**Bubble Sort** — Алгоритм просматривает массив и сравнивает каждую пару соседних элементов. Когда он встречает пару элементов, расположенных не по порядку, происходит замена двух элементов местами.

Этот алгоритм почти не применяется на практике из-за низкой эффективности: он медленно работает на тестах, в которых маленькие элементы стоят в конце массива. Однако на нём основаны многие другие методы, например, шейкерная сортировка и сортировка расчёской.

**Insertion Sort** - Этот алгоритм разделяет оригинальный массив на сортированный и несортированный подмассивы.

Длина сортированной части равна 1 в начале и соответствует первому (левому) элементу в массиве. После этого остается итерировать массив и расширять отсортированную часть массива одним элементом с каждой новой итерацией.

После расширения новый элемент помещается на свое место в отсортированном подмассиве. Это происходит путём сдвига всех элементов вправо, пока не встретится элемент, который не нужно двигать.

**Selection Sort -** Сортировка выбором тоже разделяет массив на сортированный и несортированный подмассивы. Но на этот раз сортированный подмассив формируется вставкой минимального элемента, не отсортированного подмассива в конец сортированного.

Данная сортировка неустойчива, т.к. одинаковые элементы (с точки зрения той характеристики, по которой мы сортируем элементы) могут изменить своё положение.

**Shuttle Sort** - Суть алгоритма в том, что мы итерируемся слева направо, при этом при выполнении swap элементов мы выполняем проверку всех остальных элементов, которые остались позади, не нужно ли повторить swap.

**Пирамидальная сортировка**

Для понимания работы пирамидального алгоритма сортировки нужно понять структуру, на которой он основан – пирамиду.

Пирамида или двоичная куча – это дерево, в котором каждый узел состоит в отношениях с дочерними узлами. Добавление нового узла начинается с левой позиции нижнего неполного уровня.

По мере движения вниз по дереву значения уменьшаются (**min-heap**) или увеличиваются (**max-heap**).

Двоичное дерево поиска имеет специальную структуру хранения элементов. Двоичное дерево состоит из элементов,каджый элемент имеет 3 доп. ссылки помимо полезных данных. Ссылку на левого потомка,правого потомка и на родительский элемент,где хранятся полезные данные.Элементы в двоичном дерве хранятся таким образом:если некоторый элемент является ключом дерева,то в левом поддереве у нас хранятся элементы меньше либо равные искомому элементу и в правом поддереве хранятся элементы больше либо равные родительскому элементу соответственнно.Для того чтобы осуществить поиск мы должны на каждом шаге поиска сравнить элемент с искомым и пойти либо в правое поддерево,либо в левое.В этом случае отсекается в лучшем случае половина элементов.Поиск осуществляется за счет структурирования данных,а не за счет их сортировки.Если левого или правого поддерева нет,то специальные ссылки равны 0.Ссылка parent может быть равна 0,только если этот элемент самый верхний в дереве.

Интерфейс: search-возвращает узел дерева с заданным ключом

Min-вовзращает узел дерева с минимальным ключом.

Max- вовзращает узел дерева с максимальным ключом.

Successor-возвращает узел, следующий относительно заданного

Predessor - возвращает узел, предыдущий относительно заданного

Insert –добавляет узел в дерево

Delete –удаляет узел из дерева

Разновидности двоичных деревьев поиска

Двоичное дерево поиска

АВЛ-дерево

Красно-чёрное дерево

Матричное дерево

Дерево Фибоначчи

Суффиксное дерево

Преимущества и недостатки

АВЛ- дерево

Плюсы: Гарантирует выполнение любой операции за время пропорциональное логорифму от n.

Минусы: Сложно сохранят структуру дерева при вставке

Красно-черное дерево

Плюсы: для удобства сам класс дерева имеет элемент «заглушка». Минусы: Сложно сохранят структуру дерева при вставке