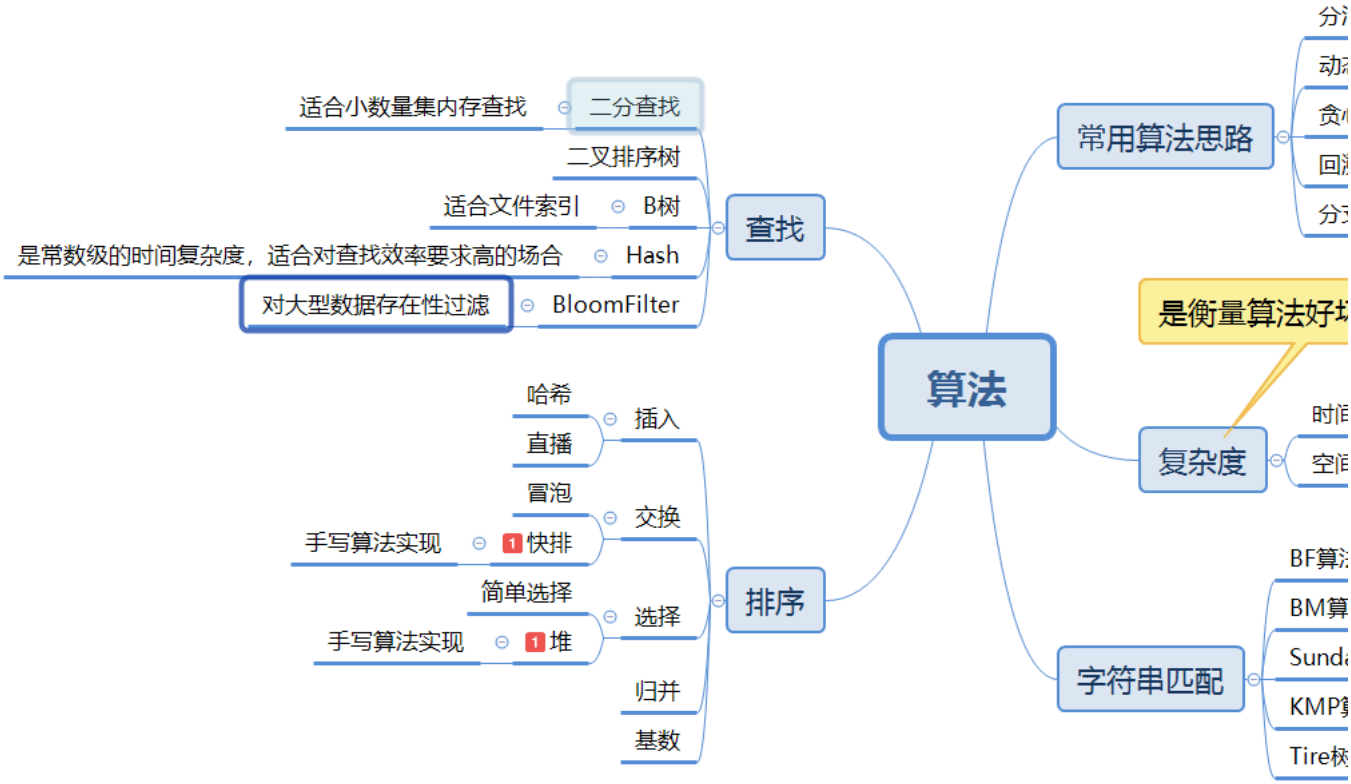
1. 数据结构知识点



2. 从搜索树到二叉树

2.1 二叉搜索树

知识点详解

```
private static final Map<Character, Char
static {
    brackets.put(')', '(');
    brackets.put(']', '[');
    brackets.put('}', '{');
}

public static boolean isMatch(String str
    if (str == null) {
        return false;
    }
    Stack<Character> stack = new Stack<>
    for (char ch : str.toCharArray()) {
        if (brackets.containsValue(ch))
            stack.push(ch);
        } else if (brackets.containsKey(
            if (stack.empty() || stack.p
                return false;
            }
        }
    }
    return stack.empty();
}
```

满足条件：

1. 每个分支只有两个数
2. 左分支小于自身，右分支大于自身

问题：随着插入删除，二叉搜索树的树高会有所变化，当只有左节点或者右节点的时候，就会退化成线性的搜索树。会大大降低查找效率
平衡二叉树可以解决这个问题
平衡二叉树保证：每个节点的左右子树的高度差的绝对值不超过1。
例如AVL树（严格的平衡二叉树），插入或者删除数据的时候可能会需要旋转来保持平衡，比较适合插入删除比较少的场景

2.2 红黑树

是一种更加实用的非严格的平衡二叉树
关注点是局部平衡，而不是整体平衡。确保没有一条路径是比其他路径长度超出两倍。所以是接近平衡的。但是减少了旋转操作，所以更加实用
java8的HashMap就使用了红黑树解决散列冲突的查找问题，TreeMap也是通过这个保证有序性的

规则：

1. 每个节点不是红色就是黑色
2. 根节点是黑色
3. 每个叶子节点都是空的黑色的节点
4. 红色节点的两个字节点都是黑色的
5. 任意节点到到其叶子节点的每条路径上都包含相同数量的黑色节点

3. 从搜索树到B+树

知识点详解

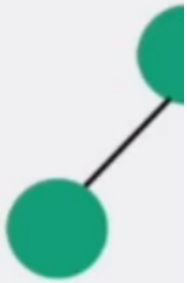
TopK问题：找出N个数中最

解法：

1、用前K个数创建大小为K的大根堆

2、剩余N-K个数跟堆顶进行比较

时间复杂度：N*logK



K

B树是一种二叉树，是一种搜索树，B树的每个节点可以存储多个树，非常适合使用在文件索引上，可以有效减少磁盘的I/O次数
B树种所有节点的**最大子节点数**称为B树的**阶**，例如上图，最大子节点数是左下角的三，所以称为三阶B数，也叫2-3树

> 5.1 B树
> B树就是二叉搜索树。具有的特点如下：
>
> （1）所有非叶子结点最多拥有两个儿子；
>
> （2）所有结点存储一个关键字；
>
> （3）非叶子结点的左指针指向小于其关键字的子树，右指针指向大于其关键字的子树。
>
> 5.2 B-树
>
> B-树的定义：
>
> （1） 任意非叶子结点的儿子的个数最多为M个；
>
> （2） 根结点的儿子的个数为[2，M]个；
>
> （3） 除根的非叶子结点的儿子数为[M/2，M]；
>
> （4） 每个结点的关键字为（M/2-1）上取整~[M-1]；
>
> （5） 非叶子结点的关键字的个数=指向儿子的指针数-1；
>
> （6） 结点内的关键字是有序的；
>
> （7） 每个指针指向的关键字不同的区域；
>
> （8） 所有叶子结点位于同一层。
>
> B-树的特点：
>
> （1） 关键字分布在整棵树上；
>
> （2） 任何一个关键字出现且出现在一个结点中；
>
> （3） 搜索有可能在非叶子结点结束；
>
> （4） 搜索性能等价于在关键字上做一次二分查找；
>
> （5） 自动层次控制。
>
> 5.3 B+树
>
> B+树的定义：
>
> （1） 其定义基本与B-树相同，除了：
>
> （2） 非叶子结点的孩子个数等于关键字的个数；
>
> （3） 非叶子结点的指针p[i]，指向[k[i]，k[i+1]]的子树；
>
> （4） 叶子指向一个链指针；
>
> （5） 所有关键字都在叶子出现。
>
> B+树的特点：
>
> （1） 关键字都出现在叶子结点的链表中，且关键字恰好有序；
>
> （2） 不可能在非叶子结点命中；
>
> （3） 非叶子结点相当于索引，叶子结点相当于存储数据的数据层；
>
> 更适合文件索引系统。

B树总结：

1. B树的关键字分布在整颗树中，一个关键字只出现在一个节点中
2. 搜索可能在非叶节点停止
3. 一般应用在文件系统

B+树总结：

1. 关键字都出现在叶子节点的链表中，所有叶节点都有指向下一个叶节点的指针
2. 不可能在非叶子节点命中，一定会查询到叶子节点
3. 叶子节点相当于存储层，保存数据。非叶节点保存指向关键字的指针，不保存数据
4. 更适合做索引系统
5. **mysql数据库的索引就是B+树的实现**