查找

可视化 https://www.cs.usfca.edu/~galles/visualization/Algorithms.html

顺序查找

顺序查找（Sequential Search），也叫线性查找。

就是在线性表中逐个查找。查找时间复杂度是O(n)。

二分查找

二分查找（Binary Search），也叫折半查找。

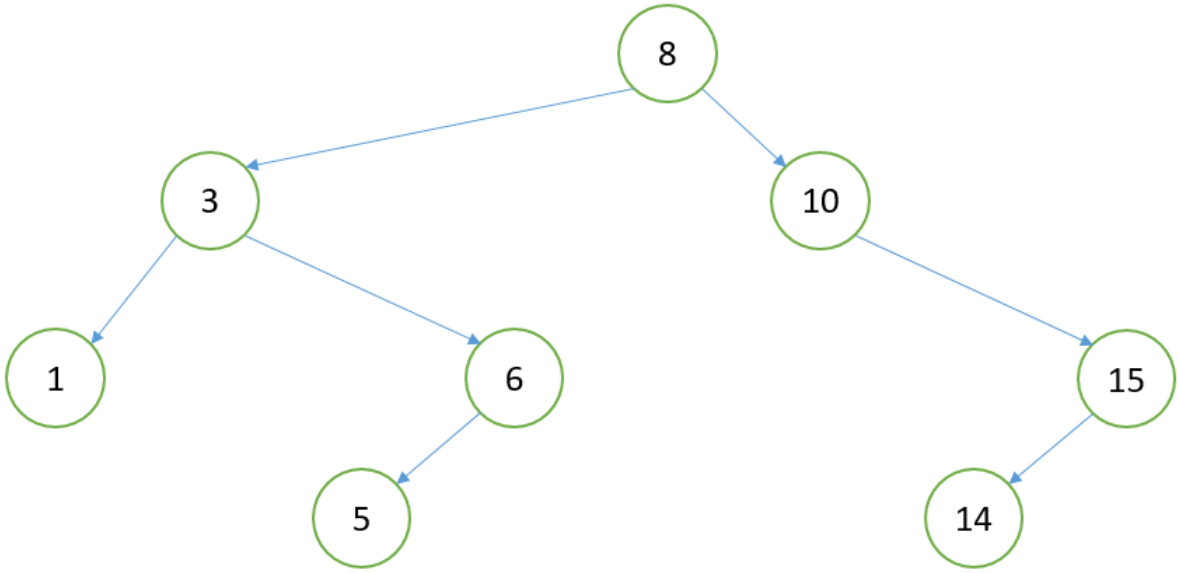
前提，必须先排序达到有序后，才能二分查找。

时间复杂度为$O(\log_2 n)$。

但是，二分查找前提是排过序，而排序非常耗时。如果频繁插入元素，不建议使用，需要重新排序，维护它代价颇高。

BST

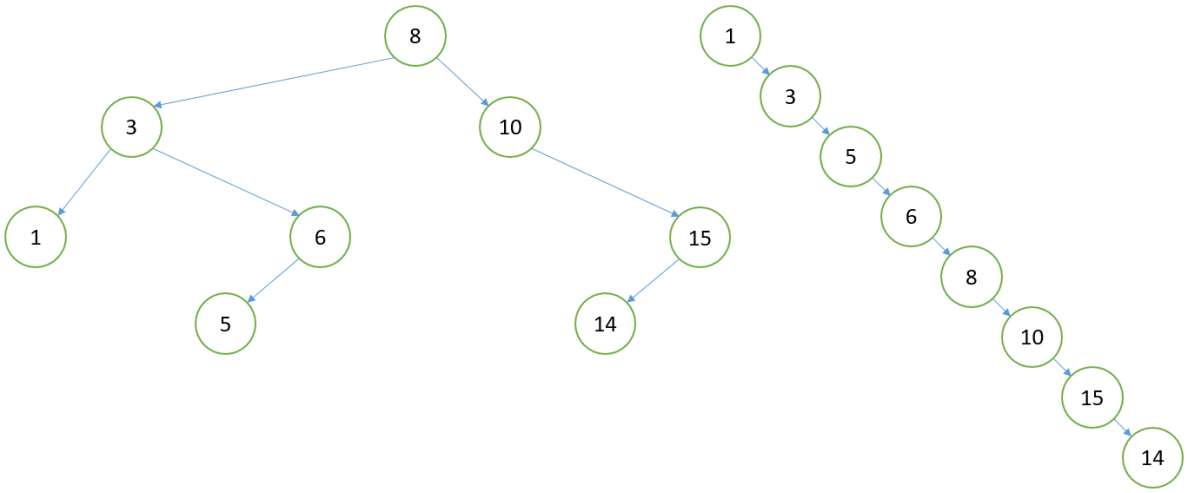
二分查找树（Binary Search Tree），也叫二叉排序树。



二叉排序树特性：

- 若有左子树，则左子树所有结点的值均小于它的根结点的值
- 若有右子树，则右子树所有结点的值均小于它的根结点的值
- 其左子树、右子树也是二叉排序树

对上面的树进行中序遍历，得到序列[1, 3, 5, 6, 8, 10, 14, 15]



如果BST左右子树较为平衡，那么搜索效率较高，O(logn)。

如果成了一棵斜树，那么搜索效率很差，O(n)。

为了使增删结点后，BST应该尽量平衡，常用的有：

- AVL，对平衡要求高，所以，查找性能高，但是增删调整多
- 红黑树，增删效率高，查找效率略低

多路查找树

BST虽然可以做到平衡，但是由于是二叉树，如果数据非常多，结点非常多，这棵二叉树将非常高。如果用它来组织大量数据存储，需要更多次数IO访问。而且每次访问磁盘读取数据较少。

B树特别适合较大数据块的存储系统，例如磁盘。通常用在数据库和文件系统。

以B树为基础，衍生出B+树。MySQL就使用了它。

索引

当数据规模大的时候，尤其是数据库，可以存储数万、数亿条数据。那么，在兼顾增删性能的同时，还要能从海量数据中快速查询数据。

例如新华字典，存储大量文字，为了快速查询某一个字，提供了拼音音节索引，还提供部首检字表。这是都是索引，通过某个关键码定位数据。利用空间换时间。

MySQL索引类型

主键索引，主键会自动建立主键索引，主键本身就是为了快速定位唯一记录的。
唯一索引，表中的索引列组成的索引必须唯一，但可以为空，非空值必须唯一
普通索引，没有唯一性的要求，就是建了一个字典的目录而已。
联合索引，多个字段组合创建索引，使用条件查询时，先匹配左边字段

全文索引，MyISAM使用，对Char、Varchar、TEXT类型使用
空间索引，SPATIAL，基本不用。

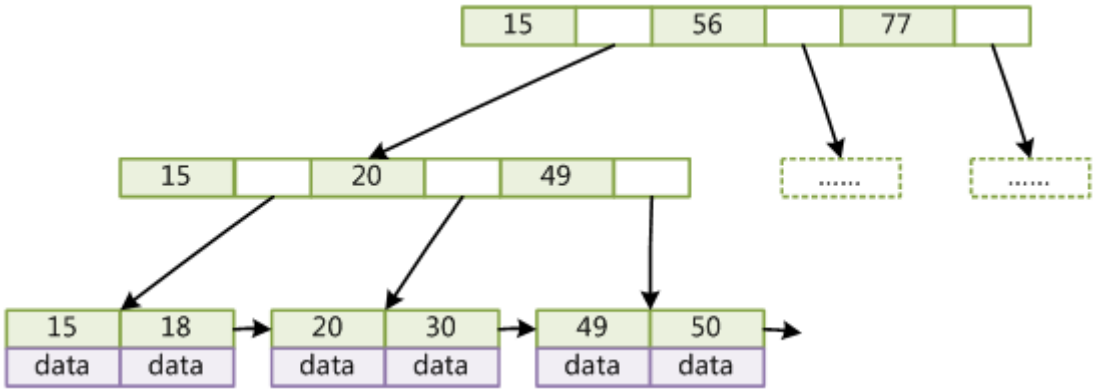
在MySQL中，InnoDB和MyISAM的索引数据结构可以使用Hash或BTree，innodb默认是BTree。

Hash时间复杂度是O(1)，但是只能进行精确匹配，也就是Hash值的匹配，比如范围匹配就没办法了，hash值无序所以无法知道原有记录的顺序。Hash问题较多。

BTree索引，以B+树为存储结构。

虽然，索引可以提高查询效率，但是却影响增删改的效率，因为需要索引更新或重构。频繁出现在where子句中的列可以考虑使用索引。要避免把性别这种字段设索引。

B+树



B+树节点组织成一棵树。节点分为内部节点和叶子节点。

内部节点不存储数据，叶子节点不存储指针。

叶子节点深度一致。叶子节点包含所有索引字段值。

每个leaf node保存数据，所有的leaf node组织成链表。假设读取16到22的数据，找到18后，顺着链表往后遍历读取即可。

InnoDB中，数据文件本身就是按主键索引存储的，叶子节点中保存的就是数据记录。

InnoDB靠主键才能组织数据存储的，所以一定要定义主键，否则MySQL帮你找一个候选键作为主键，找不到会自动增加一个主键。这个主键建议使用整数或自增整数。整数比较方便快捷。

InnoDB中主键索引和数据存在一起，称为聚簇索引、聚集索引、聚类索引、簇集索引。MyISAM索引和数据是不同文件存储，这是非聚集索引。聚簇索引效率一般高于非聚簇索引。

如果在其他字段上定义B+Tree索引，叶子节点的数据记录的是主键，这种称为辅助索引（二级索引）。