Оглавление

1. Математика

1.1. Математический анализ

Предел

Предел последовательности

Число А будем называть **пределом последовательности** $\{x_n\}_{n=1}^{n=\infty}$, если для любого $\varepsilon > 0$ можно найти номер $n_0 = n_0\,(\varepsilon)$ (зависящий от ε), начиная с которого все члены последовательности будут удовлетворять неравенству $|x_n - A| < \varepsilon$.

Обозначается: $\lim_{n\to\infty} x_n = A$

$\forall\varepsilon > 0,\ \exists n_0 = n_0(\varepsilon),\ \forall n > n_0 : |x_n - A| < \varepsilon$

Предел функции

Определение предела функции по Коши

Число А называется **пределом функции** $f(x)$ при x, стремящемся к x_0 (или в точке x_0), если для любого $\varepsilon > 0$ можно найти число $\delta = \delta(\varepsilon) > 0$ так, что для всех значений $x \in D(f)$, для которых выполнено неравенство $0 < |x - x_0| < \delta$, справедливо неравенство $|f(x) - A| < \varepsilon$.

Обозначается: $\lim_{x\to x_0} f(x) = A$

$\forall\varepsilon > 0,\ \exists\delta = \delta(\varepsilon) > 0 : \forall x\ 0 < |x - x_0| < \delta \Rightarrow |f(x) - A| < \varepsilon$

Определение предела функции по Гейне

Число А называется **пределом функции** $f(x)$ при x, стремящемся к x_0 (или в точке x_0), если для любой последовательности $\{x_n\}$ точек, взятых из области определения функции, сходящейся к x_0, но не содержащей x_0 в качестве одного из своих элементов, последовательность значений функции $f(x_n)$ будет стремиться к числу А.

Обозначения O() и o()

Пусть $f(x)$ и $g(x)$ — две функции, определенные в некоторой проколотой окрестности точки x_0, причем в этой окрестности g не обращается в ноль. Говорят, что:

- f является «O» большим от g при $x \to x_0$, если существует такая константа $C > 0$, что для всех x из некоторой окрестности точки x_0 имеет место неравенство: $|f(x)| \leq C|g(x)|$;

- f является «o» малым от g при $x \to x_0$, если для любого $\varepsilon > 0$ найдется такая проколотая окрестность U'_{x_0} точки x_0, что для всех $x \in U'_{x_0}$ имеет место неравенство: $|f(x)| < \varepsilon|g(x)|$.

Иначе говоря, в первом случае отношение $\frac{|f|}{|g|} \leq C$ в окрестности точки x_0(то есть ограничено сверху), а во втором оно стремится к нулю при $x \to x_0$.

Запись $x^2 = o(x)$ означает, что x^2 при $x \to 0$ является бесконечно малой функцией более высокого порядка, по сравнению с функцией x.

Доказательство и применение асимптотических оценок, при необходимости переформулировка в «терминах эпсилон и дельта»

Если $\lim_{x\to a} \frac{f(x)}{g(x)} = 0$, то говорят, что $f(x) = o(g(x))$ при $x \to a$.

Например, $x^2 = o(x)$ при $x \to 0$, поскольку $\lim_{x\to 0} \frac{x^2}{x} = \lim_{x\to 0} x = 0$.

Если предел отношения $\frac{|f(x)|}{|g(x)|}$ при $x \to a$ конечен, то $f(x) = O(g(x))$ при $x \to a$.

Например, $x+x^2 = O(x)$ при $x \to 0$, поскольку $\lim_{x\to 0} \frac{x+x^2}{x} = \lim_{x\to 0} 1 + x = 1$.

Непрерывность

Непрерывность в точке:

Определение 1: Пусть $x_0 \in D(f)$ - предельная точка области определения функции $f(x)$. (Предельная точка множества — это такая точка, любая проколотая окрестность которой пересекается с этим множеством.) Будем говорить, что функция $f(x)$ **непрерывна** в точке x_0, если $\lim_{x\to x_0} f(x) = f(x_0)$.

Если точка x_0 является предельной точкой области $D(f)$, но функция не является непрерывной в этой точке, то точка x_0 называется **точкой разрыва** функции $f(x)$.

Определение 2: Функция $f(x)$ **непрерывна** в точке x_0, если $\lim_{x\to x_0^{-0}} f(x) = \lim_{x\to x_0^{+0}} f(x) = f(x_0)$.

Если односторонние пределы в точке x_0 существуют и равны между собой, но функция в этой точке не определена или $f(x_0) \neq \lim_{x\to x_0^{-0}} f(x) =$

$\lim_{x\to x_0^{+0}} f(x)$, то точка x_0 называется **точкой устранимого разрыва**.

Если существуют конечные односторонние пределы, но они не равны между собой, то точка x_0, называется **точкой разрыва первого рода**.

Если в точке x_0 хотя бы один конечный односторонний предел не существует или существует и бесконечен, то эта точка называется **точкой разрыва второго рода**.

Критерий непрерывности функции в точке:

Функция $f(x)$ будет непрерывной в точке x_0 тогда и только тогда, когда ее приращение в этой точке будет стремиться к нулю, если приращение аргумента стремится к нулю.

Если $\triangle x \to 0$, то $\triangle f(x_0) \to 0$.

Непрерывность на множестве:

Определение: Будем говорить, что функция $f(x)$ непрерывна на множестве, если она непрерывна в каждой точке этого множества.

Первая теорема Вейерштрасса: Функция, непрерывная на отрезке, ограничена.

Вторая теорема Вейерштрасса: Если функция непрерывна на отрезке, то на этом отрезке она достигает своих наибольшего и наименьшего значений.

Первая теорема Коши о промежуточном значении непрерывной на отрезке функции: Пусть функция $f(x)$ непрерывна на отрезке $[a, b]$ и на концах этого отрезка принимает значения разных знаков. Тогда внутри отрезка найдется, по крайней мере,одна точка, в которой $f(x) = 0$.

Равномерная непрерывность:

Числовая функция вещественного переменного $f\colon M \subset \mathbb{R} \to \mathbb{R}$ равномерно непрерывна, если: $\forall \varepsilon > 0,\ \exists \delta = \delta(\varepsilon) > 0 : \forall x_1, x_2 \in M\ (|x_1 - x_2| < \delta) \Rightarrow (|f(x_1) - f(x_2)| < \varepsilon)$.

Производная

Пусть функция $y = f(x)$ определена в некоторой окрестности точки x_0. Допустим, что существует предел отношения приращения функции в этой точке к вызвавшему его приращению аргумента, когда последнее стремится к нулю: $\lim_{\triangle x\to 0}\frac{f(x_0+\triangle x)-f(x_0)}{\triangle x}$.Тогда этот предел называется **производной** функции в точке x_0.

Т.о. $f'(x_0)=\lim_{\triangle x\to 0}\frac{f(x_0+\triangle x)-f(x_0)}{\triangle x}=\lim_{\triangle x\to 0}\frac{\triangle f(x_0)}{\triangle x}$.

Первообразная

Первообразной для данной функции $f(x)$ называют такую функцию $F(x)$, производная которой равна f (на всей области определения f),), то есть $F'(x)=f(x)$.

Дифференциал

Дифференциал функции одной переменной:

Функция f (x) называется **дифференцируемой** в точке x_0 , если существует число A такое, что $\triangle f(x_0)=A\triangle x+o(\triangle x)$ при $\triangle x\to 0$.

Допустим, что функция $f(x)$ дифференцируема в точке x_0. Тогда выражение $f'(x_0)\triangle x$ будем называть **дифференциалом** этой функции в точке x_0 и обозначать $df(x_0)$ или df.

Дифференциал функции многих переменных:

Пусть есть функция $f(x_1...x_n)$, дифференцируемая в точке $a(x_1...x_n)$, тогда её дифференциалом будет: $df=\frac{\partial f}{\partial x_1}dx_1+\frac{\partial f}{\partial x_2}dx_2+...+\frac{\partial f}{\partial x_n}dx_n$, где $dx_i=\triangle x_i$.

Нахождение экстремума функции от одной и от многих переменных

Нахождение экстремума функции от одной переменной:

Точка x_0 называется **точкой локального максимума(минимума)** функции $f(x)$, если существует такая окрестность этой точки, что для всех x из этой окрестности выполняется неравенство: $f(x)\le f(x_0)$ $(f(x)\ge f(x_0))$.

Необходимое условие экстремума: Если функция $y=f(x)$ имеет экстремум в точке x_0, то ее производная $f'(x_0)$ либо равна нулю, либо не существует.

Первое достаточное условие экстремума: Пусть для функции $y=f(x)$ выполнены следующие условия:

- функция непрерывна в окрестности точки x_0;

- $f'(x_0) = 0$ или $f'(x_0)$ не существует;
- производная $f'(x)$ при переходе через точку x_0 меняет свой знак.

Тогда в точке $x = x_0$ функция $y = f(x)$ имеет экстремум, причем это **минимум**, если при переходе через точку x_0 производная меняет свой знак с минуса на плюс; **максимум**, если при переходе через точку x_0 производная меняет свой знак с плюса на минус.

Второе достаточное условие экстремума: Пусть для функции $y = f(x)$ выполнены следующие условия:

- она непрерывна в окрестности точки x_0;
- $f'(x_0) = 0$;
- $f''(x_0) \neq 0$.

Тогда в точке x_0 достигается экстремум, причем, если $f''(x_0) > 0$, то в точке $x = x_0$ функция $y = f(x)$ имеет **минимум**; если $f''(x_0) < 0$, то в точке $x = x_0$ функция $y = f(x)$ достигает **максимум**.

Нахождение экстремума функции от многих переменных:

Пусть функция $f(x_1...x_n)$ определена на множестве $E \in R^n$ и точка $x^0 \in E$. Точка x^0 называется **точкой локального минимума (максимума)** функции $f(x_1...x_n)$ если$\exists U(x^0) : \forall x \in U(x^0) \cap E : f(x) \geq f(x^0)(f(x) \leq f(x^0))$.

Необходимое условие экстремума: Пусть функция $f(x_1...x_n)$ определена в $U(x^0)$ и имеет локальный экстремум в точке x_0. Если $\exists \frac{\partial f}{\partial x_k}(x^0)$, $1 \leq k \leq n$, то $\frac{\partial f}{\partial x_k}(x^0) = 0 \; \forall k = 1...n$.

Достаточное условие экстремума: Пусть функция $f(x_1...x_n)$ определена в $U(x^0)$ и имеет в этой окрестности непрерывные частные производные второго порядка. Пусть $df(x^0) = 0$. Если $d^2f(x^0)$ является знакоопределенной квадратичной формой, тогда x^0 - **точка локального экстремума**, причем если $d^2f(x^0) > 0$, то x^0 - **локальный минимум**, а если $d^2f(x^0) < 0$, то x^0 - **локальный максимум**.

Формула Тейлора

Если функция $f(x)$ имеет $n+1$ производную на отрезке с концами a и x, то для произвольного положительного числа p найдётся точка ξ, лежащая между a и x, такая, что (или пусть действительная функция f определена в некоторой окрестности точки a): $f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k + \left(\frac{x-a}{x-\xi}\right)^p \frac{(x-\xi)^{n+1}}{n!p} f^{(n+1)}(\xi)$.

1.2. Дискретная математика и математическая логика

Отображения и отношения и их свойства

1

Транзитивное замыкание отношения

1

Эквивалентность

1

Отношения порядка

1

Логика высказываний

1

Кванторы

1

Метод математической индукции

1

Основные понятия теории графов

Ориентированные графы

Ориентированным графом G называется пара $G = (V, E)$, где V — множество вершин, а $E \subset V \times V$ — множество рёбер.

Ребром ориентированного графа называют упорядоченную пару вершин $(v, u) \in E$.

В графе ребро, концы которого совпадают, то есть $e = (v, v)$, называется **петлей**.

Два ребра, имеющие общую концевую вершину, то есть $e_1 = (v, u_1)$ и $e_2 = (v, u_2)$, называются **смежными**.

Если имеется ребро $(v, u) \in E$, то говорят:

- v - **предок** u;
- u и v — **смежные**;
- Вершина u **инцидентна** ребру (v, u);
- Вершина v **инцидентна** ребру (v, u).

Кратные рёбра - это два и более рёбер, инцидентных одним и тем же двум вершинам.

Неориентированные графы

Неориентированным графом G называется пара $G = (V, E)$, где V — множество вершин, а $E \subset \{\{v, u\} : v, u \in V\}$ — множество рёбер.

Ребром в неориентированном графе называют неупорядоченную пару вершин $\{v, u\} \in E$.

Простым графом G называется граф, в котором нет петель и кратных рёбер.

Степенью вершины $degv_i$ в неориентированном графе называют число рёбер, инцидентных v_i.

Изолированной вершиной в неориентированном графе называют вершину степени 0.

Часто используемые графы

Полный граф — граф, в котором каждая пара различных вершин смежна.

Регулярный граф — граф, степени всех вершин которого равны, то есть каждая вершина имеет одинаковое количество соседей.

Планáрный граф — граф, который может быть изображён на плоскости без пересечения рёбер. Иначе говоря, граф планарен, если он изоморфен некоторому плоскому графу, то есть графу, изображённому на плоскости так, что его вершины — это точки плоскости, а рёбра — непересекающиеся кривые на ней.

Ещё полезные определения:

Маршрут — чередующаяся последовательность вершин и рёбер $v_0, e_1, v_1, e_2, v_2, ..., e_k, v_k$, в которой любые два соседних элемента инцидентны. Если $v_0 = v_k$, то маршрут замкнут, иначе открыт.

Путь — последовательность рёбер (в неориентированном графе) и/или дуг (в ориентированном графе), такая, что конец одной дуги (ребра) является началом другой дуги (ребра).

Простой (вершинно-простой) путь — путь, в котором каждая из вершин графа встречается не более одного раза.

Рёберно-простой путь — путь, в котором каждое из рёбер графа встречается не более одного раза.

Цикл - замкнутый маршрут, все рёбра которого различны.

Эйлеровым путем в графе называется путь, который проходит по каждому ребру, причем ровно один раз.

Эйлеров цикл — замкнутый эйлеров путь.

Граф называется **эйлеровым**, если он содержит эйлеров цикл.

Гамильтоновым путём называется простой путь, проходящий через каждую вершину графа ровно один раз.(и дальше по аналогии со всем эйлеровым)

Лемма о рукопожатиях

Неориентированный граф

Лемма: Сумма степеней всех вершин графа — чётное число, равное удвоенному числу рёбер: $\sum_{v \in V(G)} degv = 2 * |E(G)|$

Следствие: Число рёбер в полном графе $\frac{n*(n-1)}{2}$.

Ориентированный граф

Лемма: Сумма входящих и исходящих степеней всех вершин ориентированного графа — чётное число, равное удвоенному числу рёбер: $\sum_{v \in V(G)} deg^- v + \sum_{v \in V(G)} deg^+ v = 2 * |E(G)|$

Критерий двудольности

Двудольный граф — граф, множество вершин которого можно разбить на две части таким образом, что каждое ребро графа соединяет какую-то вершину из одной части с какой-то вершиной другой части, то есть не существует ребра, соединяющего две вершины из одной и той же части.

Двудольный граф с n вершинами в одной доле и m во второй обозначается $K_{n,m}$.

Критерий двудольности Граф является **двудольным** тогда и только тогда, когда он содержит более одной вершины и все его циклы имеют четную длину.

Оценки числа ребер

- Число рёбер в **полном графе** $\frac{n*(n-1)}{2}$.
- Максимальное число ребер на n вершинах можно построить именно в случае, когда граф связный.
- Любое дерево с n вершинами содержит $n-1$ ребро.
- Не забываем про лемму о рукопожатиях.
- Если речь идет о двудольных графах, полезно помнить, что число ребер в полном двудольном графе с n_1 и n_2 вершинами в соответствующих долях равно $n_1 * n_2$.

Характеризация деревьев

Дерево — связный ациклический граф.Связность означает наличие путей между любой парой вершин, ацикличность — отсутствие циклов и то, что между парами вершин имеется только по одному пути.

Лес — граф, являющийся набором непересекающихся деревьев.

Остовное дерево — ациклический связный подграф данного связного неориентированного графа, в который входят все его вершины.

N-арные деревья:

- N-арное дерево (неориентированное) — это дерево (обычное, неориентированное), в котором степени вершин не превосходят $N+1$.

- N-арное дерево ориентированное) — это ориентированное дерево, в котором исходящие степени вершин (число исходящих рёбер) не превосходят N.

1.3. Алгебра и теория чисел

Группы

Непустое множество G с заданной на нём бинарной операцией $* : \mathrm{G}\times\mathrm{G} \to \mathrm{G}$ называется группой $(\mathrm{G}, *)$, если выполнены следующие аксиомы:

1. ассоциативность:
   $\forall(a, b, c \in G)\colon (a * b) * c = a * (b * c)$;
2. наличие нейтрального элемента:
   $\exists e \in G \quad \forall a \in G\colon (e * a = a * e = a)$;
3. наличие обратного элемента:
   $\forall a \in G \quad \exists a^{-1} \in G\colon (a * a^{-1} = a^{-1} * a = e)$.

Поля

Множество F с введёнными на нём алгебраическими операциями сложения $+$ и умножения $*$ ($+\colon F \times F \to F, \quad *\colon F \times F \to F$, т. е. $\forall a, b \in F \quad (a+b) \in F,\ a * b \in F$) называется полем $\langle F, +, *\rangle$, если выполнены следующие аксиомы:

1. Коммутативность сложения:
   $\forall a, b \in F \quad a + b = b + a$.
2. Ассоциативность сложения:
   $\forall a, b, c \in F \quad (a + b) + c = a + (b + c)$.
3. Существование нулевого элемента:
   $\exists 0 \in F\colon \forall a \in F \quad a + 0 = 0 + a = a$.
4. Существование противоположного элемента:
   $\forall a \in F\ \exists(-a) \in F\colon a + (-a) = 0$.
5. Коммутативность умножения:
   $\forall a, b \in F \quad a * b = b * a$.

6. Ассоциативность умножения:
   $\forall a, b, c \in F \quad (a * b) * c = a * (b * c)$.
7. Существование единичного элемента:
   $\exists e \in F \setminus \{0\}\colon \forall a \in F \quad a * e = a$.
8. Существование обратного элемента для ненулевых элементов:
   $\forall a \in F\colon a \neq 0)\ \exists a^{-1} \in F\colon a * a^{-1} = e$.
9. Дистрибутивность умножения относительно сложения:
   $\forall a, b, c \in F \quad (a + b) * c = (a * c) + (b * c)$.

Кольца

Множество R, на котором заданы две бинарные операции: $+$ и $*$ (называемые сложение и умножение), со следующими свойствами, выполняющимися для любых $a, b, c \in R$:

1. Коммутативность сложения:
   $a + b = b + a$.
2. Ассоциативность сложения:
   $(a + b) + c = a + (b + c)$.
3. Существование нулевого элемента:
   $\exists 0 \in R\colon a + 0 = 0 + a = a$.
4. Существование противоположного элемента:
   $\forall a \in R\ \exists(-a) \in R\colon a + (-a) = 0$.
5. Ассоциативность умножения:
   $(a * b) * c = a * (b * c)$.
6. Дистрибутивность:
   $a * (b + c) = (a * b) + (a * c)$
   $(b + c) * a = (b * a) + (c * a)$.

Факторизация

Факторизацией натурального числа называется его разложение в произведение простых множителей. Может быть выполнена, например, **перебором возможных делителей**. Способ заключается в том, чтобы последовательно делить факторизуемое число n на натуральные числа от 1 до $\lfloor\sqrt{n}\rfloor$.

Формально достаточно делить только на простые числа в этом интервале, однако, для этого необходимо знать их множество. На практике составляется таблица простых чисел и производится проверка небольших чисел (например, до 2^{16}). Для очень больших чисел алгоритм не используется в силу низкой скорости работы.

Идеал

Для кольца R идеалом называется подкольцо, замкнутое относительно умножения на элементы из R.

Идеалом кольца R называется такое подкольцо (подкольцо кольца $(K, +, *)$ рассматривается как подмножество $R \subset K$, замкнутое относительно операций $+$ и $*$ из основного кольца) I кольца R, что

1. $\forall i \in I \; \forall r \in R$ произведение $ir \in I$ (условие на правые идеалы);
2. $\forall i \in I \; \forall r \in R$ произведение $ri \in I$ (условие на левые идеалы);

Сравнения

Если два целых числа a и b при делении на m дают одинаковые остатки, то они называются сравнимыми (или равноостаточными) по модулю числа m.

Сравнимость чисел a и b записывается в виде формулы (сравнения):

$$a \equiv b \pmod m$$

. Число m называется модулем сравнения.

Алгоритм Евклида

Алгоритм Евклида – эффективный алгоритм для нахождения наибольшего общего делителя двух целых чисел.

Пусть a и b — целые числа, не равные одновременно нулю, и последовательность чисел $a > b > r_1 > r_2 > r_3 > r_4 > \ \dots \ > r_n$ определена тем, что каждое r_k — это остаток от деления предпредыдущего числа на предыдущее, а предпоследнее делится на последнее нацело, то есть:

$a = bq_0 + r_1,$
$b = r_1 q_1 + r_2,$
$r_1 = r_2 q_2 + r_3,$
$\ldots$
$r_{k-2} = r_{k-1} q_{k-1} + r_k,$
$\ldots$
$r_{n-2} = r_{n-1} q_{n-1} + r_n,$
$r_{n-1} = r_n q_n.$

Тогда НОД(a, b), наибольший общий делитель a и b, равен r_n, последнему ненулевому члену этой последовательности.

Теоремы Эйлера и Ферма

Теорема Эйлера: если a и m взаимно просты, то $a^{\varphi(m)} \equiv 1 \pmod m$, где $\varphi(m)$ — функция Эйлера (количество натуральных чисел, меньших m и взаимно простых с ним).

Малая теорема Ферма: если a не делится на простое число p, то $a^{p-1} \equiv 1 \pmod p$.

Кольцо многочленов

Многочлен от x с коэффициентами в поле k — это выражение вида $p = p_m x^m + p_{m-1} x^{m-1} + \ldots + p_1 x + p_0$, где $p_0, \ldots, p_m$ — элементы k, коэффициенты $p, a, x, x^2, \ldots$ — формальные символы («степени x»). Такие выражения можно складывать и перемножать по обычным правилам действий с алгебраическими выражениями (коммутативность сложения, дистрибутивность, приведение подобных членов и т. д.). Члены $p_k x^k$ с нулевым коэффициентом p_k при записи обычно опускаются. Используя символ суммы, многочлены записывают в более компактном виде:
$p = p_m x^m + p_{m-1} x^{m-1} + \ldots + p_1 x + p_0 = \sum_{k=0}^{m} p_k x^k.$

Множество всех многочленов с коэффициентами в k образует коммутативное кольцо, обозначаемое $k[x]$ и называемое **кольцом многочленов** над k.

Число корней многочлена

Корень многочлена (не равного тождественно нулю) $a_0 + a_1x + \ldots + a_nx^n$ над полем K — это элемент $c \in K$ (либо элемент расширения поля K), такой, что выполняются два следующих равносильных условия:

- данный многочлен делится на многочлен $x - c$;
- подстановка элемента с вместо х обращает уравнение $a_0 + a_1x + \ldots + a_nx^n = 0$ в тождество.

Число корней многочлена степени n не превышает n даже в том случае, если кратные корни учитывать кратное количество раз.

Линейные пространства и операторы

Линейное пространство $V\,(F)$ над полем F — это упорядоченная четвёрка $(V, F, +, \cdot)$, где

- V — непустое множество элементов произвольной природы, которые называются векторами;
- F — поле, элементы которого называются скалярами;
- Определена операция сложения векторов $V \times V \to V$, сопоставляющая каждой паре элементов $\mathbf{x}, \mathbf{y}$ множества V единственный элемент множества V, называемый их суммой и обозначаемый $\mathbf{x} + \mathbf{y}$;
- Определена операция умножения векторов на скаляры $F \times V \to V$, сопоставляющая каждому элементу λ поля F и каждому элементу $\mathbf{x}$ множества V единственный элемент множества V, обозначаемый $\lambda \cdot \mathbf{x}$ или $\lambda\mathbf{x}$;

причём заданные операции удовлетворяют следующим аксиомам — аксиомам линейного (векторного) пространства:

- $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$, для любых $\mathbf{x}, \mathbf{y} \in V$ (коммутативность сложения);
- $\mathbf{x} + (\mathbf{y} + \mathbf{z}) = (\mathbf{x} + \mathbf{y}) + \mathbf{z}$, для любых $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ (ассоциативность сложения);
- существует такой элемент $\mathbf{0} \in V$, что $\mathbf{x} + \mathbf{0} = \mathbf{0} + \mathbf{x} = \mathbf{x}$ для любого $\mathbf{x} \in V$ (существование нейтрального элемента относительно сложения), называемый нулевым вектором или просто нулём пространства V;
- для любого $\mathbf{x} \in V$ существует такой элемент $-\mathbf{x} \in V$, что $\mathbf{x} + (-\mathbf{x}) = \mathbf{0}$

, называемый вектором, противоположным вектору $\mathbf{x}$;

- $\alpha(\beta\mathbf{x}) = (\alpha\beta)\mathbf{x}$ (ассоциативность умножения на скаляр);
- $1 \cdot \mathbf{x} = \mathbf{x}$ (унитарность: умножение на нейтральный (по умножению) элемент поля F сохраняет вектор).
- $(\alpha + \beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x}$ (дистрибутивность умножения вектора на скаляр относительно сложения скаляров);
- $\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{x} + \alpha\mathbf{y}$ (дистрибутивность умножения вектора на скаляр относительно сложения векторов).

Линейным отображением (оператором) векторного пространства L_K над полем K в векторное пространство M_K над тем же полем K (линейным оператором из L_K в M_K) называется отображение $f\colon L_K \to M_K$, удовлетворяющее условию линейности:

- $f(x + y) = f(x) + f(y)$,
- $f(\alpha x) = \alpha f(x)$.

для всех $x, y \in L_K$ и $\alpha \in K$.

Базис, размерность, ранг

Рангом системы строк (столбцов) матрицы A с m строк и n столбцов называется максимальное число линейно независимых строк.

Число столбцов и строк задают **размерность** матрицы.

Векторы $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ называются линейно зависимыми, если существует их нетривиальная линейная комбинация, значение которой равно нулю; то есть $\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2 + \ldots + \alpha_n\mathbf{x}_n = \mathbf{0}$ при некоторых коэффициентах $\alpha_1, \alpha_2, \ldots, \alpha_n \in F$, причём хотя бы один из коэффициентов α_i отличен от нуля.

В противном случае эти векторы называются линейно независимыми.

Число элементов (мощность) максимального линейно независимого множества элементов векторного пространства не зависит от выбора этого множества. Данное число называется **рангом**, или **размерностью**, пространства, а само это множество — **базисом**. Элементы базиса именуют **базисными векторами**. Размерность пространства чаще всего обозначается символом $\dim$.

Собственные числа и собственные векторы

Пусть L — линейное пространство над полем K, $A\colon L \to L$ — линейное преобразование.

Собственным вектором линейного преобразования A называется такой ненулевой вектор $x \in L$, что для некоторого $\lambda \in K$ $Ax = \lambda x$.

Собственным значением (собственным числом) линейного преобразования A называется такое число $\lambda \in K$, для которого существует собственный вектор, то есть уравнение $Ax = \lambda x$ имеет ненулевое решение $x \in L$.

Упрощённо говоря, собственный вектор — любой ненулевой вектор x, который отображается в коллинеарный ему вектор λx оператором A, а соответствующий скаляр λ называется собственным значением оператора.

Характеристический многочлен

Для данной матрицы A, $\chi(\lambda) = \det(A - \lambda E)$, где E — единичная матрица, является многочленом от λ, который называется характеристическим многочленом матрицы A.

1.4. Теория вероятностей

Зависимые и независимые события

Два события называются **независимыми**, если появление одного из них не изменяет вероятность появления другого. Например, если в цехе работают две автоматические линии, по условиям производства не взаимосвязанные, то остановки этих линий являются независимыми событиями.

События называются **зависимыми**, если одно из них влияет на вероятность появления другого. Например, две производственные установки связаны единым технологическим циклом. Тогда вероятность выхода из строя одной из них зависит от того, в каком состоянии находится другая.

Условные вероятности

Вероятность одного события B, вычисленная в предположении осуществления другого события A, называется условной вероятностью события B и обозначается $P\{B|A\}$.

Формула полной вероятности

Если событие A наступает только при условии появления одного из событий $B_1, B_2, \dots B_n$, образующих полную группу несовместных событий, то вероятность события A равна сумме произведений вероятностей каждого из событий $B_1, B_2, \dots B_n$ на соответствующую условную вероятность события $B_1, B_2, \dots B_n$: $P\{A\} = \sum\limits_{i=1}^{n} P\{B_i\}P\{A|B_i\}$.

При этом события $B_i,\ i = 1, \dots, n$ называются гипотезами, а вероятности $P\{B_i\}$ — априорными.

Математическое ожидание

Математическое ожидание дискретной случайной величины X вычисляется как сумма произведений значений x_i, которые принимает случайная величина X, на соответствующие вероятности p_i: $M[X] = \sum\limits_{i=1}^{\infty} x_i\, p_i$.

Задание. Вероятность попадания в цель при одном выстреле равна 0,8 и уменьшается с каждым выстрелом на 0,1. Составить закон распределения числа попаданий в цель, если сделано три выстрела. Найти математическое ожидание, этой случайной величины.

Решение. Введем дискретную случайную величину $X =$ (Число попаданий в цель). X может принимать значения 0, 1, 2 и 3. Найдем соответствующие вероятности. Вероятность не попасть 3 раза: $0,2*0,3*0,4$. Вероятность не попасть 2 раза: $0,2*0,3*0,6+0,2*0,7*0,4+0,8*0,3*0,4$. И т.д. Мат. ожидание будет $0*0,2*0,3*0,4+1*0,2*0,3*0,6+0,2*0,7*0,4+0,8*0,3*0,4$ и т.д.

Второй момент

Начальным моментом s-го порядка прерывной случайной величины называется сумма вида: $\alpha_s[X] = \sum\limits_{i=1}^{\infty} x_i^s\, p_i$.

Математическое ожидание – первый начальный момент случайной величины.

Неравенства Маркова и Чебышёва

Неравенство Маркова дает вероятностную оценку того, что значение неотрицательной случайной величины превзойдет некоторую константу через известное математическое ожидание. Когда никаких других данных о распределении нет, неравенство дает некоторую информацию, хотя зачастую оценка груба или тривиальна.

Пусть X - случайная величина, принимающая неотрицательные значения, $M(X)$ - ее конечное математическое ожидание, то для любых $a > 0$ выполняется: $P(X \geq a) \leq \frac{M(X)}{a}$.

Задача: Среднее количество вызовов, поступающих на коммутатор завода в течение часа, равно 300. Оценить вероятность того, что в течение следующего часа число вызовов на коммутатор превысит 400. **Решение**: По условию $M(X) = 300$. Воспользуемся формулой (неравенством Маркова): $P(X \geq 400) \leq \frac{300}{400} = 0,75$, т.е. вероятность того, что число вызовов превысит 400, будет не более 0,75.

Неравенство Чёбышева показывает, что случайная величина принимает значения близкие к среднему (математическому ожиданию) и дает оценку вероятности больших отклонений.

$P(|X - M(X)| \geq a) \leq \frac{D(X)}{a^2}, a > 0$

2. Алгоритмы и структуры данных

Нужно уметь написать код для перечисленных ниже элементарных алгоритмов.

2.1. Оценка алгоритмов

Мы рассчитываем, что вы понимаете, какое количество операций и объём дополнительной памяти необходимы для обсуждаемых алгоритмов и из каких соображений это получается.

2.2. Простейшие алгоритмы

Поиск заданного элемента

1

Поиск наибольшего элемента

1

Сортировка вставкой

1

Сортировка пузырьком

1

Быстрая сортировка

1

Иерархические сортировки

1

2.3. Простейшие структуры данных

Массив

Массив — структура данных, хранящая набор значений (элементов массива), идентифицируемых по индексу или набору индексов, принимающих целые (или приводимые к целым) значения из некоторого заданного непрерывного диапазона.

Особенностью массива как структуры данных (в отличие, например, от связного списка) является константная вычислительная сложность доступа к элементу массива по индексу. Имеет константную длину.

Список

Связный список — базовая динамическая структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну или две ссылки («связки») на следующий и/или предыдущий узел списка. Принципиальным преимуществом перед массивом является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

Стек

Стек — абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (англ. last in — first out, «последним пришёл — первым вышел»).

Чаще всего принцип работы стека сравнивают со стопкой тарелок: чтобы взять вторую сверху, нужно снять верхнюю.

Зачастую стек реализуется в виде однонаправленного списка (каждый элемент в списке содержит помимо хранимой информации в стеке указатель на следующий элемент стека).

Очередь

Очередь — абстрактный тип данных с дисциплиной доступа к элементам «первый пришёл — первый вышел» (FIFO, англ. first in, first out). Добавление элемента (принято обозначать словом enqueue — поставить в очередь) возможно лишь в конец очереди, выборка — только из начала очереди (что принято называть словом dequeue — убрать из очереди), при этом выбранный элемент из очереди удаляется.

3. Программирование

Нужно знать базовые принципы одного из «традиционных» (C, C++, Java, Python и др.) языков программирования.

Основы синтаксиса

- Язык Java различает прописные и строчные буквы.
- Каждая команда (оператор) в языке Java должна заканчиваться точкой с запятой.
- Программа на Java состоит из одного или нескольких классов. Абсолютно вся функциональная часть программы (т.е. то, что она делает) должна быть помещена в методы тех или иных классов.
- Хотя бы в одном из классов должен существовать метод main(). Именно этот метод и будет выполняться первым.
- В простейшем случае программа может состоять из одного (или даже ни одного) пакета, одного класса внутри пакета и единственного метода main() внутри класса. Команды программы будут записываться между строчкой

  ```
  public static void main(String[] args) { }
  ```

  и закрывающей фигурной скобкой, обозначающей окончание тела метода.

Переменные

- Целочисленные (к ним относятся byte, short, int, long).
- С плавающей точкой (к ним относятся float, double).
- Символы (char).
- Логические (boolean).

Условные выражения

- Условный оператор if. Если логическое выражение в скобках правдиво, то выполняется блок кода в фигурных скобках после if. Если логическое выражение принимает значение false, то ничего не происходит.

- Условный оператор if-else. Конструкция if-else отличается от предыдущей тем, что если логическое выражение в круглых скобках принимает значение false, то выполняется блок кода, находящийся в фигурных скобках после ключевого слова else.
- Условный оператор switch — case. Условный оператор switch — case удобен в тех случаях, когда количество вариантов очень много и писать для каждого if-else очень долго. Конструкция имеет следующий вид:

```
switch (expression) {
  case value1:
    //блок кода 1;
    break;
  case value2:
    //блок кода 2;
    break;
    ...
  case valueN:
    //блок кода N;
    break;
  default:
    //блок N+1;
}
```

Циклы

- **Цикл for**.

```
  for (int i = 1; i < 9; i++)
{
    // действия
}
```

- **Цикл do** сначала выполняет код цикла, а потом проверяет условие в инструкции while. И пока это условие истинно,цикл повторяется. Например:

```java
int j = 7;
do{
    System.out.println(j);
    j--;
}
while (j > 0);
```

- **Цикл while** сразу проверяет истинность некоторого условия, и если условие истинно, то код цикла выполняется. Например:

```java
int j = 6;
while (j > 0){

    System.out.println(j);
    j--;
}
```

- Оператор **break** позволяет выйти из цикла в любой его момент, даже если цикл не закончил свою работу.
- Чтобы цикл не завершался, а просто переходил к следующей итерации, используем оператор **continue**.

Массивы

```java
// Объявление массива.
int[] myArray;

// Создание, то есть, выделение памяти
// для массива на 10 элементов типа int.
myArray = new int[10];
```

Функции

Метод — это именованный блок кода, который объявляется внутри класса и может быть использован многократно.

Хорошо написанный метод решает одну практическую задачу: находит квадратный корень из числа (как штатный метод sqrt() в Java), преобразует число в строку (метод toString()), присваивает значения полям объекта и так далее.

Новый метод сначала объявляют и определяют, затем вызывают для нужного объекта или класса.

Методы могут возвращать или не возвращать значения, могут вызываться с указанием параметров или без. Тип возвращаемых данных указывают при объявлении метода — перед его именем.

В примере ниже метод должен найти большее из двух целых чисел, поэтому тип возвращаемого значения — int:

```java
// Заголовок метода.
public static int maxFinder(int a, int b) {
    // Ниже - тело метода.
    int max;
    if (a < b)
        max = b;
    else
        max = a;
    return max;
}
```

Рекурсия

```java
private int factorial(int n) {
    int result = 1;
    if (n == 1 || n == 0) {
        return result;
    }
    result = n * factorial(n-1);
    return result;
}
```

Динамическая память

Динамическое распределение памяти — способ выделения оперативной памяти компьютера для объектов в программе, при котором выделение памяти под объект осуществляется во время выполнения программы.

Куча — это хранилище памяти, также расположенное в ОЗУ, которое допускает динамическое выделение памяти и не работает по принципу стека: это просто склад для ваших переменных. Когда вы выделяете в куче участок памяти для хранения переменной, к ней можно обратиться не только в потоке, но и во всем приложении. Именно так определяются глобальные переменные. По завершении приложения все выделенные участки памяти освобождаются. Размер кучи задаётся при запуске приложения, но, в отличие от стека, он ограничен лишь физически, и это позволяет создавать динамические переменные.

Вы взаимодействуете с кучей посредством ссылок, обычно называемых указателями — это переменные, чьи значения являются адресами других переменных. Создавая указатель, вы указываете на местоположение памяти в куче, что задаёт начальное значение переменной и говорит программе, где получить доступ к этому значению. Из-за динамической природы кучи ЦП не принимает участия в контроле над ней; в языках без сборщика мусора (C, C++) разработчику нужно вручную освобождать участки памяти, которые больше не нужны. Если этого не делать, могут возникнуть утечки и фрагментация памяти, что существенно замедлит работу кучи.

В сравнении со стеком, куча работает медленнее, поскольку переменные разбросаны по памяти, а не сидят на верхушке стека. Некорректное управление памятью в куче приводит к замедлению её работы; тем не менее, это не уменьшает её важности — если вам нужно работать с динамическими или глобальными переменными, пользуйтесь кучей.

Стек

Стек — это область оперативной памяти, которая создаётся для каждого потока. Он работает в порядке LIFO (Last In, First Out), то есть последний добавленный в стек кусок памяти будет первым в очереди на вывод из стека.

Каждый раз, когда функция объявляет новую переменную, она добавляется в стек, а когда эта переменная пропадает из области видимости (например, когда функция заканчивается), она автоматически удаляется из стека. Когда стековая переменная освобождается, эта область памяти становится доступной для других стековых переменных.

Из-за такой природы стека управление памятью оказывается весьма логичным и простым для выполнения на ЦП; это приводит к высокой скорости, в особенности потому, что время цикла обновления байта стека очень мало, т.е. этот байт скорее всего привязан к кэшу процессора. Тем не менее, у такой строгой формы управления есть и недостатки. Размер стека — это фиксированная величина, и превышение лимита выделенной на стеке памяти приведёт к переполнению стека. Размер задаётся при создании потока, и у каждой переменной есть максимальный размер, зависящий от типа данных. Это позволяет ограничивать размер некоторых переменных (например, целочисленных), и вынуждает заранее объявлять размер более сложных типов данных (например, массивов), поскольку стек не позволит им изменить его. Кроме того, переменные, расположенные на стеке, всегда являются локальными.

В итоге стек позволяет управлять памятью наиболее эффективным образом — но если вам нужно использовать динамические структуры данных или глобальные переменные, то стоит обратить внимание на кучу.