$$(x_1 \vee x_2 \vee x_3 \vee \ldots \vee x_m)$$

Введем новую переменную y_1 и заменим его на

$$(x_1 \vee x_2 \vee y_1) \wedge (\bar{y}_1 \vee x_3 \vee \ldots \vee x_m)$$

Продолжая таким же образом, получим эквивалентную 3-КНФ:

$$(x_1 \vee x_2 \vee y_1) \wedge (\bar{y}_1 \vee x_3 \vee y_2) \wedge (\bar{y}_2 \vee x_4 \vee y_3) \wedge \ldots \wedge (\bar{y}_{m-3} \vee x_{m-1} \vee x_m)$$

SAT: не более 2 вхождений каждого терма

Любая задача SAT может быть сведена к SAT, в которой каждая переменная встречается не более *трех* раз. Действительно, пусть переменная x встречается $k > 3$ раз. Заменим каждое ее вхождение на новую переменную x_1, x_2, $\ldots$ x_k и добавим набор клауз, которые заставят все эти переменные принимать одинаковые значения:

$$(x_1 \vee \bar{x}_2) \wedge (x_2 \vee \bar{x}_3) \wedge \ldots \wedge (x_{k-1} \vee \bar{x}_k) \wedge (x_k \vee \bar{x}_1)$$

Если все вхождения переменной – одинаковые термы (либо x_i, либо $\bar{x}_i$) – такая переменная называется чистой (pure literal). От чистых переменных можно избавиться: присвоить значение *true* или *false* в зависимости от того, в виде какого терма она входит.

В результате можем считать, что каждый терм входит в SAT формулу не более *двух* раз. Это будет полезно при сведении SAT к задаче о трехдольном сочетании.

3-дольное сочетание

Дано:

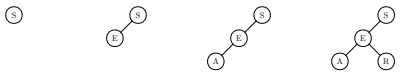
- множества $A = \{a_1, \dots a_n\}$, $B = \{b_1, \dots b_n\}$, $C = \{c_1, \dots c_n\}$;
- множество сочетаемых троек $F = \{(a_i, b_j, c_k)\} \subset A \times B \times C$.

Требуется:

выбрать n троек из F так, чтобы каждый элемент из A, B и C входил в какую-то из выбранных троек.

SAT $\to$ 3-дольное сочетание

Для каждой переменной x_i построим конструкцию:



Элементы a_{i0}, a_{i1}, b_{i1}, b_{i2} мы не будем включать в другие тройки.

Следовательно, если задача о трехдольном сочетании будет иметь решение, то оно должно будет включать в себя тройки

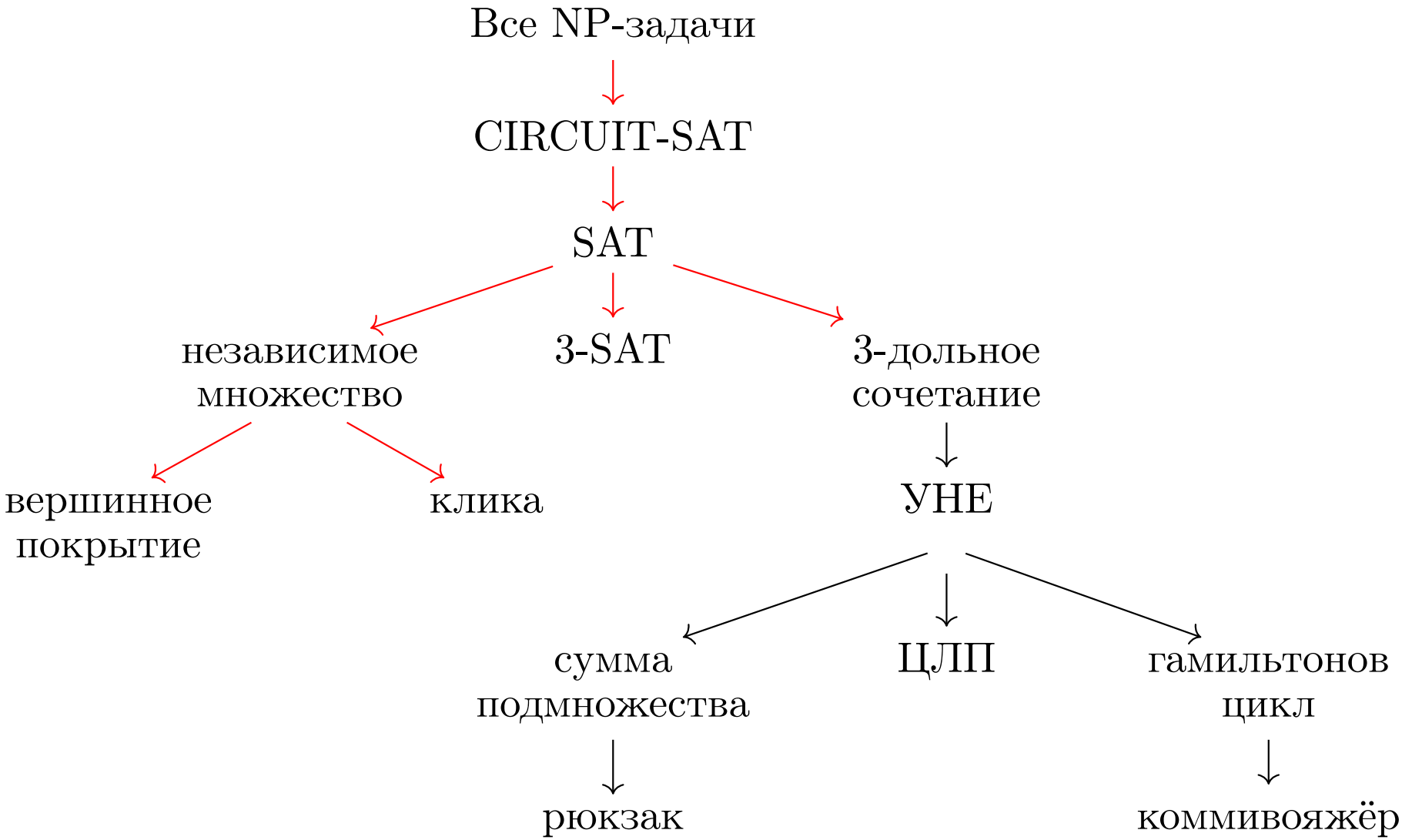
$$(a_{i0},\, b_{i0},\, c_{i00}) \text{ и } (a_{i1},\, b_{i1},\, c_{i11})$$

или

$$(a_{i0},\, b_{i1},\, c_{i01}) \text{ и } (a_{i1},\, b_{i0},\, c_{i10})$$

SAT $\to$ 3-дольное сочетание

Схема доказательства NP-полноты



Линейное программирование:

Дано: A – матрица $m \times n$,
b – вектор длины m, c – вектор длины n.

Требуется: найти вектор x длины n такой, что

- $x \geq 0$;
- $Ax \leq b$;
- $c \cdot x$ – достигает максимума.

Целочисленное линейное программирование:

Вектор x должен быть целочисленным.

Уравнения в нулях и единицах (УНЕ):

Матрица A состоит из нулей и единиц.
Надо найти x из нулей и единиц, для которого $Ax = \mathbf{1}$.

Уравнения в 0 и 1 $\to$ целочисленное ЛП

Уравнения в нулях и единицах:

$$Ax = 1, \quad x_i = 0, 1$$

Составим такую задачу целочисленного линейного программирования:

$$\begin{pmatrix} A \\ -A \\ E \end{pmatrix} x \leq \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}$$

3-дольное сочетание $\to$ уравнения в 0 и 1

Дано:

- множества $A = \{a_1, \dots a_n\}$, $B = \{b_1, \dots b_n\}$, $C = \{c_1, \dots c_n\}$;
- множество сочетаемых троек $F = \{(a_i, b_j, c_k)\} \subset A \times B \times C$.

Требуется:
выбрать n троек из F так, чтобы каждый элемент из A, B и C входил в какую-то из выбранных троек.

Введем переменную x_s для каждой тройки $s = (a_i, b_j, c_k) \in F$.

Для каждого $i = 1, \dots n$ запишем уравнение

$$\sum_{s=(a_i,\cdot,\cdot)} x_s = 1$$

То же самое для b_i и c_i. Получаем $3n$ уравнений в 0 и 1.

Задача о сумме подмножества

Переформулируем задачу о рюкзаке в виде задачи поиска:

Дано:

- Рюкзак максимальной вместимости W (по весу)
- n предметов, для которых задан вес w_i и ценность v_i
- число g (goal = цель)

Требуется: Выбрать предметы общего веса $\leq W$ и суммарной ценности $\geq g$.

Частный случай задачи о рюкзаке: все $v_i = w_i$ и $g = W$.

Дано:

- n чисел v_i
- число g

Требуется: выбрать подмножество этих чисел, чтобы общая сумма равнялась g.

Уравнения в 0 и 1 $\to$ сумма подмножества

$$Ax = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Рассмотрим столбцы матрицы A как некоторые числа, записанные в $(n+1)$-ичной системе счисления.

Уравнения в 0 и 1 $\to$ гамильтонов цикл

Сначала сведем задачу УНЕ к задаче о гамильтоновом пути с парными ребрами.

Дано: Граф $G = (V, E)$ и множество пар ребер

$$C = \{(e_1, e_2) \mid e_1, e_2 \in E,\ e_1 \neq e_2\}.$$

Требуется: Найти гамильтонов цикл в G такой, что для каждой пары ребер $(e_1, e_2) \in C$ в гамильтонов путь входит ровно одно из ребер e_1 и e_2.

УНЕ $\to$ гамильтонов цикл с парными ребрами

Составим граф: “цикл” с кратными ребрами

- каждой переменной x_i соответсвуют 2 кратных ребра (одно пометим $x_i = 0$, другое – $x_i = 1$);
- каждому уравнению $x_{j1} + \ldots + x_{jk} = 1$ соответствует k кратных ребер (пометим их $x_{j1}, \ldots x_{jk}$)
- множество пар $C = \{$(ребро переменной с пометкой $x_j = 0$, ребро уравнения с пометкой x_j)$\}$.

Гамильтонов цикл с парными ребрами $\rightarrow$ гамильтонов цикл

Будем избавляться от ограничений (парных ребер) одно за другим, заменяя каждую пару ребер $((a, b), (c, d)) \in C$ на следующий блок:

Гамильтонов цикл $\to$ коммивояжёр

Пусть G – входной граф для задачи о гамильтоновом цикле.

Присвоим длину 1 всем ребрам графа G. Проведем ребра между всеми парами вершин, между которыми еще не было ребер, и присвоим им длину $1 + \alpha$.

Количество ребер гамильтонова цикла в графе с V вершинами равна V.

Коммивояжёр может объехать за $V \Leftrightarrow$ в исходном графе есть гамильтонов цикл.

Схема доказательства NP-полноты