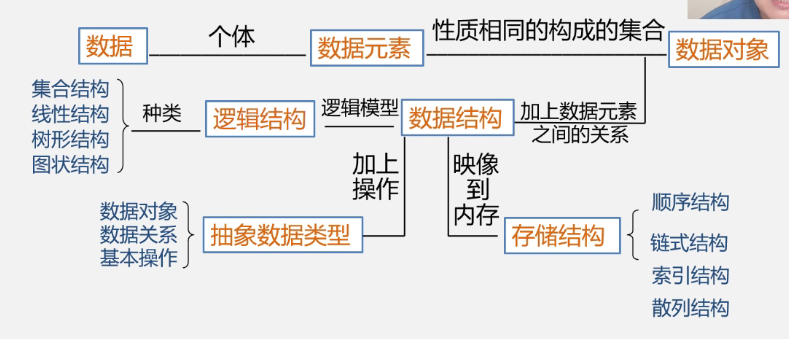
1. 数据结构；逻辑结构（数据元素之间的逻辑关系），物理结构或者存储结构（数据元素以及其关系在内存中的表示），数据的运算和实现
2. 
3. 程序=数据结构+算法
4. 算法的特性：有穷，确定，可行，输入，输出
5. 算法时间复杂度：算法每条语句的执行次数的和，取其数量级O(f(n))，渐进时间复杂度

for(i=1,i<=n,I++){

for(j=1,j<=i,j++){

}

}

复杂度计算=级数求和

1. .net 使用c#实现的常见数据结构：
2. Array数组 ：内存连续，类型相同，指定长度，插入需要移动元素，扩容存在gc
3. Arraylist 动态数组：动态添加和删除，可以类型不同都是object，存在装箱拆箱，类型不安全，不建议使用
4. List类型安全，泛型，很好用
5. Linklist 双向链表，随机存储，适合插入删除，不能随机访问
6. Stack堆栈
7. Queue队列
8. Dictionary字典，内部数组加链表存储，哈希函数映射索引，拉链法处理冲突
9. Hashtable 散列表（不建议使用），类型都是object存在大量装箱拆箱
10. Hashset 集合 不可重复，求交集，并集等集合的操作
11. 算法思想：分治思想，动态规划，如果没有好的解决办法就使用贪婪算法
12. 二分查找：数组有序，时间复杂度log(n)
13. 时间复杂度：线性时间和非线性，表示的是普遍情况的时间，从小到大log n，n，n\*logn，n^2，n!
14. 选择排序：每次循环找出一个最大值存入新的数组，时间复杂度n^2
15. 递归：使用递归程序清晰易懂，有时候循环的性能可能更好，结构基线条件：函数结束的条件，不在调用自己；递归条件：函数调用自己，有返回值的计算返回值，多个返回值的可以定义个类辅助返回；递归很适合解决链表和树的问题
16. 函数的调用栈：循环和递归存在差异，递归将会建立很长的调用栈用来存储数据，而循环是申请了托管内存，所以递归死循环会栈溢出
17. 分而治之D&C：1、找出基线条件 ，多数为数组长度小于等于1

2、确定如何缩小问题的规模，使其符合基线条件

1. 快速排序：典型的分治案例，找出一个参考值，将数组小于的放左边，将大于的放右边，分区得到左右两个数组，依次迭代直到左右数组长度小于等于1结束，时间复杂度nlogn
2. 合并排序复杂度和快速排序相同，但是系数大，还是快速排序效率高
3. 哈希函数或者散列函数，给一个输入有唯一的输出，但是不同的输入可能有相同的输出，也就是哈希冲突，查询和插入的时间复杂度都是O（1），插入和删除并不会移动内存，只是数据的填充和置空，内部存储是无序的
4. 解决冲突常用的方法就是在冲突位置创建链表，保证更少的冲突，需要合适的填充因子和哈希函数
5. 广度优先算法（breadth-first search）：非加权图，寻找最少步骤，实现方法创建图来模拟模型，然后创建队列存储要处理的数据，必须按分级顺序处理，从上层开始进队列，先进先出，还需要添加一个数组记录已经处理过的节点，处理过的直接跳过
6. 算法图：节点和边组成，使用哈希表来实现，节点作为key，节点的邻居组成数组作为value，箭头边是有向图，线是无向图
7. 狄克斯特拉算法（dijkstra）：处理加权图，使用于正加权，有向无环图，处理最快路径的问题，使用散列表构建节点权重，再嵌套一个散列表存储所有节点，每次循环找出开销最小的节点，循环条件待处理的节点不为空，处理过的节点不再处理
8. 贪婪算法：每步选择局部最优解，最终得到全局最优解（近似），处理NP完全问题，如集合覆盖问题，旅行商问题，识别NP完全问题设计集合，随着元素数量增加，速度变化很快
9. 常见贪婪算法：A\*，广度优先，狄克斯特拉
10. 动态规划算法：可以得到最优解，首先构建网格，然后将复杂的问题分解，找出最简单的情况，然后依次填充表格，提炼每次表格的计算方法