1. **设计思路与程序结构：**

本project使用C++语言以一个比较OOP的方式实现，先设计了一个纯虚的基类 KeyValueStruct，提供了插入，删除，查找等必要的接口，然后用带有两个参数的模板实现了继承该基类的红黑树(rb\_tree.h/rb\_tree.cc)和B树(b\_tree.h/b\_tree.cc)；并将这四个文件放在trees文件夹内。

然后实现了WordFrequencyList类，其内部置有一个私有的KeyValueStruct指针，通过调用其提供的接口函数实现要求的功能，并对外提供一个运行函数。故用户可以选择WordFrequencyList的实现方式是红黑树还是B树，而非实现两个WordFrequencyList分别调用。

main函数在test.cc中，其通过构造WordFrequencyList对象并调用其运行函数来运行/测试程序的功能。

此外，本project的代码风格遵循Google C++ Style Guide。

1. **使用方法**

编译全部.cc文件得到可执行文件(a.exe)后，直接运行。

首先选择单词频率表的实现方式：红黑树(输入1)、B树(输入2)；

然后会出现欢迎与提示页面，分别有9种有效输入，分别对应所要求的8种功能和退出，用户只需按照程序的提示输入指令即可；每当程序完成一次操作后，用户都可以按回车键回到主页面(即欢迎与提示界面)。

P.S.程序具有较强的鲁棒性，一般的错误或无效输入将不会影响程序的正常运行，用户只需重新输入指令即可。但插入或更新的输入以及文件操作的对象文件应该具有规范的格式，否则程序运行的结果可能会与预期不同。

1. **一些设计细节和我的思考**

红黑树和B树插入、删除等主要操作的逻辑都参考了算法导论和ppt。所以这些设计细节就不做赘述。主要说明一下一些可以变通的细节点。

首先我的树的节点都定义在树的内部，并设为私有，因为我认为树的节点不应当暴露给外部，外部只应使用键或键值对来进行操作，否则外部可能会借助节点的指针对树的内部进行强制操作从而破坏树的性质，这是一个很大的隐患。

其次我像书中一样在红黑树中使用了哨兵nil节点，而非像网上一些代码没有用哨兵，用NULL代替，前期的设计中我也没有使用哨兵nil节点，并认为这是多此一举，但在不断思考和代码的完善中我稍微发现了使用哨兵的意义所在。从定义上讲，这样红黑树到叶节点的简单路径的便是NIL节点的简单路径，这就不用像平衡二叉树中那样定义平衡性时需要考虑左右孩子节点，即对于根和叶的一些边界情况更好更易处理，例如能够更好的识别case2(待调整节点的兄弟以及兄弟的儿子都是黑色的情况)；关键在于nil保证了内部节点一定会有两个儿子，没有nil红黑树的平衡性得不到保证，故若不具体构造一个nil出来，对于nil的逻辑作用的表达会很麻烦。(个人思考)

另外，B树的最小度数是可以手动设定的，但本project没有给用户提供修改最小度数的权限，而是默认设为3。