数据结构是一种特殊的组织和存储数据的方式，可以使我们可以更高效地对存储的数据执行操作。数据结构在计算机科学和软件工程领域具有广泛而多样的用途。

几乎所有已开发的程序或软件系统都使用数据结构。此外，数据结构属于计算机科学和软件工程的基础。当涉及软件工程面试问题时，这是一个关键主题。因此，如果你需要用到编程技能，就应该对数据结构有充分的了解。

1.数组

数组是固定大小的结构，可以容纳相同数据类型的项目。它可以是整数数组，浮点数数组，字符串数组或什至是数组数组（例如二维数组）。数组已建立索引，这意味着可以进行随机访问。

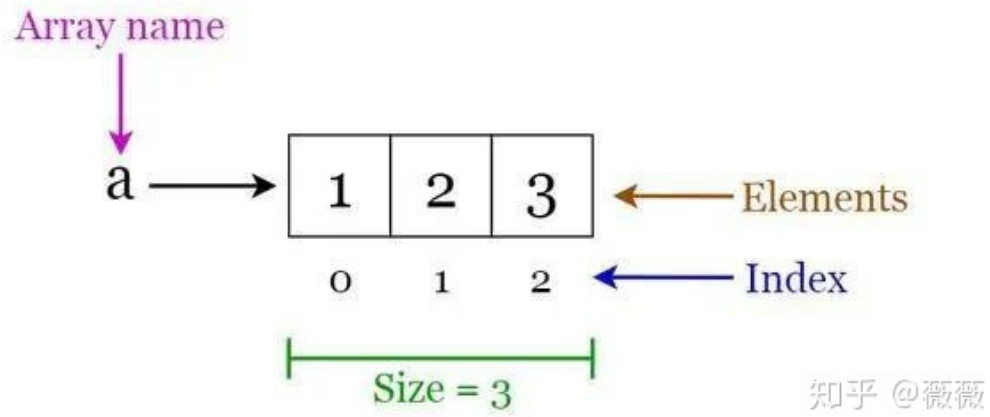

Fig 1. Visualization of basic Terminology of Arrays

数组运算

- 遍历：遍历所有元素并进行打印。
- 插入：将一个或多个元素插入数组。
- 删除：从数组中删除元素。
- 搜索：在数组中搜索元素。您可以按元素的值或索引搜索元素。
- 更新：在给定索引处更新现有元素的值。

数组的应用

- 用作构建其他数据结构的基础，例如数组列表，堆，哈希表，向量和矩阵。
- 用于不同的排序算法，例如插入排序，快速排序，冒泡排序和合并排序。

2.链表

链表是一种顺序结构，由相互链接的线性顺序项目序列组成。因此，您必须顺序访问数据，并且无法进行随机访问。链接列表提供了动态集的简单灵活的表示形式。

让我们考虑以下有关链表的术语。您可以通过参考图2来获得一个清晰的主意：

- 链表中的元素称为节点。
- 每个节点都包含一个密钥和一个指向其后继节点（称为next）的指针。
- 名为head的属性指向链接列表的第一个元素。
- 链表的最后一个元素称为尾。

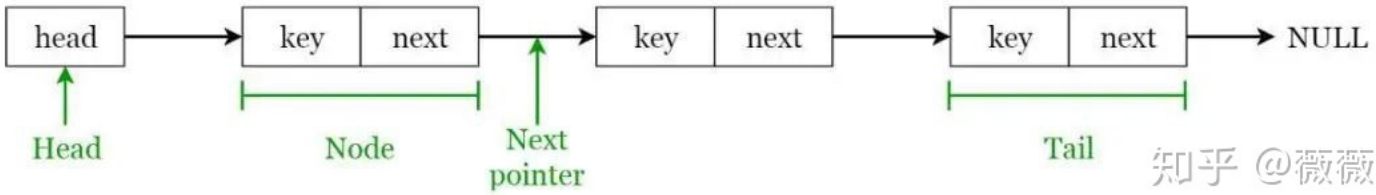

Fig 2. Visualization of basic Terminology of Linked Lists

以下是可用的各种类型的链表：

- 单链列表—只能沿正向遍历项目。
- 双链表-可以在前进和后退方向上遍历项目。节点由一个称为上一个的附加指针组成，指向上一个节点。
- 循环链接列表—链接列表，其中头的上一个指针指向尾部，尾号的下一个指针指向头。

链表操作

- 搜索：通过简单的线性搜索在给定的链表中找到键为k的第一个元素，并返回指向该元素的指针。
- 插入：在链接列表中插入一个密钥。插入可以通过3种不同的方式完成；在列表的开头插入，在列表的末尾插入，然后在列表的中间插入。
- 删除：从给定的链表中删除元素x。您不能单步删除节点。删除可以通过3种不同方式完成；从列表的开头删除，从列表的末尾删除，然后从列表的中间删除。

链表的应用

- 用于编译器设计中的符号表管理。
- 用于在使用Alt Tab（使用循环链表实现）的程序之间进行切换。

3.堆栈

堆栈是一种LIFO（后进先出-最后放置的元素可以首先访问）结构，该结构通常在许多编程语言中都可以找到。该结构被称为"堆栈"，因为它类似于真实世界的堆栈——板的堆栈。

堆栈操作

下面给出了可以在堆栈上执行的2个基本操作。请参考图3，以更好地了解堆栈操作。

- Push 推送：在堆栈顶部插入一个元素。
- Pop 弹出：删除最上面的元素并返回。

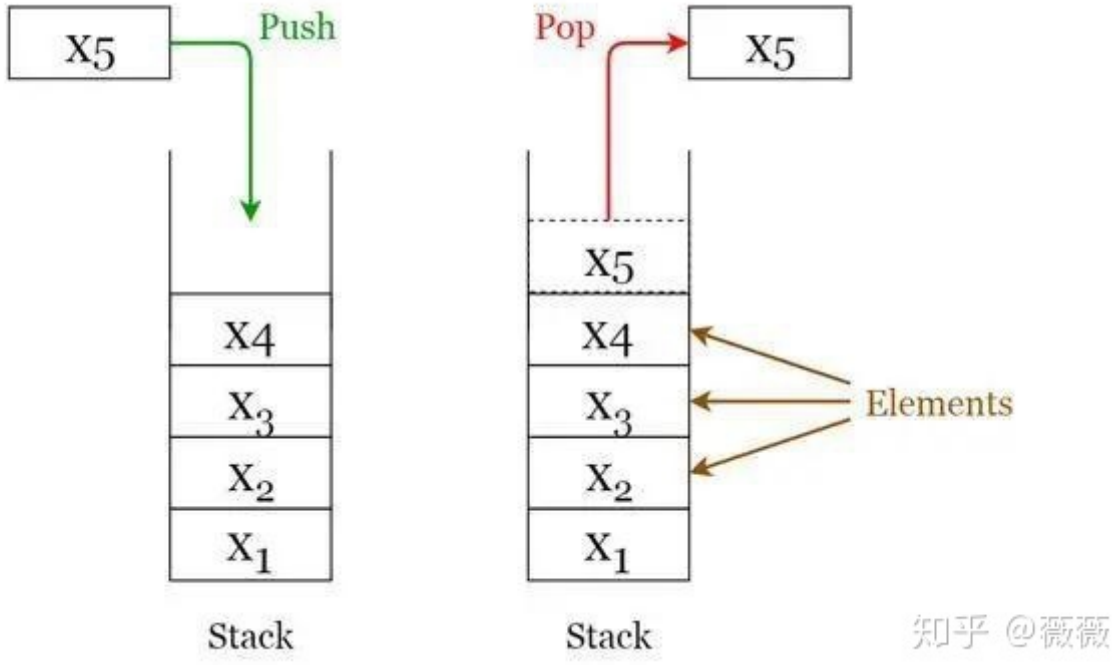

Fig 3. Visualization of basic Operations of Stacks

此外，为堆栈提供了以下附加功能，以检查其状态。

- Peep 窥视：返回堆栈的顶部元素而不删除它。
- isEmpty：检查堆栈是否为空。
- isFull：检查堆栈是否已满。

堆栈的应用

- 用于表达式评估（例如：用于解析和评估数学表达式的调车场算法）。
- 用于在递归编程中实现函数调用。

4.队列

队列是一种FIFO（先进先出-首先放置的元素可以首先访问）结构，该结构通常在许多编程语言中都可以找到。该结构被称为"队列"，因为它类似于现实世界中的队列——人们在队列中等待。

队列操作

下面给出了可以在队列上执行的2个基本操作。请参考图4，以更好地了解堆栈操作。

- 进队：将元素插入队列的末尾。
- 出队：从队列的开头删除元素。

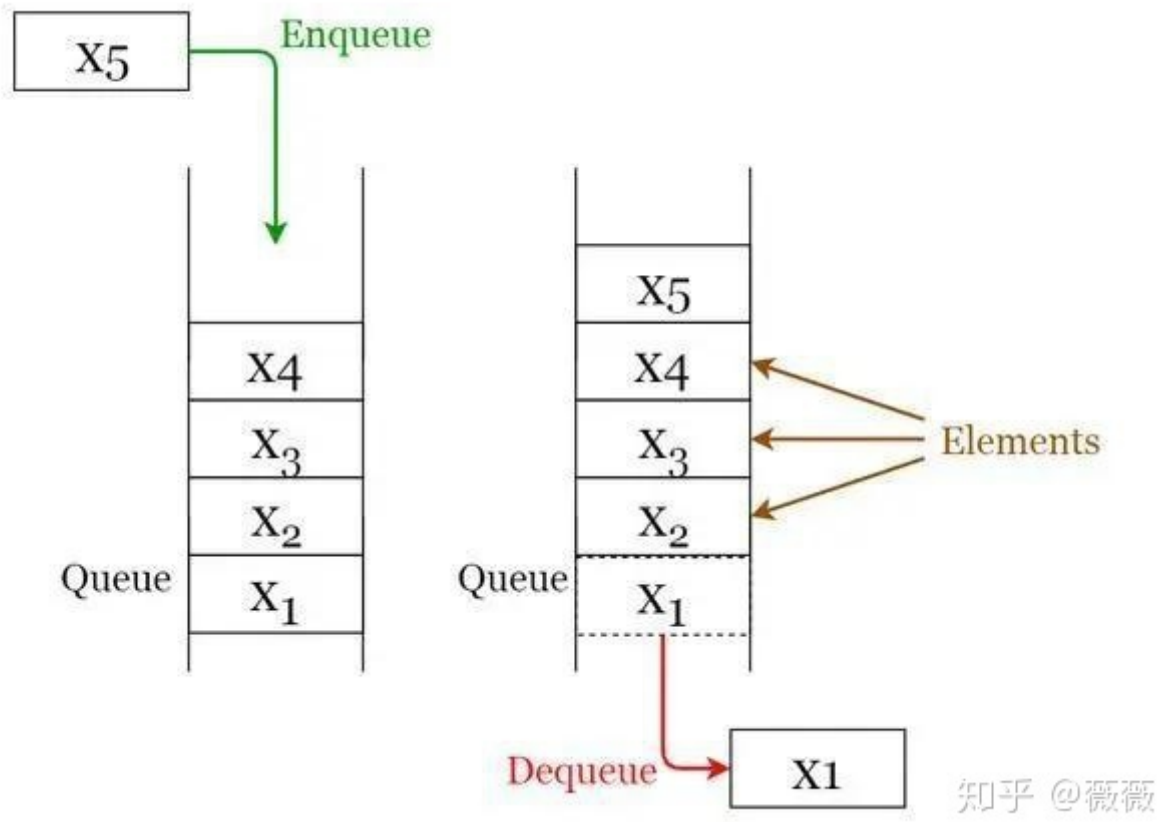

Fig 4. Visualization of Basic Operations of Queues

队列的应用

- 用于管理多线程中的线程。
- 用于实施排队系统（例如：优先级队列）。

5.哈希表

哈希表是一种数据结构，用于存储具有与每个键相关联的键的值。此外，如果我们知道与值关联的键，则它有效地支持查找。因此，无论数据大小如何，插入和搜索都非常有效。

当存储在表中时，直接寻址使用值和键之间的一对一映射。但是，当存在大量键值对时，此方法存在问题。该表将具有很多记录，并且非常庞大，考虑到典型计算机上的可用内存，该表可能不切实际甚至无法存储。为避免此问题，我们使用哈希表。

哈希函数

名为哈希函数（h）的特殊函数用于克服直接寻址中的上述问题。

在直接访问中，带有密钥k的值存储在插槽k中。使用哈希函数，我们可以计算出每个值都指向的表（插槽）的索引。使用给定键的哈希函数计算的值称为哈希值，它表示该值映射到的表的索引。

- h：哈希函数
- k：应确定其哈希值的键
- m：哈希表的大小（可用插槽数）。一个不接近2的精确乘方的素数是m的一个不错的选择。

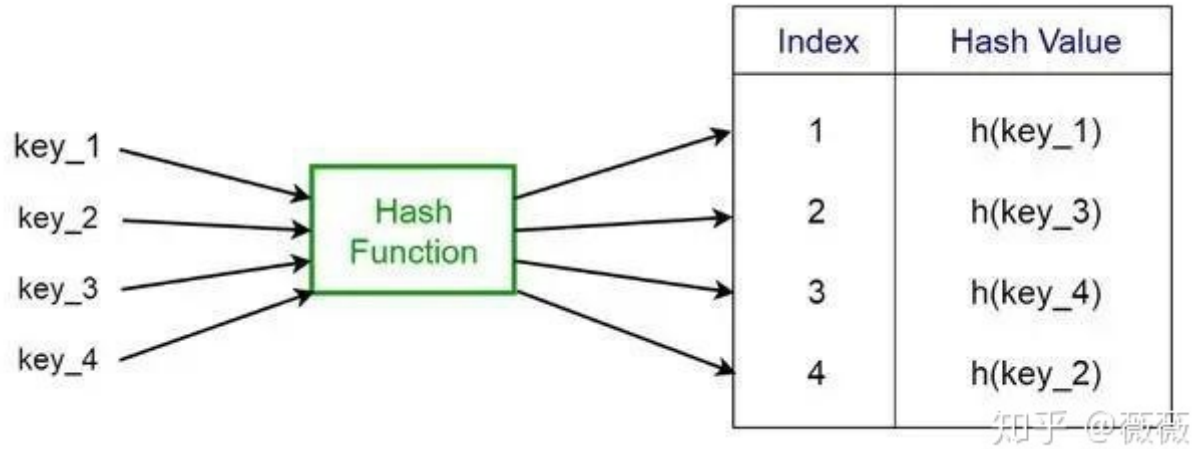

Fig 5. Representation of a Hash Function

- 1→1→1
- 5→5→5
- 23→23→3
- 63→63→3

从上面给出的最后两个示例中，我们可以看到，当哈希函数为多个键生成相同的索引时，就会发生冲突。我们可以通过选择合适的哈希函数h并使用链接和开放式寻址等技术来解决冲突。

哈希表的应用

- 用于实现数据库索引。
- 用于实现关联数组。
- 用于实现"设置"数据结构。

6.树

树是一种层次结构，其中数据按层次进行组织并链接在一起。此结构与链接列表不同，而在链接列表中，项目以线性顺序链接。

在过去的几十年中，已经开发出各种类型的树木，以适合某些应用并满足某些限制。一些示例是二叉搜索树，B树，红黑树，展开树，AVL树和n元树。

二叉搜索树

顾名思义，二进制搜索树（BST）是一种二进制树，其中数据以分层结构进行组织。此数据结构按排序顺序存储值，我们将在本课程中详细研究这些值。

二叉搜索树中的每个节点都包含以下属性：

- key：存储在节点中的值。
- left：指向左孩子的指针。
- 右：指向正确孩子的指针。
- p：指向父节点的指针。

二叉搜索树具有独特的属性，可将其与其他树区分开。此属性称为binary-search-tree属性。令x为二叉搜索树中的一个节点：

- 如果y是x左子树中的一个节点，则y.key≤x.key

- 如果y是x的右子树中的节点，则y.key≥x.key

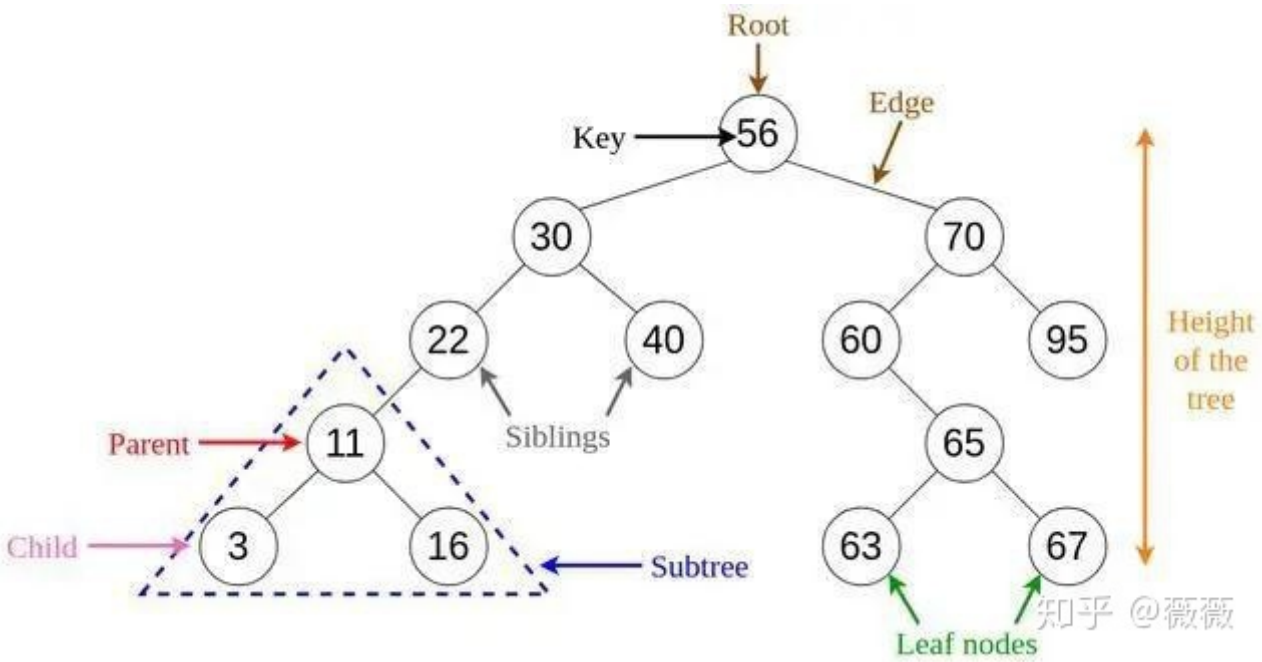

Fig 6. Visualization of Basic Terminology of Trees.

树的应用

- 二叉树：用于实现表达式解析器和表达式求解器。
- 二进制搜索树：用于许多不断输入和输出数据的搜索应用程序中。
- 堆：由JVM（Java虚拟机）用来存储Java对象。
- Trap：用于无线网络。

7.堆

堆是二叉树的一种特殊情况，其中将父节点与其子节点的值进行比较，并对其进行相应排列。

让我们看看如何表示堆。堆可以使用树和数组表示。图7和8显示了我们如何使用二叉树和数组来表示二叉堆。

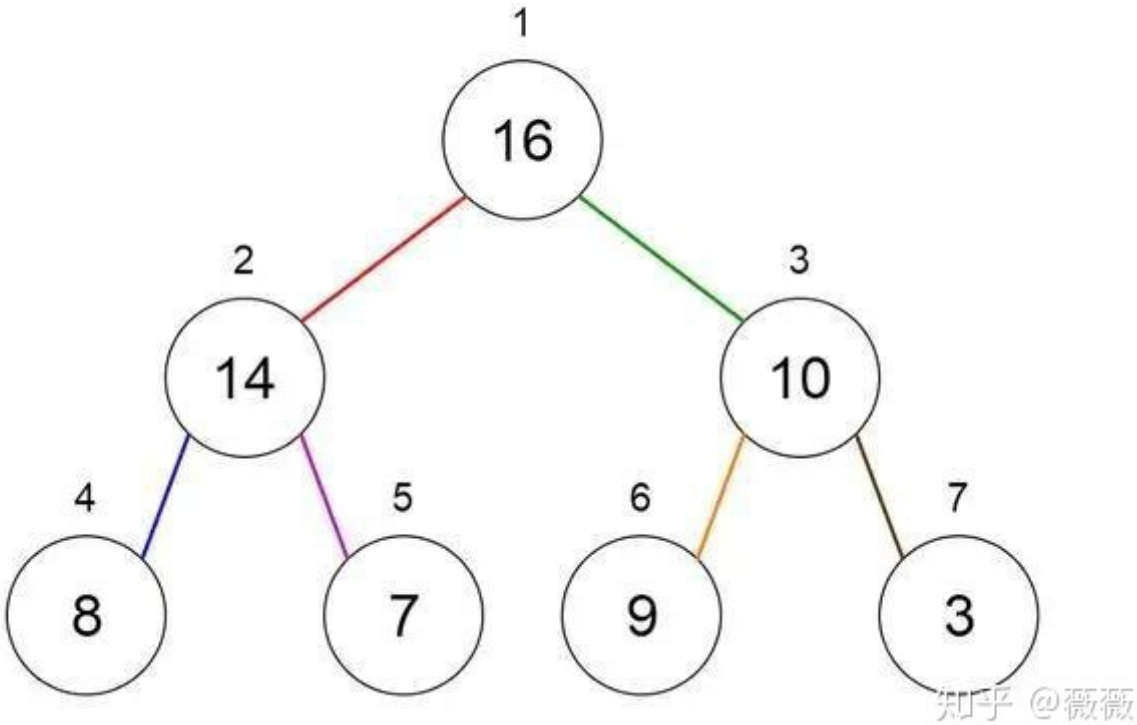

Fig 7. Binary Tree Representation of a Heap

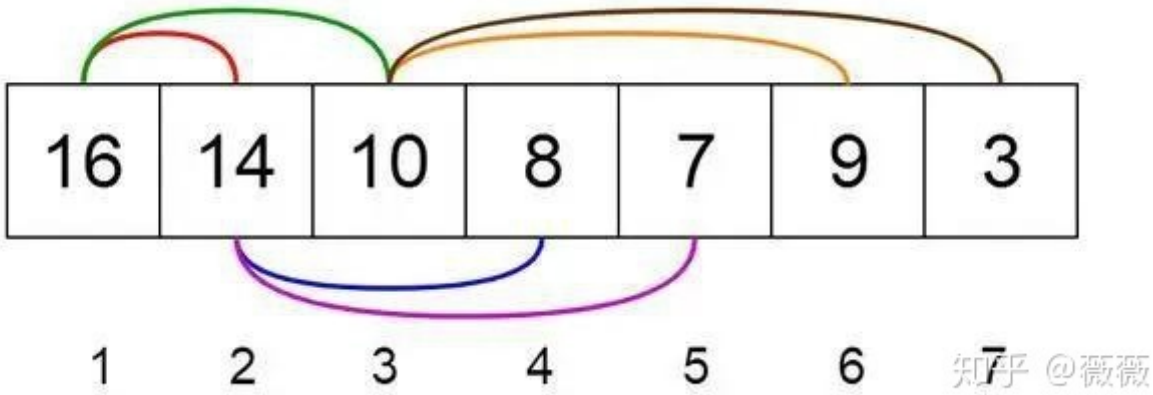

Fig 8. Array Representation of a Heap

堆可以有2种类型：

- 最小堆：父项的密钥小于或等于子项的密钥。这称为min-heap属性。根将包含堆的最小值。
- 最大堆数：父项的密钥大于或等于子项的密钥。这称为max-heap属性。根将包含堆的最大值。

堆的应用

- 用于实现优先级队列，因为可以根据堆属性对优先级值进行排序。
- 可以在O（log n）时间内使用堆来实现队列功能。
- 用于查找给定数组中k个最小（或最大）的值。
- 用于堆排序算法。

8.图

一个图由一组有限的顶点或节点以及一组连接这些顶点的边组成。

图的顺序是图中的顶点数。图的大小是图中的边数。

如果两个节点通过同一边彼此连接，则称它们为相邻节点。

有向图

如果图形G的所有边缘都具有指示什么是起始顶点和什么是终止顶点的方向，则称该图形为有向图。

我们说（u，v）从顶点u入射或离开顶点u，然后入射到或进入顶点v。

自环：从顶点到自身的边。

无向图

如果图G的所有边缘均无方向，则称其为无向图。它可以在两个顶点之间以两种方式传播。如果顶点未连接到图中的任何其他节点，则称该顶点为孤立的。

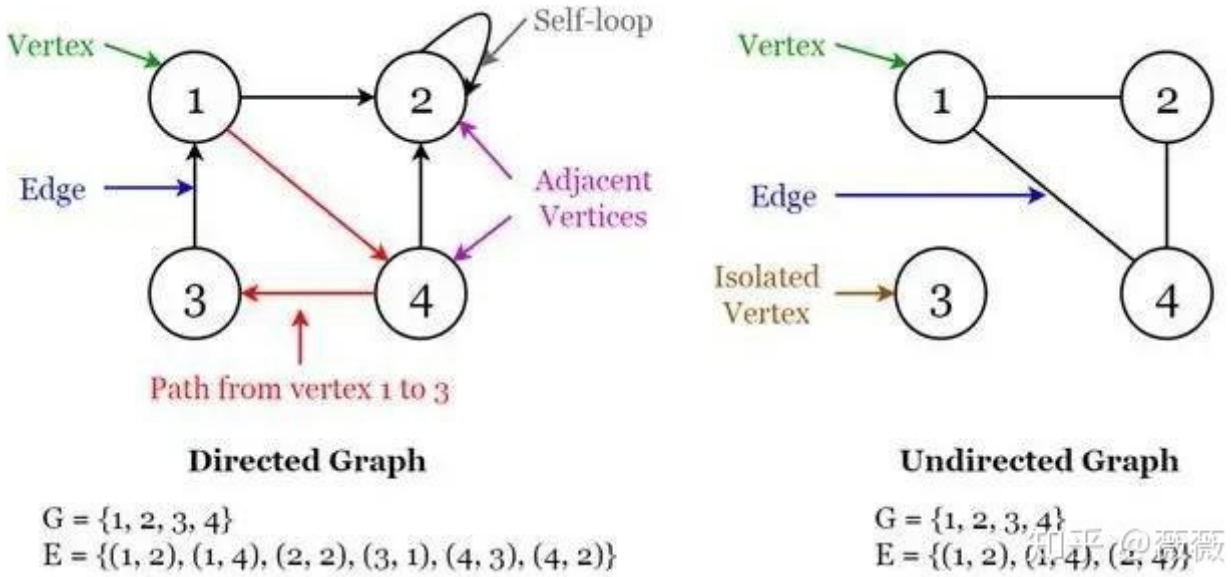

Fig 9. Visualization of Terminology of Graphs

图的应用

- 用于表示社交媒体网络。每个用户都是一个顶点，并且在用户连接时会创建一条边。
- 用于表示搜索引擎的网页和链接。互联网上的网页通过超链接相互链接。每页是一个顶点，两页之间的超链接是一条边。用于Google中的页面排名。
- 用于表示GPS中的位置和路线。位置是顶点，连接位置的路线是边。用于计算两个位置之间的最短路径。