# 数据结构

数据结构包括**线性结构**和**非线性结构**；

## 线性结构

1. 线性结构是最常见的数据结构，特点是数据元素之间存在一对一的线性关系；
2. 线性结构有两种不同的存储结构，顺序存储结构和链式存储结构：

顺序存储的线性表成为顺序表，顺序表的存储元素是连续的，如数组：依次往下保存；

链式存储的线性表成为链表，链表中的存储元素不一定是连续的，元素节点中存放数据元素以及相邻元素的地址信息；

1. 线性结构常见的有数组、队列、链表和栈；

## 非线性结构

非线性结构包括：二维数组、多维数组、广义表、树结构、图结构

# 稀疏数组和队列

## 稀疏数组

当一个数组中大部分元素都为0，或者为同一值的数组时，可以使用稀疏数组来保存该数据；

* 稀疏数组的处理方法：

1. 记录数组一共有几行几列，有多少个不同的值
2. 把具有不同值的元素的队列及值记录在一个规模数组中，从而缩小程序的规模；



第一列数据表示：6行7列8个值

PS：数组下标是从0开始的；

EG：需求保存五子棋中下棋的位置

* 二维数组转为稀疏数组的思路：

1. 遍历原始的二维数组，得到有效数据的个数sum；
2. 根据sum创建稀疏数组sparseArray int[sum+1][3];
3. 将二维数组的有效数据存入稀疏数组；

* 稀疏数组转为二维数组：

1. 先读取稀疏数组第一行，根据第一行创建原始的二位数组：

**int[][]** origin = **new int[**sparse**[**0**][**0**]][**sparse**[**0**][**1**]]**;

1. 读取稀疏数组最后几行的数据，并赋予原始的二位数组；

## 队列

* 队列的介绍

1. 队列是一个有序列表，可以用数组或者链表来实现；
2. 遵循先入先出的原则；

* 数组模拟队列：



maxSize为该队列的最大容量；front和rear作为两个指针，记录队列的前后端指标，初始位置是-1；

存入数据后，rear值为存放值的最大下标，front为已取出值下标+1；

存数据rear加，front不变；取数据rear不变，front加；

思路：

* addQueue 添加队列

1. 当rear<maxSize-1时数据存入rear为下标的元素中，然后rear上移++；
2. 当rear=maxSize-1，数组存满；

* fetchQueue 获取队列

1. 先将front指针上移+1，取出对应下标的值；
2. 如果front=rear，则队列取完；

* 数组模拟环形队列

思路：

1. front：初始值0，指向队列的第一个元素的当前位置;
2. rear： 初始值0，指向队列后一个元素的后面一个位置，为了空出来一个空间做约定，发现要不没办法判断队列是否满，因为取模实现循环，判断只能使用==。所以导致实际队列存放的数据是数组maxSize-1;
3. 队列为空的条件：front==rear；
4. 队列为满的条件：(rear+1)%maxSize==front；
5. 队列中的有效数据的个数：(rear+ maxSize - front)%maxSize

# 链表(Linked List)

链表是有序的列表，内存中存储模型如下:

小结：

1. 链表是以节点存储的；
2. 每个节点包含data域和next域：指向下一个节点；
3. 链表的各个节点不是连续存储的，即next域记录内存地址不是连续的；
4. 头指针记录第一个数据的地址；
5. 第一个数据的地址在data域获取数据，在next域获取下一个数据的地址；

3、依次往后，直到next域内容为null；

## 单链表

单链表：带有头结点，逻辑结构如下：



最后一个的next域为null，这只是逻辑结构，实际在内存中a1，a2...是不连续的；