“ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ ТЕОРИИ АЛГОРИТМОВ”

**1. Представление алгоритмов на интуитивном уровне**

Варка манной каши • Взять 500 мЛ воды. • Довести до кипения. • Соль и сахар добавит по вкусу. • Всыпать, помешивая, 100 Г манной крупы. • Довести до кипения, периодически помешивая. • Варить, помешивая, 10 минут.

**2. Определения понятия “алгоритм”**

Неформальное определение алгоритма Алгоритм – набор систематических инструкций по выполнению в строго определённом порядке необходимых действий для решения всех задач какого-либо заданного класса.

Формализованное определение Алгоритм – это заданное на некотором языке конечное предписание, задающее конечную последовательность выполнимых элементарных операций для решения задачи, общее для класса возможных исходных данных.

**3. Основные разделы теории алгоритмов**

1. Общие понятия и определения теории алгоритмов

Введены понятия:

• конструктивного объекта А.Н. Колмогоровым: слова в алфавите,

матрицы, вектора, массивы, (Б, к)-деревья, комплексы (графы),

ансамбли;

• итеративного процесса;

• локальной операции.

Теория представительных вычислительных моделей

Теория исчислений

Теория представительных порождающих моделей

Теория связей между алгоритмами и счислениямиИсследование времЕнных и ёмкостных показателей сложности

порождения и вычисления

Исследование вычислимых функций и породимых множествИсследование -рекурсивных функций

Исследование возможностей построения неразрешимого породимого

множества Изучение проблемы сводимости Поста

Исследование возможностей арифметического и диофантова

представления любого перечислимого множества

Изучение относительных алгоритмов

Исследование вычислимых операций

Изучение программ как объекта вычисления и порождения

Теория нумерации

Теория инвариантности (машинной независимости) сложных

Вычислений

Теория сложности и энтропии конструктивных объектов

Теория вычислительных моделей

**4. Математические приложения теории алгоритмов**

Исследование массовых проблем

Алгоритмическая проблема – построение алгоритма с заданными

свойствами.

Единичная проблема – состоит в требовании предъявить объект,

который удовлетворяет определённым условиям, называемый

решением проблемы.

Массовая проблема – состоит в серии (часто бесконечной) единичных

проблем и требовании решить все эти проблемы.

Конструктивная семантика

Лежит в основании теории компиляции программ

Анализ формализованных языков логики и арифметики

Лежит в основании предикатных языков

Вычислительный анализ

Нумерованные структуры

Построение псевдослучайных последовательностей

Алгоритмический подход к понятию количества информации

Оценка сложности решения отдельных задач

Влияние на алгоритмическую практику**5. Современные направления теории алгоритмов**

1. Классическая теория алгоритмов:

• формулировка задач в терминах формальных языков;

• понятие задачи разрешения, введение сложностных классов;

• исследование класса NP-полных задач.

2. Теория асимптотического анализа алгоритмов:

• критерии оценки алгоритмов;

• методы получения асимптотических оценок;

• асимптотический анализ трудоемкости или времени выполнения.

3. Теория практического анализа вычислительных алгоритмов

• получение явных функции трудоёмкости;

• интервальный анализ функций;

• построение практических критериев качества алгоритмов;

• методики выбора рациональных алгоритмов.

**6. Цели и задачи теории алгоритмов**

Дальнейшая формализация понятия “алгоритм” и исследование

формальных алгоритмических систем;

• формальное доказательство алгоритмической неразрешимости ряда

задач;

• классификация задач, определение и исследование сложностных

классов;

• асимптотический анализ сложности алгоритмов;

• исследование и анализ рекурсивных алгоритмов;

• получение явных функций трудоёмкости в целях сравнительного

анализа алгоритмов;

• разработка критериев сравнительной оценки качества алгоритмов.

**7. Аспекты применения теории алгоритмов**

**Теоретический аспект** теории алгоритмов позволяет

• определить алгоритмическую разрешимость задачи;

• определить возможность сведения алгоритмически неразрешимых

задач к задаче останова машины Тьюринга;

• определить для алгоритмической разрешимой задачи – факт

принадлежности этой задачи к классу NP–полных задач;

• оценить для члена класса NP–полных задач временных затрат для

получения точного решения для больших размерностей исходных

данных.

**Практический аспект,** основывающийся на разделах асимптотического

и практического анализа позволяют осуществить:

• рациональный выбор из известного множества алгоритмов решения

данной задачи с учетом особенностей их применения

• получение временнЫх оценок решения сложных задач;

• получение достоверных оценок невозможности решения некоторой

задачи за определенное время

• разработку и совершенствование эффективных алгоритмов решения

задач в области обработки информации на основе практического

анализа.

**ОСНОВЫ АНАЛИЗА АЛГОРИТМОВ**

**1. Принципы анализа алгоритмов**

1. Инвариантность проводимого анализа

Независимость от реализации на конкретной ЭВМ

2. Анализу подвергаются алгоритмы, решающие одну и ту же проблему

3. Алгоритмы анализируются на классах данных

Пример – алгоритм выбора максимума из N чисел

4.Учёт ресурса типа “память”

5. Поиск наилучшего, наихудшего и среднего случаев

Наилучший случай – ситуация, для которой обработка некоторого класса

данных даёт самый лучший показатель в рамках выбранной оценки.

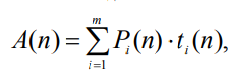
Обычно тривиален.

Наихудший случай – ситуация, для которой обработка некоторого класса

данных даёт самый плохой показатель в рамках выбранной оценки.

Позволяет определить максимальное время работы алгоритма.

Средний случай – результат статистической обработки показателей,

полученных для разных классов данных

**2. Допущения, принятые при проведении анализа**

1. В качестве вычислительного устройства, реализующего алгоритм,

принимается абстрактная машина с процессором фон-Неймановской

архитектуры

2. Каждая команда выполняется не более соответствующего

фиксированного времени

3. Исходные данные представляются в виде машинных слов по 

(байт/бит) каждое.

Конкретный входной набор задаётся N словами памяти, отсюда объём

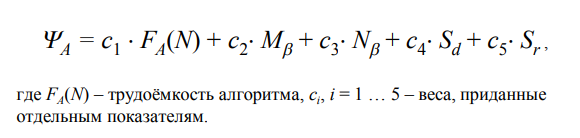
на входе алгоритма N = N  .

Часто используется термин длина входа алгоритма N, который отражает линейную размерность входных данных

4. Программный код представляется последовательностью машинных слов

5. Факультативно (опционально) учитываются дополнительные ресурсы

6. Комплексный показатель оценки алгоритма

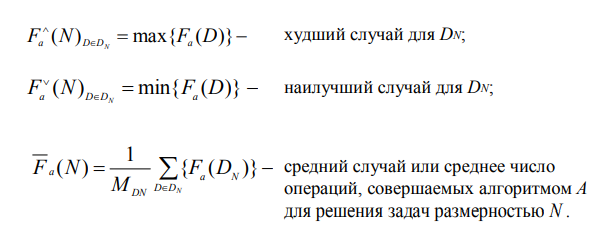
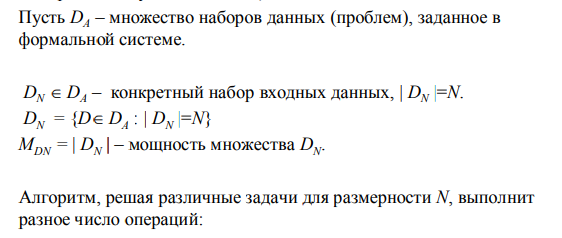


Трудоёмкость алгоритма Fа (N) – количество “элементарных” операций, совершаемых алгоритмом для решения конкретной проблемы в данной формальной системе.

**3. Формальная классификация входных данных**

Количество операций, выполняемых над входом одинаковой длины N,

зависит от конкретного набора данных



**4. Функции трудоёмкости алгоритмов**

Функции трудоёмкости есть результаты аппроксимации зависимостей трудоёмкости Fa (N) от длины входа алгоритма. Классификация функций трудоёмкости организована по виду входа.

1.Количественно-зависимые по трудоёмкости алгоритмы (класс N)

Для данной категории алгоритмов функция трудоёмкости зависит

только от объёма входных данных, а не от их конкретных

значений.



2.Параметрически-зависимые по трудоёмкости алгоритмы (класс PR)

Размерность, как правило, фиксирована, а трудоёмкость определяется

конкретными значениями компонентов вектора данных. Это, например

алгоритмы расчётов математических функций на базе степенных рядов с

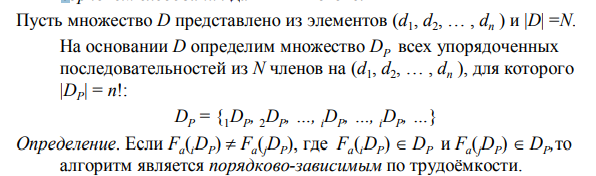
заданной точностью



3. Количественно-параметрические по трудоёмкости алгоритмы (класс NPR: NPRL – Low, NPRE – Equivalent, NPRH – High) Алгоритмы данного класса имеют функцию трудоёмкости, которая одновременно зависит как от количества (объёма) данных, так и от конкретных значений элементов данных.



4. Порядково-зависимые по трудоёмкости алгоритмы (класс NPRS) Для данных алгоритмов функция трудоёмкости определяется порядком следования данных в потоке.



**ВРЕМЕННЫЕ ОЦЕНКИ ТРУДОЁМКОСТИ АЛГОРИТМОВ**

**1. Постановка задачи временной оценки трудоёмкости алгоритма**

Временная оценка, фактически, зависит от:

• состава команд процессора, необходимых для отображения отдельных

операций алгоритма;

• количестве этих команд;

• времени выполнения отдельных типов команд.

Временная оценка определяется рядом причин.

1. Особенности формальной системы записи алгоритма на языке

программирования (например, MathLab, C++, Pascal et c).

2.Особенности трансляции исходной программы в машинный код.

3. Различие во времени выполнения разнотипных команд процессора.

4. Различие во времени выполнения однотипных команд для разных форматов

(типов) данных

5. Различия, обусловленные типами адресации.

6. Наличие у процессора особенностей архитектуры, способствующих

ускорению вычислений: конвейер, кеширование памяти, сопроцессоры и т.д.

МЕТОДЫ ПОСТРОЕНИЯ ОЦЕНОК

• Пооперационный анализ.

• Метод Гиббсона.

• Метод прямого определения среднего времени

**2. Пооперационный анализ**

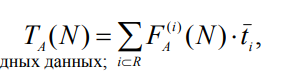
Пооперационный анализ включает

1. Построение функции трудоёмкости по каждой элементарной операции для

заданного формата данных.

2. Экспериментальное либо потактовое определение времени выполнения

соответствующей элементарной операции.

Оценка рассчитывается по формуле:

**3. Понятие “элементарной” операции**

“Элементарная” операция соотносится с укрупнённой командой языка

высокого уровня.

К “элементарным” операциям относят:

• операция присваивания: y := x;

• операция одномерной индексации: v[i]

• арифметические операции: +, –,  или (\*), /;

• операции сравнения: <, >, =,  и ;

• логические операции: or, not, and, xor и др.

**4. Алгоритмические конструкции**

**Конструкция “последовательность” (“следование”)**

2. Конструкция “ветвление” (условный оператор) if y then f t else fe

3. Конструкция “цикл” с заданным числом повторений for i := 1 to N /\* Body \*/ end for Число элементарных операций заголовка цикла: 1. инициализация управляющей переменной; 2. проверка условия окончания цикла; 3. инкремент управляющей переменной цикла; 4. присвоение нового значения управляющей переменной в теле цикла. 1 + 3⋅N

**5. Метод Гиббсона**

Включает этапы

1. Классификация отдельных задач по типам

• задачи научно-технического и расчётного

характера с преобладанием вещественных типов

данных;

• задачи целочисленной арифметики;

• задачи обработки текстовой информации.

2. Экспериментальное исследование реальных

программ с учётом типов задач на частоту

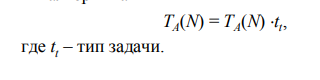
встречи тех или иных операций.

3. Определяется время типовой задачи на тестовых

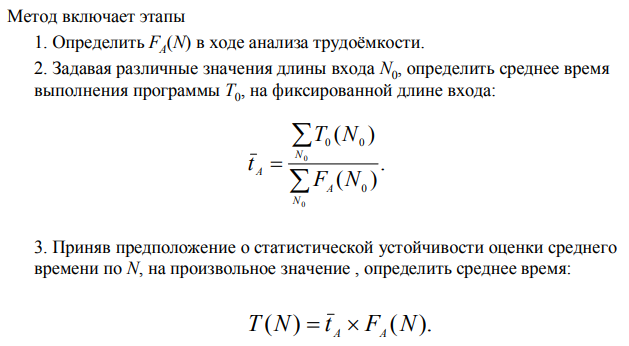
прогонах

4. Рассчитывается оценка времени работы

алгоритма

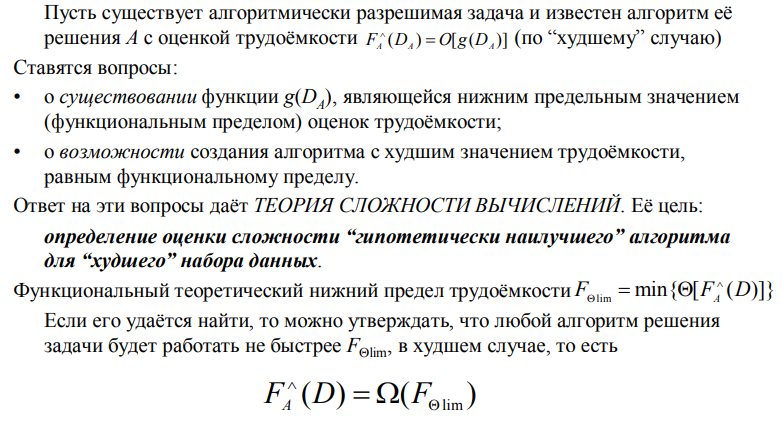
****

**6. Метод прямого определения среднего времени**

****

**СЛОЖНОСТНЫЕ КЛАССЫ ЗАДАЧ**

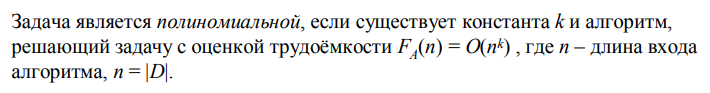
1. **Постановка задачи классификации алгоритмов**



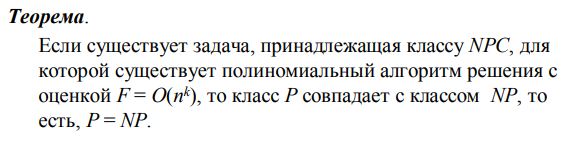
**2. Классификация сложности**

Alan Cobham & Jack Edmonds – авторы классификации задач по сложности.

1. Класс Р – задачи с полиномиальной сложностью

****

1. Класс NP – полиномиально проверяемые задачи Содержательно задача относится к классу NP, если её решение может быть полиномиально проверено.
2. Класс NP-полных задач (NPC: NP-complete) Понятие NPC было введено С. Куком (Stephen Cook) и исходит из идеи сводимости одной задачи к другой.

****

**АЛГОРИТМЫ И ИСЧИСЛЕНИЯ**

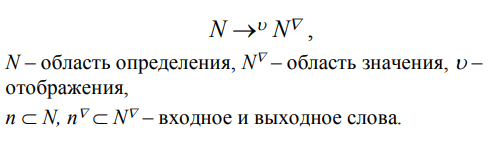
**1. Алфавитный оператор**

Абстрактный алфавит – любая конечная совокупность (≡множество) объектов, называемых буквами (≡ литерами, символами).

Слово (≡строка) в (над) алфавите (ом) W – любая последовательность символов. Примеры: 1101, 000, 1000, 10111, … abc, ccbba, cba˽aac, …

Существует пустое слово, не содержащее ни одной литеры. Число символов в слове называется его длиной.

Алфавитный оператор (≡ алфавитное отображение) – всякое соответствие, сопоставляющее словам какого-либо алфавита слова в том же либо другом алфавите. Эти алфавиты называются, соответственно, входными и выходными алфавитами. Существуют области определения и области значения оператора. Алфавитный оператор, не сопоставляющий некому входному слову ϕ ни какого, даже пустого слова, называется не определённым на слове ϕ.

****

**2. Кодирующий оператор**

Кодирующий оператор (≡кодирующее отображение) – соответствие, сопоставляющее любому символу входного алфавита некоторую конечную последовательность символов выходного алфавита.

Процесс, обратный кодированию, называется декодированием

Алгоритмы равны, когда

• равны алфавитные операторы, соответствующие им;

• совпадает система правил, задающая их действие на выходные слова.

Алгоритмы эквивалентны, когда

• алфавитные операторы совпадают;

• система правил, задающая их действие на выходные слова, не

совпадает.

Алгоритмы и соответствующие им алфавитные операторы, которые

любому входному слову соотносят только одно выходное слово,

называются детерминированными.

Алгоритмическая система – всякий общий способ задания

алгоритма. Приоритетными будут системы, характеризующиеся

свойством универсальности, которые способны задать алгоритм,

эквивалентный любому, наперед заданному.

**3. Рекурсия в вычислениях**

Рекурсия [прогр.] – способ организации вычислений при котором функция вызывает сама себя с другими аргументами

Индукция – есть разновидность рекурсии в методологии поведения доказательств

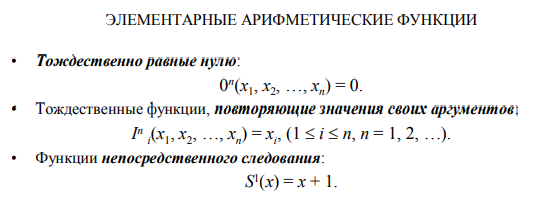
Рекурсия – способ задания функции, при котором значения определяемой функции для произвольных значений аргументов выражается известным образом через значения определяемой функции для меньших значений аргументов.

Если удаётся показать, что функция, решающая некоторую задачу, не может быть рекурсивной, то задача не разрешима.

Применение рекурсивных функций в теории алгоритмов основано на предложении перенумеровать слова в произвольном алфавите натуральными числами в порядке возрастания. В качестве способа такой нумерации, слова располагаются в порядке возрастания их длин, а слова одинаковой длины упорядочиваются в алфавитном порядке.

**4. Арифметические функции**

Если пронумеровать входные и выходные данные произвольного алфавитного оператора, последний трансформируется в функцию y = f(х), (1) где аргумент х и функция y – целые неотрицательные значения. Такие функции называются арифметическими



**5. Конструктивные приёмы**

1. Операция суперпозиции функций Сущность операции: подстановка одних арифметических функций вместо аргументов других.

2. Операция примитивной рекурсии Используется для конструирования п + 1 – местной арифметической функции по двум заданным: п-местной и (п + 2)-местной.

Примитивно-рекурсивными называются функции, которые могут быть построены из элементарных арифметических операций с помощью операций суперпозиции и примитивной рекурсии, примененных произвольное число раз и в произвольной последовательности

3. Операция наименьшего корня или операция минимизации

**6. Частично-рекурсивные функции**

Арифметические функции, которые могут быть построены с помощью операций • суперпозиции, • примитивной рекурсии, • минимизации называются частично-рекурсивными. Если частично-рекурсивная функция всюду определена, то она называется общерекурсивной. Общерекурсивная функция свободна от ограничений на • состав, • порядок и • количество применения операций.

Частично-рекурсивная функция обладает свойствами:

1. каждая стандартно заданная частично-рекурсивная функция вычислима

путём определенной процедуры механического характера;

2. какие бы классы алгоритмов не строились доныне, во всех случаях

числовые функции, вычисляемые с их помощью, являются частично

рекурсивными;

3. класс всех частично-рекурсивных функций совпадает с классом

вычисляемых на ЭВМ частичных числовых функций.

Последнее свойство называется гипотезой(тезисом) Чёрча (Church

Alonzo). Тезис не доказан и не опровергнут.

Понятием частично-рекурсивной функции пользуются для доказательства

алгоритмической разрешимости или неразрешимости какой-либо задачи.

**ВЫЧИСЛИТЕЛЬНЫЕ МОДЕЛИ**

**1. Финитный 1-процесс**

Набор инструкций применим к данной общей проблеме, если его

применение к частной проблеме не требует операции (a), когда

ячейка помечена или операции (b), если она не помечена.

• Ход выполнения инструкций трактуется как детерминированный

процесс в ходе применения к каждой конкретной проблеме, который

окончится, когда дойдёт до (с).

Набор инструкций задаёт финитный 1-процесс в связи с данной

проблемой, если он применим к этой проблеме и если определяемый

набором процесс заканчивается для каждой конкретной

проблемы: финитный 1-процесс, связанный с решением некоторой

проблемы, будем называть 1-решением этой проблемы, если

даваемый им для каждой конкретной проблемы ответ является

правильным.Пусть между классом целых чисел и классом конкретных проблем

установлено взаимно-однозначное соответствие.

Общую проблему будем называть 1-заданной, если есть

финитный 1-процесс, который в результате применения к

положительным числам, выдаёт номера проблем, образующих общую

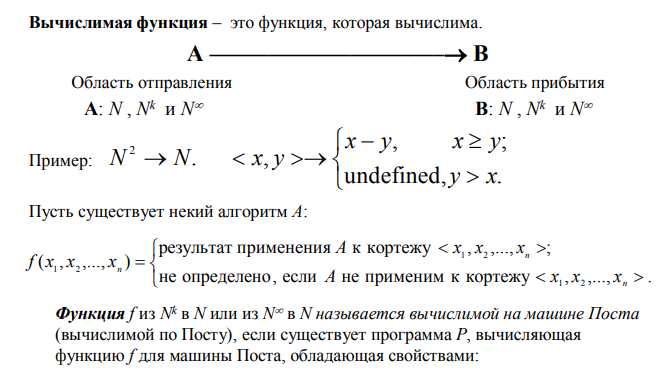
проблему.

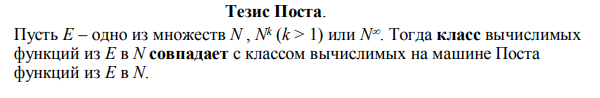
**2. Формулировка гипотезы Поста**

Пусть имеем два алгоритма с совпадающими совокупностями исходных данных. Алгоритмы 1 и 2 называются равносильными, если к любому данному из общей их совокупности они либо оба применимы, либо оба не применимы, а если применимы, то дают одинаковый результат. Гипотеза

Каждый алгоритм, все результаты которого суть числа, а областью исходных данных служат N , Nk или N∞ равносильны алгоритму с такой же совокупностью возможных исходных данных, задаваемой некоторой программой машины Поста.

**3. Вычислимые функции и тезис Поста**

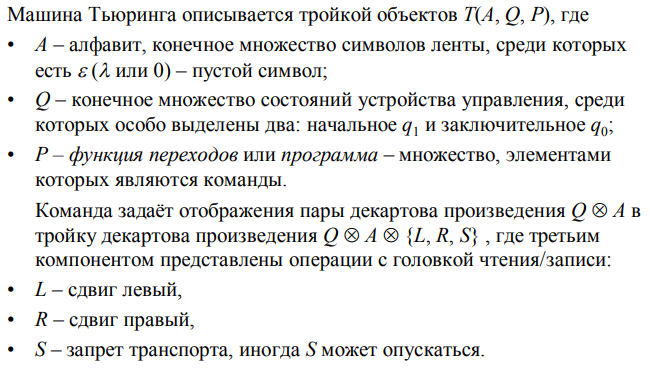
****

****

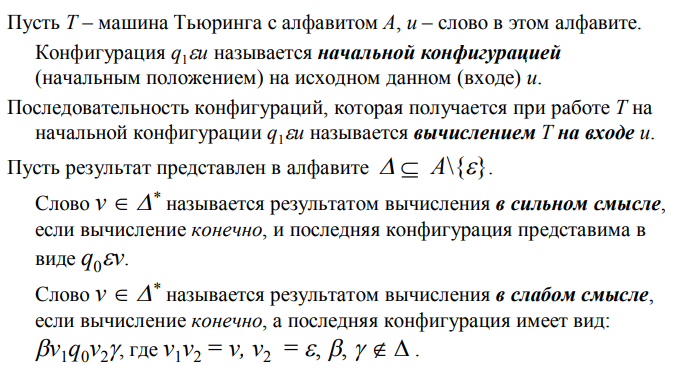
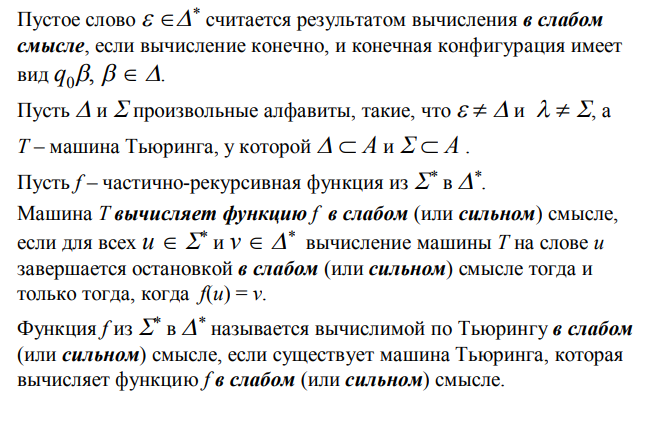
**4. Постулат Поста**

Задача на составление программы, приводящей от исходного данного к результирующему числу (и не приводящей ни к какому результату, если результирующего числа не существует), тогда и только тогда имеет решение, когда имеется какой-нибудь общий способ, позволяющий по произвольному исходному данному выписать результирующее число и не выдающий никакого – заведомо ложного – в этом случае результата, коль скоро результирующего числа не существует.

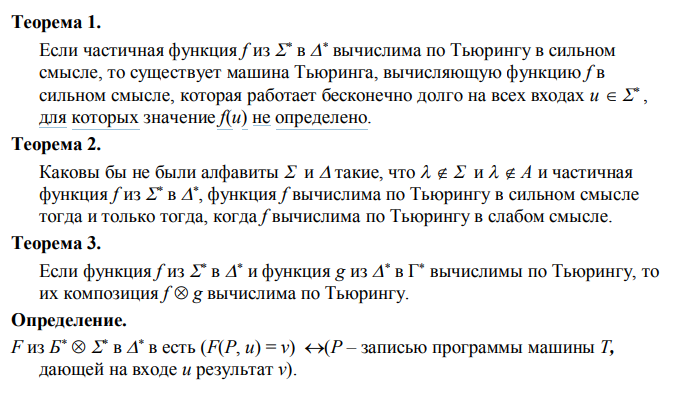
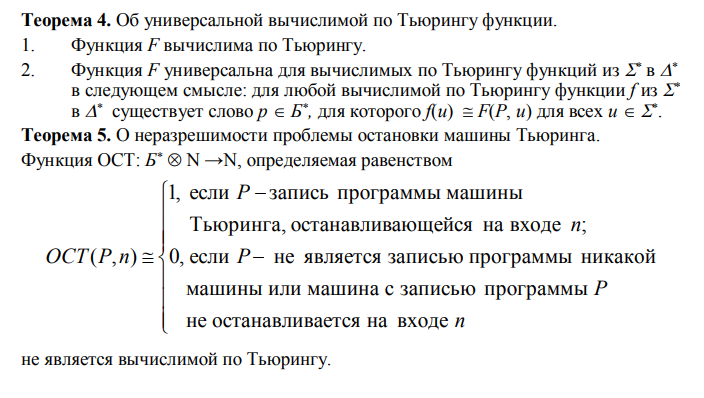
**5. Формальное описание машины Тьюринга**



**6. Вычислимость в сильном и слабом смыслах**

****

**7. Теоремы, связанные с вычислительной моделью Тьюринга**

****

**8. Тезис Тьюринга**

ТЕЗИС Для нахождения значения функции, заданной в некотором алфавите, тогда и только тогда существует какой-нибудь алгоритм, когда функция является вычислимой по Тьюрингу, т.е. когда она может быть вычислена на подходящей машине Тьюринга.

**9. Тезис Чёрча**

Формулировка 1. Числовая функция тогда и только тогда алгоритмически (машинно) вычислима, когда она частично рекурсивна.

Пост и Тьюринг одновременно с Чёрчем и не зависимо друг от друга утверждают, что класс всюду определённых функций, вычислимых в определённой модели, совпадает с классом всюду определённых вычислимых функций (для фиксированных ансамблей). С. Клини уточнил, что тезис Чёрча, в узком смысле, утверждает, что всякая вычислимая функция с натуральными аргументами и значениями частично рекурсивна.

По этой причине тезис называют тезисом Чёрча – Тьюринга – Поста – Клини.

Формулировка 2. Для любой функции f ∈ Computable(Σ \* , Σ \* ) существует алфавит Г и машина Тьюринга с алфавитом Г ⊇ Σ \* , которая вычисляет f.

Формулировка 3. Класс алгоритмически (машинно) вычислимых частичных числовых функций совпадает с классом всех частично рекурсивных функций.

**10. Неразрешимые алгоритмические проблемы**

Теорема. Не существует алгоритма (машины Тьюринга), позволяющего по описанию произвольного алгоритма и его исходных данных (и алгоритм и данные заданы символами на ленте машины Тьюринга) определить, останавливается ли этот алгоритм на этих данных или работает бесконечно.

I. Отсутствие общего метода решения задачи

1. Распределение девяток в записи числа π.

2. Вычисление совершенных чисел ( это числа, которые равны сумме своих делителей, например: 28 = 1+2+4+7+14).

3. Десятая проблема Гильберта Имеется многочлен п-ой степени с целыми коэффициентами. Необходимо выяснить, существует ли у данного уравнения целочисленное решение? Ю.В. Матиясевич в 1970 г. Доказал, что алгоритм решения проблемы не существует.

4. При этих же условиях выяснить, существует ли решение в рациональных числах? Факт существования не доказан и не опровергнут.

II. Информационная неопределённость задачи

1. Позиционирование машины Поста на последний помеченный ящик

III. Логическая неразрешимость в смысле теоремы Гёделя о неполноте Примеры

1. Проблема останова (теорема выше).

2. Проблема эквивалентности алгоритмов

3. Проблема тотальности

Теорема Гёделя

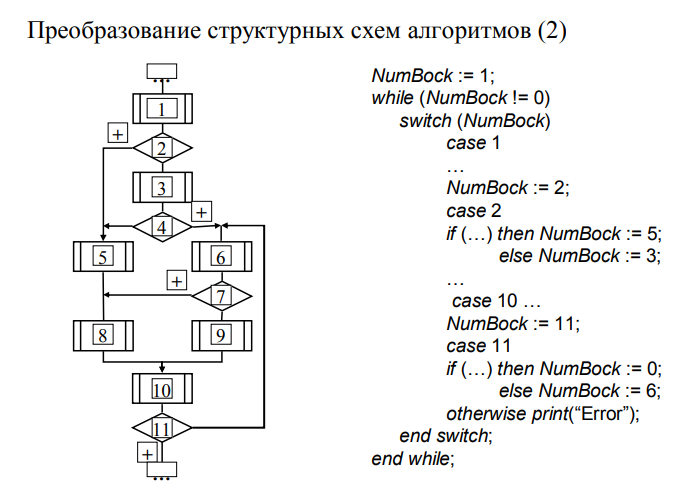
Если формальная система S непротиворечива, то формула A невыводима в S; если система S ω-непротиворечива, то формула ¬A невыводима в S. Таким образом, если система S ω-непротиворечива, то она неполна и A служит примером неразрешимой формулы.

**11. Понятие частичного алгоритма**

Решающая последовательность – конечная последовательность пар, (не обязательно из различных xi и yi ), такая, что цепочка, составленная из левых компонентов пар, совпадает с последовательностью, составленной из правых подцепочек.

Для решения задачи можно сконструировать частичный алгоритм, строящий всевозможные упорядоченные возможные последовательности, проверяющий для каждой генерации условия решения. Частичный алгоритм возможно, но не обязательно (!), находит решение проблемы. Если существует решающая последовательность, то решение может быть получено за конечное число шагов. Поскольку имеется частичный алгоритм, то задача называется частично-разрешимой. Так как общий метод определения отсутствия решающей последовательности не может быть указан, следовательно, задача сведена к проблеме останова, следовательно, алгоритмически не разрешима. Алгоритм нахождения эквивалентности и тотальности не имеют даже частичных алгоритмов

**12. Алгоритм преобразования структурных схем алгоритмов Ашкрофта-Манны**

****