Коллок по ДМ №4

by @alekshi3

1. Метод математической индукции.

Wiki:

- Метод математического доказательства, который используется, чтобы доказать истинность некоторого утверждения для всех натуральных чисел. Для этого сначала проверяется истинность утверждения с номером *1*,затем доказывается, что если верно утверждение с номером *n*, то верно и следующее утверждение с номером *n+1*.

Presentation:

МАТЕМАТИЧЕСКАЯ ИНДУКЦИЯ

Метод математической индукции является важным способом доказательства предложений (утверждений), зависящих от натурального аргумента.

Метод математической индукции состоит в следующем:

Предложение (утверждение) P(n), зависящее от натурального числа n, справедливо для любого натурального n если:

1. P(1) является истинным предложением (утверждением);
2. P(n) остается истинным предложением (утверждением), если n увеличить на единицу, то есть P(n + 1) - истинное предложение (утверждение).

Таким образом метод математической индукции предполагает два этапа:

1. Этап проверки: проверяется, истинно ли предложение (утверждение) P(1).
2. Этап доказательства: предполагается, что предложение P(n) истинно, и доказывается истинность предложения P(n + 1) (n увеличено на единицу).

Замечание В некоторых случаях метод математической индукции используется в следующей форме:

Пусть m - натуральное число, m > 1 и P(n) - предложение, зависящее от n, n ≥ m.

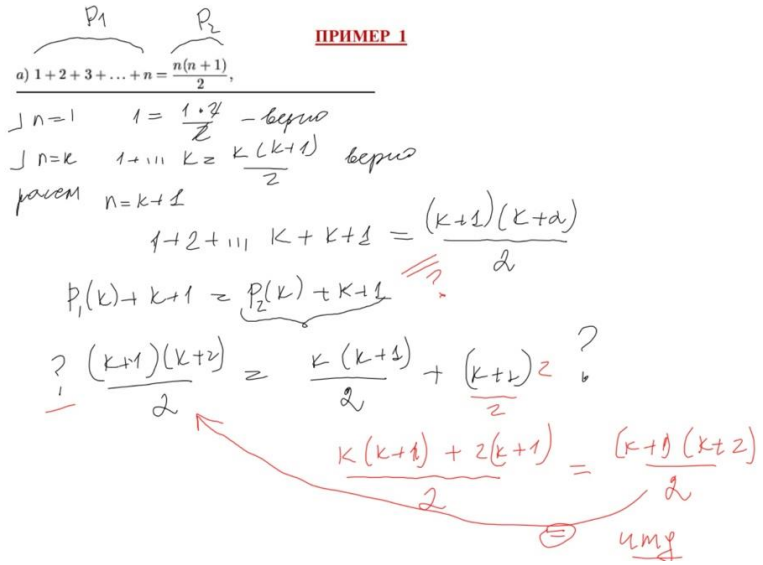
Если

1. P(m) справедливо;
2. P(n) будучи истинным предложением, влечет истинность предложения P(n + 1) для любого натурального n, n ≥ m, тогда P(n) - истинное предложение для любого натурального n, n ≥ m.

Вся информация здесь: **https://www.math.md/school/krujok/inductr/inductr.html**

ПРИМЕР 1

a) $1+2+3+\ldots+n=\frac{n(n+1)}{2}$,



2. Определение сочетания (не формулой).

Неупорядоченный набор размера k из n элементов *(C(n,k))*.

Presentation:

3. Формулы для сочетаний без повторений.

$$C(n,k) = C_n^k = \binom{n}{k} = \frac{A(n,k)}{k!} = \frac{n!}{(n-k)! \cdot k!}$$

4. Формулы для сочетаний с повторениями.

$$\dot{C}(n,k) = \dot{C}_n = \left(\dot{n}_k\right) = \frac{(n+k-1)!}{(n-1)!\cdot k!}$$

5. Определение размещения (не формулой).

– упорядоченная последовательность длины k из n элементов *(A(n,k))*.

Presentation:

Arrangement (Partitial Permutation / k-Permutation)

Размещения без повторений

Def Размещение – упорядоченная последовательность длины m из n элементов

$m < n$

Def Число размещений без повторений

$$A(n,k) = A_n^k = n\cdot(n-1)\cdot\ldots\cdot(n-k+1) = \frac{n!}{(n-k)!}$$

Пример: В чемпионате по футболу участвуют 16 команд. Сколько существует способов распределения медалей за первые три места?

$n = 16$ (алфавит)
$k = 3$ (события)

16 (E1) 15 (E2) 14 (E3) $= 16\cdot 15\cdot 14$

$\frac{16!}{(16-3)!} = \frac{16!}{13!}$

NB: Важен порядок. Элементы различны

Arrangement of k elements of a set S Permutation with repetition

Размещения с повторениями

Def Число размещений с повторениями

$$\dot{A}(n,k) = \dot{A}_n^k = n^k$$

Пример: Сколько существует трехзначных чисел, состоящих только из нечетных цифр?

$k = 3$
$n = 5$

нечет: 1, 3, 5, 7, 9 ($n = 5$)

$5 \cdot 5 \cdot 5$ ($m = 3$)

$$\dot{A}_5^3 = 5^3$$

Мощность булеана 2^n
Число наборов функц n-ар2
Кол-во булевых функций n-ар2

NB: Важен порядок. Элементы могут повторяться

6. Формулы для размещений без повторений.

$$A(n,k) = A_n^k = \frac{n!}{(n-k)!}$$

7. Формула для размещений с повторениями.

$$\dot{A}(n,k) = \dot{A}_n^k = n^k$$

8. Определение перестановки (не формулой).

это переупорядочение набора элементов *(P(n))*.

Presentation:

Permutation (without repetition)

Перестановки без повторений

Def Факториал – произведение всех натуральных чисел от 1 до n = $n!$

Def Перестановка – это переупорядочение набора элементов

Def Число перестановок без повторений: $P(n) = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1$

$$P(n) = n!$$

Пример: Сколько существует возможных способов формирования числа, переставляя цифры 1, 2, 3, 4, 5?

неповторяющиеся $P(5) = 5!$ $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$

NB: Важен порядок. Элементы различны

Permutation of multiset

Перестановки с повторениями

Def Число перестановок с повторениями

Имеется n элементов k различных типов, тогда количество различных перестановок этих элементов:

$$\dot{P}(n) = \frac{n!}{n_1! \cdot n_2! \cdot \ldots \cdot n_k!}$$

Пример: Сколько слов можно образовать слов из букв слова МАТЕМАТИКА $n=10$ $k=6$

$$\dot{P}(n) = \frac{10!}{2! \cdot 3! \cdot 2! \cdot 1! \cdot 1! \cdot 1!} = 151200$$

М – 2
А – 3
Т – 2
Е – 1
И – 1
К – 1

NB: Важен порядок. Элементы могут повторяться

9. Формула перестановок с повторениями.

$$\dot{P}(n) = \frac{n!}{n_1! \cdot n_2! \cdot \ldots \cdot n_k!}$$

10. Формула перестановок без повторений.

$$P(n) = n!$$

11. Отличие перестановок и размещений.

перестановки задействуют все элементы набора, а размещения только часть его элементов.

12. Принцип Дирихле.

если поместить $n + 1$ объект в n контейнеров, то по крайней мере в одном контейнере будет более одного объекта.

13. Принцип сложения.

Wiki:

- если событие A имеет a возможных исходов, а событие B имеет b возможных исходов, причем только одно из этих событий может произойти, то общее число возможных результатов = a+b.

Presentation:

Принцип сложения: Пусть $S_1, S_2, \dots S_m$ - попарно непересекающиеся множества. Пусть для каждого i, множество S_i содержит n_i элементов.

Тогда выбрать один элемент из них можно $n_1 + n_2 + \cdots + n_m$ способами.

На языке теории множеств: $|S_i| = |n_i|$

NB: Выбирается один элемент. Попарно непересекающиеся множества
Пример: Сколько способов выбрать один фрукт из 5 груш и 6 яблок?

14. Принцип умножения.

Wiki:

- если есть a способов сделать что-то и b способов независимо сделать что-то другое, то существует a*b способов сделать и то, и другое.

Presentation:

Принцип умножения: Пусть задана последовательность событий $E_1, E_2, \dots E_m$ таких, что E_1 осуществляется n_1 способами E_2 осуществляется n_2 способами и т.д.

Тогда вся эта последовательность (упорядоченная) может быть осуществлена $n_1 \cdot n_2 \cdot \cdots \cdot n_m$ способами.

NB: Последовательный выбор нескольких элементов (одновременно). Множества могут совпадать, пересекаться, не совпадать

Пример: Номера автомобиля, подбрасывание монеты и кости.

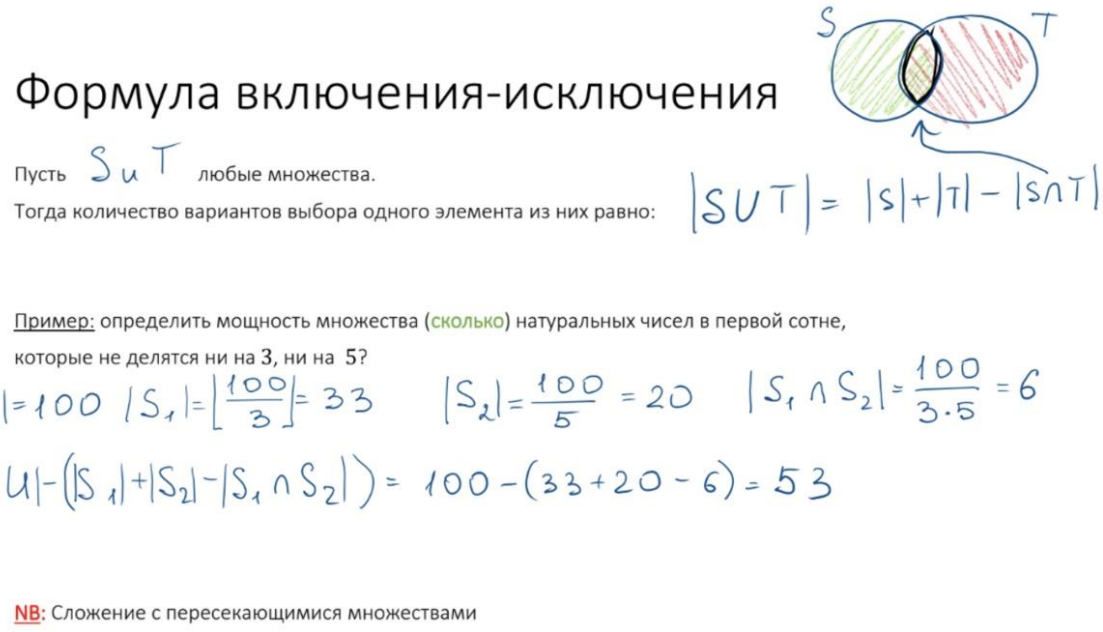
15. Формула включения-исключения.

Wiki:

- если имеются два множества, то количество элементов в их объединении равно сумме количеств элементов во множествах за вычетом количества элементов в их пересечении.

- формула для 2 множеств:
  $A \vee B = A + B - A \wedge B$
- формула для 3 множеств:
  $A \vee B \vee C = A + B + C - A \wedge B - B \wedge C - A \wedge C + A \wedge B \wedge C$

Presentation:



16. Лексикографический порядок на строках.

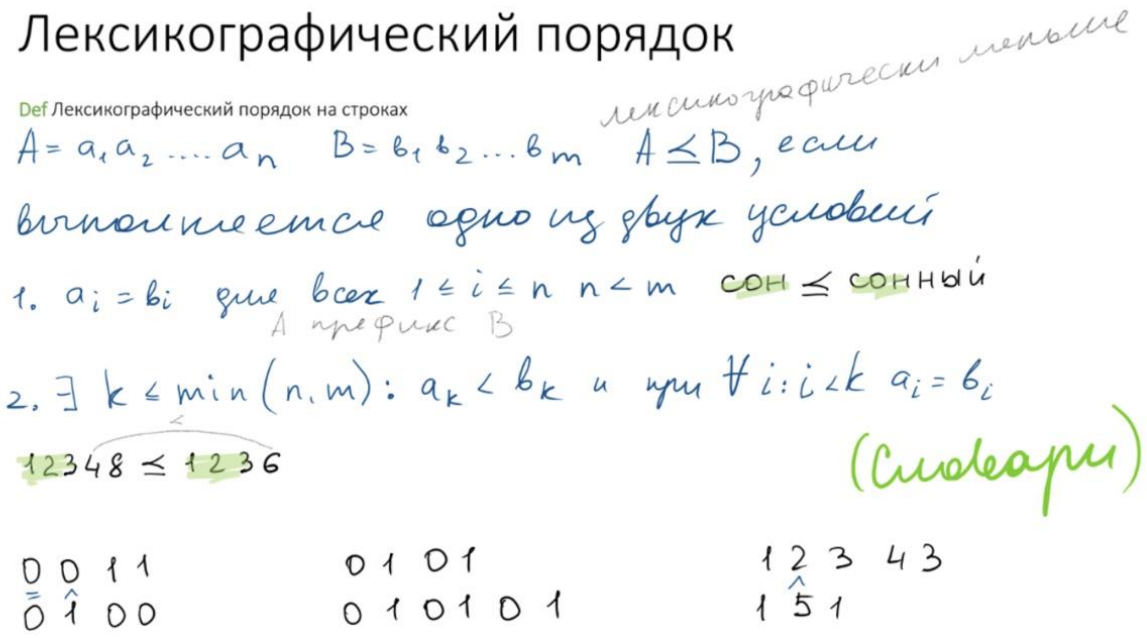
Лексикографический - название лексикографический порядок получил по аналогии с сортировкой.

Wiki:

Слова α предшествует слову β ($\alpha < \beta$), если:

- либо первые m символов этих слов совпадают, а m+1-й символ слова α меньше m+1-го символа слова β (например, АБАК < АБРАКАДАБРА, так как первые две буквы у этих слов совпадают, а третья буква у первого слова меньше, чем у второго);
- либо слово α является началом слова β (например, СОН < СОННЫЙ).

Presentation:



17. Формирование следующей в лексикографическом порядке перестановки.

Простыми словами:

- проходимся по перестановке справа налево, пока не найдем возрастающую ПАРУ.
- меняем местами МЕНЬШИЙ (левый) элемент пары с САМЫМ ПРАВЫМ относительно него БОЛЬШИМ (минимальный элемент, больший нашего и стоящим правее) элементом перестановки.
- зеркалим все, что было справа от позиции, где только что стоял меньший элемент пары.

С нерка:

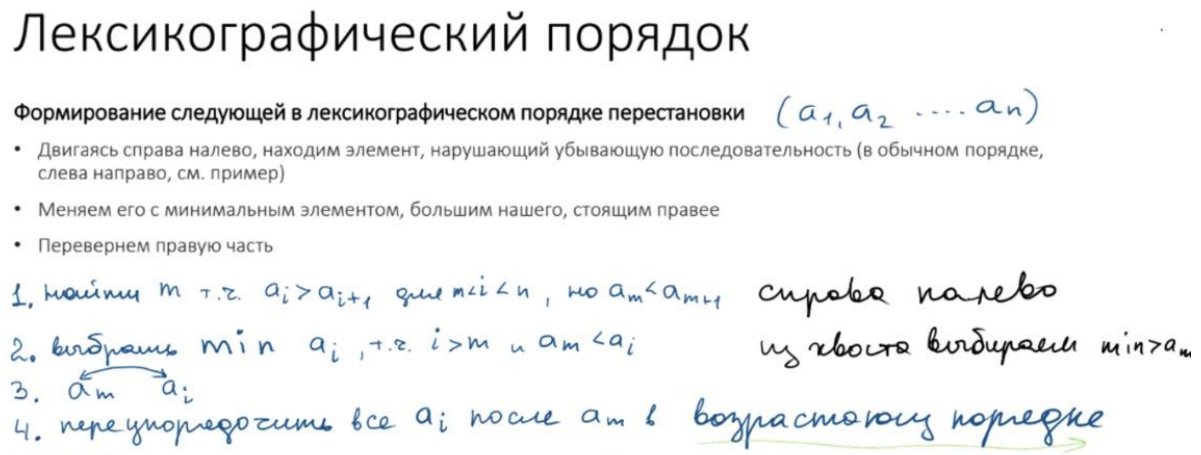
алгоритм для генерации следующей лексикографической перестановки

- Двигаясь справа налево, находим элемент, нарушающий убывающую последовательность (в обычном порядке, слева направо, см. пример)
- Меняем его с минимальным элементом, большим нашего, стоящим правее
- Перевернем правую часть

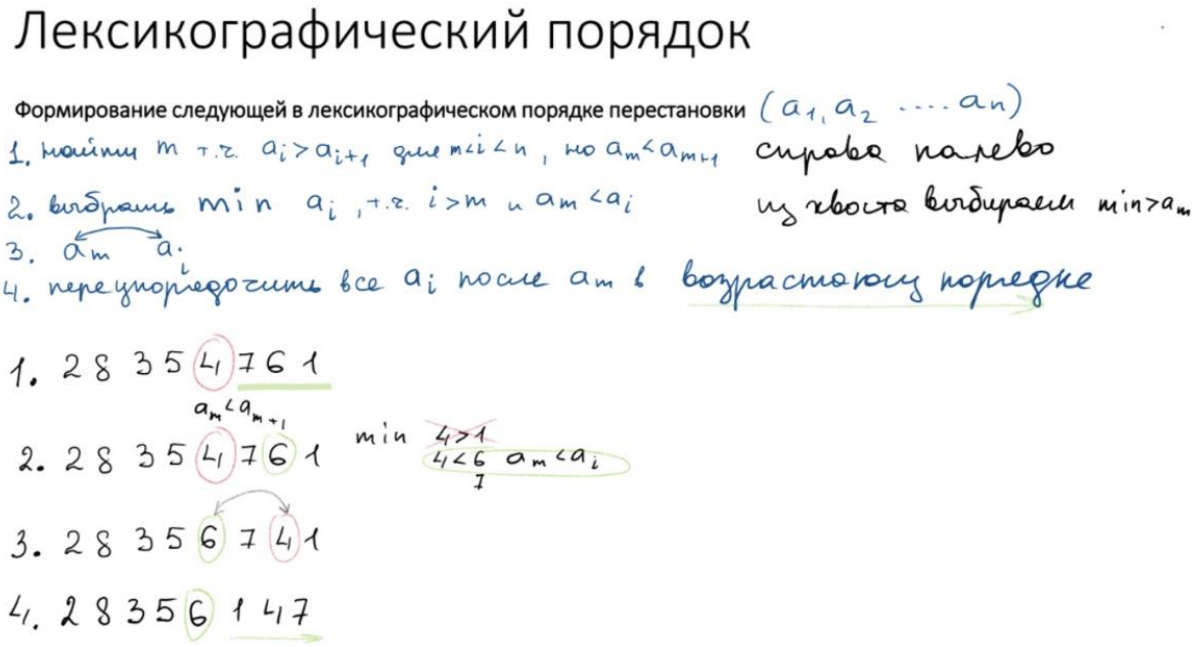
Пример работы

1	3	2	5	4	исходная перестановка
		^			находим элемент, нарушающий убывающую последовательность
				^	минимальный элемент больше нашего
1	3	4	5	2	меняем их местами
1	3	4	2	5	разворачивам правую часть
1	**3**	**4**	**2**	**5**	следующая перестановка

Presentation:



1	3	2	5	4	исходная перестановка
		^			находим элемент, нарушающий убывающую последовательность
				^	минимальный элемент больше нашего
1	3	4	5	2	меняем их местами
1	3	4	2	5	разворачивам правую часть
1	**3**	**4**	**2**	**5**	следующая перестановка



18. Формирование следующего в лексикографическом порядке сочетания той же длины.

Простыми словами:

- приписываем в конец сочетания число равное увеличенному на 1 максимальному числу данного алфавита.
- ищем ПАРУ элементов, разница между которыми >1.
- инкрементируем МЕНЬШИЙ (увеличиваем значение на 1) элемент пары.
- убираем приписанное ранее число в конце.(пункт 1)

- заменяем все, что было справа от этого элемента(приписанное число из п. 1 не учитывается) на минимальный возможный хвост.

С нерка:

алгоритм для генерации следующего лексикографического сочетания

- Добавим в конец массива с сочетанием $N + 1$ – максимальный элемент.
- Пойдём справа налево. Будем искать номер элемента, который отличается от предыдущего на 2 и больше.
- Увеличим найденный элемент на 1, и допишем в конец минимально возможный хвост, если такого элемента нет – на вход было дано последнее сочетание.

Пример работы

1	2	5	6	**7**	Дописываем 7 в конец сочетания.
1	**2**	5	6	**7**	
	^				Находим элемент i, a[i + 1] - a[i] >= 2
1	**3**	5	6	**7**	Увеличиваем его на 1.
1	3	**4**	**5**	**6**	Дописываем минимальный хвост.
1	**3**	**4**	**5**		Следующее сочетание.

Presentation:

Лексикографический порядок

k - сочетания

Формирование следующего в лексикографическом порядке сочетания (той же длины)

- Добавим в конец массива с сочетанием $N + 1$ – максимальный элемент.
- Пойдём справа налево. Будем искать номер элемента, который отличается от предыдущего на 2 и больше.
- Увеличим найденный элемент на 1, и допишем в конец минимально возможный хвост, если такого элемента нет – на вход было дано последнее сочетание.

Пример: 1 2 5 6 след?

1	2	5	6	**7**	Дописываем 7 в конец сочетания.
1	**2**	5	6	**7**	
	^				Находим элемент i, a[i + 1] - a[i] >= 2
1	**3**	5	6	**7**	Увеличиваем его на 1.
1	3	**4**	**5**	**6**	Дописываем минимальный хвост.
1	**3**	**4**	**5**		Следующее сочетание.

1 2 3 6
1 2 4 5
1 2 4 6
1 2 5 6 ~~7~~
1 3 4 5 ~~7~~
1 3 4 6 ~~7~~
1 3 5 6 ~~7~~
1 4 5 6

$S = \{1, 2, 3, 4, 5, 6\}$ $k = 4$

(1) 1 2 3 4

(6 4) 3 4 5 6

Лексикографический порядок

Формирование следующего в лексикографическом порядке сочетания с повторениями (той же длины)

1. *Сортировка* $\{C_1, C_2, \ldots, C_k\}$ *по возрастанию.*
2. $i = k; C_0 = 0.$
3. *Если* $C_i = n$, *то* $i = i - 1$ *и идти к* 3.
4. $C_i = C_i + 1; j = i + 1.$
5. $C_j = C_{j-1}; j = j + 1.$
6. *Если* $j \le k$, *то и идти к* 5.
7. $\{C_1, C_2, \ldots, C_k\}$ – *сочетание с повторениями.*

n - число элементов конечного множ

k - число элементов в сочетании с повторени

$\{C_1, C_2, \ldots C_k\}$ - сочетание с повторениями

$S = \{0 \ldots 9\}$ $n = 10$ $k = 5$

$n = 5$ $k = 3$

1 1 1
1 1 2
1 1 3
1 1 4
1 1 5
1 2 2
1 2 3
1 2 4
1 2 5
1 3 3
1 3 4
1 4 5
1 5 5
2 2 2
2 2 3
2 2 4
2 2 5
2 3 3
2 3 4
2 3 5
2 4 4
2 4 5
2 5 5
3 3 3
3 3 4
. . .
5 5 5

3 9
4 4
0 2 3 4 5
4 6
4 7

19. Теорема об эквивалентности формул задающих сочетания (правило симметрии)

Простыми словами:

сочетания из n по m элементов и из n по n-m элементов совпадают.

$$C(n,m) = C_n^m = \binom{n}{m} = \frac{n!}{m!*(n-m)!} = C(n,n-m) = \binom{n}{n-m} = \frac{n!}{(n-m)!*(n-(n-m))!}$$

Presentation:

Правило симметрии (Symmetry)

Теоремы комбинаторики

Th $0 \le m \le n$ $\quad C(n,m) = C(n, n-m)$

$$\binom{n}{m} = \frac{n!}{(n-m)! \cdot m!}$$

$$\binom{n}{n-m} = \frac{n!}{(n-(n-m))! \cdot (n-m)!} = \frac{n!}{m! \cdot (n-m)!}$$

Пример: Сколько существует трёхэлементных подмножеств множества из 10 элементов, а сколько семиэлементных?

$$\binom{10}{3} = \frac{10!}{(10-3)! \cdot 3!} = \frac{10!}{7! \cdot 3!}$$

$$\binom{10}{7} = \frac{10!}{(10-7)! \cdot 7!} = \frac{10!}{3! \cdot 7!}$$

20. Теорема Вандермонда

Presentation:

Теоремы комбинаторики

Th Вандермонда. Сколькими способами можно составить подмножество из двух множеств.

$n, m, r \ge 0 \quad r \le \min(n, m)$

$$\binom{n+m}{r} = \sum_{k=0}^{r} \binom{n}{k} \cdot \binom{m}{r-k}$$

21. Следствие из теоремы Вандермонда.

Presentation:

Следствие: $n = m = r$

$$\binom{2n}{n} = \sum_{k=0}^{n} \binom{n}{k}^2$$

$$\binom{2n}{n} = \sum \binom{n}{k} \cdot \binom{n}{n-k}$$

по правилу симметрии $= \binom{n}{k}$

22. Биномиальная теорема.

Wiki:

В элементарной алгебре **биномиальная теорема** (или **биномиальное разложение**) описывает алгебраическое разложение по степеням биномиала. Согласно теореме, можно разложить многочлен $(x+y)^n$ в сумму, включающую члены вида ax^by^c, где показатели b и c являются неотрицательными целыми числами с $b+c=n$, а коэффициент a каждого члена является определенным положительным целым числом, зависящим от n и b. Для например, для $n=4$,

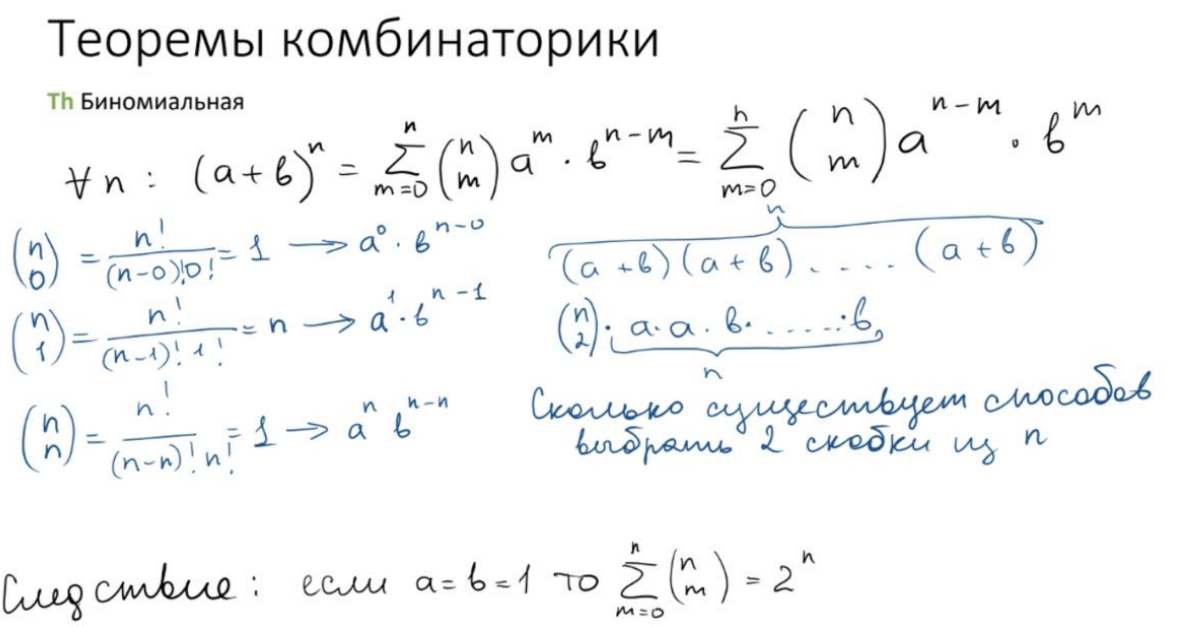
$$(x+y)^4 = x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4.$$

Коэффициент a в терминах ax^by^c известен как биномиальный коэффициент $\binom{n}{b}$ или $\binom{n}{c}$ (оба имеют одинаковое значение). Эти коэффициенты при изменении n и b могут быть расположены так, чтобы образовать треугольник Паскаля. Эти числа также встречаются в комбинаторике, где $\binom{n}{b}$ указывается количество различных комбинаций b элементов, которые могут быть выбраны из n-элементного множества. Поэтому $\binom{n}{b}$ часто произносится как "n выбирают b".

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1

Биномиальный коэффициент $\binom{n}{k}$ отображается как k-я запись в n-й строке треугольника Паскаля (подсчет начинается с 0). Каждая запись является суммой двух приведенных выше.

Presentation:



23. Теорема о сумме сочетаний (следствие из биномиальной).

если $a = b = 1$, то $\sum_{m=0}^{n} \binom{n}{m} = 2^n$

24. Теорема о коэффициентах биномиального треугольника.

$$\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m}$$

Presentation:

Теоремы комбинатории

Th Формула Паскаля (коэффициенты треугольника Паскаля) *Pascal's formula*

$1 \le m \le n$

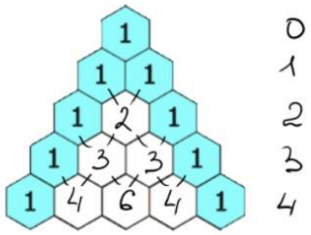
$C(n,m) = C(n-1,m-1) + C(n-1,m)$

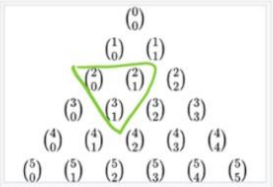
∀ объект $r \in N$ ① r - выбран ② r - не выбран

Алгебраическое доказательство:

$$\frac{(n-1)! \cdot m}{(n-1-(m-1))!\,(m-1)!} + \frac{(n-1)! \cdot (n-m)}{(n-1-m)!\,m!} = \frac{n!}{(n-m)! \cdot m!}$$

$$\binom{3}{1} = \binom{2}{0} + \binom{2}{1}$$




Six rows Pascal's triangle as binomial coefficients

Есть еще комбинаторное док-во

25. Алфавит, символ.

- **символ** — объект, имеющий собственное содержание и уникальную читаемую форму.
- **алфавит** — конечное непустое множество символов.

Presentation:

Формальные языки

Символ (англ. *symbol*) — объект, имеющий собственное содержание и уникальную читаемую форму.
Алфавит (англ. *alphabet*) — конечное непустое множество символов. Условимся обозначать алфавиты символом Σ.

Наиболее часто используются следующие алфавиты.
1. $\Sigma = \{0,1\}$ — бинарный или двоичный алфавит.
2. $\Sigma = \{a, b, \dots, z\}$ — множество строчных букв английского алфавита.
3. Множество ASCII-символов или множество всех печатных ASCII-символов

Слово / Цепочка / Строка (англ. *string*) — это конечная последовательность символов некоторого алфавита.

Длина цепочки (англ. *string length*) — число символов в цепочке. Длину некоторой цепочки ω обычно обозначают $|\omega|$

Пустая цепочка (англ. *empty string*) — цепочка, не содержащая ни одного символа. Эту цепочку, обозначаемую ε, можно рассматривать как цепочку в любом алфавите.

Степень алфавита: Если Σ — некоторый алфавит, то можно выразить множество всех цепочек определенной длины, состоящих из символов данного алфавита, используя знак степени. Определим Σ^k как множество всех цепочек длины k, состоящих из символов алфавита Σ (над алфавитом Σ).

Множество всех цепочек над алфавитом Σ принято обозначать

$$\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k$$

26. Слово / цепочка.

Это конечная последовательность символов некоторого алфавита.

27. Длина цепочки.

Число символов в цепочке (длину некоторой цепочки ω обычно обозначают |ω|).

28. Пустая цепочка.

Цепочка, не содержащая ни одного символа (обозначается за ε).

29. Степень алфавита.

Выражение множества всех цепочек определенной длины, состоящих из символов данного алфавита, используя знак степени (Σ^k — множество всех цепочек длины k, состоящих из символов алфавита Σ (над алфавитом Σ)).

30. Степень языка.

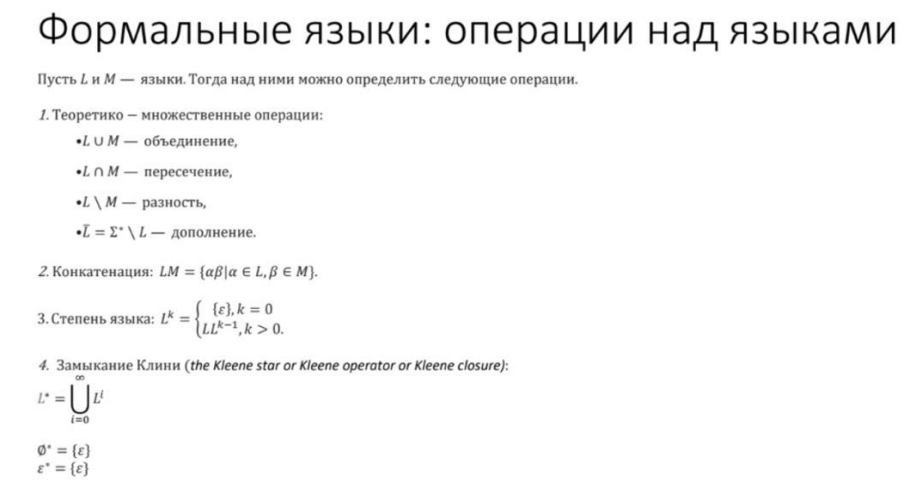
Base:

- **Множество всех цепочек** над алфавитом Σ принято обозначать $\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k$
- **Язык** L над алфавитом Σ — некоторое подмножество Σ^*.
- $\{\varepsilon\}$ — язык, содержащий одну лишь пустую цепочку.

$$L^k = \begin{cases} \{\varepsilon\}, k = 0 \\ LL^{k-1}, k > 0. \end{cases}$$

расшифровка: Пусть L - язык, тогда $L^0 = \{\varepsilon\}$.

Presentation:



Формальные языки: операции над языками

Пусть L и M — языки. Тогда над ними можно определить следующие операции.

1. Теоретико – множественные операции:
   - $L \cup M$ — объединение,
   - $L \cap M$ — пересечение,
   - $L \setminus M$ — разность,
   - $\bar{L} = \Sigma^* \setminus L$ — дополнение.
2. Конкатенация: $LM = \{\alpha\beta | \alpha \in L, \beta \in M\}$.
3. Степень языка: $L^k = \begin{cases} \{\varepsilon\}, k = 0 \\ LL^{k-1}, k > 0. \end{cases}$
4. Замыкание Клини (*the Kleene star or Kleene operator or Kleene closure*):

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

$\emptyset^* = \{\varepsilon\}$
$\varepsilon^* = \{\varepsilon\}$

31. Конкатенация цепочек.

Последовательная запись нескольких цепочек.

Presentation:

Формальные языки

Конкатенация цепочек

Пусть x и y — цепочки. Тогда xy обозначает их конкатенацию (соединение), т. е. цепочку, в которой последовательно записаны цепочки x и y.
Более строго, если
x — цепочка из i символов: $x = a_1a_2 \dots a_i$,
y — цепочка из j символов: $y = b_1b_2 \dots b_j$,
то xy — это цепочка длины $i + j$: $xy = a_1a_2 \dots a_ib_1b_2 \dots b_j$.

Пример:
Пусть $x = 01101$ и $y = 110$. Тогда $xy = 01101110$, а $yx = 11001101$.

Для любой цепочки ω справедливы равенства $\varepsilon\omega = \omega\varepsilon = \omega$

Таким образом, ε является единицей (нейтральным элементом)относительно операции конкантенации, поскольку результат ее конкатенации с любой цепочкой дает ту же самую цепочку.

32. Конкатенация языков.

Язык, состоящий из конкатенаций цепочек данных языков.

Presentation:

Формальные языки

$L\,\emptyset = L$
$\emptyset^* = \{\varepsilon\}$
$\{\varepsilon\}^* = \{\varepsilon\}$

Операции над языками
Пусть L и M — языки. Тогда над ними можно определить следующие операции.

1. Теоретико – множественные операции:
   - $L \cup M$ — объединение,
   - $L \cap M$ — пересечение,
   - $L \setminus M$ — разность,
   - $\overline{L} = \Sigma^* \setminus L$ — дополнение.

2. Конкатенация: $LM = \{\alpha\beta | \alpha \in L, \beta \in M\}$.

$|L \cdot M| \leq |L| \cdot |M|$
$L = \{0, 01\}$
$M = \{0, 10\}$
$L \cdot M = \{010, 0110, 010\} = \{010, 0110, 00\}$

3. Степень языка: $L^k = \begin{cases} \{\varepsilon\}, k = 0 \\ LL^{k-1}, k > 0. \end{cases}$

4. Замыкание Клини (*the* ***Kleene star*** *or* ***Kleene operator*** *or* ***Kleene closure***):

$L^* = \bigcup_{i=0}^{\infty} L^i$ $\quad L = \{0, 1\}$ $\quad L^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000 \dots\}$

33. Замыкание Клини.

Множество всех строк конечной длины´, порождённое элементами множества L.

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Wiki:

Если V — множество строк

то V^* — минимальное надмножество множества V, которое содержит ε (пустую строку) и замкнуто относительно конкатенации. Это также множество всех строк, полученных конкатенацией нуля или более строк из V.

Если V — множество символов

то V^* — множество всех строк из символов из V с добавлением пустой строки.

Звезда Клини [править | править код]

Замыкание Клини множества V есть

$$V^* = \bigcup_{i=0}^{\infty} V^i.$$

То есть это множество всех строк конечной длины, порождённое элементами множества V.

34. Формальный язык.

Presentation:

Формальные языки

$L\,\emptyset = L$

$\emptyset^* = \{\varepsilon\}$

$\{\varepsilon\}^* = \{\varepsilon\}$

Операции над языками

Пусть L и M — языки. Тогда над ними можно определить следующие операции.

1. Теоретико – множественные операции:
   - $L \cup M$ — объединение,
   - $L \cap M$ — пересечение,
   - $L \setminus M$ — разность,
   - $\overline{L} = \Sigma^* \setminus L$ — дополнение.

2. Конкатенация: $LM = \{\alpha\beta | \alpha \in L, \beta \in M\}$.

$|L \cdot M| \leq |L| \cdot |M|$

$L = \{0, 01\}$

$M = \{0, 10\}$

$L \cdot M = \{010, 0110, 00, 010\} = \{010, 0110, 00\}$

3. Степень языка: $L^k = \begin{cases} \{\varepsilon\}, k = 0 \\ LL^{k-1}, k > 0. \end{cases}$

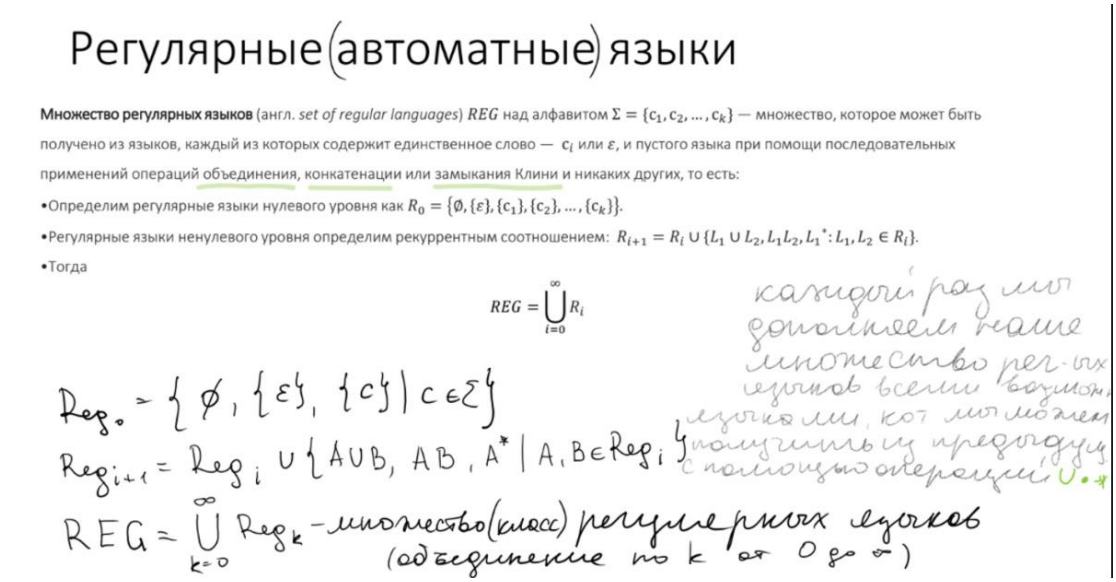
4. Замыкание Клини (*the* ***Kleene star*** *or* ***Kleene operator*** *or* ***Kleene closure***):

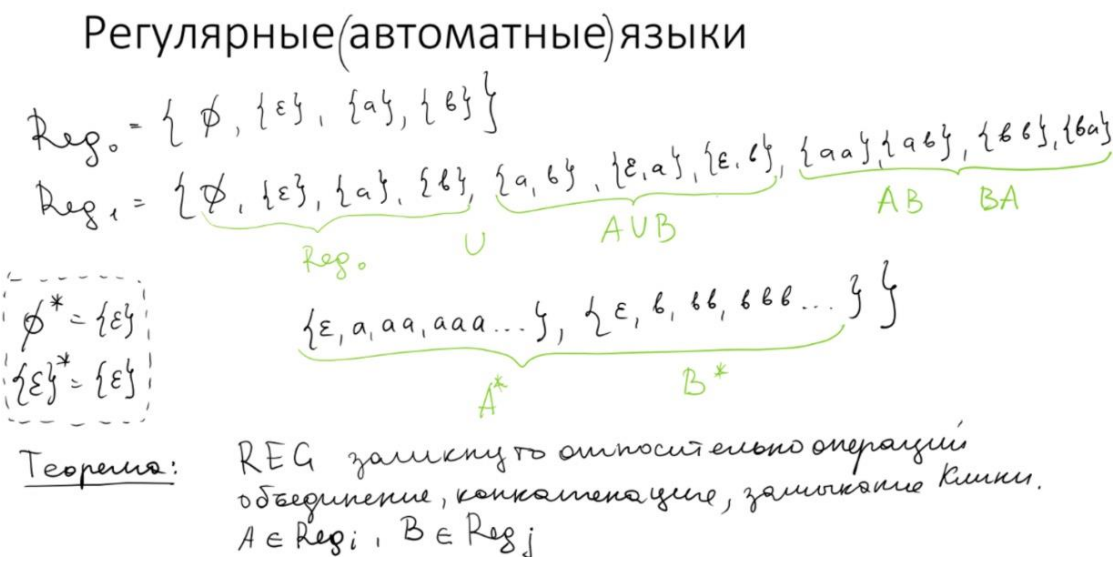
$L^* = \bigcup_{i=0}^{\infty} L^i$

$L = \{0, 1\}$ $L^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000 \ldots\}$

35. Множество регулярных языков.

Presentation:





36. Регулярное выражение.

Способ порождения языка над алфавитом.

По словам Пермякова:

— шаблон, сопоставляемый с искомой строкой слева направо.

Presentation:

Регулярные выражения

Регулярное выражение (англ. *regular expression*) над алфавитом $\Sigma = \{c_1, c_2, \ldots, c_k\}$ — способ порождения языка над Σ. Определяется рекурсивно следующим образом:

- Для любого i слово c_i является регулярным выражением, задающим язык из одного слова c_i.
- ε является регулярным выражением, задающим язык из одной пустой строки, а $\emptyset$ — пустой язык.
- Если α_1 и α_2 являются регулярными выражениями, задающими языки L_1 и L_2 соответственно, то $(\alpha_1)|(\alpha_2)$ — регулярное выражение, задающее $L_1 \cup L_2$.
- Если α_1 и α_2 являются регулярными выражениями, задающими языки L_1 и L_2 соответственно, то $(\alpha_1)(\alpha_2)$ — регулярное выражение, задающее $L_1 L_2$.
- Если α_1 является регулярным выражением, задающим язык L_1, то α_1^* — регулярное выражение, задающее L_1^*.
- Операции указаны в порядке возрастания приоритета, при этом скобки повышают приоритет аналогично арифметическим выражениям.

Регулярные выражения, входящие в современные языки программирования (в частности, PCRE), имеют больше возможностей, чем то, что называется регулярными выражениями в теории формальных языков; в частности, в них есть нумерованные обратные ссылки[ист]. Это позволяет им разбирать строки, описываемые не только регулярными грамматиками, но и более сложными, в частности, контекстно-свободными грамматиками

Академические

Компактное представление для регулярных языков

Регулярные выражения

Язык	Regex	
$\emptyset$	$\emptyset$	– пустое множество
$\{\varepsilon\}$	ε	– пустая строка
$\{c\}$	c	– язык, состоящий из одного символа
A	α	
B	β	
$A \cup B$	$\alpha \mid \beta$	← min
AB	$\alpha\beta$	← средний
A^*	α^*	← max

приоритет

$(01)^* = \{\varepsilon, 01, 0101, 010101 \ldots$

$01^* = \{0, 01, 011, 0111 \ldots$

$01^* | 1 = (0(1)^*)|1 = \{0, 1, 01, 011, 0111 \ldots$

Регулярные выражения, входящие в современные языки программирования (в частности, PCRE), имеют больше возможностей, чем то, что называется регулярными выражениями в теории формальных языков; в частности, в них есть нумерованные обратные ссылки[ист]. Это позволяет им разбирать строки, описываемые не только регулярными грамматиками, но и более сложными, в частности, контекстно-свободными грамматиками

Академические

Регулярные выражения

Пример:

$(a|bc)^* = \{\varepsilon, a, bc, aa, bcbc, abc, bca, aaa, bcbcbc \ldots$

Пример:

Слова, содержат четное число единиц

$0^* | (0^*10^*10^*)^*$

Это все описание регулярных языков через порождение.

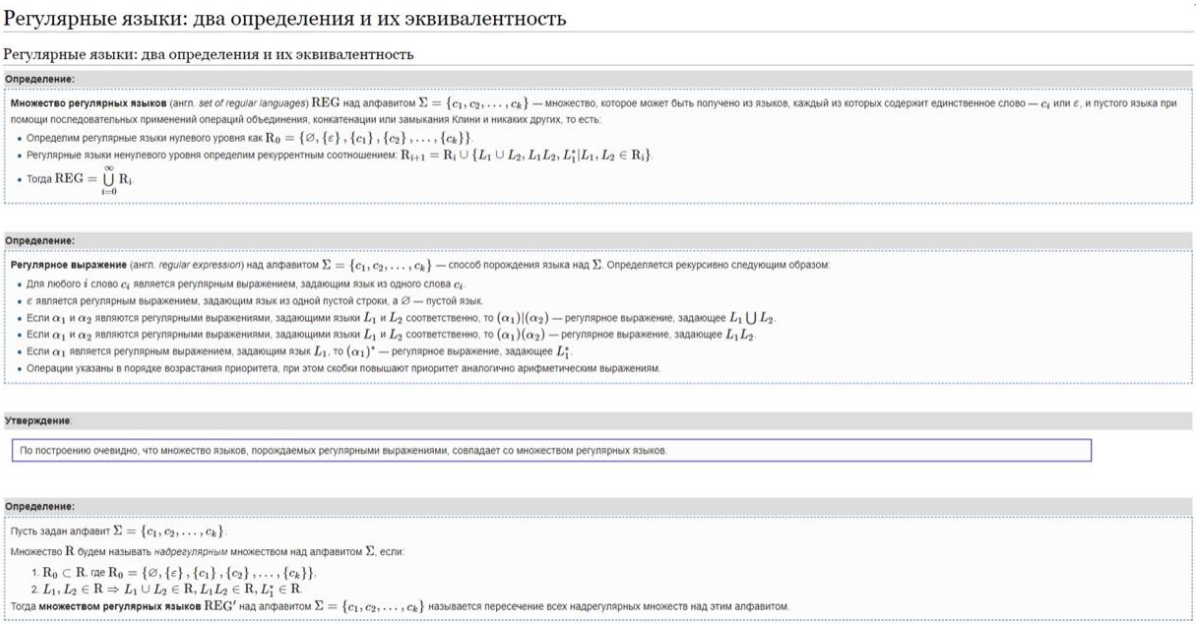
Если рассмотреть через метод распознавания, то

wiki:

Регуля́рный язык (**регуля́рное мно́жество**) в теории формальных языков — множество слов, которое распознает некоторый конечный автомат.

37. Регулярный язык.

Сперка:



Регулярные языки: два определения и их эквивалентность

Регулярные языки: два определения и их эквивалентность

Определение:

Множество регулярных языков (англ. *set of regular languages*) REG над алфавитом $\Sigma = \{c_1, c_2, \ldots, c_k\}$ — множество, которое может быть получено из языков, каждый из которых содержит единственное слово — c_i или ε, и пустого языка при помощи последовательных применений операций объединения, конкатенации или замыкания Клини и никаких других, то есть:

- Определим регулярные языки нулевого уровня как $\mathrm{R}_0 = \{\varnothing, \{\varepsilon\}, \{c_1\}, \{c_2\}, \ldots, \{c_k\}\}$.
- Регулярные языки ненулевого уровня определим рекуррентным соотношением: $\mathrm{R}_{i+1} = \mathrm{R}_i \cup \{L_1 \cup L_2, L_1 L_2, L_1^* | L_1, L_2 \in \mathrm{R}_i\}$.
- Тогда $\mathrm{REG} = \bigcup_{i=0}^{\infty} \mathrm{R}_i$.

Определение:

Регулярное выражение (англ. *regular expression*) над алфавитом $\Sigma = \{c_1, c_2, \ldots, c_k\}$ — способ порождения языка над Σ. Определяется рекурсивно следующим образом:

- Для любого i слово c_i является регулярным выражением, задающим язык из одного слова c_i.
- ε является регулярным выражением, задающим язык из одной пустой строки, а $\varnothing$ — пустой язык.
- Если α_1 и α_2 являются регулярными выражениями, задающими языки L_1 и L_2 соответственно, то $(\alpha_1)|(\alpha_2)$ — регулярное выражение, задающее $L_1 \bigcup L_2$.
- Если α_1 и α_2 являются регулярными выражениями, задающими языки L_1 и L_2 соответственно, то $(\alpha_1)(\alpha_2)$ — регулярное выражение, задающее $L_1 L_2$.
- Если α_1 является регулярным выражением, задающим язык L_1, то $(\alpha_1)^*$ — регулярное выражение, задающее L_1^*.
- Операции указаны в порядке возрастания приоритета, при этом скобки повышают приоритет аналогично арифметическим выражениям.

Утверждение:

По построению очевидно, что множество языков, порождаемых регулярными выражениями, совпадает со множеством регулярных языков.

Определение:

Пусть задан алфавит $\Sigma = \{c_1, c_2, \ldots, c_k\}$.

Множество R будем называть *надрегулярным* множеством над алфавитом Σ, если:

1. $\mathrm{R}_0 \subset \mathrm{R}$, где $\mathrm{R}_0 = \{\varnothing, \{\varepsilon\}, \{c_1\}, \{c_2\}, \ldots, \{c_k\}\}$.
2. $L_1, L_2 \in \mathrm{R} \Rightarrow L_1 \cup L_2 \in \mathrm{R}, L_1 L_2 \in \mathrm{R}, L_1^* \in \mathrm{R}$.

Тогда **множеством регулярных языков** REG' над алфавитом $\Sigma = \{c_1, c_2, \ldots, c_k\}$ называется пересечение всех надрегулярных множеств над этим алфавитом.

38. Детерминированный конечный автомат.

- Набор из пяти элементов (Σ, Q, $q_0 \in Q$, $F \subseteq Q$, $\delta : Q \times \Sigma \square Q$), где:
  Σ – алфавит.
  Q – множество состояний.
  q_0 – начальное (стартовое) состояние (их может быть несколько).
  F – множество доступных состояний.
  δ – функция перехода.

39. Недетерминированный конечный автомат.

- Набор из пяти элементов (Σ, Q, $q_0 \in Q$, $F \subseteq Q$, $\delta : Q \times \Sigma \square 2^Q$), где:
  Σ – алфавит.
  Q – множество состояний.
  q_0 – начальное (стартовое) состояние (их может быть несколько).
  F – множество доступных состояний.
  δ – функция перехода.

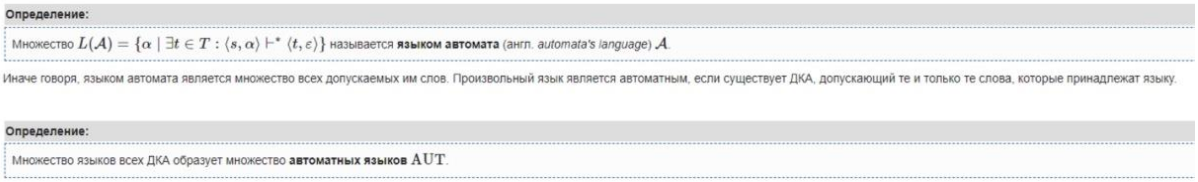
единственное отличие НКА от ДКА — существование нескольких переходов по одному символу из одного состояния.

40. Язык автомата.

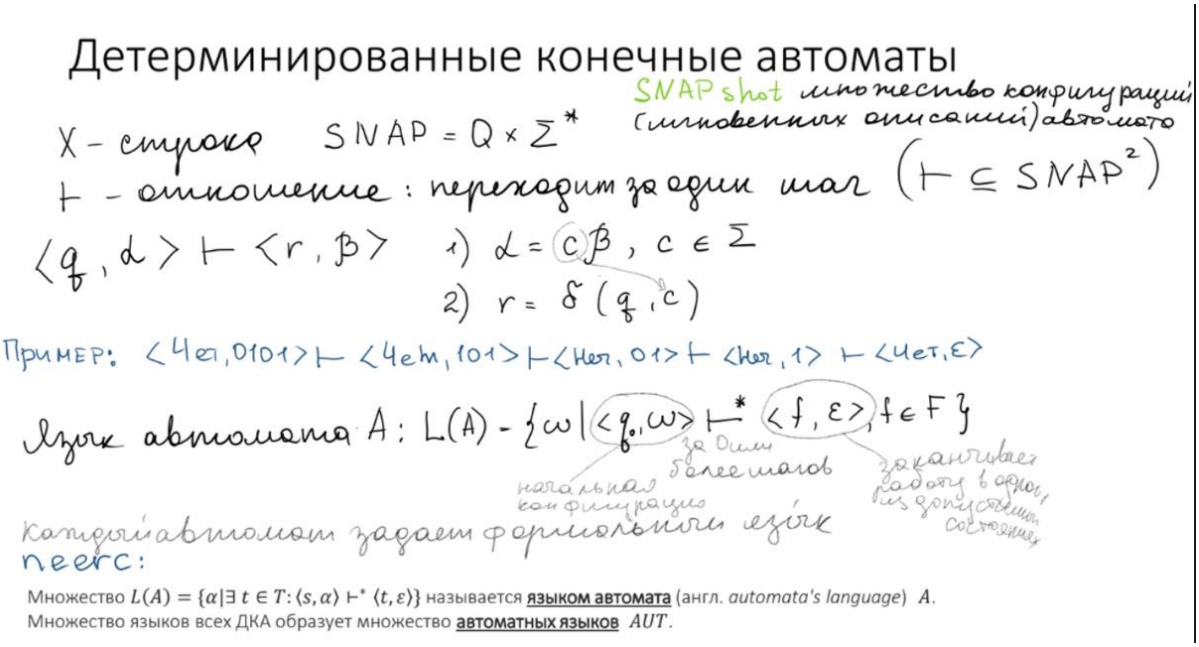
- множество всех допускаемых автоматом слов.
- произвольный язык является автоматным, если существует ДКА, допускающий те и только те слова, которые принадлежат языку.

- множество L(A)={α∃t∈T:⟨s,α⟩⊢^∗⟨t,ε⟩} называется **языком автомата.**

С нерка:



Presentation:



41. Теорема Клини.

REG = AUT

Классы регулярных и автоматных языков совпадают. иными словами, все, что можно задать с помощью регулярных выражений, можно задать с помощью конечных автоматов и наоборот.

42. Событие (достоверное, невозможное, случайное).

Примеры событий: случайное (Я выиграю в лотерее), невозможное (У собаки выросли крылья), достоверное (После среды наступит четверг).

43. Несовместные события.

События, возникающие в одно и то же время в определённых обстоятельствах, называют совместными, а не возникающие вместе — несовместными. Пример: такие события, как «начался полдень» и «посыпался град», будут совместными, а события «началось утро» и «началась ночь» — несовместными.

44. Полная группа событий.

Полная группа — группа, которая изоморфна своей группе автоморфизмов. Полная группа — так называют делимые неабелевы группы. Полная группа событий — в теории вероятностей система случайных событий такая, что непременно произойдет одно и только одно из них.

45. Вероятность события.

Вероятность события A — отношение количества благоприятствующих событию A исходов к общему количеству всех равновозможных исходов. m — количество исходов испытания, в которых наступает событие A, n — количество всех равновозможных исходов.

46. Закон сложения двух совместных событий A и B.

Правило суммы для совместных событий: чтобы найти вероятность объединения двух совместных событий, нужно из суммы их вероятностей вычесть вероятность их пересечения. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.

47. Условная вероятность события A относительно события B.

$P(A|B)$ – вероятность события A при условии, что событие B произошло. Если события A и B несовместны, то они не имеют общих точек и, следовательно, событие AB является невозможным: $P(AB) = 0$. $P(A|B) = P(A)$.

48. Независимые события A и B.

События A и B являются независимыми, если вероятность наступления одного из них не изменяется при наступлении другого. Событие A является зависимым от события B, если наступление события B изменяет вероятность наступления события A. Пример: бросают игральный кубик.

49. Геометрическое определение вероятности.

Вероятность наступления некоторого события A в испытании равна $P(A) = g/G$, где G – геометрическая мера, выражающая общее число всех равновозможных исходов данного испытания, а g – мера, выражающая количество благоприятствующих событию A исходов.

50. Формула полной вероятности.

позволяет вычислить вероятность интересующего события через вероятности его произойти при выполнении гипотез с заданной вероятностью. Формула полной вероятности требуется, когда необходимо узнать вероятность совершения некоторого события, если его совершение зависит от нескольких условий. Например, можно узнать вероятность принятия законопроекта, зная, с какой вероятностью его примет каждая партия. Ещё формула применяется в задачах о нахождении среднего качества продукции, выпускаемой цехом.

51. Формула Байеса.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

- P(A) – изначальная вероятность предложения A.
- P(B) – абсолютная вероятность наступления события B.
- P(A|B) – вероятность предложения A при наступлении события B.
- P(B|A) – вероятность наступления события B, если предложение A истинно.

52. Функция распределения.

Функцией распределения случайной величины ξ называется функция F(x), выражающая вероятность того, что ξ примет значение, меньшее чем x: F(x) = P(ξ<x).

53. Распределение Бернулли (какие в нем есть события, формула вероятности события).

Описывает ситуации, где "испытание" имеет результат "успех" либо "неуспех".

С нерка:

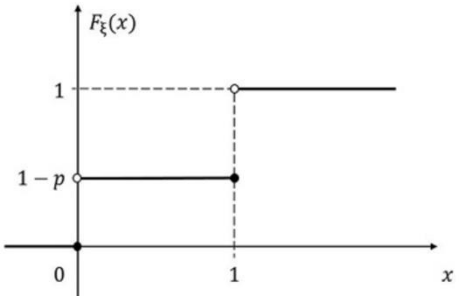
Распределение Бернулли

Определение:

Распределение Бернулли (англ. *Bernoulli distribution*) — описывает ситуации, где "испытание" имеет результат "успех" либо "неуспех".

Случайная величина ξ с таким распределением равна числу успехов в одном испытании схемы Бернулли с вероятностью p успеха : ни одного успеха или один успех. Функция распределения ξ имеет вид

$$F_{\xi}(x) = P(\xi < x) \begin{cases} 0, & x \leqslant 0 \\ 1-p, & 0 < x \leqslant 1 \\ 1, & x > 1 \end{cases}$$



54. Биномиальное распределение (какие в нем есть события, формула вероятности события).

С нерка:

Биномиальное распределение

Определение:

Случайная величина ξ имеет **биномиальное распределение** (англ. *binomial distribution*) с параметрами $n \in \mathbb{N}$ и $p \in (0,1)$ и пишут: $\xi \in \mathbb{B}_{n,p}$ если ξ принимает значения $k = 0, 1, \ldots, n$ с вероятностями $P(\xi = k) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}$.

Случайная величина с таким распределением имеет смысл числа успехов в n испытаниях схемы Бернулли с вероятностью успеха p.

Таблица распределения ξ имеет вид

ξ	0	1	…	k	…	n
P	$(1-p)^n$	$n \cdot p \cdot (1-p)^{n-1}$	…	$\binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}$	…	p^n

Формула Бернулли

Обозначим через ν_n число успехов, случившихся в n испытаниях схемы Бернулли. Эта случайная величина может принимать целые значения от 0 до n в зависимости от результатов испытаний. Например, если все n испытаний завершились неудачей, то величина ν_n равна нулю.