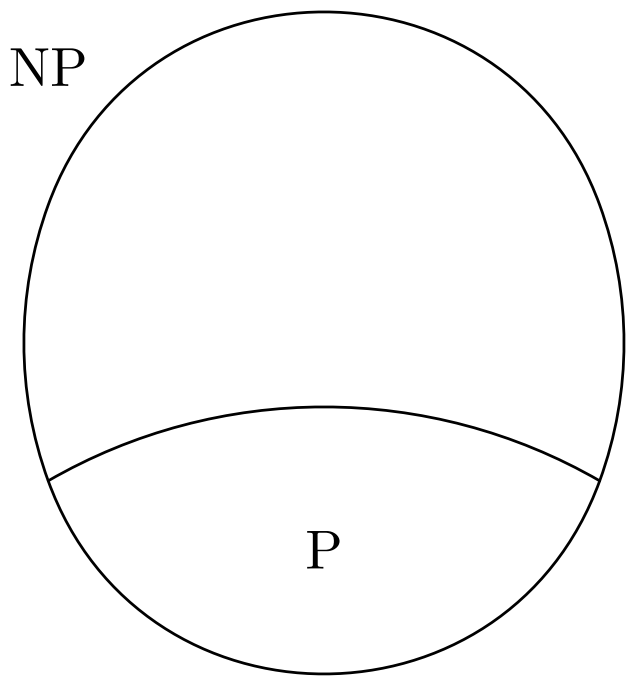
Адрианов Н.М.
Иванов А.Б.

Алгоритмы и структуры данных

NP-полные задачи

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 C, который для любых входных данных I и кандидата на решение S выполняет проверку:

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

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.

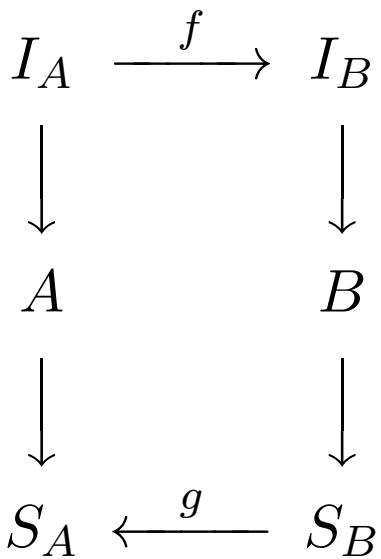
Требуется: найти гамильтонов цикл стоимости $\leq b$.

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.

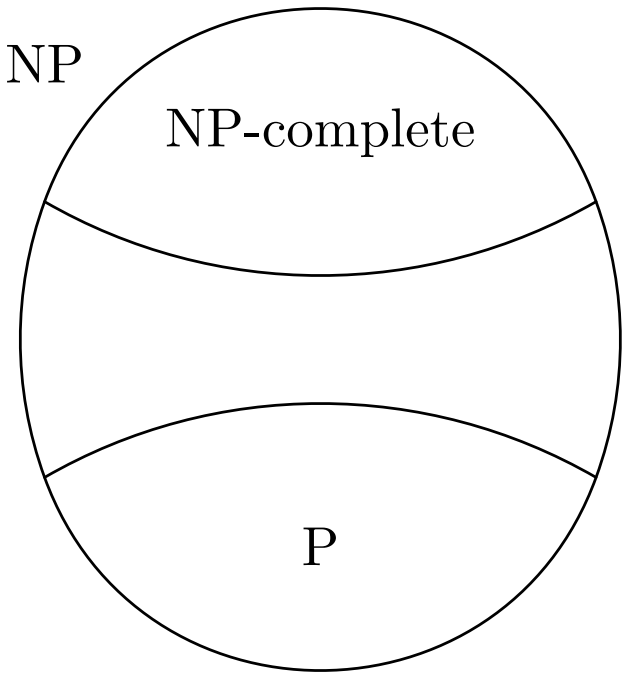
Cook: премия Тьюринга, 1982
Karp: премия Тьюринга, 1985
Левин: премия Кнута, 2012

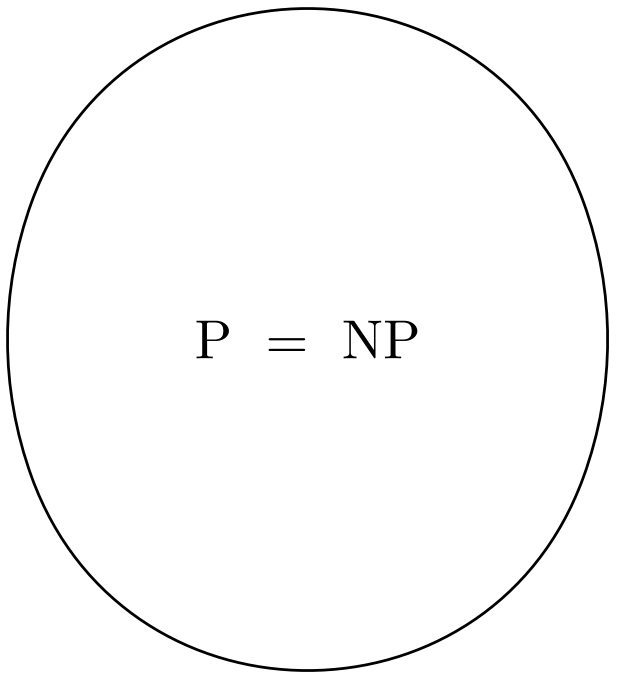
(Некоторые) NP-полные задачи

- 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?

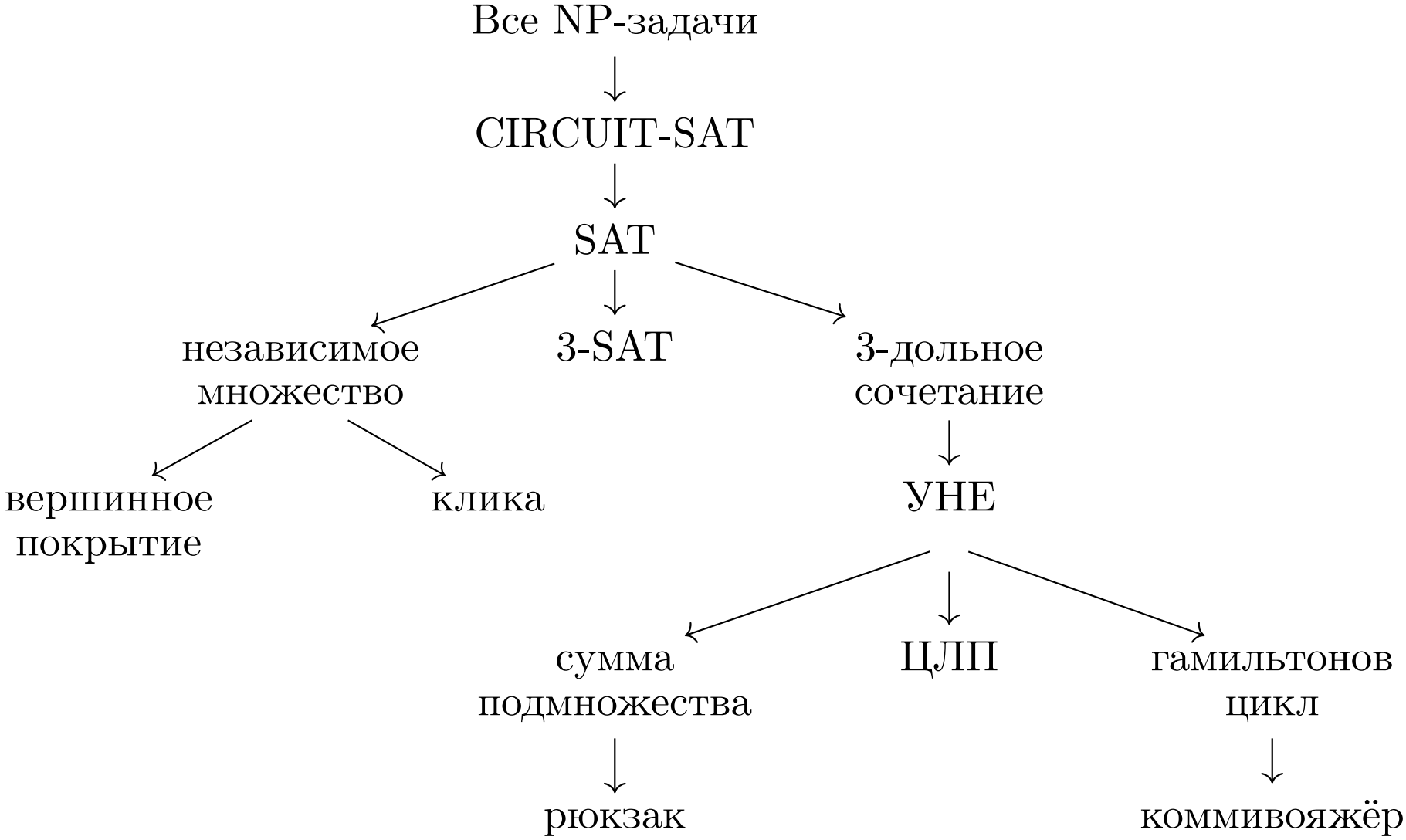




От простого до сложного – один шаг

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



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 → 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`.