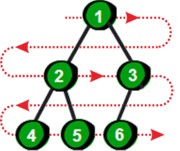
* 插入排序：现有无序数列以及有序数列，现需将无序数列插入有序数列并保持有序。新建空白数组，使其长度比有序数组大1，复制有序数组到空白数组中。从无序数组中依次拿出一个数n，遍历有序数组各个键值与该数作比较，得到目标键i，将i右侧数据全部右移一格，因此在i处空出一格，在此插入n。如此遍历无序数组，最终将无序数列内容全部织入有序数列。
* 希尔排序：插入排序遍历无序数列的每一格数据n，并携带n再去遍历有序数列的每一项做比较，但是如果能大致确定n可能会出现在有序数列的哪一段，就可以节省很多不必要的计算，希尔排序就是在此基础上进行的改进。一列很长的无序数组，首先，每5格取一次数据，能得到5串子列，分别将子列进行插入排序并还原到原数组；其次，每2格取一次数据，能得到2串子列，分别进行插入排序并还原到原数组；最后，对得到的数组进行一次插入排序，得到最终的有序数组。希尔排序在无序数列的不同层次之间分别进行了先期的插入排序，这样减少了原始插入排序消耗运算的数量级。
* 选择排序：设定一个临时变量t，将无序数列的第一个值作为t的初始值，遍历该无序数列的每一个值i并与t做比较，如果i小于t，则使用i覆盖t，最终得出当前无序数列的最小值，拿出该最小值追加到有序数列，无序数列逐渐缩小并得出有序数列。
* 冒泡排序：选择排序需要额外的空间来盛放有序数列，因此出现了更节约空间的冒泡排序算法。首先，遍历无序数列，将i与i+1格子内的数据进行对比，彼此交换位置，将值更小的数据，放置在i+1处，这样，当第一次遍历完成，该无序数列最右侧格子内的数组一定是最小值，这样的遍历按照无序数列长度n进行n次，即可逐步将小数据冒泡到数列尾部得到有序数列。
* 归并排序：一个无序数列，选连续的两个格子为一组并比较大小，放入新的数组，得到连续的长度为2的多个子列，选连续的两个子列为一组，设两个指针分别指向这两个子列的第1格并比较大小，以此比较并移位指针，所得数据逐个放入一个新数组，以此得到两个子列合并之后的新数列。如此反复，直到所有子列合并得出全新的有序数列。
* 快速排序：一列无序数据，从中选择任意一个格子得到数据n，遍历无序数列，逐个与格子内数据n进行对比，小的排左边，大的排右边，完成之后再在左、右数列内执行同样的操作，得到四个新的子数列，反复进行，直到子数列长度为2不能再分割，最终得到有序数列。
* 基数排序：回到数字本身，以十进制为例，两位数肯定大于一位数，3x肯定大于1x，xy5肯定大于xy2，基线排序就是根据数字的这些基本特点。对于一个无序数列，我们构造0~9个数字桶，从个位开始依次到最高位，第一步，遍历数列，将个位数为0~9的数字放入对应的桶中，遍历完成将所得数据串起来形成新的无序数列，但该无序数列的个位数层面为有序，第二步，遍历新数列，将十位数放入0~9的数字桶，遍历完成得到新数列，该数列在十位以及个位数层面为有序，依次执行，直到运行到最大数的最高位完成，所得数列串起来得到完全的有序数列。
* 堆排序：父节点与子节点具有树形的包含性层次关系，这样的结构称为树，族谱图、晋级图、架构图都是树。如果树中任意一个父节点拥有最多两个子节点，这种关系称为二叉树。如果每个父节点都拥有两个子节点，这称为满二叉树。如果除了最底层节点之外，每一层节点都是满的，那就是完全二叉树。我们以从左到右、从上到下的方式，将有序数列填入到二叉树中，像这种含有特殊反S型数据顺序结构的树称为堆，如果任意父节点大于任意子节点，则构成大顶堆，反之则构成小顶堆。像这种特殊的称为堆的二叉树具有更高效的数据遍历能力。在典型二叉关系中，只需要进行至多两次数据互换就能将最大或者最小数据推至顶点，而最大/最小值就在这样层层推举的过程中到达顶点，计算复杂度远远小于扁平的数列遍历，可以说，堆（数据结构）以二叉形结构优化减少了计算复杂度。
* 
* 人们发现，物质内部各个部分不同的组织结构，决定了该物质的独有特性。将结构哲学运用到复杂数据的内部组织中，人们提炼出了优化易用的数据结构设计。从读写效率的层面看：数组是非常常用的数据结构，它是内存中固定长度的一片连续区域，因为内存地址连续，读效率非常高，只需指针移动固定长度就能找到相应数据，也因为地址连续固定，写效率非常低，因为要顺延所有其他位置的内存地址；与数组对应的是链表，链表在内存中是非连续的一串节点，每一个节点都分别包含了用于存放指针和数据的区域，指针指向的是下一个节点的内存地址，这样的特殊结构易于写数据，因为这只是意味着改变某个节点的指针指向，然而，链表结构不利于读数据，因为读数据要遍历所有节点；数组通过指针偏移访问数据，链表通过逐个遍历节点来访问数据，有没有一种更高效的办法，无需比较就能直达数据？散列/哈希表便是这样的一种数据结构，它拥有键与值两部分，通过对键进行哈希运算得到一个对应的hash值，然后将数据存放到与该hash值对应的一个固定位置，这样读写数据都非常直观快速，但hash算法消耗了更多的内存运算。从数据时序的层面看：栈和队列都是一种复杂的用于存储流动性数据的容器，栈是一端封闭，另一端作为入口和出口，所以先进后出，后进先出；队列是无封闭，两端开放的，数据总是从列尾进，列头出，所以先进先出；对于普通队列进一步优化，每次进入数据时为其指定优先级，队列内部根据该优先级来决定哪个数据最先出列，这就是优先队列。有一种情况就是，数据具有层级和节点关系，就像是一棵树，层级和节点两个维度构成了指针，指向数据，这种数据结构叫做树，树用于数据具有层级关系的情况。有时候，数据具有不确定性，比如从A点到C点可以有很多路线，不同情况下，最优路线总不一样，于是，人们将多个点连成面，点与点之间的连线用来表示数据，比如ABCD四个点连成线，我们将AB线记为x，BC线记为y，AC线记为z，那么，从A到C，就是f(x, y)与f(z)的比较，这样的数据结构成为面，面用于数据具有不确定性结果的情况。