ChatGPT

Курс лекций по информатике для подготовки к ЕГЭ

Представление и обработка информации

Информация и ее количество. Информатика начинается с понимания того, что такое информация и как она представляется. **Информация** – это сведения об окружающем мире, которые могут храниться, передаваться и обрабатываться. Количество информации измеряется в **битах** – минимальных единицах, принимающих значение 0 или 1. В объёмном (алфавитном) подходе количество информации определяется логарифмически: например, по формуле Хартли $I = K \log_2 N$ (где N – число возможных сообщений из данного алфавита, K – длина сообщения) информация измеряется в битах. **Закон аддитивности информации** гласит, что суммарная информация двух независимых сообщений равна сумме информации каждого из них. На практике используются более крупные единицы: 1 байт = 8 бит, далее килобайт (~10^3 байт), мегабайт (~10^6), гигабайт и т.д.

Представление данных. Компьютеры представляют любую информацию в двоичном коде (последовательностями битов). Числовые данные (целые, действительные) кодируются двоичными числами, текст – числовыми кодами символов, изображения – кодами цветов пикселей, звук – дискретными сигналами (отсчётами звуковой волны). Например, буква латинского алфавита «Y» в памяти представляется как двоичный код `1011001`. Различные виды данных могут требовать разного объёма памяти. **Пример:** если символ кодируется 1 байтом (256 возможных символов), то текст из 100 символов займёт 100 байт. Если же используется кодировка Unicode (обычно 2 байта на символ), тот же текст потребует ~200 байт.

Обработка информации. Под обработкой понимают любые преобразования данных: вычисления, сортировка, поиск и т.д. На уровне хранения важно понимать, как оценивается необходимый объём памяти и время передачи данных. **Объём данных:** вычисляется исходя из количества элементов и бит на элемент. *Пример:* изображение 800×600 с глубиной цвета 24 бита (3 байта на пиксель) займёт $800 \cdot 600 \cdot 3 = 1\ 440\ 000$ байт ≈ 1,44 МБ. **Скорость передачи:** характеризуется пропускной способностью канала (например, в бит/с). Время передачи T приблизительно равно отношению объёма данных к скорости (с учётом, что 1 байт = 8 бит). *Пример:* файл 5 МБ по каналу 1 Мбит/с передастся примерно за $5 \cdot 8/1 = 40$ секунд. Такие расчёты актуальны для задач на вычисление времени передачи данных.

Файловая система. Данные на компьютере хранятся в виде **файлов**, организованных по каталогам (папкам). Важно понимать структуру путей и имён файлов. **Маски файлов** используются для задания шаблонов имён при поиске. Маска `*` означает любую последовательность символов, а `?` – ровно один любой символ. *Пример:* маска `Документ?.txt` охватывает файлы `Документ1.txt`, `ДокументA.txt` и т.д., но не `Документ10.txt` (две цифры вместо одной). В задачах ЕГЭ могут спрашивать, сколько файлов удовлетворяют заданной маске или какой максимальный/минимальный файл подходит под шаблон. **Пример задачи:** *«В каталоге хранятся файлы с именами вида* `A??B1.xlsx` *(где* `?` *– любая цифра). Сколько различных имён файлов возможно?»* Решение: `A` и `B1` зафиксированы, две позиции `??` – любые цифры (10 вариантов каждая). Всего вариантов: $10 \cdot 10 = 100$` возможных имён.

Системы счисления

Позиционные системы счисления. Позиционной называется система, в которой число записывается последовательностью цифр, а значение цифры зависит от ее позиции (разряда). Основание системы показывает, сколько разных символов-цифр используется и во сколько раз ценность разряда возрастает при сдвиге влево. Например, в десятичной (основание 10) цифры 0–9, в двоичной (основание 2) – 0 и 1, в шестнадцатеричной (16) – 0–9 и A–F. Развёрнутая запись числа – это сумма цифра × основание^позиция. **Пример:** $652_{10} = 6\cdot10^2 + 5\cdot10^1 + 2\cdot10^0$; $1011_2 = 1\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 = 11_{10}$.

Конвертация чисел. Необходимо уметь переводить числа из одной системы счисления в другую. Основные алгоритмы: - *Из P-ичной в десятичную:* умножаем каждую цифру на $P^{\text{позиция}}$ и суммируем. *Пример:* $1011_2 = 1\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 = 11_{10}$. - *Из десятичной в P-ичную:* делим число на основание P, записываем остатки. Остатки, прочитанные снизу вверх, дадут запись числа в новой системе. *Пример:* 45_{10} в двоичной: $45/2=22$ (ост.1), $22/2=11$ (ост.0), $11/2=5$ (ост.1), $5/2=2$ (ост.1), $2/2=1$ (ост.0), $1/2=0$ (ост.1). Остатки снизу вверх: 101101_2. - *Перевод дробей:* умножают дробную часть на основание новой системы и берут целые части. *Пример:* 0.625_{10} в двоичную: $0.625\cdot2 = 1.25$ (целая часть 1), оставшаяся дробь $0.25\cdot2 = 0.5$ (целая 0), $0.5\cdot2 = 1.0$ (целая 1). Результат: 0.101_2.

Другие системы счисления. Помимо обычных, существуют специальные системы: - *Восьмеричная (8) и шестнадцатеричная (16):* часто используются в информатике; для перевода между ними и двоичной удобно группировать двоичное число по 3 (для 8) или 4 (для 16) бита. *Пример:* 101101_2 можно группировать как $101\ 101$ – это 5 и 5 в восьмеричной, т.е. 55_8. - *Троичная уравновешенная:* содержит цифры -1, 0, 1 (обозначают как –, 0, +), позволяет без знака представлять отрицательные. - *Двоично-десятичная (BCD):* каждая десятичная цифра кодируется фиксированными двоичными разрядами. Эти специальные системы упоминаются теоретически, но в ЕГЭ в основном нужны стандартные позиционные системы.

Типовые задачи. На экзамене часто встречаются задачи: - *Признаки делимости:* например, число в системе с основанием P оканчивается на 0, если оно делится на P. *Применение:* «Какая последняя цифра в записи числа в системе счисления P указывает на четность числа?» – Ответ: 0 (для четности в любой чётной системе). - *Неизвестное основание:* например, если $X_{p} = Y_{10}$, найти p или цифры. *Пример задачи:* «При каком наименьшем основании система счисления запись числа 37 оканчивается на 0?» Решение: 37 в этой системе должно делиться на основание p, т.е. 37 должно быть кратно p. Наименьшее основание больше 7 (т.к. цифра 7 присутствует) – это 8, но 37 не кратно 8; 9 – не кратно; 10 – кратно (37 в 10-ричной оканчивается на 7, не 0). Проверяя, найдём $p=37$ (в базе 37 запись числа 37 – это цифра 1 в старшем разряде и 0 в младшем). - *Свойства позиции:* количество символов в записи, максимальная цифра – всегда $p-1$. Например, в системе основанием p число с n цифрами максимально равно (p^n-1).

Кодирование информации

Кодирование символов (текста). Для представления текстовой информации используются таблицы кодировки символов. **ASCII** – базовая 7-битная кодировка на 128 символов: латинские буквы, цифры, знаки и некоторые управляющие символы. В расширенном варианте (8-битном) ASCII кодирует 256 символов, но набор символов зависит от локальной кодировки (например, для русского языка существовали CP1251, KOI8-R и др.), что вызывало проблемы совместимости.

Unicode – универсальный стандарт, присваивающий уникальные номера более чем 10^5 символов всех языков. Кодировки UTF-8, UTF-16, UTF-32 реализуют Unicode в виде последовательностей байтов: UTF-8 хранит символы от 1 до 4 байтов (английские буквы 1 байт, русские 2 байта и т.д.), UTF-16 – обычно 2 байта на символ и т.п. В ЕГЭ нужно уметь определять информационный объём текстового сообщения: умножать число символов на количество бит/байт на символ, учитывая пробелы, знаки, кодировку. *Пример:* «Сообщение содержит 40 символов кириллицы в Unicode. Каков объём в байтах?» – Решение: Unicode (UTF-16) ~2 байта на символ, $40 \cdot 2 = 80$ байт.

Кодирование чисел и структуры данных. Целые числа в памяти представляются двоичным кодом фиксированной разрядности. Максимальное число ограничено разрядной сеткой (например, 8 бит – до 255). Для представления отрицательных целых используют знаковый бит и код Дополнения до 2 (двойной дополнительный код) – в нём отрицательные числа хранятся как $2^n - N$ в n битах. В задачах может встретиться понятие переполнения (результат не помещается в отведенные биты). Вещественные числа кодируются сложнее: в формате с плавающей запятой (IEEE 754) с выделением бит под мантиссу и порядок. В ЕГЭ обычно достаточно знать, что вещественные числа хранятся с ограниченной точностью и возможна погрешность вычислений (например, $(0.1)_{10}$ не имеет точного двоичного представления). Детали хранения (нормализация, денормализованные числа) выходят за рамки школьного курса.

Кодирование изображений и звука. Графическая информация представляется как сетка пикселей. **Разрешение** – количество пикселей по горизонтали и вертикали, **глубина цвета** – бит на пиксель (определяет количество возможных цветов, например 8 бит = 256 цветов). Объём несжатого изображения = число пикселей × бит на пиксель. *Пример:* для 100×100 пикселей, 8 бит на пиксель: $10000 \cdot 8 = 80000$ бит = 10 КВ. Звуковая информация кодируется дискретизацией аналогового сигнала: **частота дискретизации** (отсчётов в секунду, Гц) и **разрядность** (бит на отсчёт). Объём звукового фрагмента = частота × длительность × битность × число каналов. *Пример:* 1 секунда моно-звука 44,1 кГц 16 бит: $44100 \times 16 = 705600$ бит ≈ 86 КВ.

Избыточность и сжатие. В сообщениях часто встречается избыточность – повторяемость, позволяющая сжать данные без потери. Кодирование сжатия (архивирование) – отдельная тема. В теоретической части может упоминаться неравномерное кодирование: использование более коротких кодов для частых символов (например, коды Хаффмана). *Условие Фано* – критерий, при котором код однозначно декодируем (ни один код не является префиксом другого). В задачах могут дать набор кодовых слов и спросить, однозначно ли декодируется сообщение или рассчитать длину сообщения по кодам.

Криптография (шифрование данных). ЕГЭ включает ознакомление с базовыми методами шифрования. **Симметричные шифры** используют один ключ для шифрования и расшифрования (пример: классический *шифр Цезаря* – сдвиг букв алфавита на фиксированное число; *шифр Виженера* – повторяющийся ключ-слово для сдвигов). **Асимметричные шифры** используют пару ключей (открытый для шифрования, закрытый для расшифровки); пример – RSA. На экзамене могут встретиться простые вопросы: например, зашифровать или расшифровать слово шифром Цезаря с заданным сдвигом, либо понять принцип: *«Почему шифр подстановки лёгкой для взлома при сохранении структуры языка?»* (ответ: частотный анализ).

Коды обнаружения и исправления ошибок. При передаче данных возможны искажения, поэтому применяют специальные коды с избыточностью для контроля. *Пример:* добавление битов чётности (паритета) – самый простой способ обнаружения однобитовых ошибок. **Код Хэмминга** – метод добавления нескольких проверочных бит, позволяющих не только

обнаружить, но и исправить одиночную ошибку в блоке данных. Задачи могут спросить *расстояние Хэмминга* между кодовыми словами (минимальное число отличающихся битов) или максимальное число ошибок, которое код может обнаружить/исправить (если расстояние d, то обнаруживает $d-1$ ошибок, исправляет $\lfloor\frac{d-1}{2}\rfloor$ ошибок). Как правило, детали этих кодов не углубляются, достаточно понимать принцип контроля ошибок.

Пример задания: *«Ниже приведён фрагмент таблицы кодировки. В ней каждому символу соответствует двоичный код. Закодировано слово* `10110101`*. Используя принцип префиксного кода, восстановите исходную последовательность символов.»* – Решение: проверяем, какой код является префиксом других. Если кодировка удовлетворяет условию Фано (ни один код не является началом другого), разбиваем последовательность однозначно по границам известных кодов, получаем исходные символы.

Логика и алгоритмы

Логические высказывания. Логика в информатике оперирует высказываниями – утверждениями, которые могут быть либо истинны, либо ложны [1]. Простое высказывание не содержит логических связок, сложные высказывания строятся из простых с помощью **логических операций**: И (конъюнкция), ИЛИ (дизъюнкция), НЕ (отрицание), импликация (следование $A \to B$), эквиваленция ($A \leftrightarrow B$). Логические значения истина/ложь обозначают 1/0 соответственно. Зная таблицы истинности операций, можно вычислять значение сложных выражений. *Пример:* выражение ¬(A ∧ B) (не A и B) истинно, если хотя бы одно из A, B ложно (законы де Моргана). **Логические законы** (тождественно истинные равенства) позволяют упрощать выражения: например, $¬(A \wedge B) \equiv ¬A \vee ¬B$. В ЕГЭ часто встречаются задачи на **таблицы истинности**: заполнить таблицу для выражения или по заданной таблице определить подходящее выражение. Также могут быть задачи на **логические схемы** – графическое представление выражения с помощью логических элементов (AND, OR, NOT и др.). Нужно уметь сопоставлять схему и формулу (например, нарисовать схему по формуле или наоборот).

Предикаты и кванторы. Предикат – логическое высказывание с параметром (переменной), например $P(x)$: x – чётное число. При подстановке конкретного x предикат становится истинным или ложным. **Кванторы** позволят говорить об элементах множества: ∀ (для всех) и ∃ (существует). *Пример:* высказывание ∀x P(x) – «для всех x выполняется P(x)». В заданиях могут спрашивать понимание кванторов: например, отрицание ∀ превращается в ∃ с отрицанием предиката внутри ($¬(\forall x\ P(x)) \equiv \exists x\ ¬P(x)$).

Алгоритмы и их свойства. Алгоритм – это последовательность команд, предназначенная исполнителю, выполнение которых приводит к решению задачи [2]. Алгоритм должен быть понятным исполнителю и однозначным. Свойства алгоритмов: *дискретность* (разбитие процесса на шаги), *детерминированность* (каждый шаг однозначен), *результативность* (завершается за конечное число шагов), *массовость* (применимость к целому классу задач).

Способы описания алгоритмов: словесное описание, псевдокод, блок-схема, программы на языках. Для ЕГЭ важно понимать алгоритмы, заданные на человеческом языке или простейшим кодом, и уметь их выполнять мысленно. Часто встречаются задачи: *«что выведет данный псевдокод/программа при определённых входных данных»* – нужно проследить шаги и получить результат.

Управление исполнителями. В ЕГЭ популярны задачи с *исполнителями* – абстрактными объектами с набором команд. Примеры: Робот на поле (команды шагов), Чертёжник (рисует линии на плоскости), Редактор (текстовый, команды вставки/замены символов). Нужно понимать, как программа команд изменит состояние исполнителя. **Пример задачи (исполнитель «Редактор»):** дан алгоритм, который берёт строку, и пока в ней есть подстрока "01", заменяет её на "10". Что получится после выполнения алгоритма для исходной строки "001"? Решение: *001* -> находим "01" начиная слева: "0**01**" -> заменяем на "0**10**", строка становится "010"; повторяем поиск "01": теперь "010" содержит "01" в начале -> "10" -> заменяем -> получаем "100". Больше "01" нет, алгоритм останавливается. Ответ: "100". Такие задачи требуют аккуратно применять цикл или следовать шагам.

Псевдокод и программы. Задачи могут представить небольшой фрагмент кода (на понятном языке или блок-схеме) – например, цикл с условиями – и спросить результат или промежуточное значение. Нужно понимать конструкции: *ветвление* (если-то-иначе), *цикл* (for, while). Иногда предлагают определить, при каком входном значении алгоритм даст определённый результат. *Пример:* «Приведен алгоритм: `s := 0; для i от 1 до N: s := s + 2*i`. Чему будет равна переменная s после выполнения алгоритма?» Решение: этот алгоритм суммирует $2+4+...+2N = 2(1+2+...+N) = 2 * \frac{N(N+1)}{2} = N(N+1)$.

Сложность алгоритмов. Теоретически вводится понятие оценки сложности – времени работы и требуемой памяти в зависимости от размера входа. На базовом уровне важно понимать порядок роста: например, алгоритм с двойным вложенным циклом по N элементам – порядка N^2 операций (квадратичная сложность), простой один цикл – порядка N (линейная). Асимптотическая запись $O(f(N))$ используется в продвинутых задачах, но на ЕГЭ скорее качественно: *какой алгоритм эффективнее для большого N*. *Пример:* перебор всех подмножеств (экспоненциальный $O(2^N)$) быстро становится невозможным даже при умеренных N, в то время как $O(N^2)$ алгоритм работает существенно быстрее.

Алгоритмические стратегии. Для сложных задач могут использоваться *рекурсия* (функция вызывает сама себя для подзадачи), *динамическое программирование* (разбиение задачи на пересекающиеся подзадачи и сохранение результатов), *жадные алгоритмы*, *поиск в ширину/глубину* и др. В рамках ЕГЭ часто появляются рекурсивно определённые функции или программы – нужно вычислить их значение. *Пример:* рекурсивная функция: `F(n): если n<3 вернуть n, иначе вернуть F(n-1)+F(n-2)`. Вопрос: чему равен F(5)? Решение: $F(5)=F(4)+F(3)$; $F(4)=F(3)+F(2)$; $F(3)=F(2)+F(1)$; известно $F(2)=2, F(1)=1$. Последовательно: $F(3)=2+1=3$; $F(4)=3+2=5$; $F(5)=5+3=8$. Ответ: 8.

Двухигровые задачи. Особый вид алгоритмических задач – игры для двух игроков с полным знанием. Обычно описано состояние (например, куча камней, в ход можно убрать 1 или 2 камня) и нужно определить выигрышную стратегию. Решение базируется на анализе **выигрышных** и **проигрышных позиций**: если из позиции есть хотя бы один ход в проигрышную для соперника – позиция выигрышная. Такие задачи решаются рекурсивно или перебором позиций. На ЕГЭ часто дают: *«У Петика и Васи игра: ... Какая начальная позиция выигрышная?»* Нужно расписать ходы или построить дерево игры. Как правило, школьникам достаточно рассмотреть несколько шагов вперед. *Пример:* На столе 5 камней. За ход можно взять 1 или 2 камня. Выигрывает тот, кто заберёт последний камень. Анализ: при 1 или 2 камнях на столе первый игрок сразу выиграет, взяв все. При 3 камнях: первый может взять 1 (останется 2 – второй возьмёт 2 и выиграет) или 2 (останется 1 – второй возьмёт 1, выиграет) – значит 3 для первого проигрышная. 4 камня: первый ход – возьмёт 1 (останется 3, что проигрышно для того, кто ходит, т.е. для второго, значит первый получит победу) – достаточно одной выигрышной стратегии, значит 4 – выигрышная. Аналогично определяются статусы других позиций. В ЕГЭ подобные задачи (№19–21) требуют

отметить, какие позиции выигрышные, часто для более сложных правил, иногда с двумя параметрами (например, две кучи камней).

Программирование

Языки программирования. Для экзамена не требуется глубокое владение синтаксисом конкретного языка, но предполагается умение понимать код на одном из распространённых языков (чаще всего Python, нередко Pascal, C++). Основные элементы программирования, которые нужно знать: - **Типы данных:** целые (integer), вещественные (float, с ограниченной точностью), символы и строки (char, string), логический (boolean true/false). В динамически типизированном Python типы не указываются явно, но понимание отличий сохраняется. - **Переменные и операции:** запись значения в переменную (присваивание), арифметические операции (`+ - * / %` и целочисленное деление), строковые операции (конкатенация, срезы). - **Ввод-вывод:** чтение данных (в ЕГЭ часто данные заранее содержатся в файле, который нужно обработать, или в условии задачи), вывод результатов. - **Условный оператор:** `if условие then ... else ...` – выполняет разные действия в зависимости от булева условия. - **Циклы:** - *циклы с параметром* (`for i in range(n)`) – повторяют заданное количество итераций; - *циклы с условием* (`while условие do ...`) – повторяют пока условие истинно. - **Функции и процедуры:** именованные блоки кода, позволяющие переиспользовать логику. В рамках школьного курса важно понимать идею разбиения задачи на подзадачи (модульность), но писать свои функции обычно не требуется на экзамене, кроме разве что в части 2 для оптимизации кода.

Структуры данных: - **Массивы (списки):** упорядоченные коллекции элементов. Нужно уметь проходить по массиву, находить максимум, сумму, сортировать. В Python – список (`list`). - **Двумерные массивы (таблицы, матрицы):** например, таблица чисел, доступ по двум индексам. Задачи могут дать алгоритм заполнения таблицы или поиска элемента. - **Строки:** последовательности символов. Умеем получать длину строки, обращаться к символам, искать подстроку, заменять. В задачах часто требуется найти количество вхождений символа/слова или выполнить простую обработку строки. - **Словари (ассоциативные массивы):** структура вида ключ-значение. Например, частотный словарь для подсчёта, сколько раз каждый символ встречается. Это рассматривается как расширение (в ЕГЭ может потребоваться понять принцип хранения ключей через хэш-таблицы, но без деталей реализации). - **Стек и очередь:** абстрактные структуры LIFO и FIFO. Стек используется, например, при вычислении выражений в постфиксной форме (польской нотации) или при проверке корректности скобочной последовательности – задачи ЕГЭ могут включать такие проверки. Очередь – для последовательной обработки (реже встречается, но знать принцип полезно: элемент, первым пришёл – первым обработан).

Стандартные алгоритмы: - **Поиск:** линейный поиск элемента в массиве (просто перебор) и бинарный поиск в отсортированном массиве (деление диапазона пополам). - **Сортировка:** знать принципы простейших сортировок – пузырьком, вставками, выбором (все $O(n^2)$), и более эффективных – слиянием ($O(n \log n)$), быстрой (QuickSort, в среднем $O(n \log n)$). На ЕГЭ могут спросить, сколько операций сравнения выполнит сортировка пузырьком в худшем случае для n элементов (ответ: порядка n^2), или определить результат частично выполненной сортировки. - **Программирование численных методов:** например, метод перебора или половинного деления для нахождения корня уравнения, приближённое интегрирование методами прямоугольников/трапеций – эти темы встречаются скорее в углубленной программе и редко проверяются напрямую, однако знать о них стоит.

Примеры типовых задач программирования (1 часть): - Дан фрагмент кода – нужно указать, что он выведет. *Пример:* `x:=0; для i от 1 до 5: x := x + i^2; вывести x`. По шагам: i=1 (x=1), i=2 (x=5), i=3 (x=14), i=4 (x=30), i=5 (x=55). Ответ: 55. - Дан алгоритм, нужно найти входные данные, при которых выполняется условие. *Пример:* программа принимает число N и выводит "YES", если N – простое. Вопрос: при каком наименьшем N выводит "YES"? Ответ: 2 (первое простое число). - Обработка файлов: в тексте задачи может быть ссылка на файл с набором чисел или строк, и описан алгоритм их обработки – нужно понять результат. *Пример:* «В файле перечислены 1000 чисел. Приведен код, который читает файл и увеличивает счетчик при определенном условии. Что в итоге выведет программа?» Решение: понять, какое условие (например, число оканчивается на 5 и не делится на 10) и посчитать, сколько таких чисел в файле. Эти задачи требуют внимательного чтения условий.

Задачи второй части (развернутый ответ): Здесь от абитуриента ожидается самостоятельно написать программу или алгоритм. Темы таких задач: - **Перебор условий.** Например, найти наименьшее число, удовлетворяющее набору свойств (делимость, не содержит цифры, и т.д.). Решение: написать перебор натуральных чис с проверкой условий. - **Обработка данных.** Например, задача на обработку массива: дан список чисел, нужно вывести максимальную сумму пары элементов с определёнными свойствами (разная чётность индексов и т.п.). Требуется придумать эффективный алгоритм (не $O(n^2)$, а, скажем, один проход с хранением подходящих кандидатов). - **Динамическое программирование.** Например, задача на подсчёт количества способов получить сумму или пути в графе. Нужно построить рекуррентное решение. *Пример:* «Сколькими способами можно получить сумму S из монет номиналов a, b, c?» – пишется рекуррентная формула или циклический алгоритм. - **Работа с большими данными.** В задачах на 26, 27 номера проверяется умение работать с большими объемами: эффективное использование памяти и времени. Например, задача 27: в файле 1 000 000 чисел, нужно найти пару с максимальной суммой кратной 120. Полный перебор невозможен, надо придумать оптимизацию (хранить остатки по модулю 120 и комбинировать). В таких случаях нужно описать идею решения и привести итоговый результат.

Советы по программированию: Всегда обращайте внимание на **крайние случаи** (пустой список, ноль, отрицательные числа) и **эффективность**. Если задача подразумевает огромный вход, стоит подумать о методе лучше, чем перебор всех сочетаний. На ЕГЭ оценивается не только правильность, но и обоснование – пишите комментарии или объяснения к коду, если требуется, чтобы эксперт понял ваш ход мысли.

Работа с таблицами и графами

Электронные таблицы (Excel). Навыки работы с табличными процессорами проверяются на базовом уровне. Нужно понимать, как организованы данные в таблице (ячейки по строкам и столбцам, адреса типа A1, C5 и т.д.), знать основные функции: суммирование диапазона (`SUM`), среднее (`AVERAGE`), поиски максимума/минимума, подсчёт количества значений по условию, вычисление корреляции и др. Часто в задачах дают фрагмент таблицы с формулами и просят вычислить значение в конкретной ячейке или определить, какое условие заложено в формуле фильтра.

Пример: дан фрагмент таблицы:

```
A    | B    | C
1 | 4    | =A1+B1
2 | 5    | =A2+$B$1
```

Вопрос: чему равны формулы в C1 и C2? Решение: C1 = A1+B1 = 1+4 = 5; C2 = A2 + B1 = 2 + 4 = 6 (`$B$1` – абсолютная ссылка на B1).
Другой тип заданий – анализ результатов копирования формул. *Пример:* формула из одной ячейки протягивается вниз, как изменится адресация. Нужно помнить разницу между относительной и абсолютной ссылкой (с `$`).

Также распространены задачи, где описано содержимое таблицы (например, база данных в Excel) и нужно вручную выполнить запрос: *«Отфильтровать только записи с таким-то значением в столбце X и вычислить сумму по столбцу Y»*. Такая задача сводится к базовому пониманию фильтрации и агрегирования – перебрать нужные строки и просуммировать указанный столбец.

Графы. Граф – это математическая модель, состоящая из **вершин** и **рёбер** (связей) между ними. Граф может быть **ориентированным** (направленные связи) или **неориентированным**, **взвешенным** (ребру приписан вес/длина) или нет. Формы представления графов: **список рёбер**, **матрица смежности** (таблица $n \times n$, где 1 в клетке (i,j) означает ребро из i в j), **список смежности** (для каждой вершины перечислены соседние). В ЕГЭ задачи на графы обычно проверяют умение анализировать сети и пути: - Найти кратчайший или оптимальный путь между вершинами (сумма весов минимальна). Для небольшого графа можно перебрать все пути или использовать алгоритм Дейкстры (теоретически знать принцип полезно, но экзамен рассчитан на небольшой перебор). *Пример:* в таблице городов и расстояний между ними найти минимальный путь из A в D – можно вручную просуммировать варианты. - Подсчитать количество различных путей от одной вершины к другой, особенно в **ориентированном ациклическом графе (DAG)**. Здесь часто применяют рекурсивное соотношение: число путей в конечную вершину = сумме путей через соседей. *Пример:* сколько различных маршрутов из города A в город D, если граф разветвляется? Решение: если из A можно попасть в B и C, а из них – в D, то общее число путей = (пути B->D) + (пути C->D), с учётом путей внутри подграфа. - Деревья – частный случай графа без циклов. **Бинарное дерево** – структура, где каждый узел имеет не более 2 потомков. В задачах могут представить арифметическое выражение в виде дерева (каждый внутренний узел – операция, листья – операнды) и спросить значение выражения, или перечислить узлы в порядке обхода (прямой, симметричный, обратный обход). - **Обход графа/дерева.** Нужно понимать поиски в глубину и ширину – как они в порядке обходят вершины. Задачи могут дать фрагмент алгоритма обхода и спросить, в каком порядке посетятся вершины.

Примеры задач: - Дан фрагмент **сети** (например, схема городов с дорогами) – найти по таблице смежности количество дорог в городе X (что равняется количеству единиц в строке X матрицы смежности), либо степень вершины. - Дан ориентированный граф без циклов – подсчитать число различных путей определённой длины. Например: «Сколько различных путей длины 3 существует из вершины 1 в вершину 4?» Решение: выписать все маршруты из 1 в 4, состоящие из 3 рёбер. - **Задача на минимальное остовное дерево:** если дан взвешенный граф, найти суммарный минимальный вес, соединяющий все вершины (алгоритм Крускала или Прима). Как правило, граф маленький, можно просто выбрать наименьшие рёбра, не образуя циклов. - **Задача 27 (продвинутая):** может потребовать обработать граф алгоритмически, но чаще это остаётся в теории. Однако, знание, что, например, поиск в глубину можно использовать для топологической сортировки (порядок выполнения зависимых задач), лишним не будет.

Важный навык – моделировать прикладные задачи графами. Иногда задача про файлы и папки решается через дерево (иерархия каталогов), задача про таблицу переходов – через граф состояний. Уметь распознать графическую модель в тексте – большой плюс.

Модели и формализация

Понятие модели. Модель – упрощённое представление реального объекта или процесса, сохраняющее существенные для задачи характеристики. Модели бывают материальные (макеты) и информационные (описания, схемы). Информационные модели представляют данные и связи в удобной форме: схемы, таблицы, графики, формулы, диаграммы. В курсе информатики делается акцент на формализации – переводе реальной задачи на язык математики или алгоритмов.

Примеры моделей: - Таблица или база данных для хранения сведений о студентах – модель информационной системы университета. - График функции – модель зависимости одной величины от другой. - Карты и граф-схемы – модель дорог (граф городов). - Формула – модель соотношения между величинами (например, $s = v \cdot t$ – модель равномерного движения). - Алгоритм или программа – модель процесса (например, вычисления или бизнес-логики).

Адекватность модели. Модель должна достаточно точно отражать объект в контексте цели моделирования. Проверка адекватности – сравнение результатов модели с реальностью. В задачах ЕГЭ могут спрашивать: *«Почему полученная модель не подходит для...»* – ожидается ответ про упрощения, не учитывающие чего-то важного.

Дискретизация процессов. Многие реальные процессы непрерывны, но для моделирования на компьютере их дискретизируют (разбивают на шаги по времени или по значению). *Пример:* моделирование движения автомобиля: вместо непрерывного уравнения движения рассчитывают положение каждую секунду. Шаг должен быть достаточно малым, чтобы модель была точной. Более сложный пример – **метод Монте-Карло**: использование случайных чисел для имитации работы системы (например, система массового обслуживания, очереди).

Прогнозирование и данные. В рамках “анализа данных” рассматривают построение моделей по экспериментальным данным. Например, метод **наименьших квадратов** – способ подобрать линейную зависимость к набору точек (минимизируя отклонения). В ЕГЭ могут быть вопросы: *«Какой вид зависимости лучше описывает приведённые результаты эксперимента?»* или *«Что обозначает коэффициент корреляции, вычисленный для двух рядов данных?»*. Нужно знать, что корреляция (от –1 до 1) измеряет степень линейной связанности двух величин (0 – нет линейной связи).

Формализация условий задач. Умение переводить условия задачи на язык формул – важный навык. Например, текст «число X кратно 3 или 5, но не 15» можно формализовать логическим выражением: $(X \bmod 3 = 0 \vee X \bmod 5 = 0) \wedge X \bmod 15 \neq 0$. Такие условия часто встречаются в программировании, но могут и прямо спрашиваться на ЕГЭ: *«Запишите логическое выражение для условия отбора: товар дешевле 100 рублей или тяжелее 1 кг.»* – $(price < 100) \vee (mass > 1)$.

Имитационное моделирование. Иногда задачи описывают алгоритмическую модель какого-то процесса. Например, игра «Жизнь» (клеточный автомат) – модель популяции клеток по заданным правилам. Или моделирование экономического процесса: задан алгоритм, имитирующий работу кассы магазина (приход товара, продажа, очередь) – нужно определить итоговое состояние.

Подход к таким задачам: внимательно воспроизвести описанные шаги, вести учёт состояния (можно на черновике таблицу).

Пример задачи (моделирование): *«На складе 100 единиц продукта. Каждый день продаётся 20, каждые 3 дня приходит поставка 30. Смоделируйте процесс на 6 дней и определите запас к концу 6-го дня.»* Решение: День 1: остаток 80; День 2: остаток 60; День 3: продажа до 40, приход +30 -> 70; День 4: 50; День 5: 30; День 6: продажа до 10, приход +30 -> 40. Ответ: 40 единиц.

Два слова про компьютеры: Под раздел “Модели” иногда подпадает материал об устройстве компьютеров как моделей вычислителей. Например, **машина Тьюринга** – абстрактная модель вычислительной машины (лента и набор правил). Современные компьютеры – реализация этой модели с архитектурой фон Неймана (разделение на память, процессор, ввод-вывод). Может встретиться вопрос: *«Что является основной моделью вычислений и почему универсальная вычислительная машина важна?»* – ответ про машину Тьюринга как абстракцию, на основе которой построены языки программирования.

В целом, тема “Модели и формализация” охватывает умение представить информацию в нужной форме, будь то таблица, граф, схема или формула, а также понимание, как строятся и анализируются модели различных процессов.

Базы данных и запросы

Понятие базы данных. База данных (БД) – организованная совокупность данных, обычно в виде таблиц. **Реляционная БД** состоит из таблиц, между которыми могут быть связи. Таблица представляет однотипные сущности, строки – записи (экземпляры), столбцы – поля (атрибуты). В каждой таблице выделяют **ключ** – один или несколько столбцов, однозначно идентифицирующих запись (например, номер паспорта для таблицы граждан). В связях между таблицами ключ одной (первичный ключ) выступает как **внешний ключ** в другой, обеспечивая целостность – каждая ссылка указывает на реальную запись.

Пример: Есть таблица `Ученики` (Поля: ID, Фамилия, Класс) и таблица `Олимпиады` (ID_олимпиады, Название). Есть таблица связки `Участия` (Поля: ID_ученика, ID_олимпиады, Диплом), где (ID_ученика, ID_олимпиады) – составной ключ, указывающий кто в какой олимпиаде участвовал. Здесь `ID_ученика` – внешний ключ на `Ученики.ID`, а `ID_олимпиады` – на `Олимпиады.ID_олимпиады`.

Запросы на выборку (SELECT). Чтобы получить данные из БД по условию, используются запросы. Нужно представлять себе простейший синтаксис:

```
SELECT поля
FROM таблица(ы)
WHERE условие;
```

В ЕГЭ формат могут упростить до описательного. Важно уметь фильтровать по условию (например, *WHERE возраст > 18*), выполнять сортировку (ORDER BY), находить агрегаты (COUNT, SUM, AVG) и группировать данные (GROUP BY). Часто задачи дают описание запроса и спрашивают результат (например: *«Сколько записей вернёт запрос...?»*).

Пример задачи: *Дана таблица "Сотрудники" с полями (Фамилия, Отдел, Оклад). Сколько сотрудников из отдела "Продажи" имеют оклад более 50 000?* Решение: отфильтровать строки, где Отдел="Продажи" и Оклад>50000, затем посчитать такие строки. Если их, скажем, 3 – ответ 3.

Многотабличные запросы. Запрос может объединять таблицы посредством соединения (JOIN) по внешнему ключу. В ЕГЭ могут описать это словами: *«найдите всех учеников и названия олимпиад, где они получили диплом I степени»* – подразумевается соединение таблицы учеников с таблицей участий, отбор по Диплому и выбор полей Фамилия, Название олимпиады. В решении нужно сконцептуально выполнить соединение: сопоставить записи. При небольших таблицах можно вручную выписать комбинации.

SQL-подобный синтаксис: - `INNER JOIN` – соединение по соответствующим записям (оставляет только тех, у кого есть связь). - `LEFT JOIN` – все из левой таблицы + данные из правой при наличии (для ЕГЭ это редкость). - Условия соединения указываются либо через `WHERE` (старый стиль) либо `ON`. Главное – правильно понять связь: напр. `... FROM Ученики U JOIN Участия X ON U.ID = X.ID_ученика JOIN Олимпиады O ON X.ID_олимпиады = O.ID_олимпиады WHERE X.Диплом='I'`.

Обработка и анализ данных в БД. Кроме выборки, нужно знать про сортировку (например, *«вывести трех самых старших студентов»* – значит выбрать и отсортировать по возрасту убыванию, взять топ-3) и про вычисляемые поля (например, запрос может содержать выражение `Цена*Колво AS Сумма` – виртуальное поле). В ЕГЭ часто ограничиваются подсчётами. Пример: таблица продаж с полями (Товар, Месяц, Выручка). *Вопрос:* «Какой суммарный доход за год принес каждый товар?» – требуется сгруппировать по полю Товар и суммировать Выручку.

Пример задачи: По двум таблицам – `Продукты (ProdID, Название, Цена)` и `Продажи (ProdID, Количество)` – составлен запрос: *«SELECT Название, Цена*Количество AS Выручка FROM … WHERE Количество>10»*. Нужно указать, что он выведет. Решение: запрос соединяет таблицы по ProdID, фильтрует продажи свыше 10 шт., и выводит название товара и произведение Цена×Количество (названное Выручка). Если даны конкретные данные, высчитываем для подходящих строк.

Нормализация и структура БД. Теоретически, программа включает понятия нормальных форм БД (убрать дублирование данных по разным таблицам, связи 1:м, м:м через промежуточные таблицы), но глубоко это не проверяется. Скорее спросят: *«Зачем нужна ключевая поле?»* – чтобы однозначно идентифицировать запись. Или: *«Чем чревато хранение повторяющейся информации?»* – несогласованностью при изменениях.

Практический аспект: На ЕГЭ 2025 электронного формата возможно, что будет доступен электронный интерфейс для запросов, но в любом случае готовят так, чтобы ученик понимал логику запросов без непосредственного их исполнения.

Компьютерные сети

Основы сетей. Компьютерная сеть – совокупность компьютеров, соединённых для обмена данными. Основные характеристики: **топология** (шина, звезда, кольцо и пр. – физическая структура соединений), **протоколы** – набор правил обмена данными. В Интернете используется стек протоколов **TCP/IP**, включающий разные уровни: IP для маршрутизации пакетов, TCP для надёжной доставки, UDP для быстрых передач без гарантии, HTTP/FTP/SMTP и прочие – на прикладном уровне. Задачи ЕГЭ могут затрагивать знание названий и функций основных

протоколов (например: HTTP – протокол веб-передачи гипертекста, SMTP – отправка почты, FTP – передача файлов).

IP-адресация. Каждый узел сети Интернет имеет уникальный **IP-адрес** (IPv4 – 32-битное число). Записывается в виде четырех десятичных чис 0–255 через точки, например `192.168.1.34` [3]. IP-адрес делится на две части: **номер сети** и **номер узла в сети** [4]. Определяется это с помощью **маски подсети** – маскирующего числа, которое показывает, какие биты адреса относятся к сети. Маска часто записывается как тоже 4 числа (например, `255.255.255.0`). Биты маски, равные 1, означают бит сети, 0 – бит узла. Например, маска 255.255.255.0 в двоичном виде `11111111.11111111.11111111.00000000` – первые 24 бита адреса это сеть, последние 8 – узел. Для адреса `192.168.1.34` с этой маской номер сети – `192.168.1.0`, номер узла – 34. Все адреса вида `192.168.1.x` принадлежат одной подсети. Количество узлов в подсети зависит от маски: если маска /24 (255.255.255.0), то 2^8 = 256 адресов (минус зарезервированные адрес сети и широковещательный, остаётся 254 реальных узла). Если маска имеет меньше единичных бит, сеть крупнее (больше узлов), если больше – сеть разбивается на более мелкие подсети.

На ЕГЭ нововведение – умение работать с маской подсети. Возможны задачи: - Определить, принадлежат ли два адреса одной подсети (например, IP1= `10.1.12.5`, IP2= `10.1.13.100`, маска `/24`. Решение: сравнить первые 24 бита – для первых трех октетов: `10.1.12` vs `10.1.13` – отличаются, значит разные сети). - Рассчитать число адресов в подсети по маске (например, маска `/26`, то есть 32-26=6 бит под узлы, всего 2^6=64 адреса, 62 доступны узлам). - Найти адрес сети для данного IP и маски (обнулить биты узла). *Пример:* IP `192.168.1.130`, маска 255.255.255.128 (/25). Маска /25 значит первые 25 бит – сеть, то есть 128+64=192 в четвертом октете границы. Адреса сети будут `192.168.1.0`–`192.168.1.127` для первой подсети и `192.168.1.128`–`192.168.1.255` для второй. Наш IP 1.130 попадает во вторую подсеть, адрес сети = 192.168.1.128.

DNS и доменные имена. Поскольку IP-адреса трудно запоминать, введена система доменных имён (DNS). **Доменное имя** – удобный текстовый адрес (например, `school.edu.ru`), который резолвится (преобразуется) в IP-адрес сервера. DNS – иерархическая система: имена состоят из частей, разделённых точками, от частного к общего (ресурс.домен.домен верхнего уровня). В задачах могут спросить: *«Для чего нужна система доменных имён?»* – для сопоставления буквенных адресов компьютеров их числовым IP. Либо дать URL и попросить указать доменное имя (например, в `http://www.site.com/page.html` доменное имя – `www.site.com`).

Сетевые устройства. Полезно знать, что в локальных сетях устройства соединяются через **коммутаторы (switch)**, а для объединения сетей служат **маршрутизаторы (router)**, которые пересылают пакеты между сетями на основе IP-адресов. **Точки беспроводного доступа (Wi-Fi роутеры)** – совмещают функции маршрутизатора, коммутатора и точки Wi-Fi. Модем – устройство для связи с провайдером (ADSL, оптика и т.п.). На экзамене могут быть общие вопросы: *«Какой сетевой прибор обеспечивает выход домашних ПК в Интернет?»* – маршрутизатор. *«Что служит для разделения collision domain в Ethernet?»* – коммутатор (но это углублённо, скорее всего не спросят).

Скорость и объем передаваемых данных. Нужно не путать биты и байты при оценке скорости: 100 Mbit/s = ~12.5 MByte/s. Тоже, что и в разделе про обработку информации: понимать, сколько времени займёт загрузка файла при данном канале.

Протоколы и порты. Каждое сетевое приложение идентифицируется **номером порта** в TCP/UDP (например, веб-сервер – порт 80 для HTTP). Эта деталь может упоминаться, но редко

спрашивается явно. Скорее спросят назначение конкретного протокола: FTP – file transfer, SMTP – почта, DHCP – автоматическое получение IP, DNS – преобразование имен в адреса, HTTP/HTTPS – гипертекст (веб), SSL/TLS – шифрование трафика.

Пример задачи: *«Почему при вводе адреса сайта в браузере зачастую не нужно указывать номер порта?»* – Ответ: потому что используется порт по умолчанию (80 для http, 443 для https) и браузер подставляет его автоматически.

Компьютерная безопасность (в контексте сетей): Основы безопасности могут быть включены: шифрование соединений (HTTPS – HTTP + SSL), понятия брандмауэра, антивируса, социальной инженерии. Например, могут спросить: *«Назовите способ, которым злоумышленник может получить пароль, не взламывая технически систему»* – ответ: метод социальной инженерии (обманным путем выведать у пользователя).

Общие советы: В задачах на сети внимательно читайте формулировку – иногда, помимо вычислительных моментов, проверяется знание терминов. Если спрашивают про "адрес узла" – это часть IP, определяемая маской. Если про "широковещательный адрес подсети" – это адрес, где все биты узлов =1 (например, для сети 192.168.1.0/24 широковещательный 192.168.1.255).

Каждая из перечисленных тем соответствует разделам кодификатора ЕГЭ по информатике. Для успешной подготовки рекомендуется прорабатывать теорию и сразу применять её на практических заданиях ФИПИ или демоверсиях – так вы закрепите понятия и научитесь решать типичные задачи. Каждый раздел курса снабжен примерами задач, аналогичных экзаменационным, с подробным разбором, что позволит ученику с нулевой подготовки постепенно освоить весь материал и уверенно чувствовать себя на экзамене.**

---

1 Определение истинности логического выражения • Информатика | Фоксфорд Учебник
https://foxford.ru/wiki/informatika/opredelenie-istinnosti-logicheskogo-vyrazheniya?srsltid=AfmBOopBfqhXUyxNim_XJPui06kzxJz0s-d2UbdMArxi8xrD3FBB8Tnh

2 Алгоритм: что это такое и его свойства в информатике
https://foxford.ru/wiki/informatika/algoritm-i-ego-svoystva?srsltid=AfmBOooqCBm4Cqv2khOWIuUsgNgQHfcdrYpgnPx_6I7FW6d1cIVedWPxv

3 4 Что такое IP-адрес и маска подсети: простыми словами и на понятных примерах / Skillbox Media
https://skillbox.ru/media/code/chto-takoe-ipadres-i-maska-podseti-i-zachem-oni-nuzhny/