Теория

1. Языки программирования. Классификация. Компилятор и интерпретатор.

Языки программирования делятся на процедурные и непроцедурные. **Процедурные** - представляют собой систему предписаний, для решений конкретных задач. Роль компьютера сводится к механическому выполнению этих предписаний. Нужно детальное описание. Основные понятия - оператор и данные. **Непроцедурные** делятся на объективно-ориентированные и декларативные. Процедурные делятся на Низкоуровневые и Высокоуровневые языки программирования. **Низкоуровневый язык** - язык программирования ориентирован на конкретный тип процессора и учитывает его особенности. **Высокоуровневый язык** - Особенность конкретных компьютерных архитектур в них не учитывается, поэтому создаваемые программы, на уровне исходных текстов, легко переносимые на другие платформы, для которых создан транслятор этого языка.

**Компилятор** переводит программу с высокого на низкий уровень. **Интерпретатор** выполняет код без перевода. Программа остается на исходном языке.

1. Парадигмы программирования

Это совокупность идей и понятий, определяющая стиль написания программ. Основные парадигмы программирования: Императивное, Декларативное, Структурное, Объектно-ориентированное, Функциональное и Логическое программирование

1. Типы данных

Это характеристика набора данных, которая определяет: диапазон возможных значений данных из набора; допустимые операции, которые можно выполнять над этими  
значениями; способ хранения этих значений в памяти. Примеры: целые, вещественные, комплексные, логические, символьные

1. Структуры данных

Это контейнер, который хранит данные в определенном макете. Этот «макет» позволяет структуре данных быть эффективной в некоторых операциях и неэффективной в других. Основные структуры данных: массив, стеки, очереди, деки и т.д.

1. Алгоритм. Алгоритмизация. Свойства алгоритма. Способы записи

**Алгоритм** – последовательность инструкций, правильное выполнение которых ведет к решению поставленной задачи. **Свойства:** Однозначность, дискретность, результативность, массовость, конечность. **Способы записи:** Словесно - формульное описание, Блок-схема, Алгоритмические языки, Операторные схемы, Псевдокод.

1. Сложность алгоритма

Это количественная характеристика, которая говорит о том, сколько времени потребуется на выполнение алгоритма. Порядок роста: Константный(О(1)), Линейный(О(n)), Логарифм.(О(log n)), Квадратичный(О(n^2)).

1. Массивы.

Это структура однотипных данных, занимающих непрерывную область памяти. Массив имеет размер - кол-во эл в нем, каждый эл. имеет свой номер - индекс, обращение к элементу массива осуществляется к указанию его индекса. **Одномерный массив** содержит одну строку, **двумерный массив** имеет более одной строки, т.е. матрица. В **динамическом массив**е могут изменяться размеры массива, в **статическом** нет.

1. Пузырьковая сортировка

На каждом шаге мы находим наибольший эл из двух соседних и ставим этот эл в конец пары, получается, что при каждом прогоне цикла большие эл будут всплывать к концу массива.

1. Сортировка вставкой

Второй элемент массива сравнивается с пред. элементом, если второй элемент меньше, произойдет замена местами. Третий элемент сравнивается с перед ними, если третий меньше другого, замена. Если третий больше первого, но меньше второго, вставка в 2-й слот. Сортировка повторяется n - 1 раз (n кол-во элементов).

1. Сортировка выбором

Алгоритм сортировки выбором заключается в поиске на необработанном срезе массива мин значения и в дальнейшем обмене этого значения с первым эл необработанного среза. На след шаге необработанный срез уменьш. на 1 эл, продолжать выполнять поиск и обмен пока не будет достигнут конец списка или массив.

1. Сортировка Шелла

Сравнение разделенных на группы элементов, находящихся друг от друга на некотором расстоянии. Это расстояние = D или n/2 (n общее кол-во элементов). Расстояние сокращается в два раза после 1 круга. Круг состоит из того, что берется первое число и число через n/2 элементов, потом они сравниваются и меняются местами и т. д.

1. Сортировка Хоара

Сначала выбирают элемент массива и назвать его опорным, потом элементы перераспределяются так, что элементы меньше опорного будут перед ним, а большие или равные после. Рекурсивно применить первые 2 шага к двум подмассивам (слева и справа от опорного).

1. Сортировка слиянием

Делим массив пополам, каждый из них сортируем слиянием и потом соединяем оба массива. Каждый разделённый массив тоже нарезаем на два подмассива до тех пор, пока в каждом не окажется по одному элементу. Соединение отсортированных элементов между собой, причём тоже хитрым способом: раз оба массива уже отсортированы, то нам достаточно сравнивать элементы друг с другом по очереди и заносить в итоговый массив данные по порядку.

1. Бинарный поиск

Используется только для отсортированных по возр или убыв последовательностей. Заключается в определении содержит ли массив искомое значение, а также в определении места его нахождения. Сначала находится средний элемент, для этого первый и последний индексы связываются с переменными, а индекс среднего вычисляется. Потом значение среднего элемента сравнивается с искомым значением, после дальнейший поиск будет либо слева, либо справа. Если значение равно искомому, поиск завершается.

1. Стек

Это структура данных, в которой элементы добавляются и удаляются в вершине стека. Массив элементов организован по принципу LIFO. Элементы забирают и добавляют сверху. Поскольку операции вставки и извлечения осуществляются только с одного конца массива, то для моделирования стека можно использовать лист, который имеет аналогичные методы

1. Дек

Это структура данных, представляющая из себя список элементов, в котором добавление новых элементов и удаление существующих производится в обоих концов массива. Это позволяет многие задачи решать непосредственно решать с помощью дека, абстрагируясь до нужной структуры данных. Такие структуры как очередь и стек создаются на основе дека.

1. Очередь

Это структура данных, которой добавление элементов возможно лишь в конец очереди возможно лишь в конец очереди, а выборка либо взятие элемента только из начала очереди при этом выбранный элемент из очереди удаляется. Массив элементов организован по принципу FIFO.

1. Связный список

Это структура данных, состоящая из узлов, содержащих данные и ссылки (связки) на следующий или предыдущий узел списка. Принципиальным преимуществом перед массивом является структурная гибкость, те порядок элементов связного списка может не совпадать с порядковым расположением расположения данных в памяти компьютера, а порядок обхода списков всегда явно задается его внутренними связями. Списки бывают: односвязный, двусвязный, кольцевой.

1. Граф

Это совокупность точек, соединённых в линии. Точками называют вершинами или узлами, а линии рёбрами или дугами. Виды графов: ориентированный и неориентированный (Зависит от того есть ли направления), Взвешенный или невзвешенный (Имеют ли значения ребра).

1. Поиск в глубину и ширину на графе

В процессе поиска идем в **глубь** графа настолько на сколько это возможно. Следуя алгоритму, последовательно обойдем все вершины графа, которые доступны изначальному. Если ребро ведет в не пройденную до этого момента вершину, то алгоритм запускается с нее. В случае если ребер, которые ведут в не пройденную вершину больше нет, то идет обратно

Алгоритм позволяет найти кратчайший или содержащий наименьшее кол-во ребер путь из одной вершины графа для всех остальных вершин. В нем сначала посещаются все вершины смежные с текущей, а потом все потомки.

1. Алгоритм Дейкстры

Поиск кратчайшего пути по схеме, где множество точек соединены между собой отрезками. Алгоритм: В графе на всех узлах выставляется значение бесконечности, кроме начального узла. Посещаются все соседние узлы от начального и на узлах значение меняется на значение ребра. Если значение имелось, то прибавляется значение к уже имевшимуся и сравниваются числа. Алгоритм повторяется до тех пор пока не посетим все узлы графа

1. Дерево

Это связный граф без петель. Виды деревьев: **Двоичное** - это иерархическая структура данных каждый узел имеет не более 2 потомков, как правило данный узел называется родительским узлом, а дети называются левыми и правыми наследник. **Сбалансированное** - это дерево, в котором заполнены все уровни возможно кроме последнего чем ближе дерево поиска ближе к сбалансированному тем эффективнее операции с ним. Добавление или удаление элемента дерева поиска легко нарушает сбалансированность. Поддержание сбалансированности слишком дорого, на практике чаще используется почти сбалансированные деревья

1. Методы обхода дерева

Основной рекурсивный подход для обхода дерева: Начиная с узла N делаем следующее: (L) Рекурсивно обходим левое поддерево. Этот шаг завершается при попадании опять в узел N. (R) Рекурсивно обходим правое поддерево. Этот шаг завершается при попадании опять в узел N. (N) Обрабатываем сам узел N.

1. Двоичная куча

Это такое двоичное дерево, для которого выполнены условия: значение в любой вершине е больше чем в начальной, на случайном слое вершин 2\*\*i кроме последнего, слои нумеруются с нуля, последний слой заполнен слева направо. Высота кучи это: О(log n), где n кол-во узлов дерева

Практика

1. Составить блок-схемы к различным алгоритмам
2. Показать различные алгоритмы по шагам (итерации)
3. Реализовать алгоритм на Python без использования интернета и заготовок
4. Привести пример какого-то алгоритма на графе, дереве, двоичной куче