数据（data）是对客观事物的符号表示，在计算机科学中是指所有能输入到计算机中并被计算机程序处理的符号的总称。

数据元素（data element）是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。

数据项：有时，一个数据元素可由若干个数据项（data item）组成。数据项是数据的不可分割的最小单位。

数据对象（data object）是性质相同的数据元素的集合，是数据的一个子集。

|  |  |  |
| --- | --- | --- |
| 数据结构（data structure）是相互之间存在一种或多种特定关系的数据元素的集合。数据元素相互之间的关系称为结构（structure）。  形式定义为：数据结构是一个二元组Data\_Structure。其中，是数据元素的有限集，是上关系的有限集。 | | |
| ┣ | 逻辑结构：结构定义中的“关系”描述的是数据元素之间的逻辑关系，因此又称为数据的逻辑结构。 | |
| ┃ | ┣ | 集合：结构中的数据元素之间除了“同属于一个集合”的关系外，别无其他关系。 |
| ┃ | ┃ | 线性结构：结构中的数据元素之间存在一个对一个的关系。 |
| ┃ | ┃ | 树形结构（树型结构）：结构中的数据元素之间存在一个对多个的关系。 |
| ┃ | ┗ | 图状结构（网状结构）：结构中的数据元素之间存在多个对多个的关系。 |
| ┗ | 物理结构（存储结构）：数据结构在计算机中的表示（又称映像）称为数据的物理结构，又称存储结构。  位：在计算机中表示信息的最小单位是二进制数的一位，叫做位（bit）。  元素（结点）：在计算机中，我们可以用一个由若干位组合起来形成的一个位串表示一个数据元素，通常称这个位串为元素（element）或结点（node）。  数据域：当数据元素由若干数据项组成时，位串中对应于各个数据项的子位串称为数据域（data field）。 | |
|  | ┣ | 顺序映像的特点是借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系。  顺序存储结构 |
|  | ┗ | 非顺序映像的特点是借助元素存储地址的指针（pointer）表示数据元素之间的逻辑关系。  链式存储结构 |

任何一个算法的设计取决于选定的数据（逻辑）结构，而算法的实现依赖于采用的存储结构。

|  |  |  |
| --- | --- | --- |
| 数据类型（data type）是和数据结构密切相关的一个概念，它最早出现在高级程序语言中，用以刻画（程序）操作对象的特性。数据类型是一个值的集合和定义在这个集合上的一组操作的总称。 | | |
| ┗ | 按值的不同特性（!） | |
|  | ┣ | （非结构的）原子类型：原子类型的值是不可分解的。 |
|  | ┗ | 结构类型：结构类型的值是由若干成分按某种结构组成的，因而是可以分解的，并且它的成分可以是非结构的，也可以是结构的。 |

固有数据类型：处理器中已经定义并实现的数据类型（原子类型、结构类型）。

|  |  |  |
| --- | --- | --- |
| 抽象数据类型（Abstract Data Type,ADT）是指一个**数学模型**以及定义在该模型上的一组操作。  抽象数据类型的定义仅取决于它的一组逻辑特性。  抽象数据类型和数据类型实质上是一个概念。  和数据结构的形式定义相对应，抽象数据类型可用以下三元组表示。其中，是数据对象，是上的关系集，是对的基本操作集。 | | |
| ┗ | 按值的不同特性（！） | |
|  | ┣ | 原子类型（atomic data type）：属原子类型的变量的值是不可分解的。 |
|  | ┣ | 固定聚合类型（fixed-aggregate data type）：属该类型的变量，其值由确定数目的成分按某种结构组成。 |
|  | ┗ | 可变聚合类型（variable-aggregate data type）：和固定集合类型的值相比较，构成可变聚合类型“值”的成分的数目不确定。 |

固定聚合类型和可变聚合类型统称结构类型。

多形数据类型（polymorphic data type）是指其值的成分不确定的数据类型。

|  |  |  |  |
| --- | --- | --- | --- |
| 算法（algorithm）是对特定问题求解步骤的一种描述，它是指令的有限序列，其中每一条指令表示一个或多个操作。 | | | |
| ┣ | 特性 | | |
| ┃ | ┣ | 有穷性 | |
| ┃ | ┣ | 确定性 | |
| ┃ | ┣ | 可行性 | |
| ┃ | ┣ | 输入 | |
| ┃ | ┗ | 输出 | |
| ┣ | 描述 | | |
| ┃ | ┣ | 文字 | |
| ┃ | ┣ | 流程图 | |
| ┃ | ┗ | 代码 | |
| ┗ | 要求 | | |
|  | ┣ | 正确性 | |
|  | ┣ | 可读性 | |
|  | ┃ | ┣ | 标识符名称 |
|  | ┃ | ┣ | 空白字符 |
|  | ┃ | ┗ | 注释 |
|  | ┣ | 健壮性 | |
|  | ┗ | 效率与低存储量需求 | |

线性表：个数据元素的有限序列。

一个数据元素可由若干个数据项（item）组成。常把数据元素称为记录（record），含有大量记录的线性表又称文件（file）。

线性表的长度：线性表中元素的个数（）。

空表：。

线性表的顺序表示：用一组地址连续的存储单元依次存储线性表的数据元素。

线性表的起始位置（基地址）：第一个数据元素的存储位置。

线性表的顺序存储结构或顺序映像（sequential mapping）：...。通常，称这种存储结构的线性表为顺序表。

~~线性表的存储结构~~顺序表是一种随机存取的存储结构。

线性表的链式存储结构的特点是用一组任意的存储单元存储线性表的数据元素（这组存储单元可以是连续的，也可以是不连续的）。

结点（node）：数据元素本身的信息和指示其直接后继的信息（即直接后继的存储位置）组成数据元素的存储映像。

数据域：存储数据元素信息的域。

指针域：存储直接后继存储位置的域。

指针（链）：指针域中存储的信息。

链表（线性表的链式存储结构、线性链表、单链表）：个结点（（）的存储映像）链结。

单链表是非随机存取的存储结构。

静态链表：用数组描述的链表。

循环链表（circular linked list）

双向链表（double linked list）

栈（stack、后进先出（last in first out）的线性表|LIFO结构）：限定仅在表尾进行插入或删除操作的线性表。

栈顶（top）：表尾端。

栈底（bottom）：表头端。

空栈：不含元素的空表。

假设，则称为栈底元素，为栈顶元素。

顺序栈，即栈的顺序存储结构是利用一组地址连续的存储单元依次存放自栈底到栈顶的数据元素，同时附设指针top指示栈顶元素在顺序栈中的位置。

队列（queue）是一种先进先出（first in first out，缩写为FIFO）的线性表。它只允许在表的一端进行插入，而在另一端删除元素。

队尾（rear）：允许插入的一端。

队头（front）：允许删除的一端。

假设队列为，那么，就是队头元素，则是队尾元素。

双端队列（deque）：限定插入和删除操作在表的两端进行的线性表。这两端分别称做端点1和端点2。

链队列：用链表表示的队列。一个链队列需要两个分别指示队头和队尾的指针（分别称为头指针和尾指针）才能唯一确定。

循环队列：将顺序队列臆造为一个环状的空间。

二维数组有两种存储方式:一种以列序为主序（column major order）的存储方式；一种是以行序为主序（row major order）的存储方式。

每维长度，元素下标，元素地址等于乘第维每个元素大小（第维的一个元素的首地址）加乘第维每个元素大小（选中的第维的元素里面第维的一个元素的首地址）加……加到乘第维每个元素大小，等于乘sizeof(元素类型)

压缩存储：为多个值相同的元只分配一个存储空间；对零元不分配空间。

特殊矩阵：值相同的元素或者零元素在矩阵中的分布有一定规律。

稀疏矩阵：否则。

对称矩阵的压缩存储：以行序为主序存储其下三角（包括对角线）中的元。

一维数组长度是，给元素在矩阵中的下标、（都从开始。。。），求元素在一维数组中的下标：。

这种压缩存储的方法同样也适用于三角矩阵。

下（上）三角矩阵：矩阵的上（下）三角（不包括对角线）中的元均为常数或的阶矩阵。除了和对称矩阵一样，只存储其下（上）三角中的元之外，再加一个存储常数的存储空间即可。

对角矩阵：所有的非零元都集中在以主对角线为中心的带状区域中，即除了主对角线上和直接在对角线上、下方若干条对角线上的元之外，所有其它的元皆为。对这种矩阵，可按某个原则（或以行为主，或以对角线的顺序）将其压缩存储到一维数组上。

稀疏矩阵：非零元较零元少，且分布没有一定规律。

矩阵的稀疏因子（||）：在的矩阵中，有个元素不为，。

时称为稀疏矩阵。

稀疏矩阵的压缩存储：只存储稀疏矩阵的非零元，除了存储非零元的值之外，还同时记下它所在的行和列的位置。一个三元组唯一确定了矩阵的一个非零元。稀疏矩阵可由表示非零元的三元组和其行列数唯一确定。

三元组顺序表（有序的双下标法）：以顺序结构表示三元组表。data[0]未用。data域中表示非零元的三元组是以行序为主序排列的。

稀疏矩阵快速转置：开一个数组存储原矩阵每列元素个数（转置矩阵每行元素个数），再开一个数组求那个数组的前缀和（转置矩阵每行第一个元素在三元组顺序表中下标）。注意矩阵中元素下标从开始，所以不会有差一错误。转置时，遍历原矩阵的三元组顺序表，按开的第二个数组指示的位置将元组填入转置矩阵的三元组顺序表。首先转置矩阵的三元组顺序表中“行号”这一列是有序的。因为原矩阵的三元组顺序表是以行序为主序的，所以转置矩阵的三元组顺序表中行号相同的元组中“列号”这一列也是有序的。

行逻辑链接的顺序表：将各行第一个非零元下标的数组固定在稀疏矩阵的存储结构中。

稀疏矩阵相乘：

对的每一行

开一个数组，长度为的列数，初值全0

对的每一行的每个非零元

在中用“那个”数组找到第行在三元组顺序表中的范围，对的第行的每个非零元

加到刚开的数组中下标为的元素中。

对刚开的数组中的每个元素

如果不为0，加入

树（Tree）：（）个结点的有限集。在任意一棵非空树中：（1）有且仅有一个特定的称为根（Root）的结点；（2）当时，其余结点可分为（）个互不相交的有限集，其中每一个集合本身又是一棵树，称为根的子树（SubTree）。

树的结点包含一个数据元素及若干指向其子树的分支。

结点的度（Degree）结点拥有的子树数。

叶子（终端结点）：度为的结点。

非终端结点（分支结点）：度不为的结点。

内部结点：除根节点之外，分支结点。

树的度：树内各节点的度的最大值。

结点的孩子（Child）、孩子的双亲（Parent）：结点的子树的根、该结点。

同一个双亲的孩子之间互称兄弟（Sibling）。

结点的祖先：从根到该结点所经分支上的所有结点。

结点的子孙：以某结点为根的子树中的任意结点。

结点的层次（Level）：根为第层。若某结点在第层，则其子树的根在第层。

双亲在同一层的结点互为堂兄弟。

树的深度（Depth）（高度）：树中结点的最大层次。

有序树：将树中结点的各子树看成从左至右是有次序的（即不能互换）。

无序树：否则。

第一个孩子：有序树中最左边的子树的根。

最后一个孩子：有序树中最右边的子树的根。

森林（Forest）：（）棵互不相交的树的集合。

二叉树（Binary Tree）：每个结点至多有棵子树，子树有左右之分。

空二叉树：。

在二叉树的第层上至多有个结点。

深度为的二叉树至多有个结点。

对任何一棵二叉树，如果其终端结点数为，度为的结点数为，则。

满二叉树：一棵深度为且有个结点的二叉树。

对满二叉树的结点进行连续编号，约定编号从根节点起，自上而下，自左至右。

完全二叉树：深度为的，有个结点的二叉树，每一个结点都与深度为的满二叉树中编号从至的结点一一对应。

具有个结点的完全二叉树的深度为、。

如果对一棵有个结点的完全二叉树的结点按层序编号,则对任一结点，有

（1）如果，则结点是二叉树的根；如果，则其双亲是结点。

（2）左孩子是结点（如果）。

（3）右孩子是结点（如果）。

顺序存储结构：用一组地址连续的存储单元依次自上而下、自左至右存储完全二叉树上的结点元素，即将完全二叉树上编号为的结点元素存储在如上定义的一维数组中下标为的分量中。

链式存储结构：二叉链表和三叉链表。链表的头指针指向二叉树的根节点。

遍历二叉树（traversing binary tree）：按某条搜索路径巡访树中每个结点，使得每个结点均被访问一次，而且仅被访问一次。

先（根）序遍历、中（根）序遍历、后（根）序遍历、层序遍历

线索链表：结点结构构成的二叉链表作为二叉树的存储结构。

线索：线索链表中指向结点直接前驱和直接后继的指针。

线索二叉树：加上线索的二叉树。

先序线索二叉树、中序线索二叉树、后序线索二叉树

线索化：对二叉树以某种次序遍历使其变为线索二叉树的过程。

树的存储结构

双亲表示法：以一组连续空间存储树的结点，同时在每个结点中附设一个指示器指示其双亲结点在链表中的位置。

孩子表示法：用多重链表，即每个结点有多个指针域，其中每个指针指向一棵子树的根节点。链表中的结点右两种结点格式。或将每个结点排在一个数组中，每个结点附一个链表存所有孩子。

孩子兄弟表示法（二叉树表示法、二叉链表表示法）：以二叉链表作为树的存储结构，链表中结点的两个链域分别指向该结点的第一个孩子结点和下一个兄弟结点。

树的遍历：先根（次序）遍历、后根（次序）遍历

森林的遍历：先序遍历、中序遍历

路径长度：从树中一个结点到另一个结点之间的分支构成这两个结点之间的路径，路径上的分支数目。

树的路径长度：从树根到每一结点的路径长度之和。

结点的带权路径长度：从该结点到树根之间的路径长度与结点上权的乘积。

树的带权路径长度（||）：树中所有叶子结点的带权路径长度之和。

赫夫曼（Huffman）树（最优树、最优二叉树）：有个权值，构造一棵有个叶子结点的二叉树，每个叶子结点带权为，其中带权路径长度最小的二叉树。

前缀编码：任一个字符的编码都不是另一个字符的编码的前缀。

严格的（strict）二叉树（正则的二叉树）：没有度为的结点。

顶点（Vertex）：在图中的数据元素。

：顶点的有穷非空集合。

：两个顶点之间的关系的集合。

从到的一条弧（Arc）（||）：。

弧尾（Tail）（初始点（Initial node））：。

弧头（Head）（终端点（Terminal node））：。

有向图（Digraph）

和之间的一条边（Edge）（||）：，即是对称的，以无序对代替这两个有序对。

无向图（Undigraph）

图的谓词（||）：从到的一条单向通路。

图中顶点数目（||）

边或弧的数目（||）

完全图（Completed graph）：有条边的无向图。

有向完全图：具有条弧的有向图。

稀疏图（Sparse graph）：有很少条边或弧（如）的图。

稠密图（Dense graph）：否则。

权（Weight）：与图的边或弧相关的数。

网（Network）：带权的图。

（的）子图：，，，。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |