参考文档：https://blog.csdn.net/ityqing/article/details/82838524

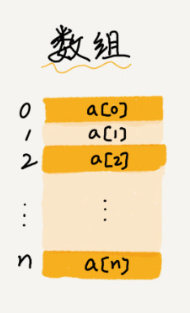
线性表的数据属于1对1关系

线性表中的数据是线性排列的，每个数据存在一个前驱和后驱数据（首尾除外），数据连起来就像一条线，称为线性表，如下是线性表的不同存储方式

**数组**

**数据结构**

数组是最简单的线性表了，其数据结构如下



数组数据的位置与物理存储位置相同

**时间复杂度**

试想，当我们在第1个位置插入数据时，我们需要将原先的1到n个数据往后移一位，当我们删除第1个数据是，将2到n个数据往前移一位，数据量n和移动时间t成正比，所以

插入删除时间复杂度为O(n)

同理

通过比较查找数据时间复杂度为O(n)

通过索引查找数据时间复杂度为O(1)

**各个语言的实现**

大多数语言不支持数组的插入和删除操作，因为这需要重新申请内存

C#：datatype[] arrayName;

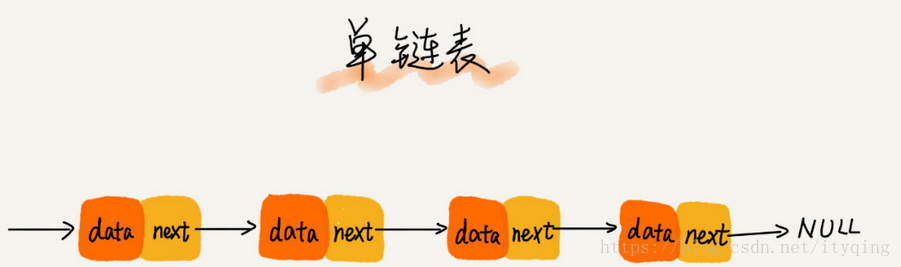
C++：string list[] = {"ABC", "DEF"};

JS：

注：不要问我为什么没有java，因为我不会

**链表**

**数据结构**



链表的每一个数据都有一个指针next，指向下一个数据

**时间复杂度**

如果要删除第i个数据，我们需要从第一个位置开始，一直next i次才能找到该位置，同理其他，所以

删除，插入，查找的时间复杂度为O(n)

**各个语言的实现**

C#：List

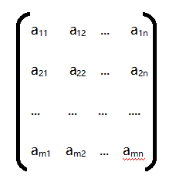
C++：Vector

Js：[]

**二维数组（矩阵）**

**数据结构**

二维数组是一个m行n列的数据结构通过 list[i][j] 来索引数据，矩阵一般使用二维数组存储



**时间复杂度**

二维数组元素的修改和查找一般都是根据索引进行

根据索引修改、查找元素为O(1)

**各个语言实现**

C#：int[,] array = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

C++：int a1[2][3] = {1,2,3,4,5,6};

Js：var a = [[1,2,3],[2,3,4]]

**稀疏矩阵**

矩阵中多数元素为0，称为稀疏矩阵

**数据结构**

稀疏矩阵就是矩阵，由于矩阵中多数元素都是0，所以稀疏矩阵一般使用3元组表示矩阵的元素，3元组的数组代表矩阵，如3元组 (1, 3, 10) 代表第1行，第3列的元素值为10，如 [(1, 3, 10), (1, 10, 2), (2, 4, 8), (2, 9, 7), (3, 5, 6)] 代表一个稀疏矩阵

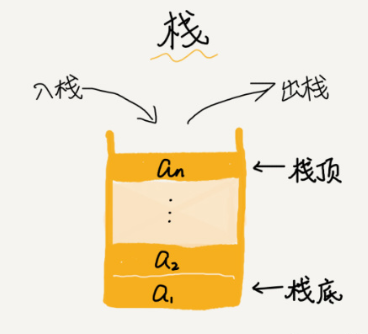
**各个语言的实现**

想啥呢，没有对应实现，要实现只能自己实现

**栈**

栈只能在数据的尾部进行插入和删除操作

**数据结构**



如上，我们只能在栈顶插入数据（入栈）或者移除数据（出栈）

**各个语言的实现**

C#：Stack

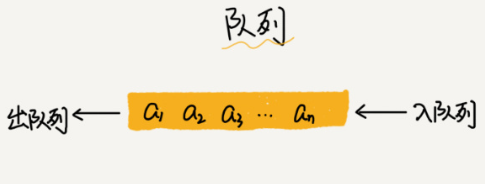
C++：stack

Js：[]，不要怀疑，js的数组实现了栈的操作

**队列**

队列是种先进先出的线性表

**数据结构**



我们只能在尾端插入数据，在头部移除数据，这样的数据结构称为队列

**各个语言的实现**

C#：Queue

C++：queue

Js：你可以使用数组实现队列，因为数组也提供了对应的队列方法