**1.Понятие алгоритма**

Аль-Хорезми (арабский Математик). Алгоритм - конечная совокупность точно заданных правил решения произвольного класса задач или набор инструкций, описывающих порядок действий исполнителя для решения некоторой задачи

# 2. Три признанные формальные модели алгоритма

Машина Тьюринга, Нормальный алгоритм Маркова, Рекурсивная функция

# 3. Определение типовой машины Тьюринга

Абстрактная вычислительная машина, созданная для формализации понятия алгоритма, состоящая из пишущей головки бесконечной ленты, Управляющего Устройства

# 4. Представление алгоритма в виде команд машины Тьюринга

В МТ фиксируются состояния и действия, относящиеся к состоянию в зависимости от прочитанного символа:

Множество состояний Q = {Q1, Q2, Q3, …}

Множество действий D = {R, L, U}

Qi Si -> Qk Si dm

Qi – текущее состояние

Sj – прочитанный символ

Qk – новое состояние, в которое переходит машина Тьюринга

Si – записываемый в текущую ячейку ленты символ

dm – один из символов множество D = {R, L, U}, задающий направления движения к очередной ячейке ленты.

#### **5. Понятие конфигурации машины Тьюринга**

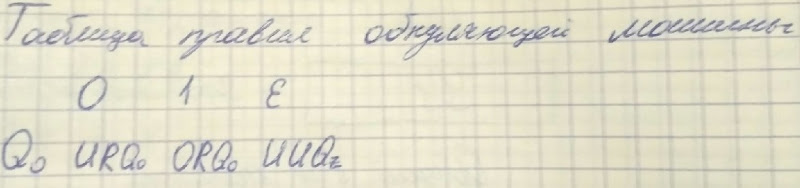
формула : альфа\*Qi\*бета

Конфигурация машины Тьюринга -представляет собой совокупность внутреннего состояния, состояния ленты, положения автомата на ленте. Конфигурацию машины Тьюринга будем записывать в виде https://lh3.googleusercontent.com/7Za6J6EGnO48LBPu3tV63n5TUl9wRobqyFhz5eyohdacx8r6oI4p6QIRIyCE7DWdZtSRNUG5E2udYBzwxFU1kJ3wC3NusV1kvL7KBcN4xzfpfrE5rndHinwcXUOArMndShxmq4Jv=s800, гдеhttps://lh3.googleusercontent.com/mClZSgoYPUev_2m80zMNT9KPajc530ffi-h0epissdmWIdDYA_Tf4IZTSOfiB83Rb4kvey0CPWycL3s2Nlf0NVp-bbOuvQmpJdfWdzLM2PzNkCaJ6m6Nf2w9eoV9JfLefJwLU6BC=s800  – текущее состояние, https://lh5.googleusercontent.com/jzeJCWPInRxmtnV4TOn-a4ol6l0pPLVrGtopXXiPcd_fs2FYW4CGrO5Jfsthj6FATXMsF8WJZxPCW7m3DcUT1FCj0heQ87Eh6HmsaHUAAfDdGyV89-2FpV425lXritBoUXoCjBCt=s800 – слово, находящееся слева от символа, https://lh3.googleusercontent.com/VK1t40-pOgIexQZh8jSHX5KiNeI8y5duF7q5cpu9ffbohTM2QWaSMhQD9ZgMkb2kPQyr9BUCA4vlklAyG0Ef9WNn980J2rSusuNidYm0JTtaemhOvhkXeiyn3GnKuswoRRkwlT4h=s800 – слово, находящееся справа от символа.

# 6. Понятие стандартной конфигурации машины Тьюринга

Конфигурация машины Тьюринга представляет собой совокупность внутреннего состояния, состояния ленты, положения автомата на ленте. Головка указывает на первый символ слова

# 7. Пример табличной записи команд обнуляющей машины Тьюринга



# 8. Определение функции через два множества точек

Говорят что на множестве Х имеется функция f из множества Y если каждому элементу x из множества X поставлен в соответствие по правилу f  некоторый элемент y из множества Y

# 9. Понятия: «вычислимая функция», «частично вычислимая функция» и «невычислимая функция»

Вычислима, если ее значение можно определить с помощью какого-либо алгоритма в какую точку отображена любая точка из множества Х(ее значение). Частично вычислима, если есть точки, которые нельзя отобразить. Не вычислима, если есть хотя бы одна точка x для которой есть отображение в y, но это отображение нельзя найти, нельзя построить МТ.

Функция f с натуральными аргументами и значениями называется вычислимой, если существует алгоритм, её вычисляющий, то есть такой алгоритм A, что

• если f(n) определено для некоторого натурального n, то алгоритм A останавливается на входе n и печатает f(n);

• если f(n) не определено, то алгоритм A не останавливается на входе n.

Несколько замечаний по поводу этого определения:

1. Понятие вычислимости определяется здесь для частичных функций (областью определения которых является некоторое подмножество натурального ряда). Например, нигде не определённая функция вычислима (в качестве A надо взять программу, которая всегда зацикливается).

2. Можно было бы изменить определение, сказав так: «если f(n) не определено, то либо алгоритм A не останавливается, либо останавливается, но ничего не печатает». На самом деле от этого ничего бы не изменилось (вместо того, чтобы останавливаться, ничего не напечатав, алгоритм может зацикливаться).

3. Входами и выходами алгоритмов могут быть не только натуральные числа, но и двоичные строки (слова в алфавите {0, 1}), пары натуральных чисел, конечные последовательности слов и вообще любые, как говорят, «конструктивные объекты». Поэтому аналогичным образом можно определить понятие, скажем, вычислимой функции с двумя натуральными аргументами, значениями которой являются рациональные числа.

*n*-аргументная функция *ϕ*частично вычислима по Маркову тогда и только тогда, когда существует нормальный алгоритм, позволяющий вычислить значение *ϕ*(*k1,k2,...,kn*) для любых совокупностей значений *х1=k1, х2=k2,..., xn = kn*, при которых *ϕ*(*k1,k2,...,kn*) существует.

Проблему остановки[3] можно сформулировать как: не существует общего алгоритма, который бы мог определить, остановится ли программа, по ее описанию и входным данным.  
Т.е. функция определения остановки невычислима

**10 Понятие функции, вычислимой по Тьюрингу**

***Определение 1.***Ф-я наз-ся *вычислимой по Тьюрингу,*если существует машина Тью­ринга, вычисляющая ее, т.е. такая машина Тьюринга, которая вычисляет ее значения для тех наборов значений аргументов, для которых функция определена, и работающая вечно, если функ­ция для данного набора значений аргументов не определена.

Остается договориться о некоторых условностях для того, что­бы это определение стало до конца точным. Во-первых, напом­ним, что речь идет о функциях (или возможно о частичных фун­кциях, т. е. не всюду определенных), заданных на множестве нату­ральных чисел и принимающих также натуральные значения. Во-вторых, нужно условиться, как записывать на ленте машины Тью­ринга значения *х1,, х2, ..., хп*аргументов функции *f(x1, x2,*..., *хп),*из какого положения начинать переработку исходного слова и, на­конец, в каком положении получать значение функции

# 11. Тезис Тьюринга

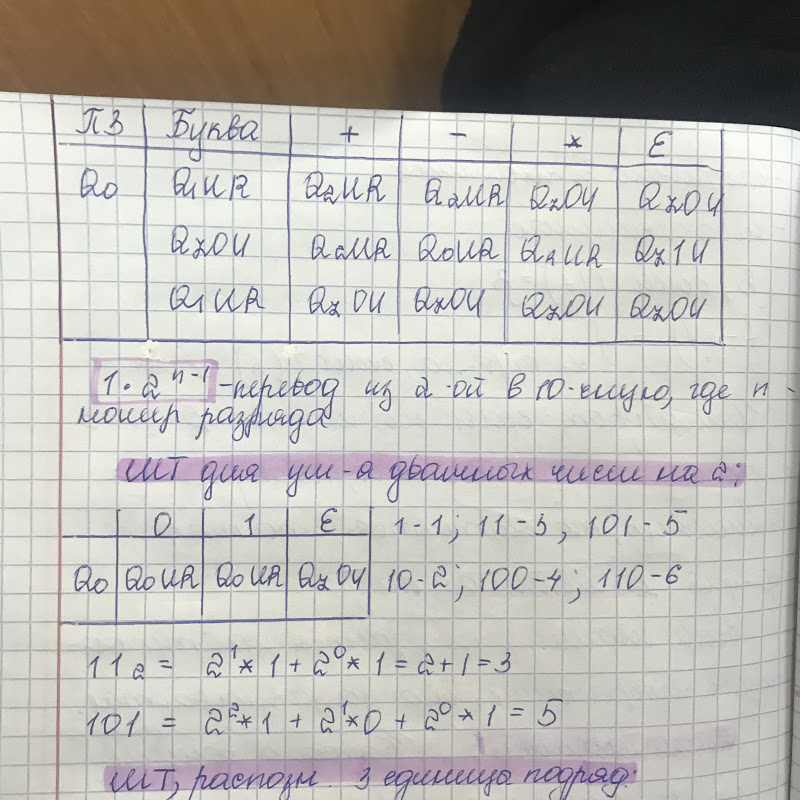
Каждая вычисляемая функция вычисляется с помощью некоторой МТ и каждая МТ вычисляет некоторую функцию.

# 12. Типы машин Тьюринга

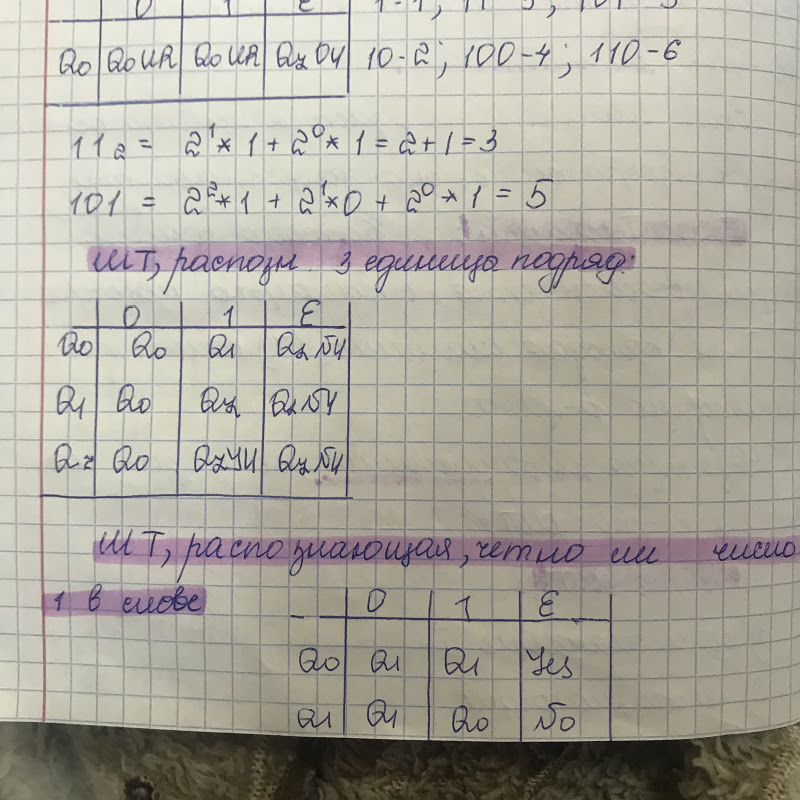
1) вычисляющие (действия над числами)

2) распознающие (отвечает на вопрос относ. задаваемого слова (да/нет))

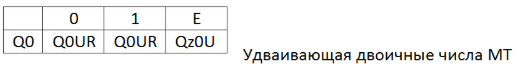
Распознающая:



Вычисляющая:



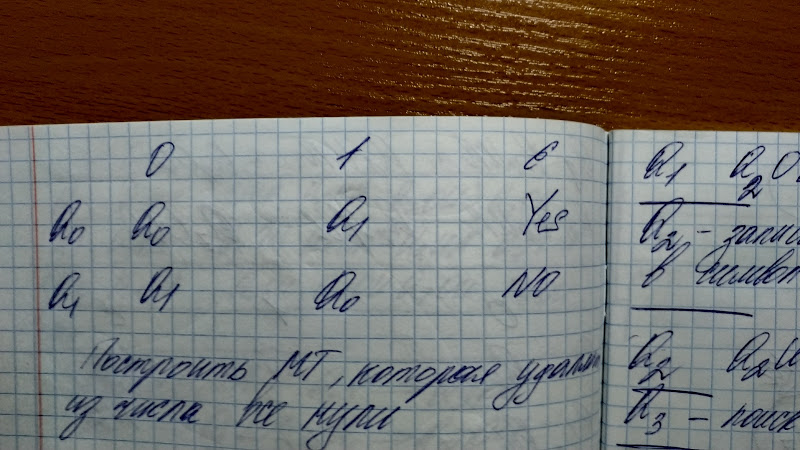
# 13. Пример вычисляющей машины Тьюринга для удвоения чисел



# 14. Пример распознающей машины Тьюринга  https://lh6.googleusercontent.com/H9-OeynXFIhmgLeII1h2_jAj_Iztm4xtIN10Fjy75kOBQbGcwbG4O9lqVpMtIJnFoFZFp4HeSUiwQPErmF8gTuDEpw5yiwAZlYle5OXzCtpMNxC52aqZ9KEZ4FzOhlATDj99soMF=s800

# 15. Особенности редуцированной таблицы правил распознающей машины Тьюринга

Отсутствует операция “записать что-либо”



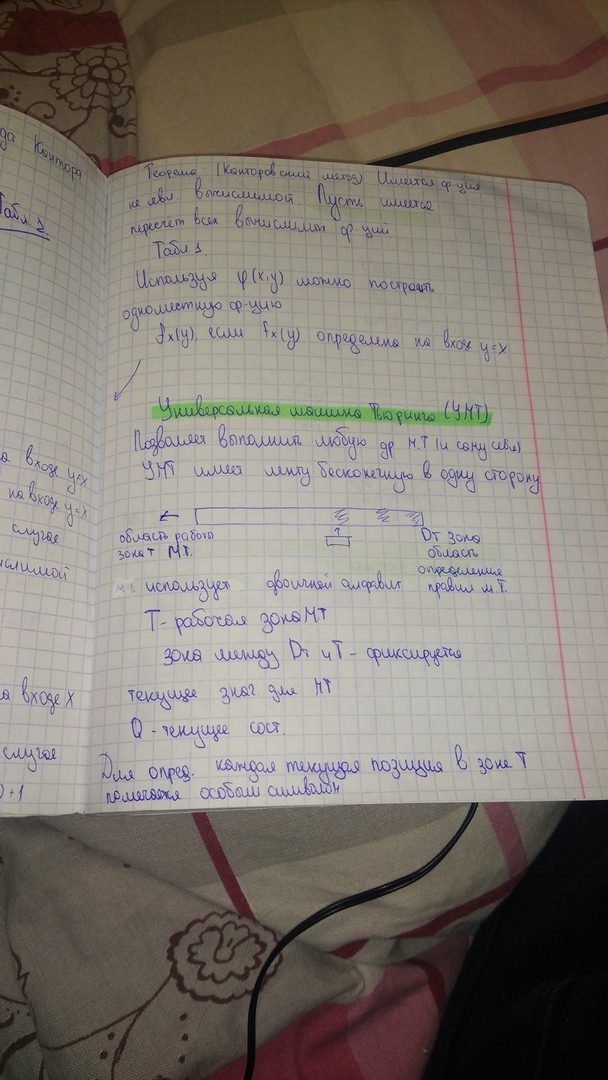
# 16. Гёделев номер машины Тьюринга (принцип его вычисления)

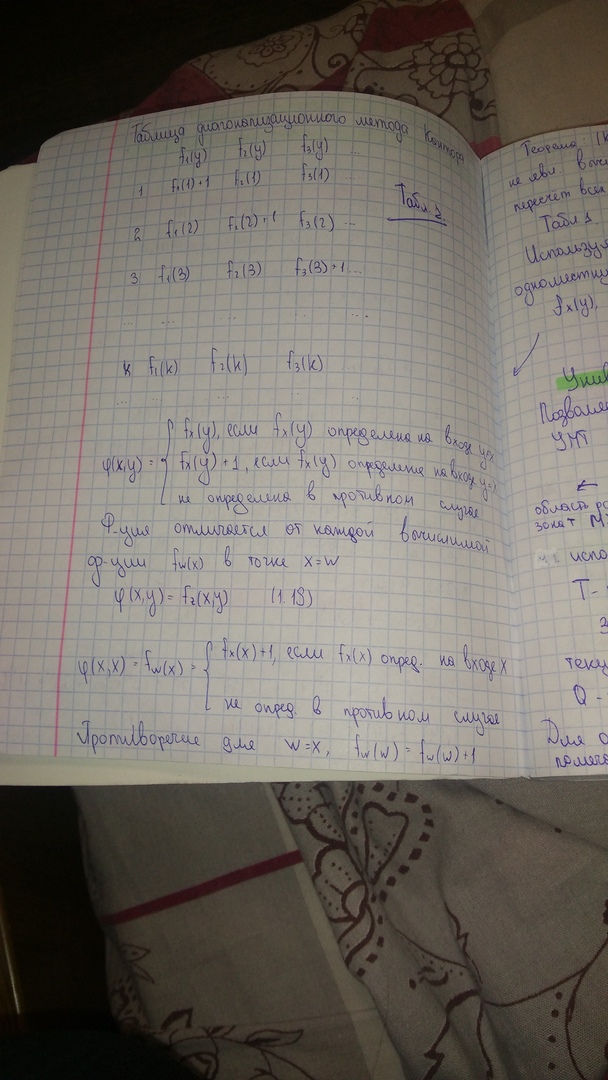
Любое число может быть представлено как произведение простых сомножителей. A – действие, каждый символ правила пронумерован числом. Эти числа подставляем как степени простых сомножителей (2^11 \* 3 ^29 \* 5^19…)

https://lh6.googleusercontent.com/2W-RETB3sU8PIzl6cNcDIXZNOoxbrwdUbj2UdFjbDgi_XUtR6aU2vIl53er9MIhgTjhKrFxGFAIj-oDgyaUQVFhSiUSRVp58Td-HyrM9AKvem6O7XBmD8iEmtTDvI332gkfECHt6=s800

# 17. Цель использования канторовского диагонализационного метода

 Для обозначения невычислимых функций(показать что нельзя построить МТ для некоторой функции)





# 18. Общее понятие универсальной машины Тьюринга

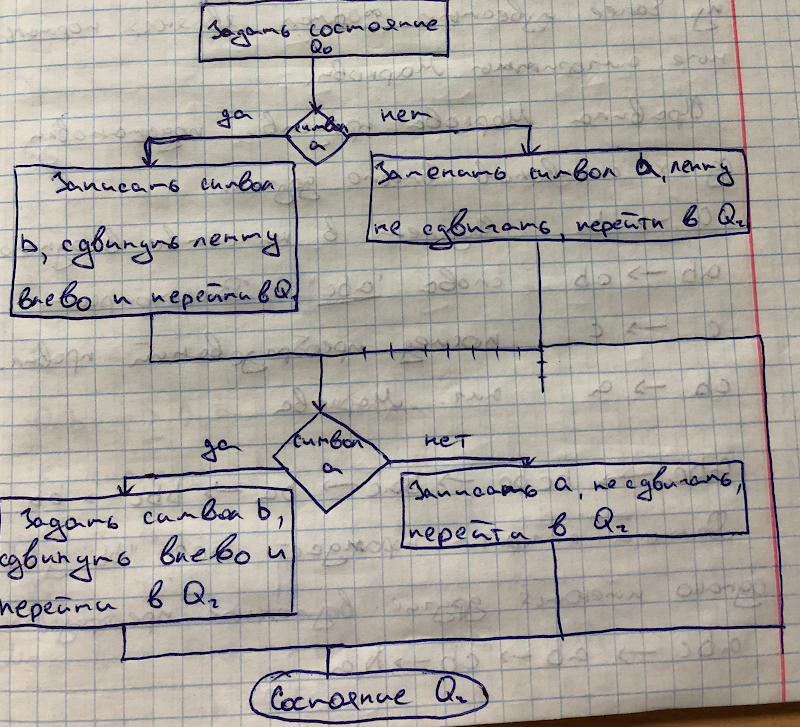
УМТ – это МТ, которая позволяет любую др мт (и саму себя). Используется бинарный алфавит +головка, УУ(уст-во упр-я). (Она достигает такого результата читая описание моделируемой машины, а также ввод данных для этой машины) Она имеет ленту, бесконечную в одну сторону. Справа(Дт(т - индекс) - область определения правил машини т; слева(Т) - область работы, между ними - текущее состояние, прочитанный символ .в зоне слева помечается состояние головки. умт ищет в зоне Дт правило перемещается в Т и выполняет действие, фиксированное в промежутке

# 19. Принципы построения универсальной машины Тьюринга

Получаем на вход Описание правил МТ, которую хотим симулировать, и входные данные, которые хотим симулировать. Между ними область работы и прочитанный символ.

# 20.   Представление алгоритма в виде графаhttps://lh4.googleusercontent.com/0o1Lvdc8cAQnIcqgNdPmcvolBownbIq3qYgNX_2A6k1Y8Dy3jAhnUYr2GW7OoCDq8F3q6aOnHsQ_kh-fyjPIAn5nOfZocc2OICQirUD3w6guRTdNklKI5tP52ESqfczsjXZts6jL=s800

# 21.   Блок-схема



где ромбики (писать значком) - условия

прямоугольники (писать значком) - действия или операции

# 22. Сущность правил алгоритмов Маркова

Правила Маркова представляют подстановку одних символов вместо других

a → bc

ab → cb

c → E

cb → a

Возьмем в кач-ве входного слова “abc”. Рассмотрим последний преобразованный правильный алгоритм маркова

abc → bcbc → bac → ba→ bbc → bb

Данный алгоритм порождает слово бб из абс. Однако имеются другие возможные преобразования

abc -->ab → cb → a

и тд тп

# 23. Нормальные алгоритмы Маркова

# Позволяют решить задачу детерменированно.

# Норм алгоритм маркова предполагает четкое упорядочивание правил подстановки. Для подстановки всегда выбирается 1е подходящее правило.

# Работа алг заканчивается, если:

# На каком-то шаге замен применилось правило с точкой в правой части (результативный останов)

# Входное слово просмотр. До конца и не удалось применить ни одного правила(нерезультативный останов)

# Зацикливание алг маркова

# 24. Взаимосвязь нормальных алгоритмов Маркова и машины Тьюринга

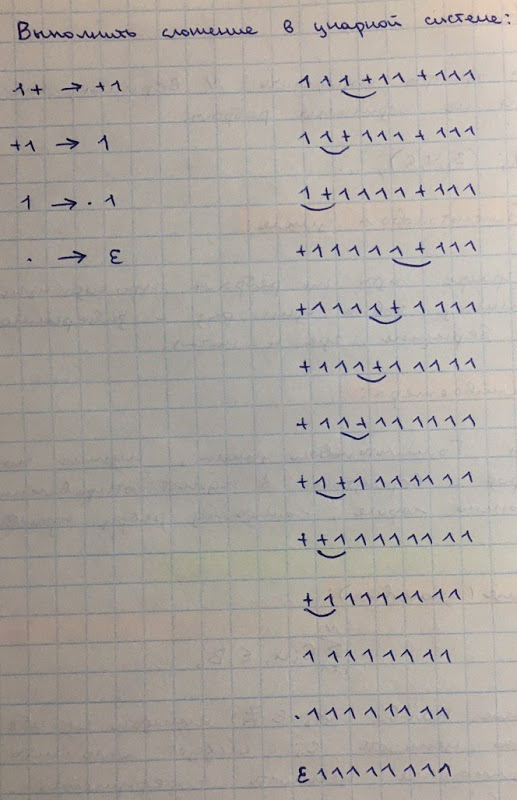
 МТ и НАМ по сути близки и их использование приводит к одинаковым результатам

# 25. Понятие самоприменимого нормального алгоритма Маркова

Самоприменимость алгоритмов маркова: Если процесс обработки входного слова, которое является описанием алгоритма Маркова, останавливается то такой алгоритм называется самоприменимым.

Доказано, что задача определения того, является алгоритм самоопределимым или нет, неразрешима.

# 26. Пример решения задачи сложения унарных чисел с помощью нормального алгоритма Маркова



# 27. Сущность идеи, на которой основано применение рекурсивных функций в теории алгоритмов

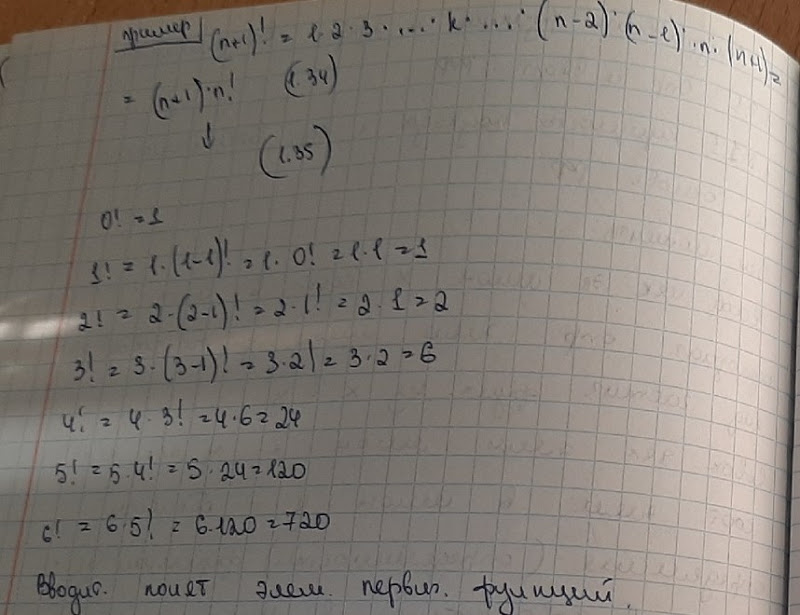
 Если некоторый элемент множества х поставить в соответствие однозначному определению элемента множества y, то говорят, что задана функция из х в y. Совокупность тех элементов множества х, у которых есть соответствующий элемент в множестве у - область определения (определённость) функции из х в у. А совокупность соответствующих значений в множестве у называется областью значений функции из х в у.

Если область определения функции из х в у совпадает с множеством х, то функция называется всюдураспределённой функцией из х в у.

# 28. Понятие рекурсии и пример рекурсивной функции

Рекурсия - способ задания функции, при котором значение определяемой функции для произвольных значений её аргумента определяется известным образом через значения определяемой функции для меньших значений аргумента.

**Пример**



# 29. Понятие первичной арифметической функции

Первичную функцию нельзя получить из других функций, но из первичных можно строить новые функции.

# 30. Типы первичных арифметических функций

Всего выделяют 3 первичных функции:

1)        Функция, тождественно равная нулю(ф-ция константы ноль)

               0n=f(x1,x2,…,xn)=0

2)         Функция тождества, повторяющие значения своих аргументов.

Ini=f(x1,x2,…,xn)=xi(1≤i≤n;n=1,2,3…)

(пример, частный случай:  I11=f(x)=x )

3)        Функция непосредственного следования(“последователь”)

S1=f(x)=x+1

# 31.Сущность и пример применения операции суперпозиции (подстановки)

 Суперпозиция - вместо одних переменных функции подставляют другие ФУНКЦИИ о тех же  или других переменных

Пусть заданы n-функций f1,f2,f3,…,fn от m переменных каждая, а так же функция f(x1,x2,…,xn) от n-переменных {xn}

Тогда операцией сyперпозиции f1,f2,f3,…,fn с f(x1,x2,…,xn)   будет порождена новая функция g(x1,x2,…,xn)= f(f1(x1,x2,…,xm),…, fn(x1,x2,…,xm))

Пример суперпозиции первичных функций f(x)=0 и g(x)=x+1:

h(x)=g(f(x))=g(0)=0+1=1.

# 32. Сущность операции примитивной рекурсии

Представим в общем виде эту операцию. Она позволяет строить (n+1) местную функцию на основании известных n и n + 2 местных функций.

Пусть заданы n местная функция g(n) и (n+2) местная функция h(n+2). Говорят, что (n+1) местная функция (f) возникает применением рекурсии из g( ) и  h( ), если для всех натуральных значений (x1,x2,…,xn,y) справедливо:

f(x1,x2,…,xn,0)=g(x1,x2,…,xn,0)

f(x1,x2,…,xn,y+1)=h((x1,x2,…,xn, y, f(x1,x2,…,xn,y))

Для каждого g( ) и h( ) есть f( ) и она является единственной, т.е.

Rn(g,h)

При том f в конкретной точке ищется механически по строгим правилам, т.е. для нахождения f(a1,a2,…an,m+1) достаточно последовательно посчитать число согласно :

b0=g(a1,a2,…,an)

b1=h(a1,a2,…,an,0,b0)

b2=h(a1,a2,…,an,1,b1)

………………………………

bm+1=h(a1,a2,…,an,m,bm)

Само определение:

*Если считать некую const функцией от нуля переменных (в уравнениях выше вместо n поставить 0), то можно определить одноместную функцию f, которая возникает примитивной рекурсией из a=const и двуместной частичной функцией в виде:*

*f(   ,a)=f(a)=0     и     f(   ,y+1)=f(y+1)=h(y,f(y))*

# 33. Пример применения операции примитивной рекурсии

пример: построить с помощью о. примит. рекурсии двуместную ф-цию суммирования f+ (+ - индекс, если что)(x,y)=x+y

Отметим, что прибавить y к x - это тоже самое, что прибавить 1 к x-y y раз. Искомая ф-ция определяется на основе тождественной ф-ции g(x)=x и h(x,y,z)=z+1.

f+(x,0)=g(x)= I11(x)=x

f+(x,1)=h(x,0, f+(x,0) )=h(x,0,x)=x+1

f+(x,2)=h(x,0, f+(x,1))=h(x,1,x+1)=x+2

…

f+(x,y-1)=h(x,y-2, f+(x,y-2))=h(x,y-2,x+y-2)=x+y-1

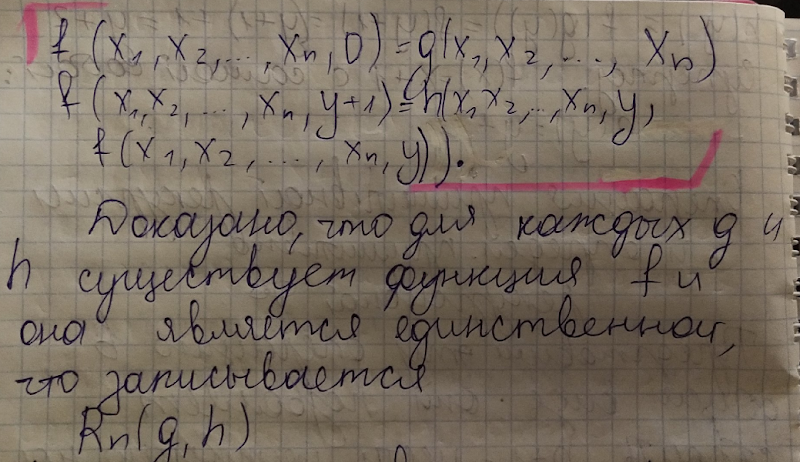
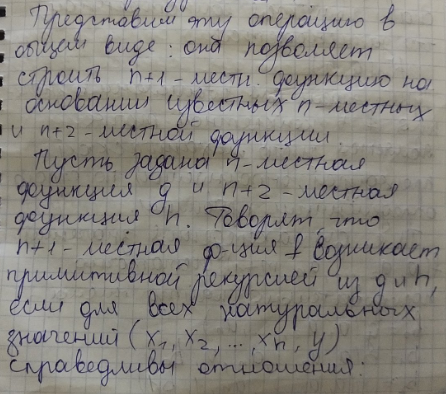
f+(x,y)=h(x,y-1, f+(x,y-1))=h(x,y-1,x+y-1)=x+y

# 34. Определение примитивно рекурсивных функций

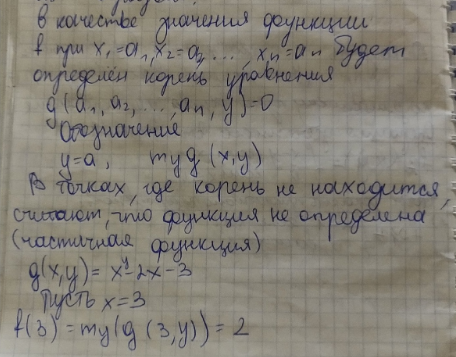
Функции, которые могут быть получены из первичной функции с помощью операций суперпозиции и примитивной рекурсии, выполняются сколько угодно раз в любой последовательности называются примитивно рекурсивные функции.

?!Примитивно рекурсивные функции всюду определимы

# 35. Как доказать примитивную рекурсивность функции



# 36. Сущность операции минимизации (взятия минимального корня)?

Операция минимизации позволяет определить новую арифметическую функцию f(x1, x2, …, xn) на основании известной g(x1, x2, …, xn, y) следующим образом:

## **37. Определение частично рекурсивной функции.**

Функции, которые могут быть получены из первичных функций с помощью операции суперпозиции, примитивной рекурсии и минимизации, выполненных сколько угодно раз, называются частично рекурсивными.

## **38. Определение общерекурсивной функции.**

Частично-рекурсивная функция, которая всюду определена, называется общерекурсивной.

### **39. Понятие множеств. Типы множеств в зависимости от их размера.**

Множество – совокупность объектов, объединённых по какому-либо признаку. Объекты множества называются элементами.

Конечные множества задаются перечислением элементов; бесконечные множества задаются правилами, определяющими свойства элементов этого множества; свойства элементов множества задают с помощью формул или функций.

# 40. Способы задания множеств

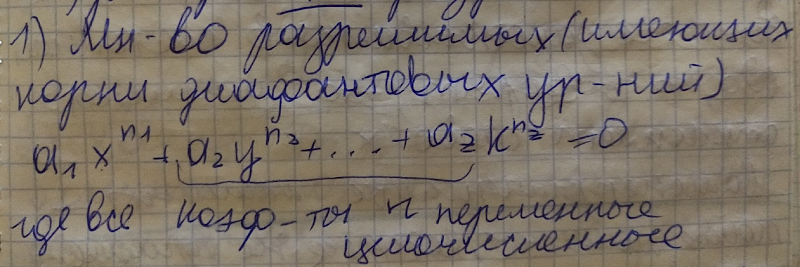
# Возможны два способа задания мн-ва: 1) Функцией разрешения. Когда члены множества определяются путем проверки: удовлетворяет ли кандидат в члены множества некоторому условию или нет. Такое мн-во - разрешимое.

2) Функцией перечислений. Когда члены множества определены по определенному правилу с помощью функции - генератора множеств. Это перечислимое множество.

# 41.Определение рекурсивного множества и особенность такого множества

Мн-во называется рекурсивным, если его характеристическая функция общерекурсивная. Особенность: для любых х с помощью алгоритма можно определить Е(пишется как эпсилон) принадлежащий элемент мн-ву или нет

#### 42. Пример не рекурсивного множества



## 43.Определение рекурсивно перечислимого множества

Множество является рекурсивно перечислимым, если его характеристическая функция частично рекурсивна:

# 44.Тезис Черча и цель его применения

отметим след положения:

1.каждая стандартнозаданная частично рекурсивная функция  вычислима путем определенной процедуры механического характера

2. какие бы классы ф-ций до сих пор не строились во всех случаях непременно оказывалось что числовые функции вычисляемые посредством алгоритмов этих классов были частично рекурсивными

на основе этого была сформулирована гипотеза - тезис Черча

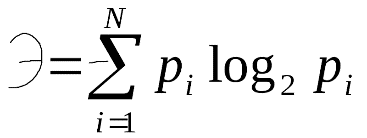
Тезис Черча: «Класс всех алгоритмически (или машинно) вычислимых частичных функций над положительными числами совпадает с классом всех частично рекурсивных функций.»

Если имеем функцию f(x) и она вычислима (соответственно и рекурсивна), то y=f(x)+1 будет рекурсивна и для нее можно построить МТ (которая не зациклится).

# 45.Что требуется, чтобы функция j, определяемая через другую функцию (функции), была вычислимой?

Чтобы функция y, определяемая через другие функции, была вычислима, нужно, чтобы эти функции сами были вычислимы, там, где они не определены, не определена и y

# 46.Варианты способов определения сложности алгоритма

1)Информационная сложность (понятие ввел К. Шеннон) - ,

 где пэ-итое - вероятность, с которой процесс вычисления пойдёт по и-тому пути, N-число вариантов решения задач.

2)алгоритмическая сложность (понятие ввел А.М.Колмогоров) - сложность определяется через размер программы, которая реализует соотв алгоритм .

3) вычислительная сложность

суть: хар-ся числом тактов, которое должна реализовать МТ для получения решения задачи на МТ.

# 47.Понятие информационной сложности алгоритма

# информационная сложность:

Существует несколько (N) путей реализации задачи, Pi – вероятность, с которой пойдёт решение задачи по i-му способу; формула выше.

## **48.Понятие алгоритмической сложности алгоритма**

Сложность определяется по размеру программы для выполнения алгоритма;

# 49. Понятие вычислительной сложности алгоритма

Характеризуется числом тактов, которое должна реализовать МТ для получения решения задачи.

##### **50 Понятия эффективного и неэффективного алгоритма**

Эффективный алгоритм - алгоритм с полиномиальной временной сложностью.

Неэффективный алгоритм - алгоритм с экспоненциальной сложностью вычислительной функции (труднорешаемая задача). Задача “подоконник”.

#### **51.Определение класса задач NP**

Класс задач распознавания типа да/нет с хорошим алгоритмом проверки решения наз классом НП. Решения можно получить за полиномиальное время на распознающей машине Тьюринга

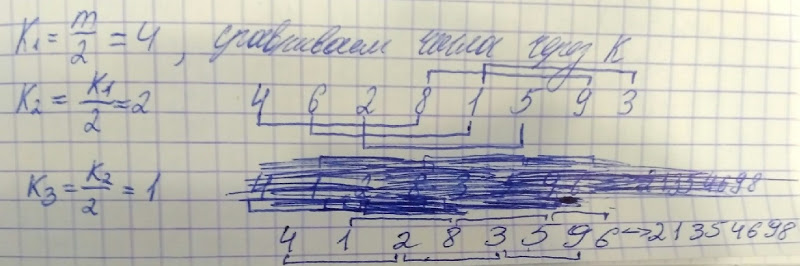
Задача из NP, к которой можно свести любую другую задачу из этого класса за полиномиальное время называется NP-полной

Класс NP включает NP-полные задачи и задачи, которые сводятся к NP-полным.

Класс NP-трудных задач включает NP-полные задачи и задачи, которые сложнее их (то есть те задачи к которым могут быть сведены NP-полные задачи.)

### **52. Суть метода простой сортировки списка целых чисел**

Массив: 4,6,8,2,1,4,9,3

Каждый раз сравниваем числа вдвое меньше их количества (m)

1,2,3,4,5,6,8,9

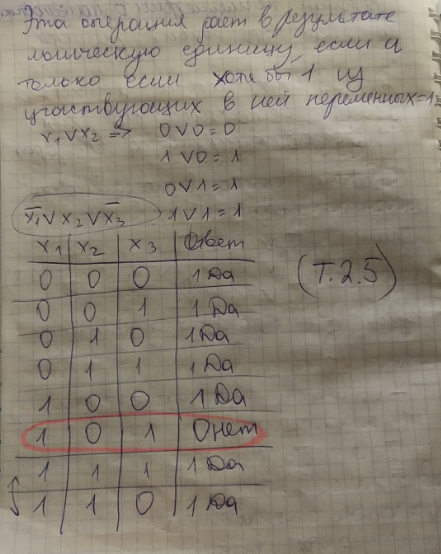
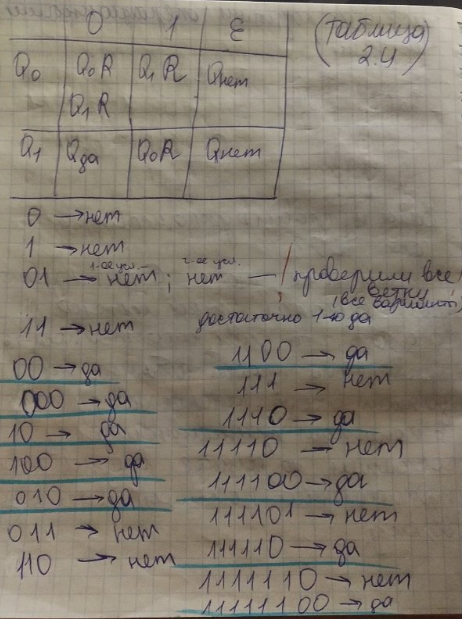
Число приходов z=log2(⁡m)(логарифм ⁡m по основанию 2)

Общее максимальное количество проверок:О(m\*log2(⁡m))

#### **53. Функция вычислительной сложности алгоритма простой сортировки списка целых чисел**

(n-1) + (n-2) + … + 2 + 1 = (𝑛2−𝑛)2 -> O(𝑛2)

###### **54.  Усовершенствованный метод сортировки списка целых чисел**



###### **55. Функция вычислительной сложности усовершенствованного алгоритма сортировки списка целых чисель**

 𝑧=𝑙𝑜𝑔2𝑛 𝑝=(𝑛−𝑘𝑖) (𝑛−𝑘𝑖)𝑙𝑜𝑔2𝑛 𝑂(𝑛𝑙𝑜𝑔2𝑛)

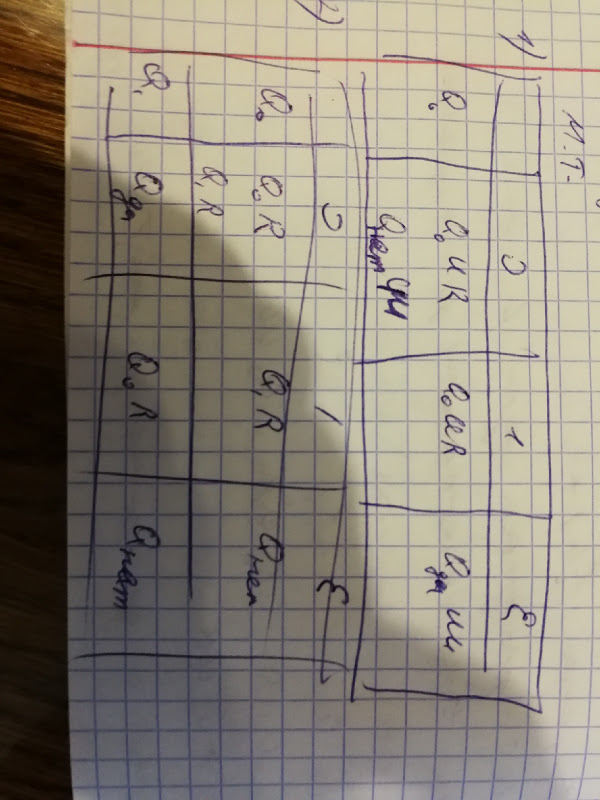
##### **56. Алгоритмическое определение проблемы распознавания языка**

Язык – множество слов над некоторым алфавитом. Он задаётся характеристической функцией. Алгоритмическая формулировка проблемы распознавания языка: построить МТ, вычисляющую характеристическую функцию данного языка.

#### **57. Определение недетерминированной машины Тьюринга**

В недетерминированной машине Тьюринга разрешается иметь несколько правил с одинаковыми условиями и разными операциями.

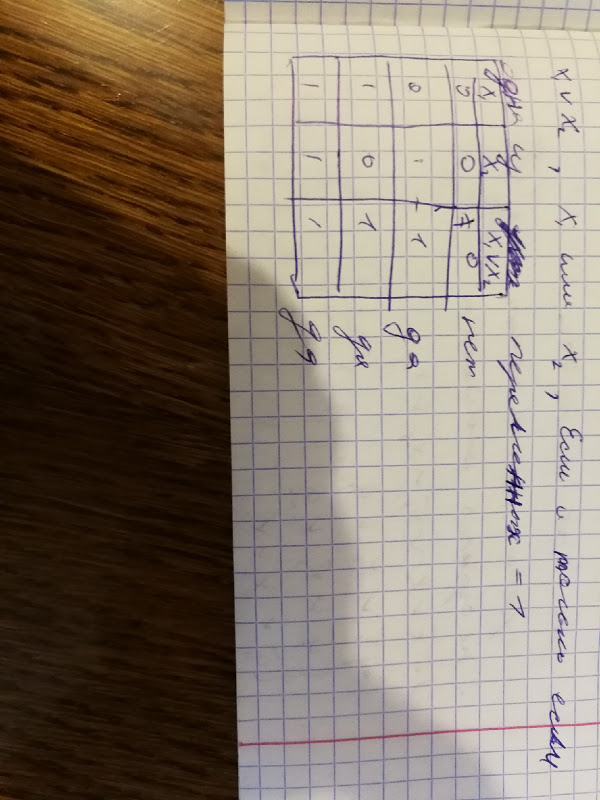
#### **58. Пример недетерминированной М.Т.**



### **59. Понятие дизъюнкта**

Дизъюнкт – слово языка выполнимости bool(true/false)

## **60. Понятие таблицы истинности дизъюнкта**





### **61. Определение понятия выполняющей интерпретации**

Интерпретация - набор логических значений для переменной. Выполняющая интерпретация – интерпретация, при которой дизъюнкт равен 1(единице). Словами языка ВЫП являются дизъюнкты.

##### **62. Определение задачи ВЫПОЛНИМОСТЬ**

Задача ВЫП заключается в следующем:

Имеется множество дизъюнктов. Спрашивается, есть ли для этого мн-ва дизъюнктов хотя бы одна общая выполняющая интерпретация? Если да, то множество дизъюнктов называется выполнимым, если нет, то невыполнимым.

##### **63. Понятие языка ВЫПОЛНИМОСТЬ и проблема его распознавания**

Языки ВЫП содержат мн-во слов, представляющих все выполнимые системы дизъюнктов.

Проблема распознавания языка ВЫПОЛНИМОСТЬ связана с распознаванием выполнимости данной системы дизъюнктов. Если система дизъюнктов выполнима, то это легко проверить, используя значения переменных выполняющей интерпретации. Проверку можно реализовать на недетерминированной в общем случае машине Тьюринга.

##### **64.Каким способом оценивается сложность проверки варианта решения** **задачи ВЫПОЛНИМОСТЬ?**

Сложностью проверки называется функция для числа тактов проверочной процедуры в зависимости от длины описания задачи (размера описания задачи). Оказывается, по критерию сложности все машины Тьюринга можно разделить на два класса: машины с функцией сложности, ограниченной некоторым полиномом и машины с функцией сложности, не ограниченной полиномом. Первый тип машин называется также эффективным.

## **65.Определение класса языков NP**

Класс NP - трудные включающие NP-полные задачи и задачи, которые сложнее их, к которым могут быть сведены NP-полные задачи (NP-легкие).

В теории сложности Класс NP — класс языков (задач), ответ на которые можно проверить за полиномиальное время. (инет)

## **66. Определение эффективной сводимости задачи А к задаче В**

Задача А эффективно сводится к задаче В, если:

1) задача А эффективно сводится к задаче B, если решение задачи В даёт решение задачи А;

2) сложность сведения ограничена полиномиальной функцией.

### **67. Сведение задачи ВЫПОЛНИМОСТЬ к задаче 3-ВЫПОЛНИМОСТЬ**

Задача 3-выполнимисть - задача выполнимость, каждый дизъюнкт которой содержит не более 3ех переменных.

При замене используется следующее правило:

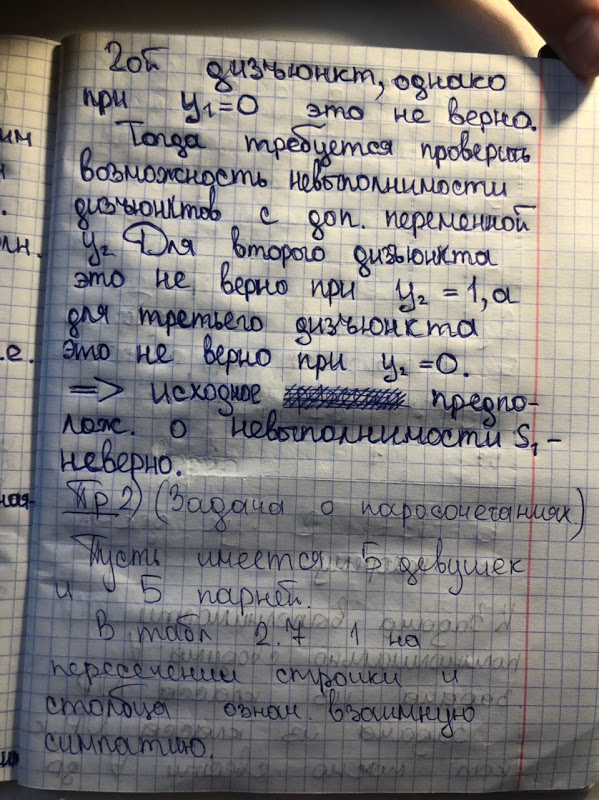
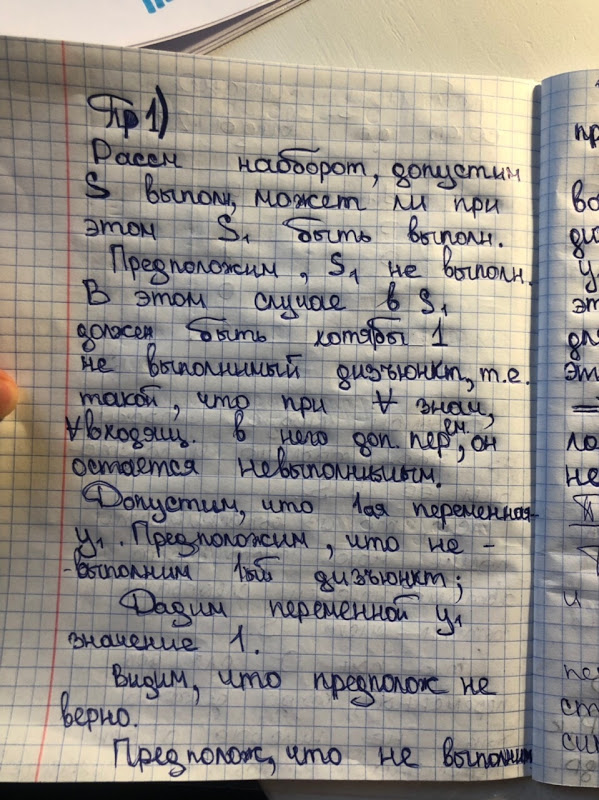
1)В начале дизъюнкта, содержащего 3+ переменных, вводятся доп. переменные.

2)Отсчитываем 3 первые переменные и формируем из них дизъюнкт-тройку.

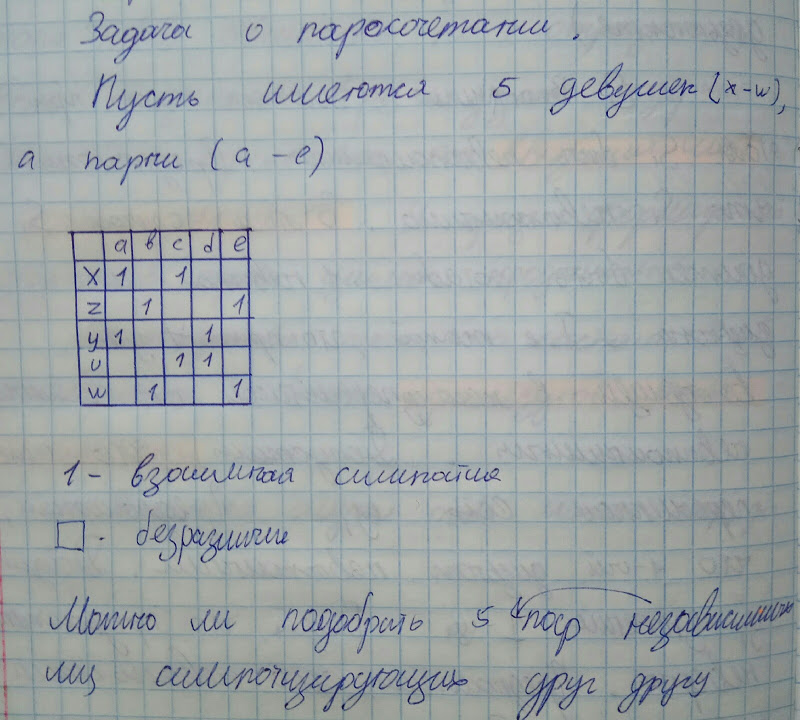
3)В начало оставшегося дизъюнкта добавляется та же новая переменная, но с отриц.(если получается в результ. дизъюнкт, содерж. 3+ переменных, то вводится след. новая переменная и процесс формир. дизъюнктов-троек продолжается. )

### **68.Принципы доказательства эквивалентности дизъюнкта его представлению** **в виде системы трехместных дизъюнктов**

 Для док-ва, что системы эквивалентны обычно используются таблицы истинности (из и-нета: - таблица, описывающая логическую функцию.)



##### **69.Суть задачи о паросочетаниях и принцип ее решения.**



### **70.Теорема С.Кука и определение NP-полной задачи.**

Теорема кука: К задаче ВЫПОЛНИМОСТЬ полиномиально сводится любая задача из класса NP.

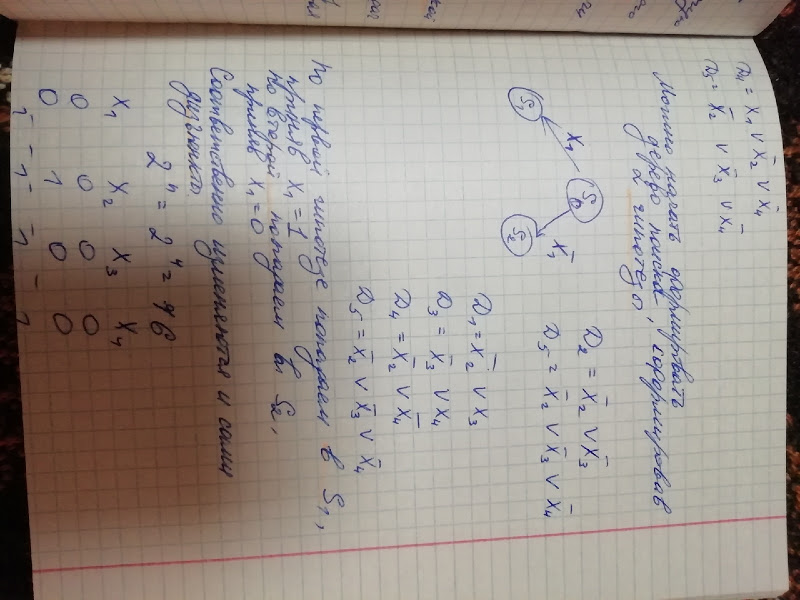
     NP-полн: Задача ВЫП яв-ся NP-полной, если для нее существует полиномиальный детерменированный алгоритм решения.

##### **71.Понятие NP-трудной задачи.**

Задача называется *NP*-*трудной*если каждая задача из *NP*полиномиально сводится к ней.

## **72.Суть стратегии поиска решения задачи ВЫПОЛНИМОСТЬ на основе дерева состояний задач**https://lh3.googleusercontent.com/NLaIH0DQKF5ZDP4AfonHxwkEDirue5o8NITQ5W-f8n-baUunMuzPzKVVuwS3ZB5FaE8IfoVM3n_UbjXblvYQcmGrokPrgGIPHDAlTg8vNg7gNOMn8oC_kr0fyiycSO4Qielqvuld=s800https://lh4.googleusercontent.com/RFFFQEtSbAt1DKoxLD89x6zyaNp3HcUUeGx36uNPw_-nSs4l6sW-0tdhA7MbpCaQv-IZMpBvE5jzhtM-00AAAUqmduG6paODrKKvh-_M8W3vaX03p7hRJUqrwRVw70rktToMIkZB=s800

## 73. Пример эвристической функции для оценки перспективности выбора вершин на дереве состояний



F = (https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%5Cfrac%7B%7Bk%7D_%7B1%7D%7D%7Bm%7D%5C+%2B%5C+%5Cfrac%7Bn%5C+-%5C+%7Bk%7D_%7B2%7D%7D%7Bn%7D)

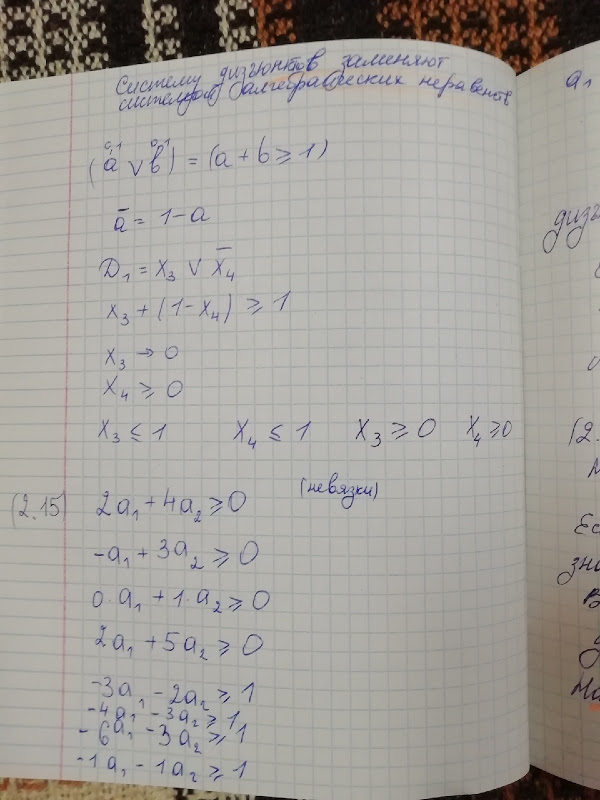
m - кол-во дизъюнктов в исходной системе

n - число разноименных переменных в исходной системе

https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%7Bk%7D_%7B1%7D- кол-во дизъюнктов в рассматриваемой вершине

https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%7Bk%7D_%7B2%7D- среднее число переменных в дизъюнктах данной вершины

## 74. Цель и правила замены логических формул (дизъюнктов задачи ВЫПОЛНИМОСТЬ) их алгебраическими эквивалентными представлениями



## **75. Выводы о разрешимости задачи ВЫПОЛНИМОСТЬ на основе анализа результатов решения соответствующей ей системы линейных алгебраических неравенств**

Если в системе нет невязок (положительная правая часть), то её решение достигается нулевым решением переменных.

### **76. Суть алгоритма метода устранения невязок для решения системы линейных алгебраических неравенств**

Если в системе нет невязок, то ее решения доставляются нулевыми значениями переменной. В противном случае используется алгоритм, который обеспечивает устранение невязок. Выделяется две фазы в алгоритме.

### **77. Задача, решаемая на первой фазе метода устранения невязок**

На первой фазе получают систему базисных неравенств вида ai >= 0 путём устранения невязок

## **78. Действия, выполняемые на второй фазе метода устранения невязок**

Действия выполняются так же, как и на первой фазе, но уже при наличии базисных неравенств Если на 2-й фазе в процессе интерации встречается невязка, причём все коэффициенты в левой части неположительны, то устанавливается факт неразрешимости (несовместности) системы неравенств.

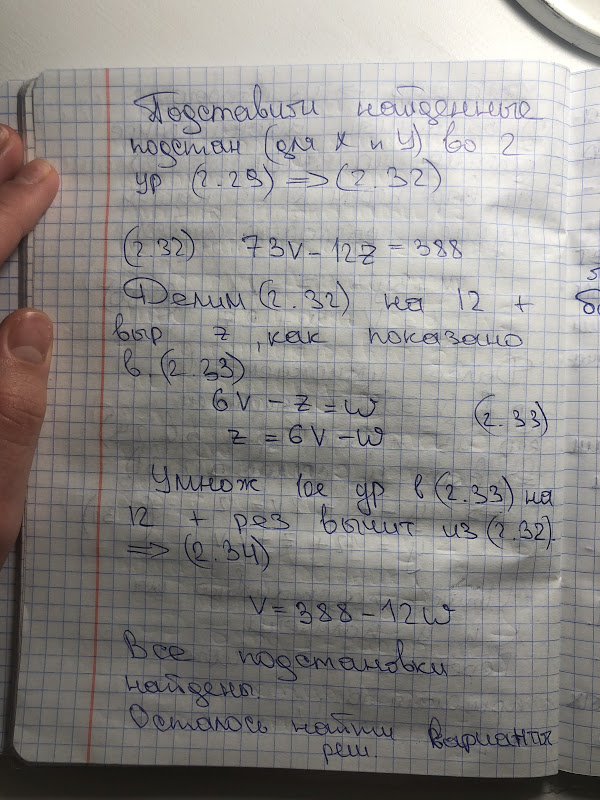
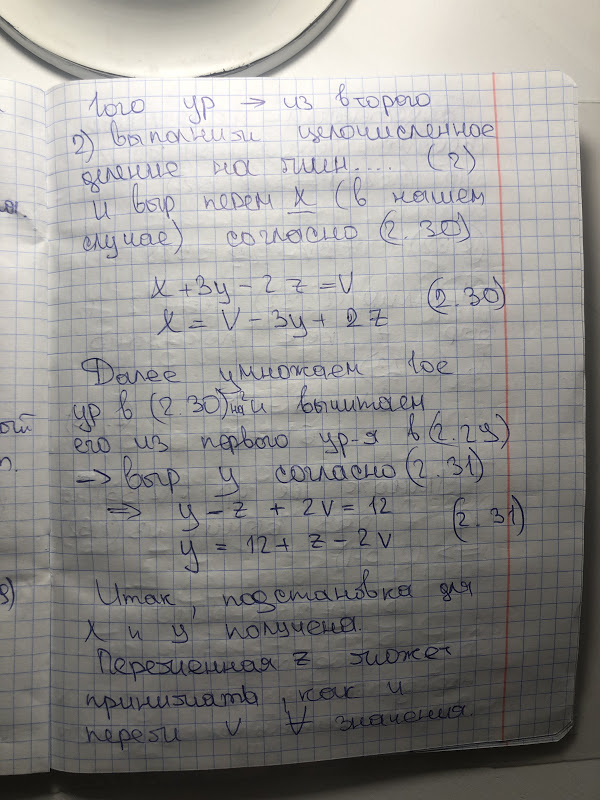
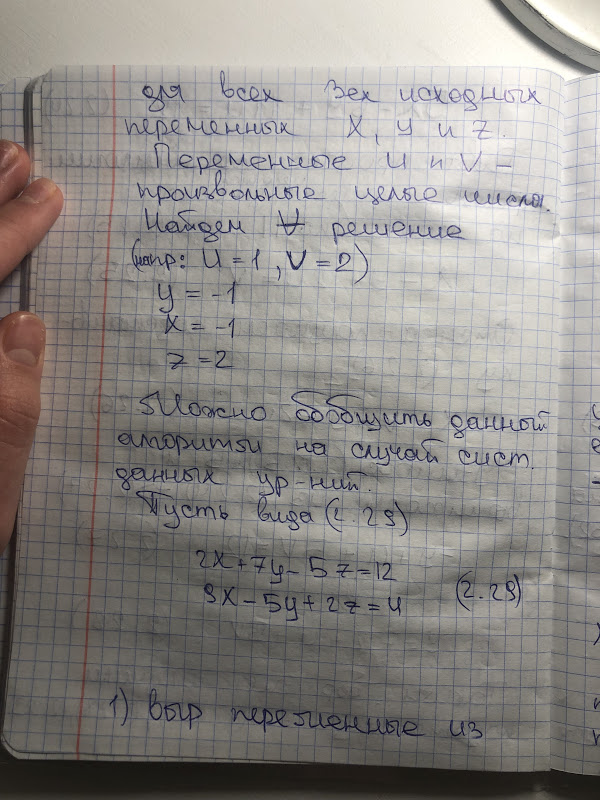
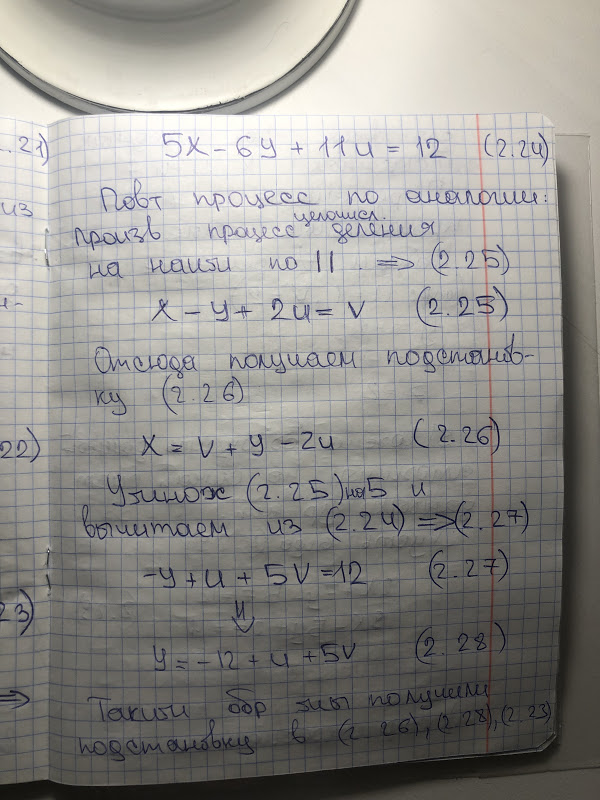
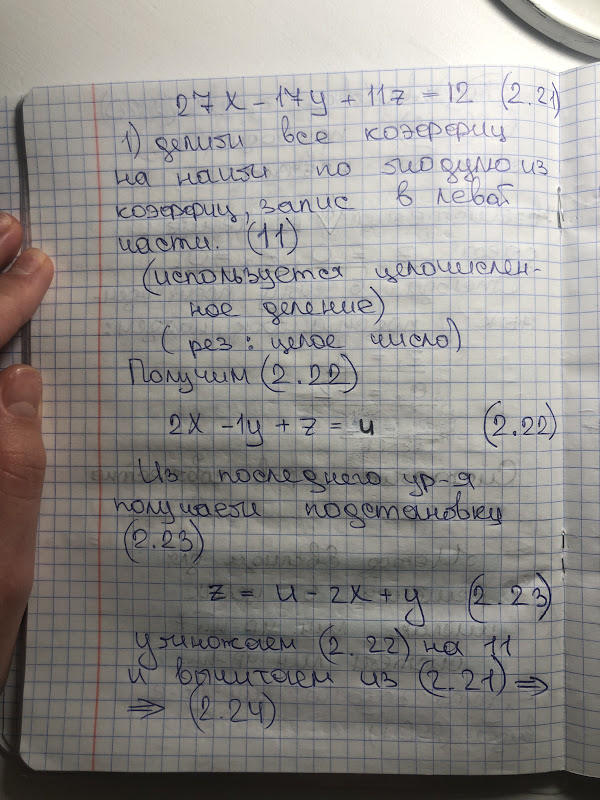
### **79. Суть метода Евклида для решения линейных уравнений**

Пусть задано линейное неравенство. Целочисленно делим все коэффициенты на наименьший (по модулю) из левой части (вместо целого числа в правой части ставим какую-либо константную переменную). Выражаем какую-либо переменную. Далее умножаем полученное после деления уравнение на коэффициент, на который делили ранее, и это уравнение вычитаем из первоначального линейного уравнения. Повторяем действия, пока не получим подстановку всех переменных.

## **80. Суть метода Евклида для решения систем линейных уравнений**

Используем метод Евклида для решения каждого из линейных уравнений в системе.

Пусть задано линейное уравнение вида (2.21)



## **81. Постановка задачи «Вершинное покрытие»**

Вершина i покрывает некоторое ребро, если данная вершина является концевой вершиной рассматриваемого ребра. «Вершинное покрытие» - множество вершин графа, если для любого ребра графа найдется хотя бы одна из вершин этого множества, которая её покрывает. Задача «Вершинное покрытие»: найти минимальное вершинное покрытие графа.

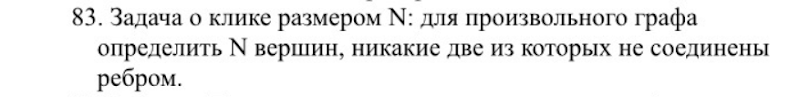
##### **82. Жадный алгоритм решения задачи «Вершинное покрытие»**

На каждом шаге алгоритма производим следующие действия:

* Выбираем случайное ребро графа e=(u, v).
* Добавляем в решение S обе выбранные вершины u и v.
* Удаляем из графа все ребра, инцидентные вершинам u или v.

Повторяем этот шаг до тех пор, пока не удалим все ребра графа.

## **83**



### **84. Постановка задачи о гамильтоновом цикле и задачи коммивояжёра**

**О гамильтоновом цикле**: Имеется ли в графе путь по ребрам, проходящий через каждую вершину ровно один раз, который бы завершился в той же вершине, из которой он начинался

**Коммивояжер**: Развитие задачи о гамильтоновом цикле. Необходимо посетить каждый город ровно один раз в пределах некоторой территории и возвратиться в пункт отправления, при этом путь должен быть как можно короче.

### **85. Определение и пример комбинаторной задачи**

*Комбинаторной задачей* называется задача, если множество её значений получается как различные комбинации значений объектов, участвующих в постановке этой задачи.

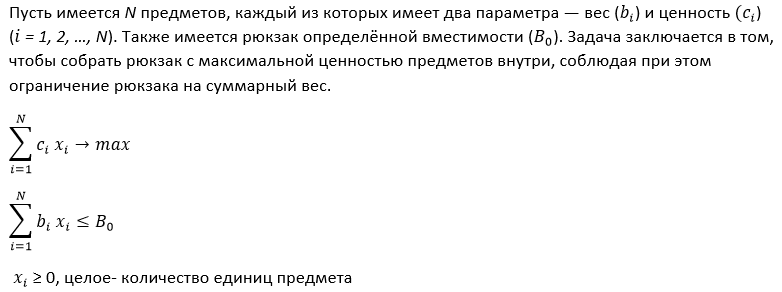
Пример. Необходимо посетить каждую вершину в пределах одного графа и вернуться в пункт отправления, при этом путь должен быть минимальным.

#### **86. Определение и пример эвристического алгоритма**

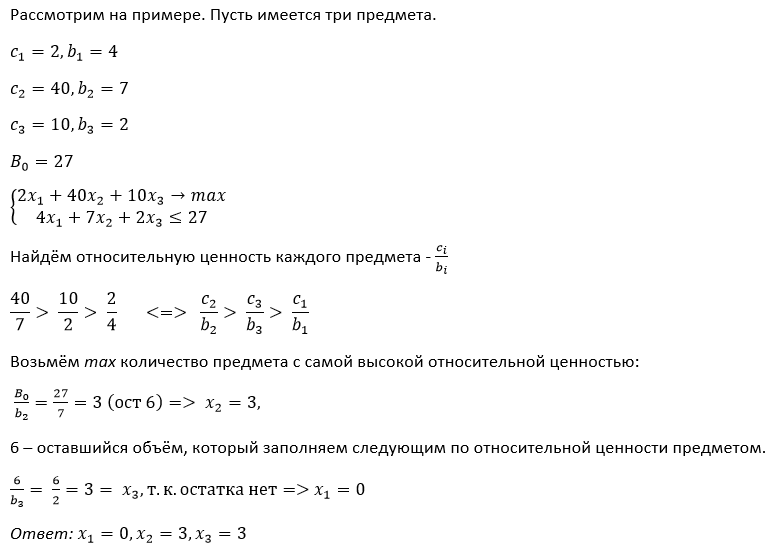
*Эвристический алгоритм*- это алгоритм решения задачи, использующий практический метод, не являющийся гарантированно точным или оптимальным, но достаточным для решения поставленной задачи.

Пример. Задача о ранце (см. №87, 88)

#### **87. Постановка задачи о ранце.**



#### **88. Эвристический алгоритм решения задачи о ранце.**

* 

### **89. Определение матроида и его связь с жадным алгоритмом**

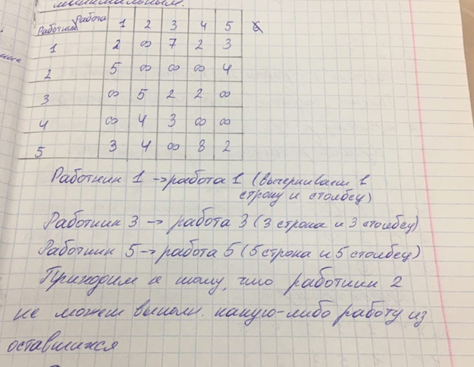
*Матроид -*структура,поиск решения на которой может быть сведен к последовательному отбору элементов.При этом число элементов в решении всегда должно быть одним и тем же.

Если доказать, что объект является матроидом, то жадный алгоритм будет работать корректно и выдавать правильный результат.

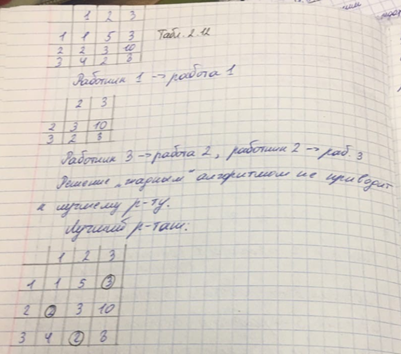
## 90. Постановка задачи о составлении расписания работ (для n работ и n работников)

Суть: имеется n работ и n работников. Известно время выполнения работ работниками. Если работник не может выполнить работу, время =∞. Требуется распределить работу так, чтобы каждый работник получил 1 работу и при этом суммарное время выполнения всех работ должно быть минимальным.

Пример 1:



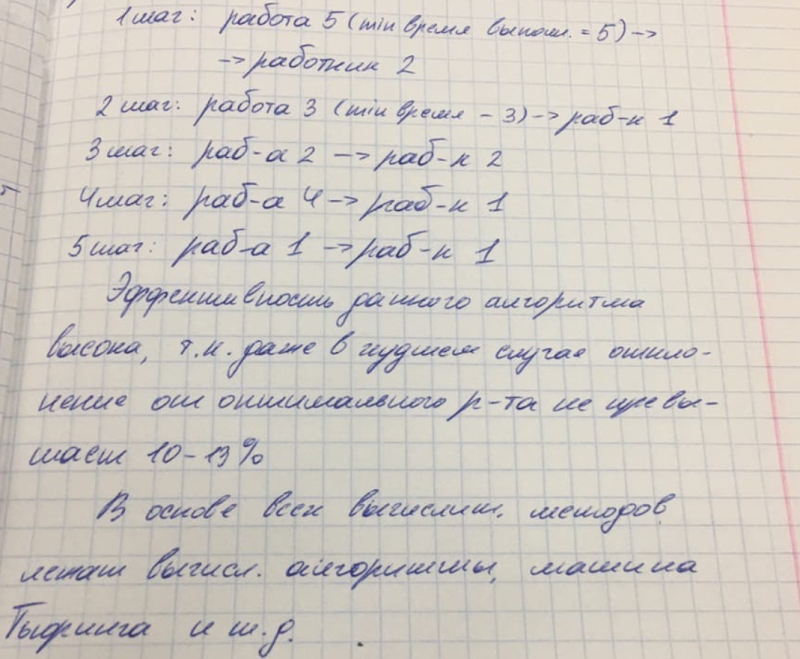
Ещё примеры:



### **91. Суть жадного алгоритма для решения задачи о составлении расписания** **работ (для n работ и n работников)**

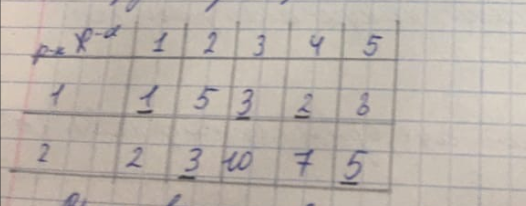
Суть: время выполнения всех работ должно быть минимальным. Применим “жадный” алгоритм: на каждом шаге выбираем работу, минимальное время которой максимально. Назначаем ее тому работнику, который заканчивает работу раньше.

Решение:



### **92. Постановка задачи о составлении расписания работ (для n работ и 2-х работников)**

Суть: имеется n работ и 2 работника. Известно время выполнения работ работниками. Если работник не может выполнить работу, время =∞. Требуется распределить работу так, чтобы каждый работник получил 1 работу и при этом суммарное время выполнения всех работ должно быть минимальным.



## **93. Суть жадного алгоритма для решения задачи о составлении расписания работ (для n работ и 2-х работников)**

На каждом шаге выбираем работу, минимальное время которой максимально. Назначаем ее тому работнику, который заканчивает работу раньше.