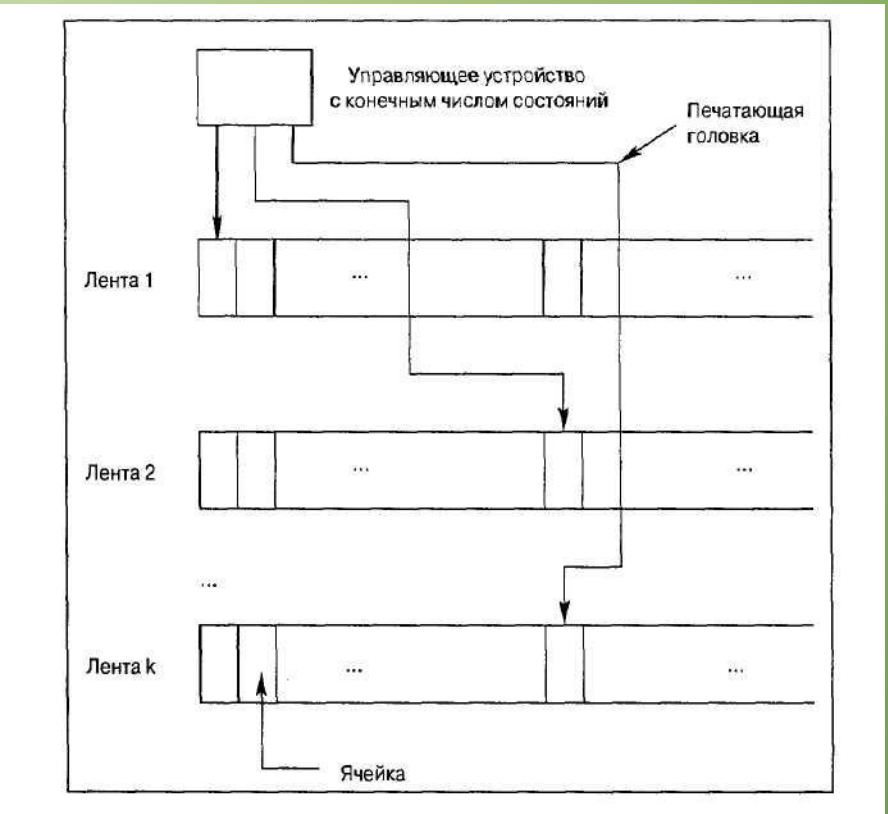
Сложность проблем

Теория сложности также классифицирует и сложность самих проблем, а не только сложность конкретных алгоритмов решения проблемы.

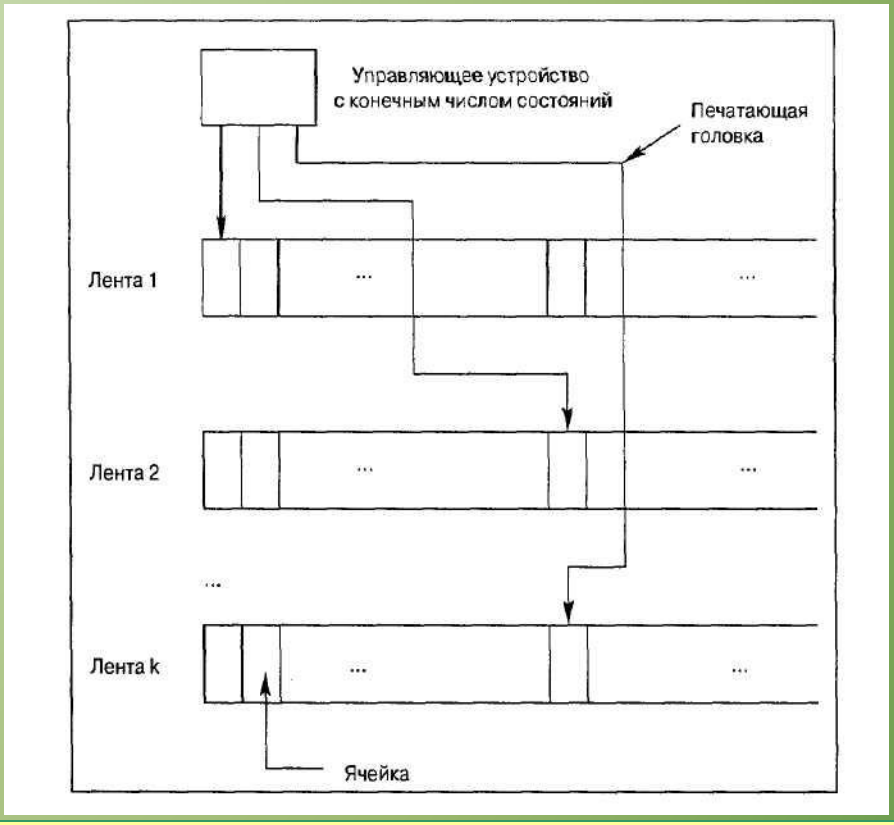
Теория рассматривает минимальное время и объем памяти, необходимые для решения самого трудного варианта проблемы на теоретическом компьютере, известном как **машина Тьюринга.**

Машина Тьюринга представляет собой конечный автомат с бесконечной лентой памяти для чтения-записи и является реалистичной моделью вычислений.



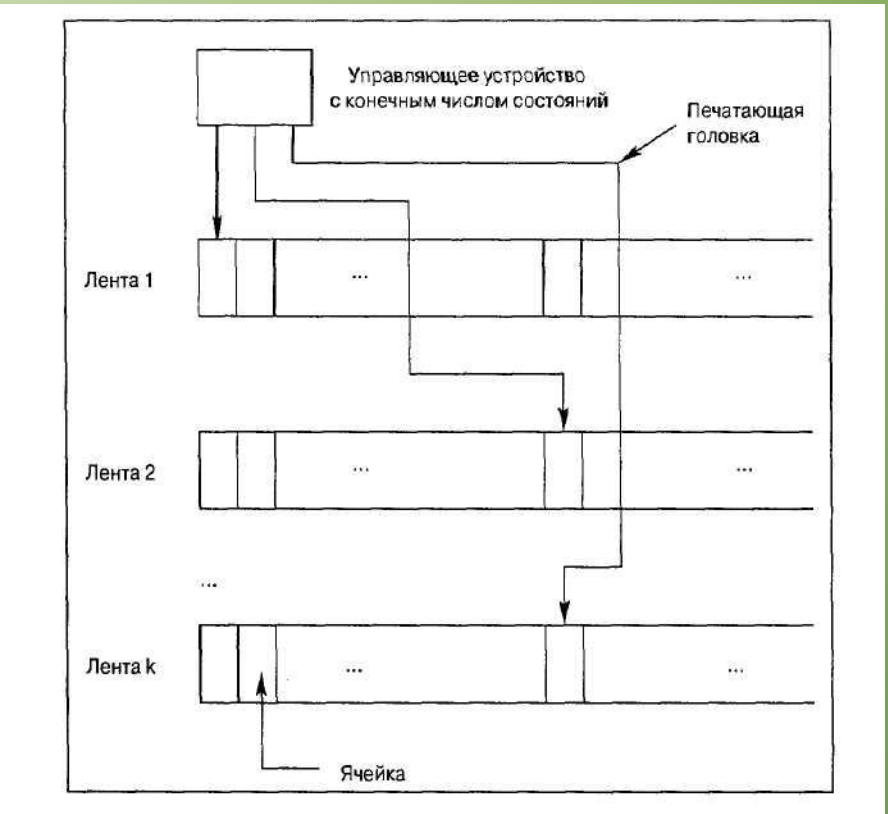
В нашем варианте машина Тьюринга состоит из управляющего устройства с конечным числам состояний (finite-state control unit), k≥1 лент (tapes) и такого же количества *головок* (tapeheads).

Управляющее устройство контролирует операции, выполняемые головками, которые считывают информацию с лент или записывают ее на ленту.

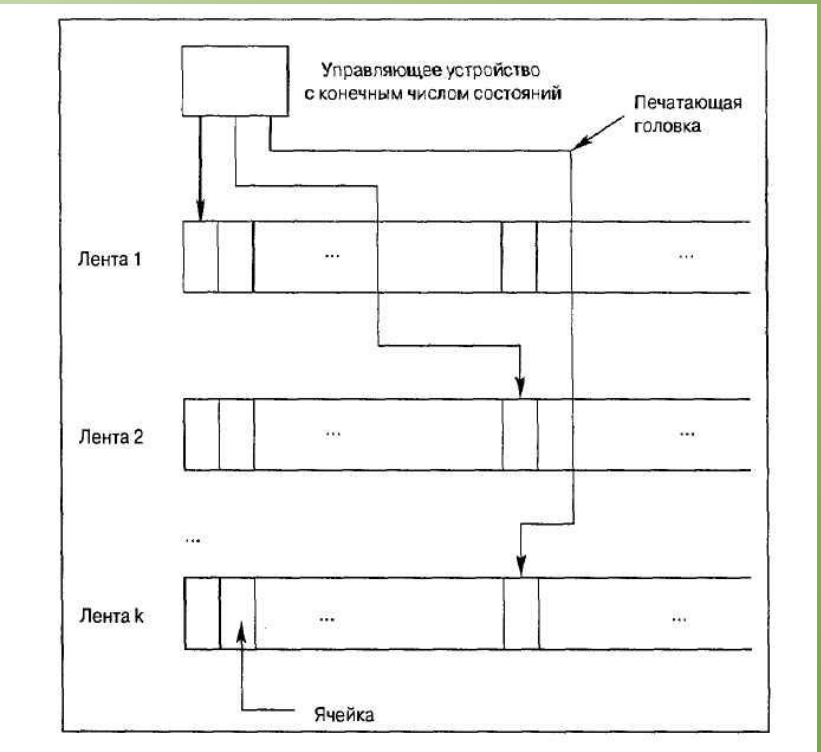


Каждая головка имеет доступ к *своей* ленте и может перемещаться вдоль нее влево и вправо.
Каждая лента разделена на бесконечное количество *ячеек* (cells).

Машина решает задачу, перемещая головку вдоль строки, состоящей из конечного количества символов, расположенных последовательно, начиная с крайней левой ячейки.

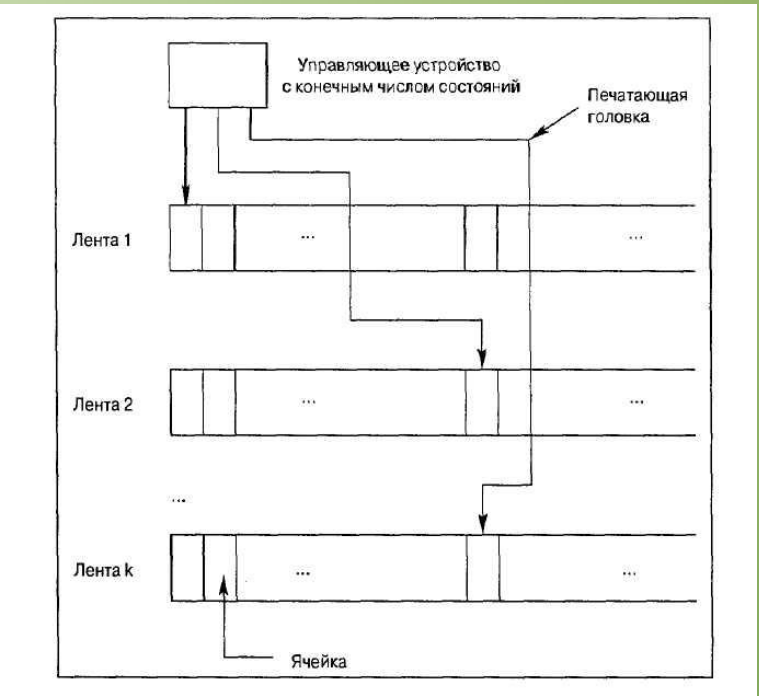


Каждый символ занимает одну ячейку, а оставшиеся ячейки ленты, расположенные справа, остаются *пустыми* (blank).
Описанная выше строка называется *исходными данными задачи* (input).
Сканирование начинается с крайней слева ячейки ленты, содержащей исходные данные, когда машина находится в предписанном *начальном состоянии* (initial state).



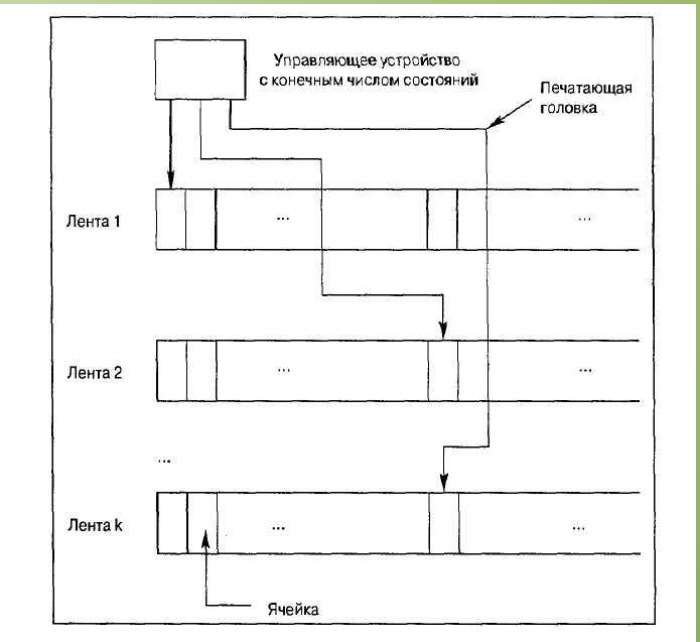
В каждый момент времени только одна из головок имеет доступ к своей ленте.

Операция доступа головки к своей ленте называется *тактом* (legal move).



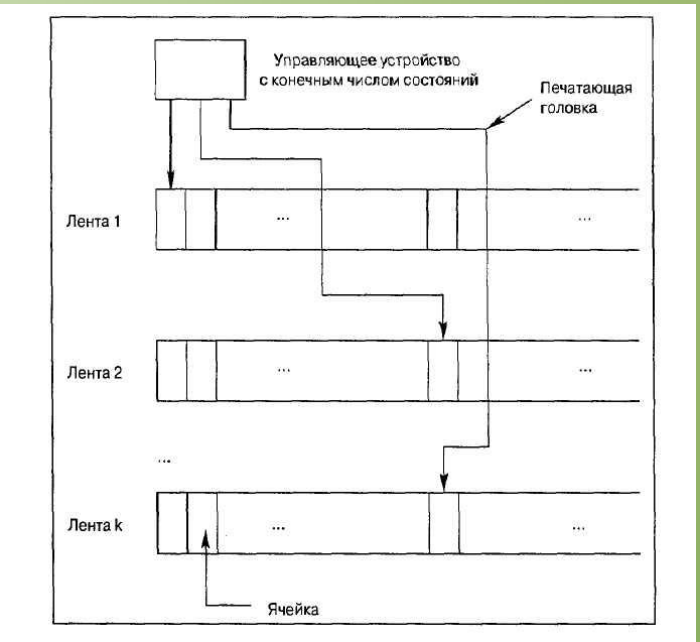
Если машина начинает работу с начального состояния, последовательно выполняет такты, сканирует исходную строку и завершает работу, достигая *заключительного состояния* (termination condition), говорят, что машина *распознает* (recognize) исходные данные.

В противном случае в некоторый момент машина не может выполнить очередной такт и останавливается, не распознав исходные данные.



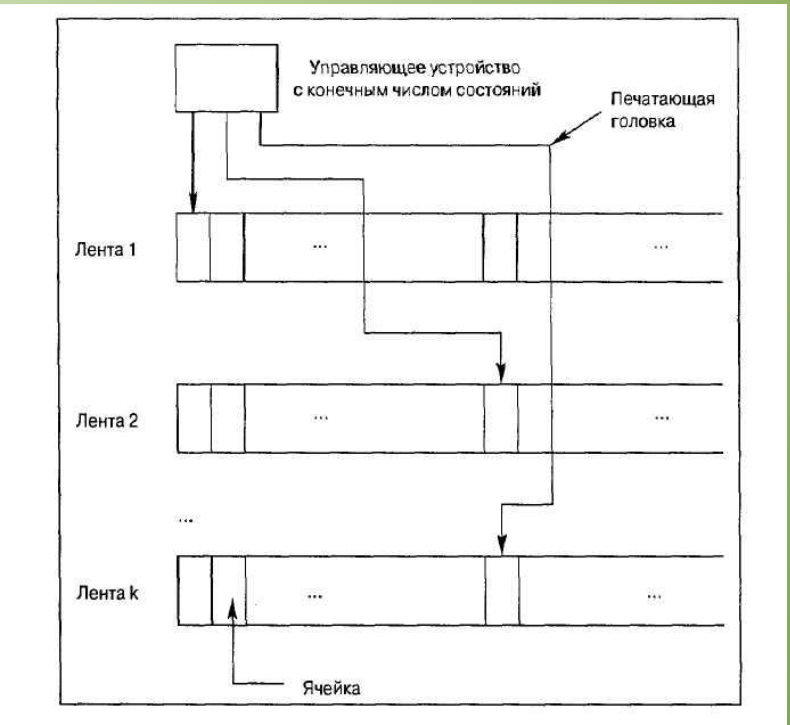
Исходные данные, распознаваемые машиной Тьюринга, называются *предложением* (instance) распознаваемого *языка* (language).
Для каждой конкретной задачи машину Тьюринга можно полностью определить с помощью функции, описывающей работу управляющего устройства с конечным числом состояний.

Такая функция может иметь вид таблицы, в которой для каждого состояния указана операция, выполняемая на следующем такте.



Количество тактов $T_м$, которые машина Тьюринга *M* должна выполнить при распознавании исходной строки, называется продолжительностью работы или *временной сложностью* (time complexity) машины *M*.

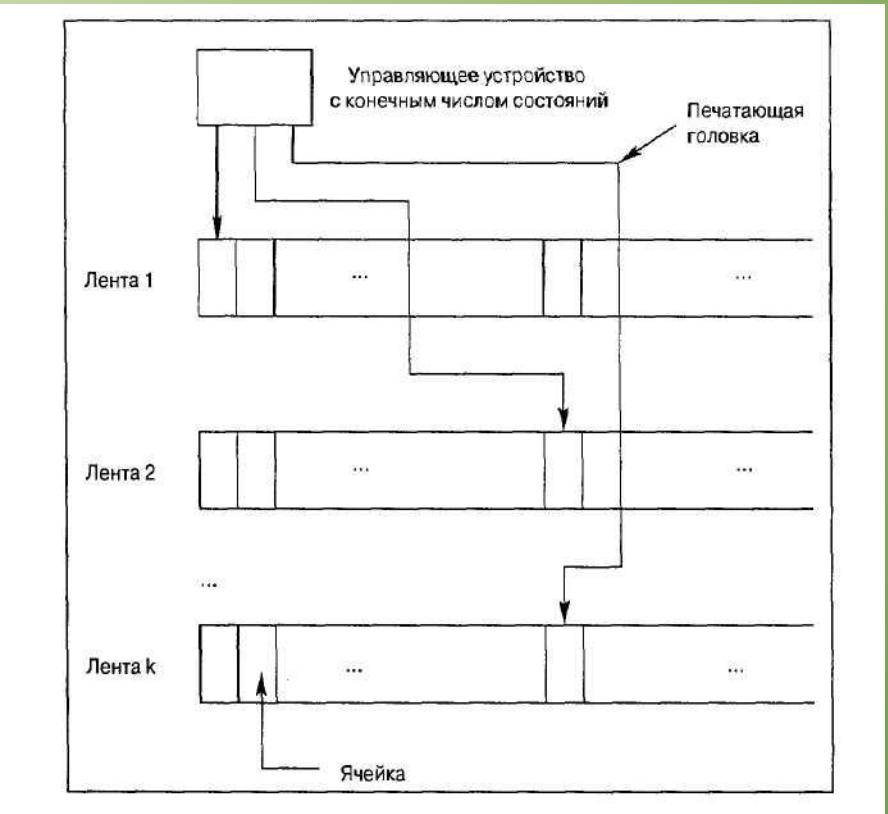
Величину $T_м$ можно представить в виде функции $T_м(n) : \mathrm{N} \rightarrow \mathrm{N}$, где *n* — *длина* (length), или *размер* (size), исходного предложения, т. е. количество символов, из которых состоит исходная строка, когда машина *M* пребывает в начальном состоянии.



Легко видеть, что $T_{\text{м}}(n) \geq n$.

Кроме временных ограничений, машина M имеет ограничения памяти S_M, представляющие собой количество ячеек, которые доступны для записи.

Величину S_M также можно представить в виде функции $S_{\text{м}}(\text{n}) : \text{N} \rightarrow \text{N}$, которая называется *пространственной сложностью* (space complexity) машины M.



Если машина начинает работу с начального состояния, последовательно выполняет такты, сканирует исходную строку и завершает работу, достигая *заключительного состояния* (termination condition), говорят, что машина *распознает* (recognize) исходные данные.

Детерминированное полиномиальное время

Функция *p(n)* является полиномиальной по целому аргументу *n,* если она имеет вид:

$$p(n) = c_k n^k + c_{k-1} n^{k-1} + \cdots + c_1 n + c_0,$$

где числа *к* и c_i ($i = 0,1,2,\ldots ,к$) — целые константы, причем $c_i \neq 0$.

Число $к \geq 0$ называется степенью (degree) полинома и обозначается как deg(p(n)),
а числа c_i называются коэффициентами (coefficients) полинома *p(n)*.

Определение : **Класс** □.

Символом □ обозначается класс языков, имеющих следующие характеристики.

Язык L принадлежит классу □, если существует машина Тьюринга M и полином p(n), такие что машина M распознает любое предложение $I \in L$ за время $T_м(n)$,

где $T_м(n) \leq p(n)$ для всех неотрицательных целых чисел n, а n — параметр, задающий длину предложения I.

В этом случае говорят, что язык L распознается за полиномиальное время.

Языки, распознаваемые за полиномиальное время, считаются *"всегда* простыми", а полиномиальные машины Тьюринга — *"всегда* эффективными".

Все машины Тьюринга, распознающие язык *L* из класса □, называются детерминированными (deterministic).

Детерминированная машина Тьюринга порождает результат, который полностью предопределен исходными данными и начальным состоянием машины.

Иначе говоря, повторный запуск машины Тьюринга с теми же исходными данными и начальным состоянием приводит к идентичным результатам.

В работах, посвященных теории вычислительной сложности, задача считается решенной, только если любой экземпляр данной задачи можно решить с помощью одной и той же машины Тьюринга (т.е. с помощью одного и того же метода).

Только в этом случае алгоритм считается достаточно общим и может называться методом.

Пример - язык DIV3

Пусть DIV3 — множество неотрицательных целых чисел, кратных трем.
Покажем, что DIV3 $\in P$.

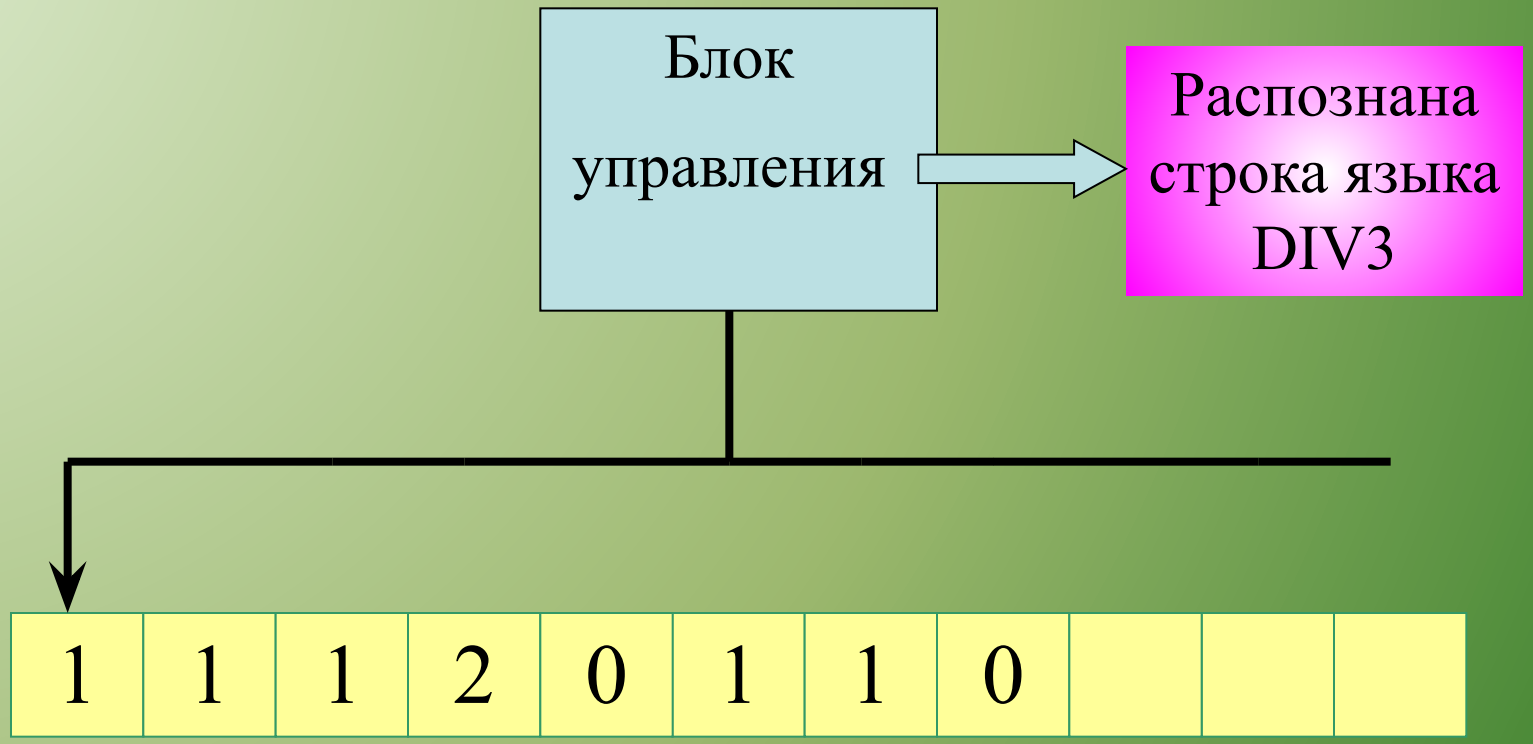
BaseForm[3333,3]

11120110_3

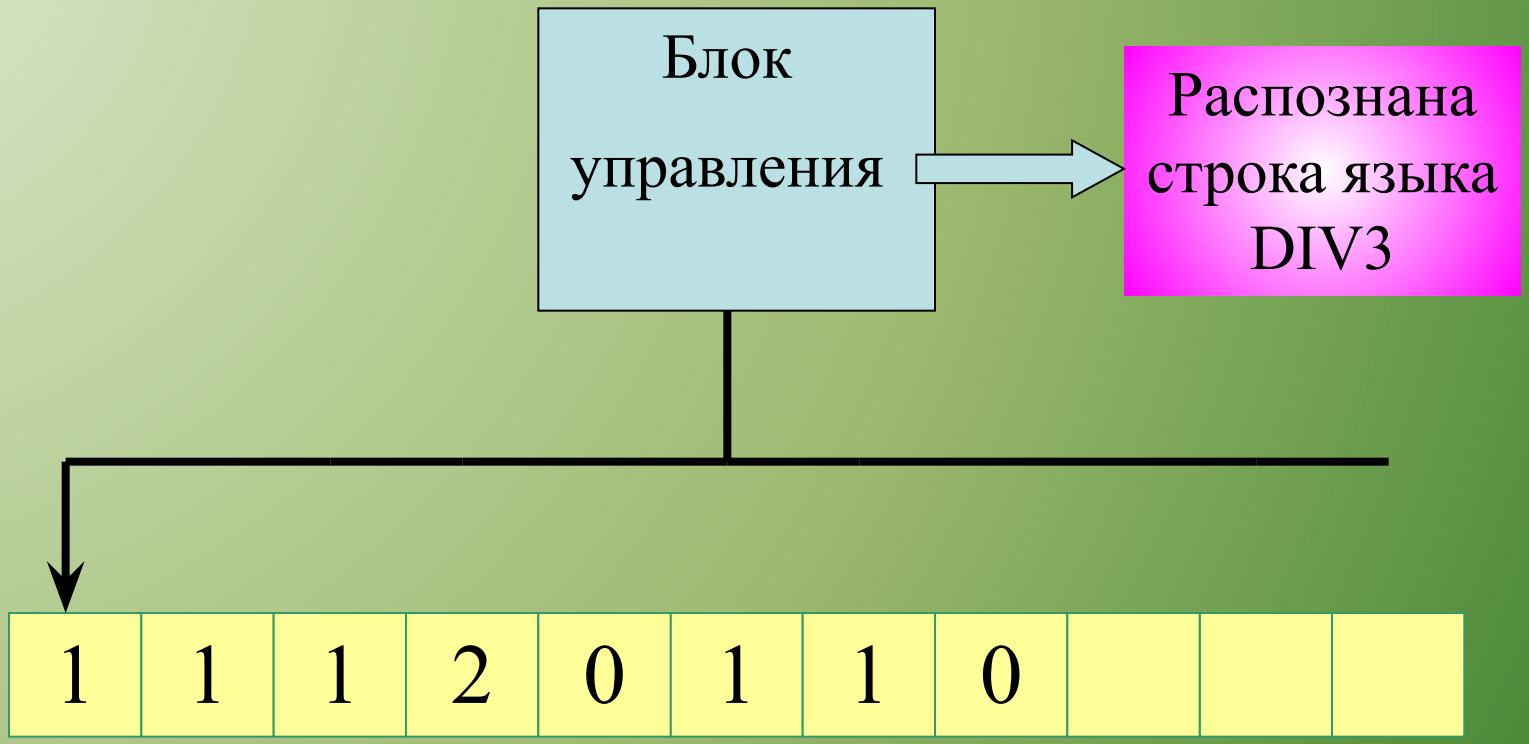
Для того чтобы распознать язык DIV3 за полиномиальное время, построим машину Тьюринга с одной лентой.

Если записать исходные данные в виде троичных чисел (в позиционной системе счисления по основанию 3), т.е. в виде строки символов из множества {0,1,2}, то задача распознавания становится тривиальной:

Исходная строка *x* принадлежит языку DIV3 тогда и только тогда, когда последняя цифра в строке *x* равна нулю.



Следовательно, создаваемая машина должна просто перемещать головку вправо, пока не обнаружит пустой символ.
Машина должна остановиться и выдать ответ "ДА", если и только если последний непустой символ был равен нулю.



Очевидно, что данная машина может распознавать любое предложение, состоящее из цифр, причем **количество шагов алгоритма равно длине исходной строки.**

Следовательно, DIV3 $\in P$.

$T_{DIV3}(n) = n$. Машина распознает язык DIV3 за полиномиальное время.

Полиномиальные вычислительные задачи

По определению класс является классом языков, распознаваемых за полиномиальное время.

Задача распознавания языка является задачей **принятия решений** (decisional problem).

При любых исходных данных результатом решения такой задачи является ответ "ДА" или "НЕТ".

Однако класс является более широким и содержит **полиномиальные вычислительные задачи** (polynomial-time computational problems).

При любых исходных данных результатом решения таких задач является более общий ответ, чем "ДА" и "НЕТ".

Поскольку машина Тьюринга может записывать символы на ленту, она позволяет решать такие задачи.

Вычислительное устройство, имеющее неймановскую архитектуру (иначе говоря, всем известную современную компьютерную архитектуру),
состоит из счетчика, памяти и центрального процессора (central processor unit — CPU),
поочередно выполняющего элементарные команды, называемые *микрокомандами.*

Load (Загрузить)
Store (Сохранить)
Add (Сложить)
Comp (Дополнение)
Jump (Перейти)
JumpZ (Условный переход)
Stop (Остановиться)

Хорошо известно ,что перечисленных выше микрокоманд достаточно для создания алгоритмов, решающих любые арифметические задачи на неймановском компьютере.

Можно доказать ,что каждую микрокоманду из указанного выше набора, можно имитировать на машине Тьюринга за полиномиальное время.

Следовательно, задачу, которую можно решить за полиномиальное время на неймановском компьютере (т.е. количество микроинструкций, используемых в алгоритме, представляет собой значение полинома, зависящего от размера исходных данных),
можно решить за полиномиальное время и с помощью машины Тьюринга.

Это возможно благодаря тому, что для любых полиномов $p(n)$ и $q(n)$ произвольные арифметические комбинации $p(n)$, $q(n)$, $p(q(n))$ и $q(p(n))$ также являются полиномами, зависящими от аргумента n.

□-символика
(order notation)

Символом □(*f*(*n*)) *обозначается функция* *g*(*n*), *для которой существует константа* c > 0 *и натуральное число N,*
такие что |*g*(*n*)| ≤ c| *f*(*n*)| *для всех* n ≥ *N*.

Используя O – символику можно отобразить временну*ю* сложность алгоритмов, рассмотрим следующую теорему:

Теорема.
Наибольший общий делитель $\gcd(a, b)$ *можно вычислить с помощью не более чем* $2\max(|a|, |b|)$ *операций модулярной арифметики.*

Эту теорему впервые доказал Ж. Ламе (1795-1870) (G. Lame). Ее можно считать первой теоремой в теории вычислительной сложности.

$\lceil x \rceil$ - минимальное целое число, большее или равное числу x;

функция **Ceiling[x]** *в пакете Mathematica*

$\lfloor x \rfloor$ - максимальное целое число, не превосходящее число x;

функция **Floor[x]** *в пакете Mathematica*

Обозначение $|x|$ - *длина целого числа* x, равная $1+\lfloor \log_2 x \rfloor$ для $x \geq 1$, или модуль числа x.

Таким образом, временная сложность алгоритма вычисления наибольшего общего делителя (при условии a > b) может быть определена как: □(log a).
Не указывая явно основание логарифма.

□-символика для поразрядных вычислений

Для оценки сложности **поразрядных** (bitwise) арифметических операций используется модифицированная □-символика.
В поразрядных вычислениях все переменные принимают значения нуль или единица, а операции носят не арифметический, а логический характер:
∧ (для операции AND),
∨ (для операции OR),
⊕ (для операции XOR, т.е. "исключающего или") и
¬ (для операции NOT).

В рамках модели поразрядных вычислений на сложение и вычитание двух целых чисел i и j затрачиваются $\max(|i|, |j|)$ побитовых операций, т.е. порядок временной сложности равен $\Theta(\max(|i|, |j|))$.

На умножение и деление двух целых чисел i и j затрачиваются $|i| \cdot |j|$ побитовых операций, т.е. порядок их временной сложности равен $\Theta(\log i \times \log j)$.

Поразрядные оценки сложности основных операций в модулярной арифметике

Операция над $a, b \in_U [1, n)$	Сложность
$a \pm b \pmod n$	$O_B(\log n)$
$a \cdot b \pmod n$	$O_B((\log n)^2)$
$b^{-1} \pmod n$	$O_B((\log n)^2)$
$a/b \pmod n$	$O_B((\log n)^2)$
$a^b \pmod n$	$O_B((\log n)^3)$

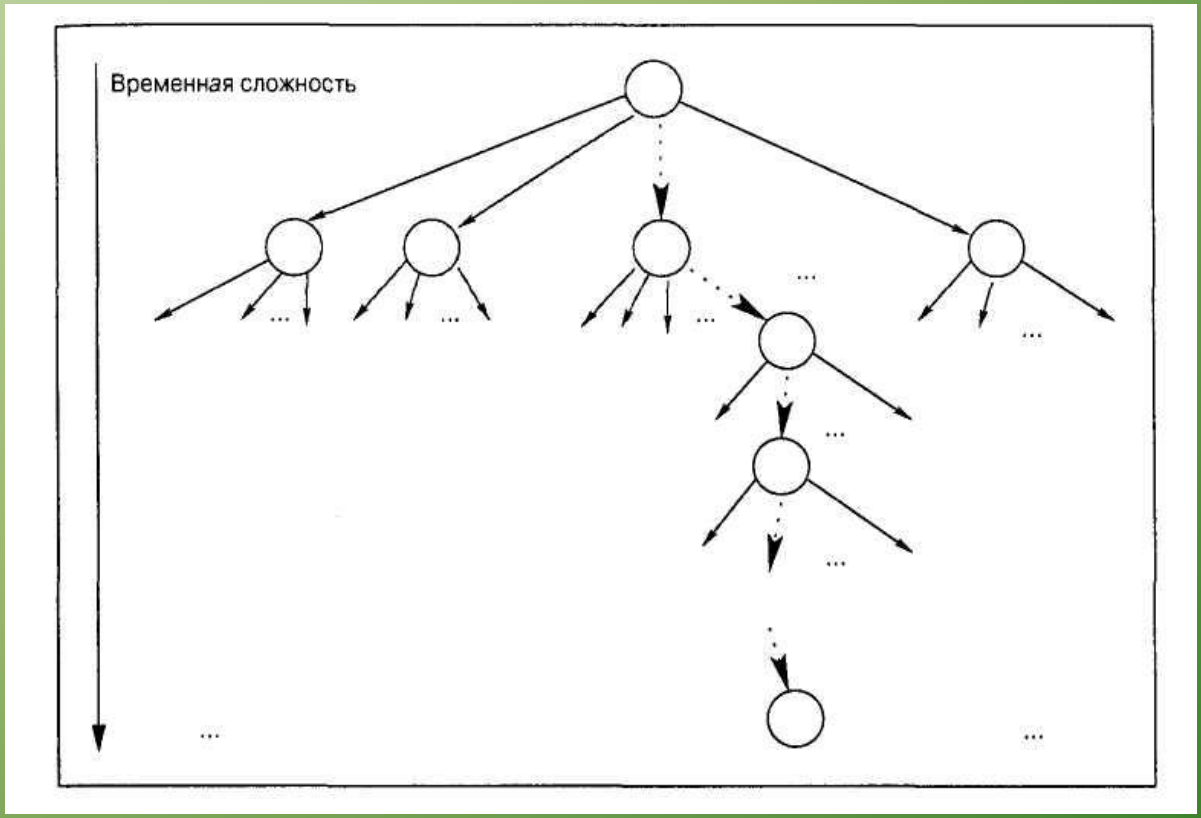
Недетерминированное полиномиальное время

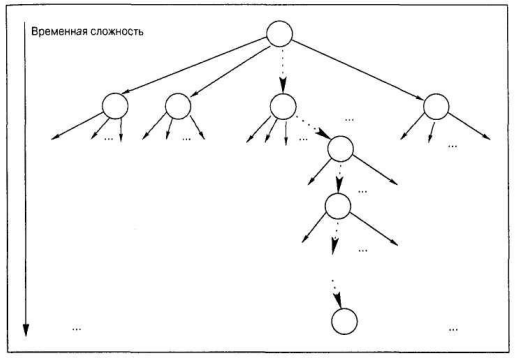
Недетерминированной машиной Тьюринга (non-deterministic Turing machine) называется устройство, на каждом такте работы которого существует конечное количество вариантов следующего такта.

Входная строка считается распознанной, если существует хотя бы одна последовательность разрешенных тактов, начинающаяся считыванием первого символа строки и завершающаяся считыванием последнего символа.

Такая последовательность тактов называется *последовательностью распознавания* (recognition sequence).

Работу недетерминированной машины Тьюринга можно представить в виде серии догадок.
В этом случае последовательность распознавания представляет собой серию правильных догадок.
Таким образом, все возможные такты образуют дерево, называемое **вычислительным деревом** (computational tree) недетерминированной машины Тьюринга.





Размер (количество узлов) этого дерева экспоненциально зависит от размера входа.

Однако, поскольку количество тактов в последовательности распознавания входной строки равно глубине дерева *d,* получаем, что *d* = □ (log (количество узлов дерева)) и количество тактов в последовательности распознавания **полиномиально** зависит от размера входной строки.

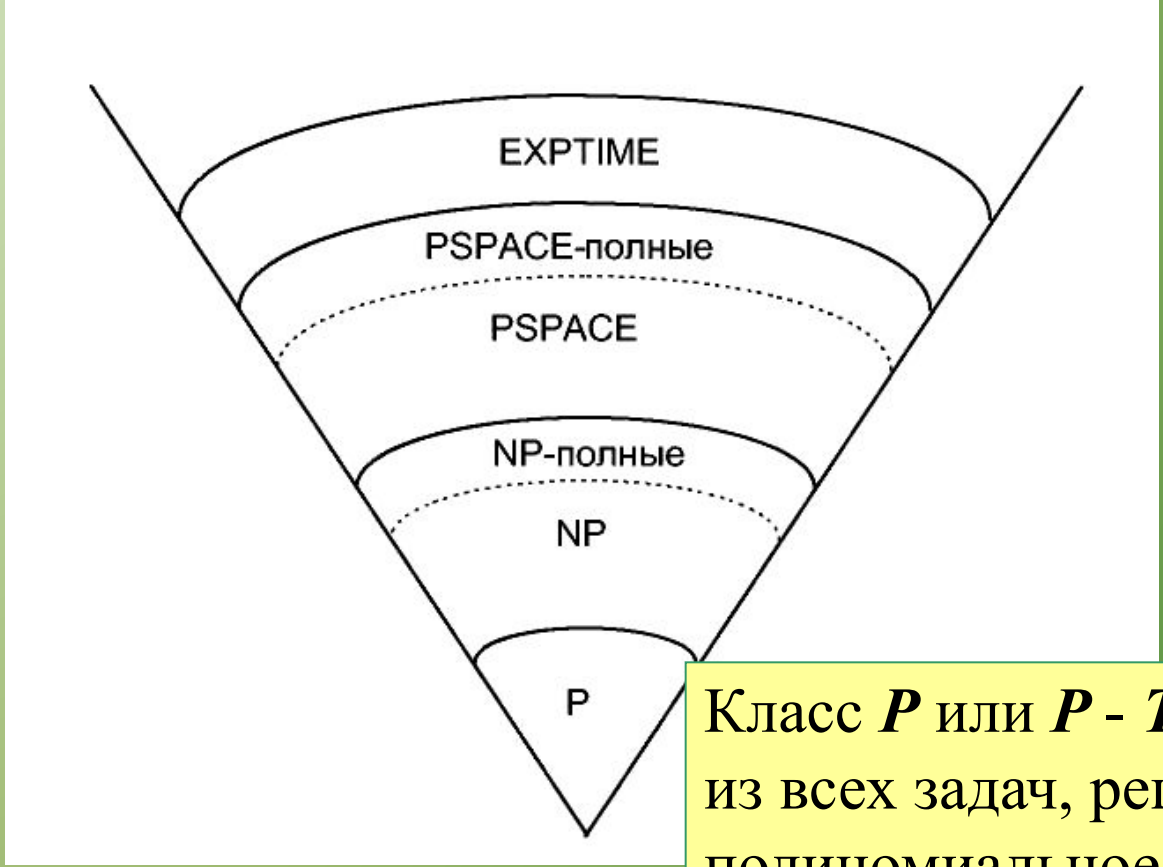
Итак, временная сложность распознавания строки с помощью серии правильных догадок полиномиально зависит от размера исходных данных.

Язык принадлежит классу □□, если он распознается недетерминированной машиной Тьюринга за полиномиальное время.

Итак, временная сложность распознавания строки с помощью серии правильных догадок **полиномиально** зависит от размера исходных данных.

Определение : ***Язык принадлежит классу □□, если он распознается недетерминированной машиной Тьюринга за полиномиальное время.***

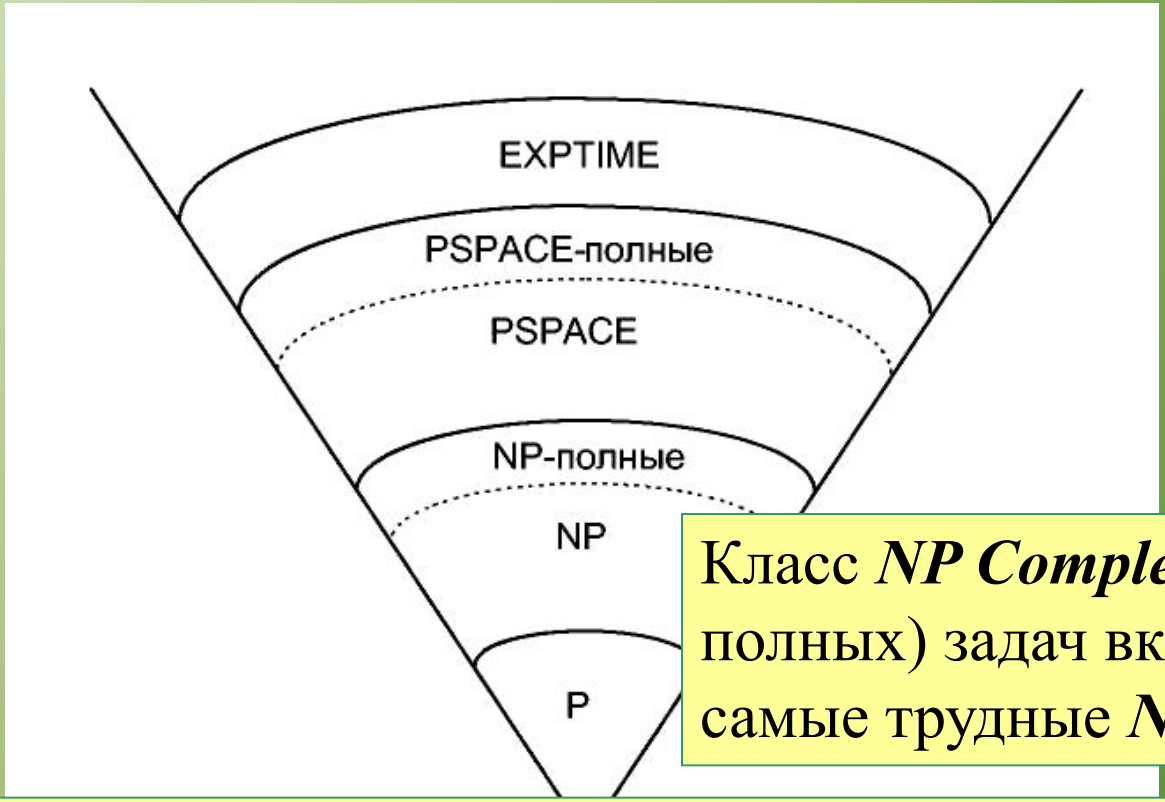
Классы сложности



Класс ***P*** или ***P - TIME*** состоит из всех задач, решаемых за полиномиальное время

Задачи, которые решаются за полиномиальное время называются ***решаемыми***, так как они обычно могут быть решены для задач достаточно большой размерности ***n***.

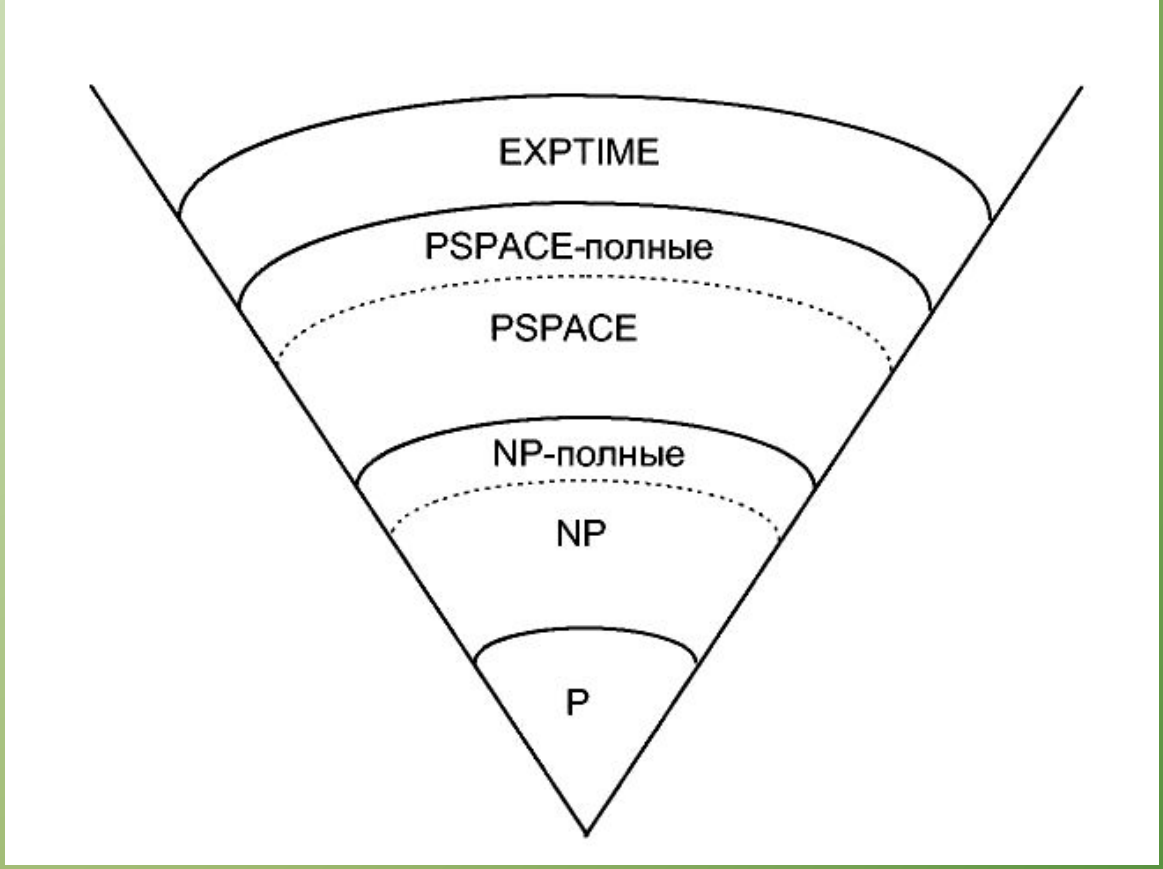
Классы сложности



Класс ***NP Complete*** (***NP*** - полных) задач включает все самые трудные ***NP*** задачи.

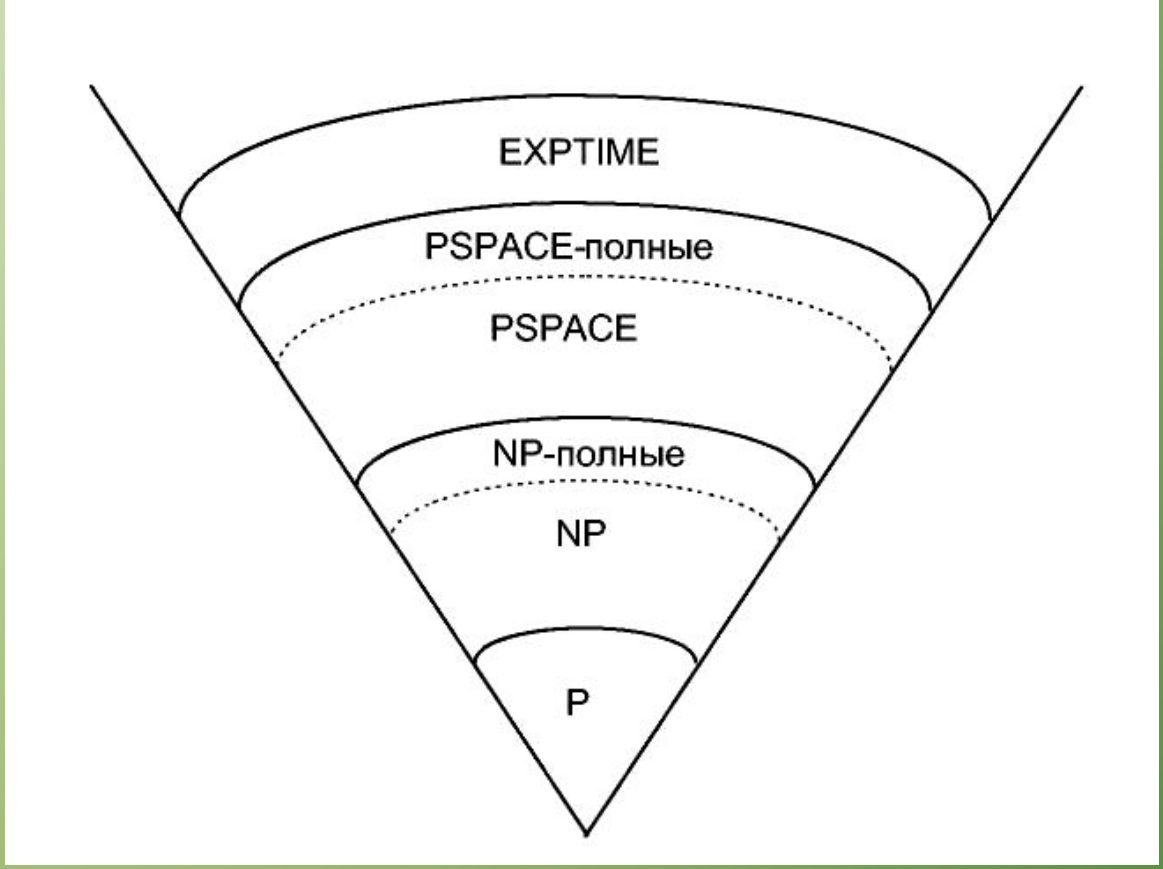
Класс ***NP*** или ***NP - TIME***, состоит из
всех задач решаемых за полиномиальное время на недетерминированной машине Тьюринга, способной параллельно выполнять
неограниченное количество независимых вычислений.

Классы сложности



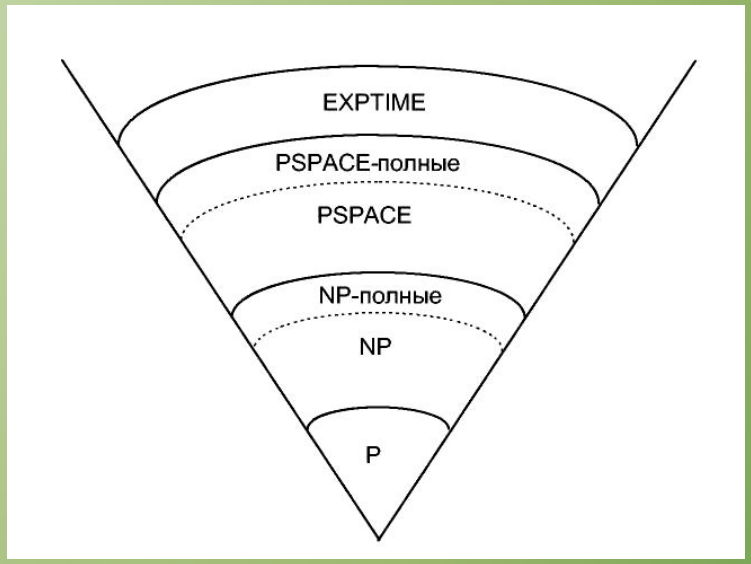
Класс ***PSPACE*** состоит из задач, требующих полиномиальных объемов машинной памяти, но не обязательно решаемых за полиномиальное время.

Классы сложности



Класс EXPTIME – эти задачи решаются за экспоненциальное время

Классы сложности



Таким образом, выбор наиболее сложных задач, для которых известно решение, и использование их в основе построения криптосистемы, позволяет создавать практически устойчивые шифры, раскрытие которых в принципе возможно, но для этого потребуется столько времени, что эта процедура дешифрования теряет практический смысл.

Основные понятия теории вероятностей

Пусть **S** — произвольное, но фиксированное множество точек,

называемое **полем вероятностей** (probability space) или

выборочным пространством (sample space).

Любая точка $x \in S$ называется **выборочным элементом** (sample point) или
исходом (outcome),
простым событием (simple event),а также
неразложимым событием (indecomposable event).

Для краткости мы будем называть этот элемент просто **точкой** (point).

Событие (составное, или разложимое)
является подмножеством множества **S** и обычно обозначается прописной буквой (например, ***E)***.

Эксперимент (experiment), или наблюдение, представляет собой извлечение точки из множества **S**.

Событие ***E*** происходит, если в результате эксперимента выясняется, что некоторая точка ***x*** из **S** принадлежит множеству ***E***.

Событие (составное, или разложимое)
является подмножеством множества **S** и обычно обозначается прописной буквой (например, ***E)***.

Эксперимент (experiment), или наблюдение, представляет собой извлечение точки из множества **S**.

Событие ***E*** происходит, если в результате эксперимента выясняется, что некоторая точка ***x*** из **S** принадлежит множеству ***E***.

Пример. Рассмотрим эксперимент, в ходе которого из "идеальной" колоды извлекается игральная карта (термин "идеальная" означает, что карта извлекается случайным образом).
Перечислим некоторые примеры поля вероятностей, точек и событий.

1. $\mathbf{S_1}$: пространство состоит из 52 точек — по одной на каждую карту.

Допустим, что событие $E1$ обозначает "туз"

то есть: E_1 = {Т♠,Т♥,Т♦,Т♣}.

Оно происходит, если из колоды извлекается туз любой масти.

2. $\mathbf{S_2}$ — {красная масть, черная масть}.

Допустим, что E_2 = {красная масть}.

Это событие происходит, если из колоды извлекается карта красной масти.

3. $\mathbf{S_3}$: пространство состоит из 13 точек — 2, 3, 4,..., 10, В, Д, К, Т.

Допустим, что E_3 = {числа}.

Это событие происходит, если из колоды извлекается карта 2, или 3, ..., или 10.

Классическое определение вероятности.

Допустим, что в ходе эксперимента извлекается одна из ***n*** **= #S**
равновероятных точек и что в результате каждого эксперимента обязательно извлекается одна точка.

Обозначим через ***m*** *количество точек, принадлежащих событию* ***E****.*

Тогда вероятностью события ***E*** *называется число* ***m/n****.*

Эта вероятность обозначается следующим образом:

Prob[***E***] = m/n

В примере вероятности событий E_1, E_2, E_3 таковы:

$\text{Prob}[\boldsymbol{E_1}] = 4/52$

$\text{Prob}[\boldsymbol{E_2}] = 1/2$

$\text{Prob}[\boldsymbol{E_3}] = 9/13$

Статистическое определение вероятности.

Допустим, что при одинаковых условиях проводятся ***n*** *экспериментов, в которых* ***μ*** *раз происходит событие.*

Если при достаточно больших значениях ***n*** *величина* ***μ*** *становится и остается устойчивой, то говорят, что событие* ***E*** *происходит с вероятностью*

$$\text{Prob}[\boldsymbol{E}] \approx \mu/n$$

Свойства

1. Поле вероятностей само по себе является **достоверным событием** (sure event).

Например, S= {ОРЕЛ, РЕШКА}.

Тогда: Prob[S] = 1.

2. Обозначим через *O* событие, не содержащее ни одной точки (т.е. событие, которое никогда не происходит, например, черные бубны).

Это событие называется **невозможным** (impossible event).

Вероятность невозможного события равна нулю.

$$Prob[O] = 0.$$

3. Вероятность любого события удовлетворяет неравенству:

$$0 \leq \text{Prob}[E] \leq 1.$$

4. Если $E \subseteq F$, то говорят, что событие F содержит событие E, и если происходит событие E, то происходит и событие F.

$$\text{Prob}[E] \leq \text{Prob}[F]$$

5. Обозначим через $\bar{E} = S \setminus E$ событие, **дополнительное** (complementary) по отношению к событию E.

Тогда: $\text{Prob}[E] + \text{Prob}[\bar{E}] = 1.$

Основные вычисления

Обозначим через $E \cup F$ сумму событий E и F (происходит одно из двух событий),
а через $E \cap F$ — произведение событий E и F (происходят оба события).

Правила сложения

1. $\text{Prob}[E \cup F] = \text{Prob}[E] + \text{Prob}[F] - \text{Prob}[E \cap F]$.

2. Если $E \cap F = O$, говорят, что события E и F являются взаимно исключающими, или несовместными, и $\text{Prob}[E \cup F] = \text{Prob}[E] + \text{Prob}[F]$.

3. Если $\bigcup_{i=1}^{n} E_i = S$ и $E_i \cap F_j = O$ (i≠j), то:

$$\sum_{i=1}^{n} \text{Prob}[E_i] = 1$$

Условная вероятность

Пусть E и F — два события, причем событие E имеет ненулевую вероятность.

Вероятность события F при условии, что произошло событие E, называется условной вероятностью события F при условии события E и вычисляется по формуле

$$\text{Prob}\,[F|E] = \frac{\text{Prob}\,[E \cap F]}{\text{Prob}\,[E]}.$$

Пример. Рассмотрим семьи с двумя детьми. Обозначим буквами g (girl) и *b* (boy) пол ребенка (девочка и мальчик соответственно), причем первая буква обозначает пол старшего ребенка.

Существует четыре возможности gg, *gb*, *bg* и bb. Эти точки образуют поле вероятностей **S**.
Вероятность извлечь каждую из этих точек из поля вероятностей равна1/4.

Обозначим через *E* событие "в семье есть девочка",
а через *F* — событие "в семье есть две девочки".

Чему равна вероятность события *F* при условии, что произошло событие *E*,
то есть Prob[F|E]?

Определение
Независимые события

События E и F называются независимыми, тогда и только тогда, когда

$$\text{Prob}[F|E] = \text{Prob}[F].$$

Событие $E \cap F$ представляет собой точку gg,
поэтому Prob[$E \cap F$] = 1/4 .
Поскольку событие E состоит из точек gg, gb или bg,
Prob[E]= 3/4

Следовательно, по определению для условной вероятности Prob[$F|E$] = 1/3.
Действительно, в одной трети случаев в семьях, имеющих двух детей, одна из которых — девочка, оба ребенка являются девочками.

Определение
Независимые события
События E и F называются независимыми, тогда и только тогда, когда

$$\text{Prob}[F|E] = \text{Prob}[F].$$

Событие $E \cap F$ представляет собой точку gg,
поэтому Prob[$E \cap F$] = 1/4 .
Поскольку событие E состоит из точек gg, gb или bg,
Prob[E]= 3/4

Следовательно, по определению для условной вероятности Prob[$F|E$] = 1/3.
Действительно, в одной трети случаев в семьях, имеющих двух детей, одна из которых — девочка, оба ребенка являются девочками.

Определение
Независимые события
События E и F называются независимыми, тогда и только тогда, когда

$$\text{Prob}[F|E] = \text{Prob}[F].$$

Правила умножения

1. $\text{Prob}[E \cap F] = \text{Prob}[F|E] \times \text{Prob}[E] = \text{Prob}[E|F] \times \text{Prob}[F]$.

2. Если события E и F являются независимыми, то
$\text{Prob}[E \cap F] = \text{Prob}[E] \times \text{Prob}[F]$.

Правила умножения

1. $\text{Prob}[E \cap F] = \text{Prob}[F|E] \times \text{Prob}[E] = \text{Prob}[E|F] \times \text{Prob}[F]$.

2. Если события E и F являются независимыми, то
$\text{Prob}[E \cap F] = \text{Prob}[E] \times \text{Prob}[F]$.

Вернемся к примеру 1. Предположим, что события *E1* и *E2* являются независимыми.
Их вероятности равны 1/13 и 1/2 соответственно.
Поскольку эти события независимы, применяя второе правило умножения, получаем, что вероятность их одновременной реализации (из колоды извлекается туз красной масти) равна 1/26.

Закон полной вероятности

Если $\bigcup_{i=1}^{n} E_i = S$ и $E_i \cap E_j = O$ (i≠j), то для любого события A, которое может произойти только вместе с одним из попарно несовместных событий $E_1, E_2, \ldots, E_n$, образующих полную группу :

$$\text{Prob}[A] = \sum_{i=1}^{n} \text{Prob}[A \mid E_i]\,\text{Prob}[E_i].$$

Случайные величины и распределения вероятностей

Допустим, что дискретное пространство S содержит конечное или счетное количество изолированных точек: $x_1, x_2. \ x_{\#s}$.

Дискретная случайная величина и ее функция распределения.

1. Дискретная случайная величина является числовым результатом эксперимента.

Она представляет собой функцию, определенную на дискретном выборочном пространстве.

2. Пусть S — дискретное пространство вероятностей, а ξ— случайная величина.

Функция распределения дискретной случайной величины ξ представляет собой отображение S→R, заданное перечислением вероятностей

$$\mathrm{Prob}[\xi = x_i] = p_i (i = 1, 2, \ldots, \#\mathbb{S})$$

удовлетворяющих следующим условиям:

а) $p_i \geq 0$;

б) $\sum_{i=1}^{\#\mathbb{S}} p_i = 1.$

Равномерное распределение

Наиболее часто в криптографии применяются случайные величины, имеющие **равномерное распределение** (uniform distribution):

$$\mathrm{Prob}[\xi = x_i] = \frac{1}{\#\mathbb{S}} (i = 1, 2, \ldots, \#\mathbb{S}).$$

Равномерное распределение

Наиболее часто в криптографии применяются случайные величины, имеющие **равномерное распределение** (uniform distribution):

$$\mathrm{Prob}[\xi = x_i] = \frac{1}{\#\mathbb{S}} (i = 1, 2, \ldots, \#\mathbb{S}).$$

Пример. Пусть *S* — множество неотрицательных чисел, состоящих из не более чем ***к*** бит (бинарных цифр).
Выберем из множества *S* случайную точку x, придерживаясь равномерного распределения.
Покажем, что вероятность извлечь число, состоящее из ***к*** бит, равна1/2.

Множество $S = \{0,1,2,..., 2^k - 1\}$ можно разбить на два непересекающихся подмножества:
$S_1 = \{0,1,2,..., 2^{\kappa - 1} - 1\}$ и
$S_2 = \{2^{\kappa-1}, 2^{\kappa-1} + 1,..., 2^{\kappa} - 1\}$,
где множество S_2 состоит из всех ***k***-разрядных чисел,
$\#S_1 = \#S_2 = \#S / 2$.

Применяя второе правило сложения вероятностей, получаем следующее:

$$\text{Prob}[x \in S_2] = \text{Prob} \bigcup_{i=2^{k-1}}^{2^k-1} \{x = i\} = \sum_{i=2^{k-1}}^{2^k-1} \text{Prob}[x = i] =$$

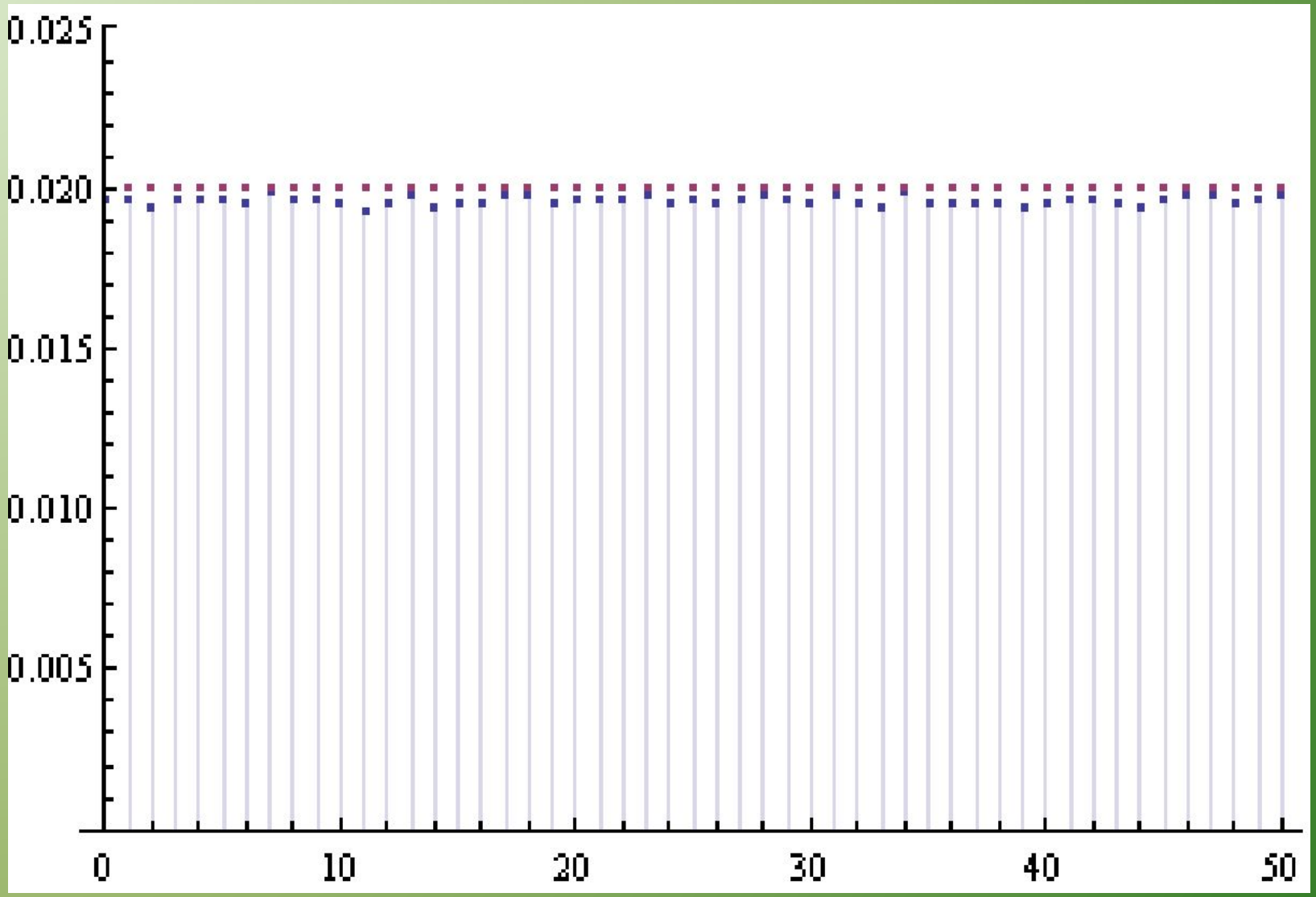
$$= \sum_{i=2^{k-1}}^{2^k-1} \frac{1}{\#S} = \frac{\#S_2}{\#S} = \frac{1}{2}.$$

Множество $S = \{0,1,2,..., 2^k - 1\}$ можно разбить на два непересекающихся подмножества:
$S_1 = \{0,1,2,..., 2^{\kappa - 1} - 1\}$ и
$S_2 = \{2^{\kappa-1}, 2^{\kappa-1} + 1,..., 2^{\kappa} - 1\}$,
где множество S_2 состоит из всех ***k***-разрядных чисел,
$\#S_1 = \#S_2 = \#S / 2$.

Применяя второе правило сложения вероятностей, получаем следующее:

$$\text{Prob}[x \in S_2] = \text{Prob} \bigcup_{i=2^{k-1}}^{2^k-1} \{x = i\} = \sum_{i=2^{k-1}}^{2^k-1} \text{Prob}[x = i] =$$

$$= \sum_{i=2^{k-1}}^{2^k-1} \frac{1}{\#S} = \frac{\#S_2}{\#S} = \frac{1}{2}.$$


0.025
0.020
0.015
0.010
0.005
0
10
20
30
40
50

Биномиальное распределение

Допустим, что эксперимент имеет только два исхода — успех или неудача: "ОРЕЛ" или "РЕШКА" при подбрасывании монеты, причем их вероятности постоянны.

Повторяющиеся независимые эксперименты, удовлетворяющие этим условиям, называются **испытаниями Бернулли** (Bernoulli trials).

Предположим, что в отдельном испытании:

$$\text{Prob}[\text{“успех”}] = p, \text{Prob}[\text{“неудача”}] = 1 - p.$$

Тогда:

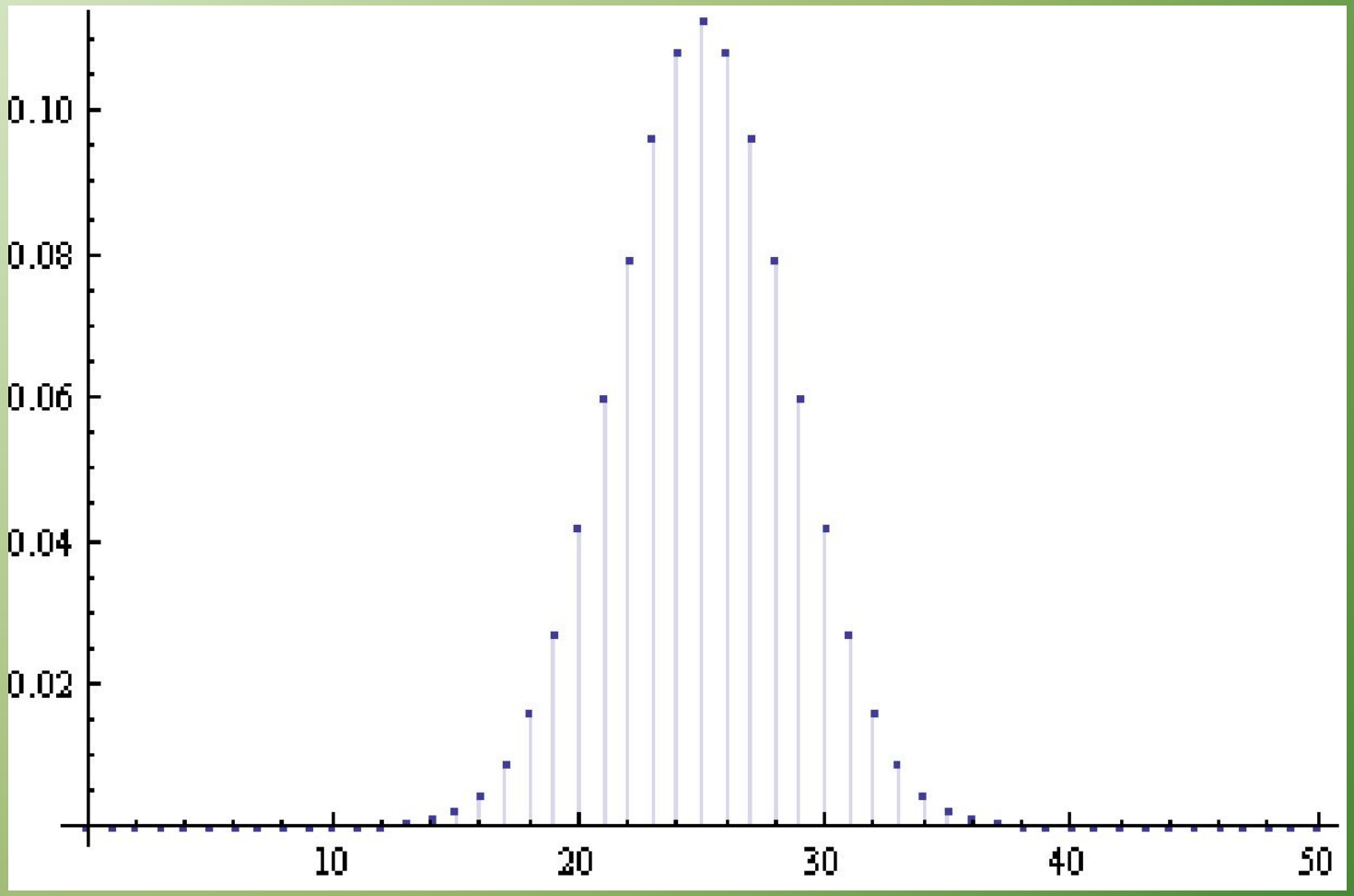
$$\text{Prob}[\text{“}k \text{ успехов в } n \text{ испытаниях”}] = \binom{n}{k} p^k (1-p)^{n-k}$$

где $\binom{n}{k}$ обозначает число сочетаний из *n* элементов по *к*.

Если случайная величина ξ_n принимает значения 0,1,..., *n,* и для величины p $\in$ (0,1) выполняется условие:

$$\text{Prob}[\xi_n = k] = \binom{n}{k} p^k (1-p)^{n-k} \; (k = 0, 1, \ldots, n),$$

то говорят, что величина ξ_n ,, имеет **биномиальное распределение** (binomial distribution).


0.10
0.08
0.06
0.04
0.02
10
20
30
40
50

Закон больших чисел

Допустим, что в схеме испытаний Бернулли вероятность успеха равна p, а случайная величина ξ_n представляет собой количество "успехов" среди n испытаний.

Тогда ξ_n/n равна среднему количеству "успехов" среди n испытаний.

В соответствии со статистическим определением вероятности величина ξ_n/n должна быть близкой к числу p.

Отсюда следует **закон больших чисел** (law of large numbers):

$$\lim_{n\to\infty} \text{Prob}\left[\left|\frac{\xi_n}{n} - p\right| < \alpha\right] = 1.$$

Парадокс дней рождений

Рассмотрим следующую задачу:
для произвольной функции $f : X \bullet Y$,
где Y — множество, состоящее из n элементов,

Найти величину $к$ для оценки вероятности ε $(0 < \varepsilon < 1)$,
такую что для $к$ попарно разных элементов $x_1, x_2, \ldots x_k \in X$
набор, состоящий из $к$ значений функции $f(x_1), f(x_2), \ldots, f(x_k)$,
удовлетворяет неравенству:

$$\text{Prob}[f(x_i) = f(x_j)] \geqslant \varepsilon \text{ для некоторых } i \neq j.$$

Иначе говоря, при $к$ вычислениях функции коллизия возникает с вероятностью не меньше ε.

Можно предположить, что вычисление функции в этой задаче порождает n разных и равновероятных точек.

Эти вычисления можно отождествить с извлечением шара из урны, содержащей n разноцветных шаров, после чего цвет записывается, и шар возвращается в урну.

Задача заключается в поиске такого числа k, при котором какой-нибудь цвет повторялся бы с вероятностью ε.

На цвет первого шара не налагаются никакие ограничения.

Пусть y_i — цвет шара, извлеченного при i-й попытке.

Цвет второго шара не должен совпадать с цветом первого шара, поэтому вероятность того, что $y_2 \neq y_1$, равна $1 — 1/n$,

вероятность того, что $y_3 \neq y_1$ и $y_3 \neq y_2$, равна $1 — 2/n$ и т.д.

При извлечении k-го шара вероятность того, что до этого не возникнет коллизия, равна:

$$\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\dots\left(1-\frac{k-1}{n}\right).$$

При достаточно большом числе n и относительно малом числе x выполняется следующее соотношение:

$$\left(1+\frac{x}{n}\right)^n \approx e^x,$$

или

$$1+\frac{x}{n} \approx e^{x/n}.$$

Итак:

$$\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\dots\left(1-\frac{k-1}{n}\right)=\prod_{i=1}^{k-1}\left(1-\frac{i}{n}\right)\approx$$

$$\approx\prod_{i=1}^{k-1}e^{-i/n}=e^{-\frac{k(k-1)}{2n}}.$$

Полученное число представляет собой вероятность извлечь *к* шаров без коллизии.
Следовательно, вероятность возникновения по крайней мере одной коллизии равна:

$$1-e^{-\frac{k(k-1)}{2n}}.$$

Приравнивая эту величину к числу ε, получаем:

$$e^{-\frac{k(k-1)}{2n}} \approx 1-\varepsilon,$$

$$k^2 - k \approx 2n \log \frac{1}{1-\varepsilon}.$$

$$k \approx \sqrt{2n \log \frac{1}{1-\varepsilon}}.$$

Итак, для случайной функции, отображающей множество *X* на множество *Y,* необходимо выполнить по крайней мере ***к*** вычислений,
чтобы обнаружить коллизию с заданной вероятностью **ε**.

Из полученного соотношения вытекает, что даже если число ε достаточно велико (т.е. очень близко к единице),
величина log (1/(1- ε)) остается весьма малой,
а значит, в принципе, число *к* прямо пропорционально •*n*.

Если ε = 1/2, то:

$$k \approx 1{,}1774\sqrt{n}.$$

Зависимость числа *к* от величины *n* означает, что для случайной функции,
пространство исходов которой состоит из *n* точек,
чтобы обнаружить коллизию со значимой вероятностью,
необходимо выполнить всего •*n* вычислений.

Этот факт оказал значительное влияние на разработку криптосистем и криптографических протоколов.

Например, если квадратный корень, извлеченный из размера фрагмента данных (скажем, криптографического ключа или сообщения),
скрытого в качестве прообраза криптографической функции (которая, как правило, является случайной),
не является достаточно большой величиной,
данные можно расшифровать с помощью случайных вычислений значений функции.

Такая атака получила название **атака по методу квадратного корня** (square-root attack),
или **атака на основе "парадокса дней рождений"** (birthday attack).

Второе название возникло из-за внешне парадоксального явления:
положив n =365 в формуле $k \approx 1{,}1774\sqrt{n}.$

мы получим $к \sim 22{,}49$.

Иначе говоря, для того, чтобы с вероятностью более 50% обнаружить двух людей,
родившихся в один и тот же день и находящихся в комнате, заполненной случайными людьми,
достаточно, чтобы в комнате было всего лишь 23 человека.

Эта величина кажется слишком маленькой по сравнению с интуитивно ожидаемой.

Применение парадокса дней рождений: алгоритм кенгуру Полларда для индексных вычислений

Пусть p — простое число. При определенных условиях ,**функция возведения в степень по модулю** (modulo exponentiation)

$$f(x) = g^x \pmod p$$

является, по существу, случайной.

Иначе говоря, для чисел $x = 1,2,\ldots ,p$— 1 значения $f(x)$ случайным образом разбросаны по всему интервалу [1, p — 1].

Эта функция широко применяется в криптографии, поскольку она является однонаправленной:
вычислить $y = f(x)$ очень легко,
однако найти обратную функцию, т.е. вычислить

$$x = f^{-1}(y),$$

чрезвычайно трудно для практически всех значений

$$y \in [1, p - 1].$$

Иногда для значения $y = f(x)$ известно, что $x \in [a, b]$ при некоторых значениях a и b.

Очевидно, что, вычисляя значения $f(a)$, $f(a + 1)$, можно найти число x не более чем за $b - a$ шагов.

Если число $b - a$ слишком велико, то такой метод перебора является непрактичным.

Однако, если число не слишком велико (например, если $b - a \approx 2^{100}$, то $\sqrt{b - a} \approx 2^{50}$, что является вполне разумной величиной),
то при обращении функции $f(x)$ за $b - a$ шагов проявляется парадокс дней рождений.

Используя этот факт, Поллард изобрел метод индексных вычислений, получивший название **λ-метод** или **метод кенгуру** (kangaroo method). Смысл этих названий станет ясен позднее.

Поллард описал свой алгоритм, используя в качестве персонажей двух кенгуру — домашнего, *T*, и дикого, *W*.

Задача вычисления индекса *x* с помощью функции

$$f(x) = g^x \pmod p$$

сводилась к ловле кенгуру *W* с помощью кенгуру *T*.

Эту задачу можно решить, позволив обоим кенгуру прыгать вокруг по определенным траекториям.

Пусть S — множество целых чисел, состоящее из J элементов $J = \log_2(b - a)$,
а значит, является достаточно малым числом:

$$\mathcal{S} = \{s(0), s(1), s(2), \ldots, s(J-1)\} = \{2^0, 2^1, \ldots, 2^{J-1}\}.$$

Каждый раз один из кенгуру прыгает на расстояние, которое случайным образом извлекается из множества *S,* причем общее расстояние, пройденное каждым из кенгуру, регистрируется с помощью счетчика.

Кенгуру Т начинает свой путь из известной точки

$$t(0) = g^b (mod p).$$

Поскольку кенгуру Г является домашним, в качестве его дома можно рассматривать точку *b*.

Его путь описывается следующей формулой:

$$t(i+1) = t(i) g^{s(t(i) (\mathrm{mod}\ J))} (\mathrm{mod}\ p) \text{ для } i = 0, 1, 2, \ldots \qquad (3.6.3)$$

Допустим, что кенгуру T делает ***n*** прыжков, а потом останавливается.

Требуется определить, сколько прыжков должен сделать кенгуру T, чтобы поймать кенгуру *W*.

После n-го прыжка счетчик пути, пройденного кенгуру T, показывает величину

$$d(n) = \sum_{i=0}^{n} s(t(i) (\mathrm{mod}\, J)).$$

Используя это значение, мы можем переписать выражение (3.6.3) для следов кенгуру T в следующем виде:

$$t(n) = g^{b+d(n-1)} (\mathrm{mod}\, p).$$

Кенгуру *W* начинает свой путь из неизвестной точки

w(0) = g^x(modp).

Этой неизвестной точкой является число *x,* и именно поэтому кенгуру *W* считается диким.

Его путь описывается следующей формулой:

$$w(j+1) = w(j) g^{s(w(j)(\mathrm{mod}\, J))} (\mathrm{mod}\, p) \text{ для } j = 0, 1, 2, \ldots \qquad (3.6.4)$$

Счетчик пути, пройденного кенгуру *W,* регистрирует величину :

$$D(j) = \sum_{k=0}^{j} s(w_k (\mathrm{mod}\, J)).$$

Аналогично выражению (3.6.3) выражение (3.6.4) для следов кенгуру *W* можно переписать в следующем виде:

$$w(j) = g^{x+D(j-1)} \bmod p$$

Очевидно, что следы обоих кенгуру, *t(i)* и *w(j),* представляют собой случайные функции.
Первая из этих функций задается в точках ***i***,
а вторая — в точках ***j***.

Согласно парадоксу дней рождений после $n \approx \sqrt{b \boxtimes a}$ прыжков, сделанных кенгуру T и W соответственно, при некоторых значениях $\xi \leq n$ и $\eta \leq n$ должна произойти коллизия

$$t(\xi) = w(\eta).$$

Именно в этот момент кенгуру T и W встретятся в одной точке, т.е. кенгуру W попадет в капкан, установленный кенгуру T.

Итак, кенгуру W пойман.

Если количество случайных прыжков, сделанных обоими кенгуру, превышает $\sqrt{b \boxtimes a}$ вероятность коллизии быстро стремится к единице.

В соответствии с формулами (3.6.3) и (3.6.4), когда возникает коллизия $t(\xi) = w(\eta)$,
выполняются равенства $t(\xi+1) = w(\eta+1)$,. $t(\xi+2) = w(\eta+2)$.. и т.д.

Иначе говоря, при некоторых числах $n \approx m$ конце концов будет выполнено равенство $w(m) = t(n)$.

Поскольку после встречи оба кенгуру продолжают прыгать по одному и тому же пути, ведущему к равенству $w(m) = t(n)$, коллизию $t(\xi) = w(\eta)$ можно интерпретировать как точку, в которой пересекаются две палочки греческой буквы λ (напомним, что кенгуру T выполняет фиксированное количество прыжков n).
По этой причине алгоритм получил название " λ -метод".

После возникновения коллизии выполняется равенство:

$$g^x = g^{b+d(n-1)-D(m-1)} (\mathrm{mod}\, p).$$

Отсюда следует, что:

$$x = b + d(n-1) - D(m-1).$$

Поскольку оба счетчика показывают величины *d(m* — 1) и *D(n — 1)* соответственно, величину *x* можно вычислить, используя длину пути, пройденного каждым кенгуру.

Следует отметить, что описанный алгоритм является **вероятностным** (probabilistic), т.е. он может не обнаружить коллизию (иначе говоря, вычислить искомую величину индекса).

Несмотря на это, благодаря высокой вероятности обнаружить коллизию вероятность отказа можно сделать достаточно малой.

Смещая стартовую точку кенгуру W на известную величину δ и повторяя алгоритм, после нескольких итераций мы обязательно обнаружим коллизию.

Условием, обеспечивающим практическую целесообразность λ -алгоритма, является относительная малость величины $\sqrt{b - a}$.

Математические модели открытого текста

Один из естественных подходов к моделированию открытых текстов связан с учетом их **частотных характеристик**, приближения для которых можно вычислить с нужной точностью, исследуя тексты достаточной длины.

Основанием для такого подхода является устойчивость частот *к* -грамм или целых словоформ реальных языков человеческого общения (то есть отдельных букв, слогов, слов и некоторых словосочетаний).

Учет частот ***к*** -грамм приводит к следующей модели открытого текста:

Пусть $\boldsymbol{P^{(к)}(A)}$ представляет собой массив, состоящий из приближений для вероятностей $\boldsymbol{p(}b_1 b_2 \ldots b_k\boldsymbol{)}$

появления ***к*** –грамм $b_1 b_2 \ldots b_k$ в открытом тексте,

A — $\{a_1,\ldots,a_m\}$ — алфавит открытого текста,

$b_i \in A$
$i = 1,\ldots k.$

Источник "открытого текста" генерирует последовательность $c_1, c_2,...,c_k,c_{k+1}$ знаков алфавита A,

в которой:

k - грамма $c_1,c_2,...,c_k$ появляется с вероятностью

$p(c_1,c_2,...,c_k) \in P^{(к)}(A)$,

следующая **k**-грамма $c_2,...,c_k,c_{k+1}$ появляется с вероятностью $p(c_2...c_{k+1}) \in P^{(к)}(A)$ и т. д.

Назовем построенную модель открытого текста

вероятностной моделью ***к*** *-го приближения.*

Таким образом, простейшая модель открытого текста - ***вероятностная модель первого приближения*** –

представляет собой последовательность знаков

$c_1, c_2, \ldots$, в которой каждый знак c_i , i = 1,2,… появляется с вероятностью

$p(c_i) \in P^{(1)}(A)$, независимо от других знаков.

Эта модель также называется *позначной моделью открытого текста.*

В такой модели открытый текст $c_1 c_2 \ldots c_l$ имеет вероятность

$$p(c_1 c_2 \ldots c_l) = \prod_{i=1}^{l} p(c_i)$$

В вероятностной модели **второго приближения** первый знак c_1 имеет вероятность:

$$p(c_1) \in P^{(1)}(A),$$

а каждый следующий знак c_i , зависит от предыдущего и появляется с вероятностью:

$$p\left(c_i / c_{i-1}\right) = \frac{p\left(c_{i-1}c_i\right)}{p\left(c_{i-1}\right)}$$

где:

$$p\left(c_{i-1}c_i\right) \in P^{(2)}\left(A\right),$$

$$p\left(c_{i-1}\right) \in P^{(1)}\left(A\right)$$

В такой модели открытый текст $c_1c_2...c_l$ имеет вероятность

$$p(c_1c_2...c_l) = p(c_1)\cdot\prod_{i=2}^{l} p(c_i / c_{i-1})$$

Таблица частот биграмм русского языка

	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
А	2	12	35	8	14	7	6	15	7	7	19	27	19	45	5	11
Б	5					9	1		6			6		2	21	
В	35	1	5	3	3	32		2	17		7	10	3	9	58	6
Г	7				3	3			5		1	5		1	50	
Д	25		3	1	1	29	1	1	13		1	5	1	13	22	3
Е	2	9	18	11	27	7	5	10	6	15	13	35	24	63	7	16
Ж	5	1			6	12			5					6		

Модели открытого текста более высоких приближений учитывают зависимость каждого знака от большего числа предыдущих знаков.

Чем выше степень приближения, тем более "читаемыми" являются соответствующие модели.

Проводились эксперименты по моделированию открытых текстов с помощью ЭВМ.

1. (Позначная модель) *алисъ проситете пригнуть стречи разве возникл;*
2. (Второе приближение) *н умере данного отствии официант простояло его то;*
3. (Третье приближение) *уэт быть как ты хоть а что я спящихся фигурой куда п;*
4. (Четвертое приближение) *ество что ты и мы сдохнуть пересовались ярким сторож;*
5. (Пятое приближение) *луну него словно него словно из ты в его не полагаете помощи я д;*
6. (Шестое приближение) *о разведения которые звенел в тонкостью огнем только.*

Как видим, тексты вполне "читаемы".