Понятие сложности задачи.

Будем говорить, что задача *полиномиально разрешима*, если существует алгоритм решающий эту задачи при любых начальных данных, и количество выполняемых алгоритмом элементарных операции ограниченно некоторым полиномом от длины исходных данных.

Для всех задач, которые мы будем рассматривать в нашем курсе, множество возможных решений является конечным, поэтому существует алгоритм перебора, но мощность множества возможных решений обычно является экспонентой от длины исходных данных.

Пример. Задана булева функция от n переменных. Определить существует ли набор значений выполняющих эту функцию.
Будем считать, что значение функции вычисляется за n операций. Возможных значений переменных 2^n. Поэтому задачу можно решить, выполнив $2^n n$ операций. При n=100, получается $2^{100} \cdot 100 \approx 10^{32}$ операций.

Проанализируем полученное решение. Рекорд быстродействия компьютера на 2013 год 10^{15} операций в секунду. В году около $3{\cdot}10^7$ секунд. Получается $\frac{10^{32}}{10^{15}\cdot 3\cdot 10^7}\approx 3\cdot 10^9$ лет непрерывной работы рекордного компьютера.

Пример. Пусть два алгоритма решают задачу на 100 переменных за приемлемое время. Трудоемкость первого ограничена экспонентой 2^n, а второго — многочленом n^2. Какой размерности задачи будут решаться, если быстродействие компьютера увеличить в 1000 раз.
Ответ. 110 и 316.

Поэтому можно считать задачи, для которых существует полиномиальный алгоритм простыми, а те для которых не существует — сложными.

Определение. Полиномиальный алгоритм будем называть *эффективным*.

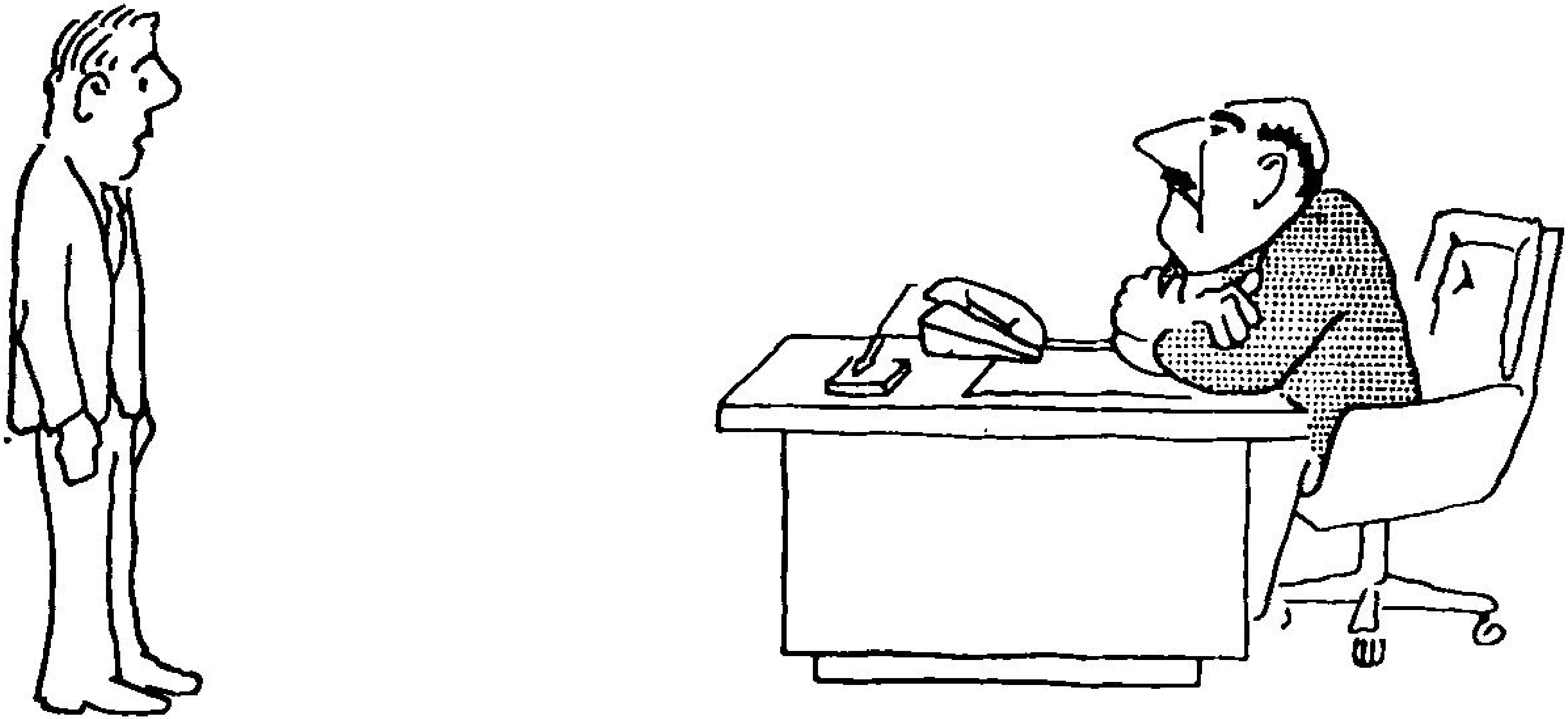
Близкие по постановке полиномиально разрешимые и трудные задачи.

Пример. Задачи об Эйлеровом маршруте и Гамильтоновом пути.

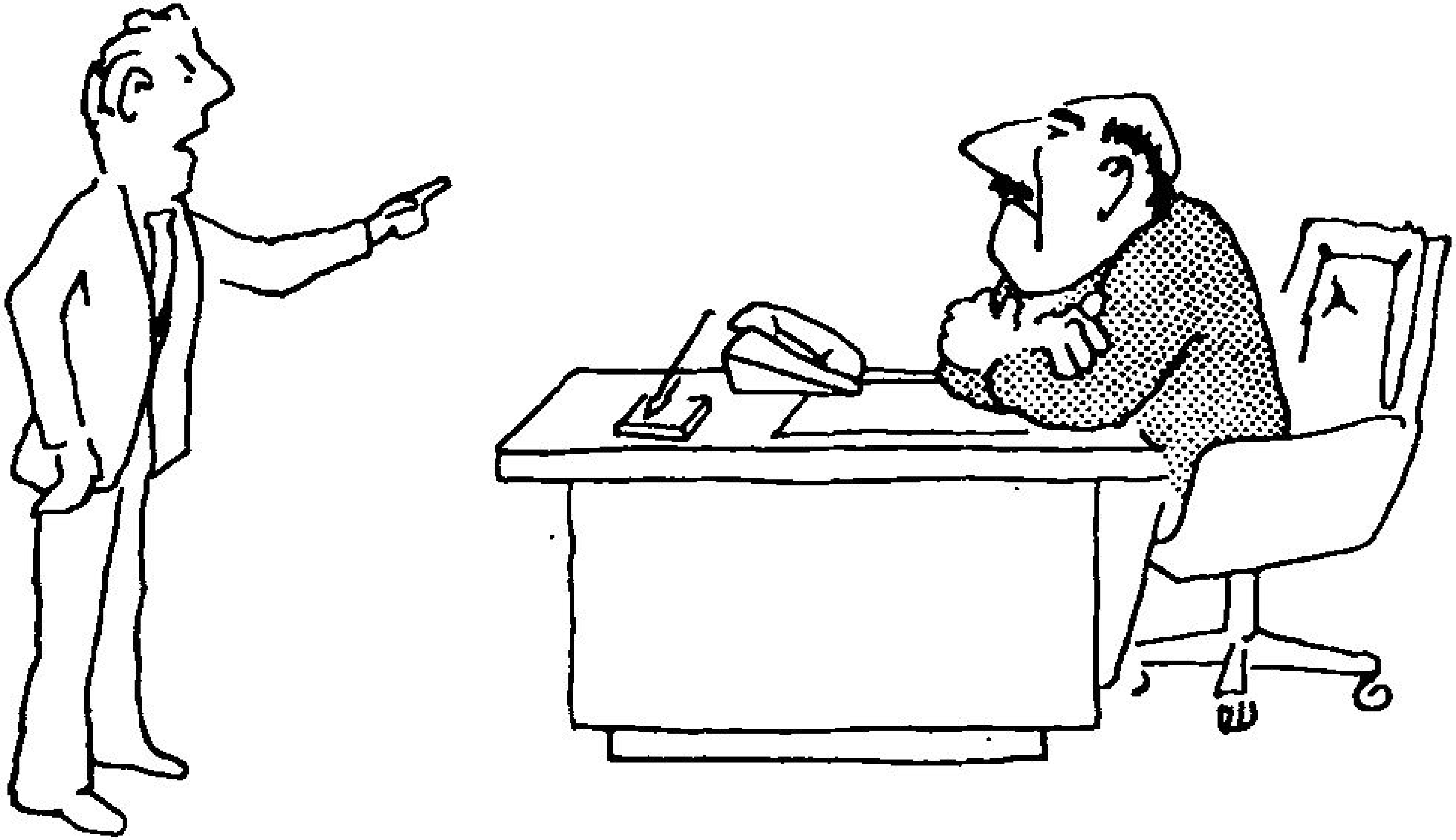
Пример. Поиск самого короткого и самого длинного по количеству ребер простого пути между двумя вершинами графа.

Определение. Будем говорить, что булева формула записана в *k-конъюктивной нормальной форме*, если она записана в конъюктивной нормальной форме и каждое слагаемое содержит ровно k переменных или их отрицаний.

Пример. Задача выполнимости 2-CNF формулы полиномиально разрешима. Для задачи выполнимости 3-CNF формулы полиномиальный алгоритм неизвестен.



"Я не могу найти эффективного алгоритма, боюсь, что я для этого слишком туп".



“Я не могу найти эффективного алгоритма, потому что такого алгоритма не существует!”

"Я не могу найти эффективного алгоритма, но этого не может сделать и никто из этих знаменитостей".

Массовая задача.

Определение. Под *массовой задачей* (или просто *задачей*) мы понимаем общий вопрос, на который следует дать ответ. Массовая задача содержит параметры, значения которых неопределенны. Задача определяется следующей информацией:

(1) общим списком всех параметров,

(2) описанием свойств, которым должен удовлетворять ответ.

Индивидуальная задача i (instance) получается из массовой, если всем параметрам присвоить конкретные значения.

Пример. Массовая задача:

(1) параметры a и b;

(2) найти $a + b$.

Индивидуальная задача 2+3

Пример. Задача о существовании гамильтонового цикла.
(1) Граф G=(V, E),
(2) Да если существует простой цикл, включающий все вершины графа.

Определение. *Задачей распознавания* свойств назовем задачу, решением которой может быть только да или нет.

Определение. *Алгоритм* — это точно и однозначно понимаемая последовательность операций, при помощи которой решается определённая вычислительная задача.

Существует ряд формальных определений алгоритма. Машина Тьюринга, рекурсивные функции, схемы функциональных элементов и др. Множества функций вычислимых с помощью разных определений практически совпадают.

Кодирование данных.

Определение. *Кодированием* множества индивидуальных задач массовой задачи Π назовем отображение множества индивидуальных задач на множество бинарных строк. *Длинной входа* назовём длину кода, соответствующего индивидуальной задаче.

Определение. Трудоёмкостью алгоритма A решения массовой задачи Π будем называть количество элементарных шагов (арифметических операций, операций сравнения, операций вызова процедуры и т.д.) необходимых для выполнения алгоритма на гипотетической ЭВМ. Обозначим $T_A(l)$ максимальную трудоемкость алгоритма на всех входах длины l.

Определение. Будем говорить, что алгоритм A полиномиальный, если $T_A(l)=O(l^k)$ для некоторого натурального k.

Определение зависит от использованной кодировки.

Пример. Определить является ли натуральное число $n>2$ простым. Существует простой алгоритм решения данной задачи.

```
i:=2;
repeat if n делится на i
       then begin write("n составное"); exit end
       else i:=i+1
until i²>n;
write("n простое").
```

В стандартной двоичной кодировке длина входа $l=\lceil \log_2 n \rceil$. Трудоёмкость этого алгоритма $O\left(\sqrt{n}\right)=O\left(\sqrt{2^{\log_2 n}}\right)=O\left(\sqrt{2}^{\,l}\right)$ является экспоненциальной.

Если использовать унарную кодировку, то длина входа равна n и алгоритм полиноминален.

Определение. Говорят, что функция $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ *полиномиально вычислима*, если существует полиномиальный алгоритм A, который для произвольных данных x из $\{0, 1\}^*$ вычисляет $f(x)$.

Определение. Говорят, что кодировки φ_1 и φ_2 массовой задачи Π *полиномиально связанными*, если существуют две полиномиально вычислимые функции f_{12} и f_{21} такие, что $f_{12}(\varphi_1(i))=\varphi_2(i)$ и $f_{21}(\varphi_2(i))=\varphi_1(i)$ для любой индивидуальной задачи i.

Лемма. Пусть φ_1 и φ_2 две полиномиально связанные кодировки задачи Π. Полиномиальный алгоритм решения задачи для кодировки φ_1 существует тогда и только тогда, когда существует полиномиальный алгоритм для кодировки φ_2.

Доказательство. Пусть в кодировке φ_1 трудоемкость решения задачи алгоритмом A равна $O(x^k)$. Пусть данные индивидуальной задачи i записаны в кодировке φ_2 строкой длины l. Вначале с помощью алгоритма вычисления функции f_{21} найдем запись $\varphi_1(i)$. Пусть трудоемкость вычисления функции f_{21} равна $O(x^m)$. Тогда длина записи $\varphi_1(i)$ не превосходит l^m. С помощью алгоритма A задача решается за время $O((l^m)^k)=O(l^{mk})$. Доказательство в обратную сторону аналогично.

Таким образом, определение класса полиномиально разрешимых задач не зависит от выбора «разумной» кодировки. Все такие кодировки полиномиально связаны.

Пример. Пусть граф задан списком смежности, длина записи $O(|V|+|E|\log|V|)$. С трудоёмкость $O(|V||E|)$ по ней можно построить матрицу инцидентности.

Независимость определения класса *P* от выбора модели ЭВМ.

Моделируемая машина *B*	Моделирующая машина *A*		
	1МТ	kМТ	МПД
Машина Тьюринга с 1 лентой (1МТ)	—	$O(T(n))$	$O(T(n)\log T(n))$
Машина Тьюринга с k лентами (kМТ)	$O(T^2(n))$	—	$O(T(n)\log T(n))$
Машина произвольного доступа (МПД)	$O(T^3(n))$	$O(T^2(n))$	—

В таблице приведено время требуемое машиной *A* для моделирования выполнения алгоритма трудоёмкость $T(n)$ на машине *B*.

Задачи, труднорешаемость которых доказуема.

Замечание. Если ответ задачи невозможно выразить полиномом от начальных данных, то задача полиномиально неразрешима.

Пример. Найти все Гамильтоновы циклы графа G.

Второй класс труднорешаемых задач, это задачи для которых доказано невозможность построения алгоритма решения, а, следовательно, и полиномиального алгоритма.

Пример. Теорема Тьюринга. Не существует алгоритма, который по произвольной программе определяет, остановится ли она на произвольном входе.

Пример. Десятая проблема Гильберта. Теорема Матиясевича. Не существует алгоритма проверяющего разрешимость в целых числах произвольного полиномиального уравнения.

Полиномиально проверяемые задачи распознавания.

Будем рассматривать только задачи распознавания. Задачу распознавания можно рассматривать как функцию $f : I \to \{0,1\}$, где I — множество всех индивидуальных задач.

Определение. Будем говорить, что массовая задача M принадлежит классу P, если существует полиномиальный алгоритм её решения. Обозначение $M \in P$.

Определение. Назовем *алгоритмом верификации* A массовой задачи M, алгоритм от двух аргументов, один из которых это код индивидуальной задачи x, а второй бинарная строка y, называемая сертификатом. Алгоритм $A(x,y)$ *проверяет* (*верифицирует*) задачу x, если существует сертификат y, такой что $A(x,y)=1$.

Определение. Будем говорить, что массовая задача M принадлежит классу полиномиально проверяемых задач, если существует полиномиальный от длины входа x алгоритм верификации $A(x, y)$, такой что для $\forall i \in I \ \left[\left(f(i)=1\right) \Rightarrow \left(\exists y\left(A(x,y)=1\right)\right)\right] \wedge \left[\left(f(i)=0\right) \Rightarrow \left(\forall y\left(A(x,y)=0\right)\right)\right]$. Обозначение $M \in NP$.

Замечание. Длина сертификата полиномиально ограничена длиной входных данных.

Пример. Задача о существовании гамильтонового цикла принадлежит NP. В качестве сертификата можно взять сам гамильтонов цикл. Очевидно, что за полином можно проверить удовлетворяет ли произвольная последовательность вершин условию простого цикла.

Определение. Задача $\bar{f} : I \to \{0,1\}$ называется *двойственной* к задаче $f : I \to \{0,1\}$.

Пример. Двойственная задача о том, что в графе нет гамильтонового цикла, имеет неустановленный статус. Неизвестно, что может являться сертификатом отсутствия гамильтонового цикла.

Пример. Задача n — составное число принадлежит NP. Сертификатом является любой делитель.

Пример. Только в 2003 году было доказано, что задача n — простое число принадлежит NP.

Обозначение NP связано с другим определением класса, как задач полиномиально разрешимых на недетерминированных машинах Тьюринга.

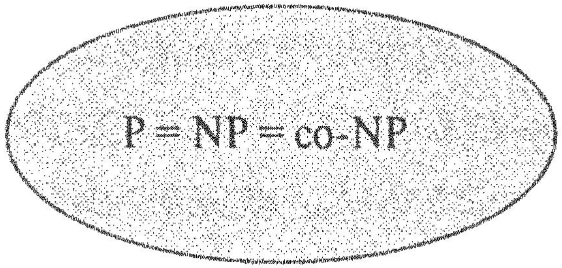
Утверждение. $P \subset NP$.

Доказательство. Пусть $M \in P$ и полиномиальный алгоритм $A(i)$ решает индивидуальные задачи i. Рассмотрим алгоритм верификации $A(i,y)$ с пустым сертификатом y. Он, очевидно, является алгоритмом верификации задачи M.

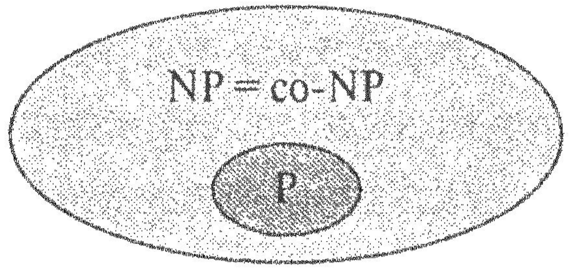
Определение. Будем говорить, что задача M принадлежит к классу co-NP, если двойственная задача принадлежит NP.

Утверждение. $P \subset co - NP$.

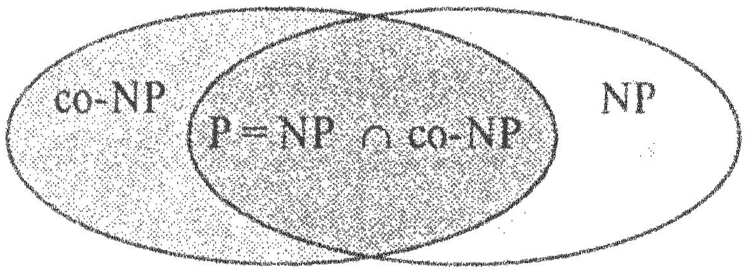
Ниже приведены 4 возможных соотношения между классами *P*, *NP* и *co-NP*.



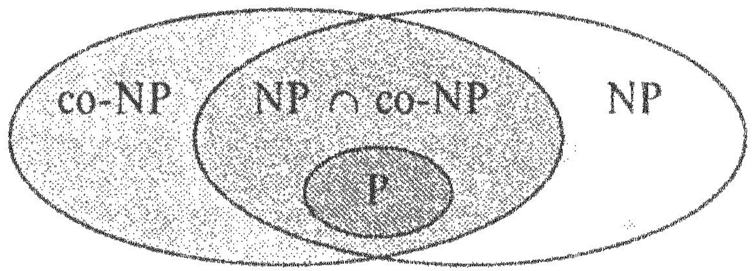
а)



б)



в)



г)

Сводимость.

Определение. Задача M_1 полиномиально сводится к задаче M_2, если существует полиномиально вычислимая функция $\varphi : I_1 \to I_2$, такая, что $f_1(i)=1$ тогда и только тогда, когда $f_2(\varphi(i))=1$. Обозначение $M_1 \leq_P M_2$.

Теорема. Если $M_1 \leq_P M_2$ и $M_2 \in P$, то $M_1 \in P$.
Доказательство. Пусть A_φ полиномиальный алгоритм, вычисляющий φ, а A_2 полиномиальный алгоритм, решающий задачу M_2. Алгоритм A_1, состоящий из последовательного применения A_φ и A_2, корректно решает задачу M_1.Оценим его трудоемкость. Пусть i — произвольная индивидуальная задача в M_1. Обозначим $|i|$ длину входа индивидуальной задачи. Тогда $|\varphi(i)| \leq |i|^k$ и $T_{A_1} = O\left(|\phi(i)|^m\right) = O\left(\left(|i|^k\right)^m\right) = O\left(|i|^{km}\right)$.

Определение. Задача M называется NP-полной если:

(1) $M \in NP$,

(2) $\forall M' \in NP \;\; M' \leq_P M$.

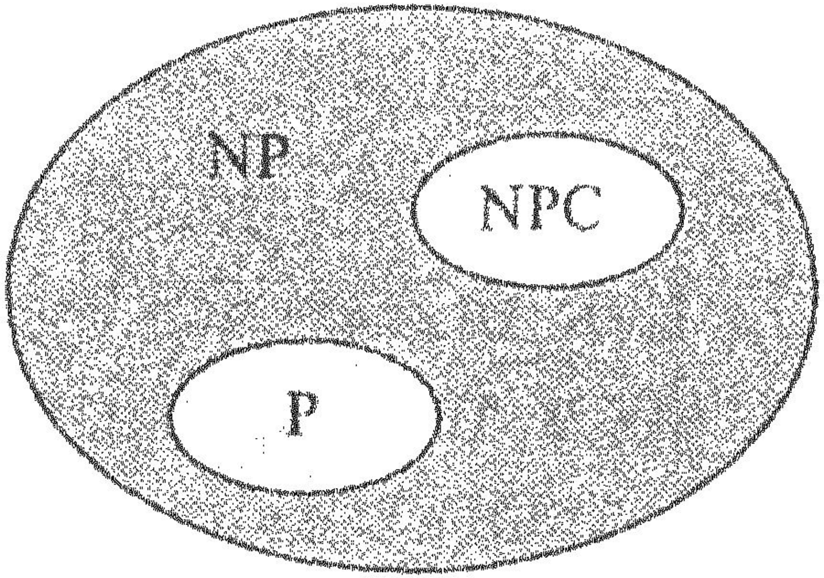
Обозначение $M \in NPC$.

Теорема. Если некоторая NP-полная задача полиномиально разрешима, то $P=NP$.

Доказательство. Из второй части определения и предыдущей теоремы следует, что для любой задачи существует полиномиальный алгоритм решения.

Пока остаётся открытым вопрос о существовании NP-полных задач. Основным достижением в теории NP-полноты является теорема Кука о существовании NP-полных задач.

Современное представление о соотношении классов P, NP и NPC.



Следствие. Если $M \in NPC$ и $P \neq NP$, то не существует полиномиального алгоритма решения задачи M.