**Динамическое программирование.**

Данный раздел состоит из задач, которые решаются методами динамического программирования.

Суть метода динамического программирования заключается в выделения для задачи следующих составляющих:

1. Состояния задачи –- набор параметров задачи, однозначно характеризующих ее.
2. Целевая функция параметром которой является состояние задачи. Предполагается, что при помощи посчитанной целевой функции для всех состояний задачи, можно легче решить исходную задачу.
3. Базовый случай --- значение целевой функции для не которых базовых состояний.
4. Множество переходов между состояниями задачи с возможностью пересчета значений целевой функции.

Примеры задач: задача о рюкзаке, задаче коммивояжера, подсчет количества сочетаний, алгоритм Форда-Беллмана поиска кратчайших путей в графе.

**Жадные алгоритмы.**

Данный раздел состоит из задач, которые решаются при помощи жадных алгоритмов.

Суть любого жадного алгоритма заключается в оптимизации некоторой целевой функция методом принятии локально оптимальных решений.

Заметим, что, вообще говоря, локально оптимальные решения не всегда приводят к глобальному оптимуму. Например, если рассматривать человеческую жизнь, как череду решений, а целевую функцию --- некоторый показатель успешности, то типичный жадный алгоритм призывает человека как можно раньше начать работать, чтобы стать локально более успешным (т.е. более успешным, чем окружающие сверстники), однако очевидно, что такое решение скорее всего принесет только убытки в будущем относительно потенциально имеющихся возможностей.

Примеры задач: градиентный спуск в случае одного оптимума, алгоритм Крускала нахождения наименьшего остовного дерева, алгоритм Дейкстры поиска кратчайших путей в графе, нахождение паросочетания наименьшего веса в вершинно-взвешенном двудольном графе.

**Реализация.**

Данный раздел состоит из задач, идея решения которых не представляет больших сложностей, однако реализация которых на определенном языке программирования представляет архитектурно сложную задачу в силу подводных камней / тонкостей задачи.

Впрочем, любая задача включает в себя сложности реализации, поэтому данный раздел призван помочь новичку без опыта программирования улучшить навыки владения языком и составления элементарных программ, при этом без необходимости в составлении сложного алгоритма.

Примеры задач: разбор выражений, моделирование поведения системы (например, взаимодействие банкомат — пользователь).

**Вычислительная математика.**

Данный раздел состоит из задач, в которых необходимо произвести математические вычисления на компьютеры, как-то: вычислительная геометрия, нашедшая широкое применение в написании графических движков; использование компьютера для проверки существующих теорем теории чисел и нахождения новых фактов, и т. д.

Примеры задач: вычисление функции Эйлера, факторизация числа, проверка числа на простоту, нахождение площади выпуклого многоугольника, проверка нахождения точки внутри многоугольника, нахождение длины пересечения отрезков.

**Конструктив**

Данный раздел состоит из задач, в которых необходимо построить на компьютере некоторую конструкцию, обладающую указанными свойствами, и предоставить эту конструкцию в удобном для человеческого восприятия виде.

Примеры задач: построение стратегии некоторой игры двух игроков, построение выпуклого многоугольника с точками в целочисленных координатах, нахождение строки с заданным количество различных подстрок, нахождение графа определенной вероятностной структуры с заданным математическим ожиданием длины кратчайшего пути.

**Графы**

Данный раздел состоит из задач, для решения которых используется понятие графа.

Математически граф — это пара <V, E>, где V --- множество вершин графа, E --- множество ребер, соединяющих вершины графа.

Данное простое определение можно расширять, получая кардинально разные интерпретации: от потока вещества по трубам до дружеских взаимоотношений в социальной сети.

Примеры задач: поиск кратчайших путей, поиск потока, построение остовного дерева, поиск паросочетания, построение компонент связности / сильной связности.

**Структуры данных**

Данный раздел состоит из задач, для решения которых используются структуры данных.

Структура данных --- некоторая абстракция, которая принимает на вход набор данных, обрабатывает их, сохраняет некоторую информацию о данных в сжатом виде, имеет возможность при помощи этой информации отвечать на определенные запросы к данным. Часто описанный случай модифицируется таким образом, что данные подаются не один раз структуре на вход, а параллельно с запросами в режиме онлайн.

Примеры структур данных: стэк, очередь, куча, дерево поиска, дерево геометрического поиска.

Примеры задач: нахождение k-ой порядковой статистики с возможностью модификации элементов, нахождение минимального элемента в множестве с возможностью модификации элементов, нахождение количества элементов в заданном диапазоне с возможностью модификации элементов.

**Строки**

Данный раздел состоит из задач, для решения которых используется понятие символьной последовательности или, иначе говоря, строки.

Строка –- это не что иное, как массив символов. Произвольный текст или книга также является строкой. ДНК человека --- еще один пример строки. Бинарная запись числа, как, впрочем, и само число в десятичной привычной записи --- очередная строка.

Примеры задач: нахождение всех вхождений одной строки в другую, нахождение количества различных подстрок заданной строки, нахождение максимального по длине подпалиндрома строки.