|  |  |  |
| --- | --- | --- |
| Linux实验报告 | | |
| 班级 | 学号 | 姓名 |
| 硬件三班 | 161403323 | 李凤明 |

* **实验目的**

1. 了解Linux编程环境。
2. 掌握在Linux环境下编程的常用工具，例如：shell，vim，make，gedit，git

及各种语言的集成开发环境。

1. 了解Linux系统的内存、进程、线程、同步、通信的基本原理和其程序在实际中的应用。
2. 掌握Linux环境下的应用程序设计、开发和项目管理。
3. 通过8学时的实验，在Linux环境下设计并实现一个代码量不少于2000行的项目。

* **实验背景**

在Linux环境下运用gedit工具，通过C语言编程实现四个程序，分别为：堆排序、用栈实现表达式求值、B+树和红黑树。

* **实验原理**

**1、堆排序**

利用大顶堆（或小顶堆）堆顶记录的是最大关键字（或最小关键字）这一特性，使得每次从无序中选择最大记录（或最小记录）变得简单。

以大顶堆为例，首先要根据该数组元素构建一个完全二叉树，之后构造初始堆，从最后一个非终端节点调整到根结点，每次调整都是从父节点、左孩子节点和右孩子节点三者中选择最大者跟父节点进行交换，交换之后可能造成呗交换孩子节点不满足堆的性质，因此每次交换之后要重新对被交换的孩子节点进行调整。构造了初始堆之后再进行排序，程序完成。

**2、用栈实现表达式求值**

使用的算法叫做“算符优先法”。

根据算术四则运算规则得出相继出现的两个运算符a和b的优先关系，如下表所示

a>b 表示a的优先权高于b

a<b 表示a的优先权低于b

a=b 表示a的优先权等于b

空白则表示该关系不存在

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| b  a | + | - | \* | / | ( | ) | # |
| + | > | > | < | < | < | > | > |
| - | > | > | < | < | < | > | > |
| \* | > | > | > | > | < | > | > |
| / | > | > | > | > | < | > | > |
| ( | < | < | < | < | < | = | > |
| ) | > | > | > | > |  | > |  |
| # | < | < | < | < | < |  | = |

使用两个工作栈，分别是OPTR用来寄存运算符，OPND 用来寄存操作数或运算结果，算法基本思想如下：

1. 首先置操作数栈为空栈，表达式起始符‘#’为运算符栈的栈底元素
2. 依次读入表达式中的数字和运算符，若是操作数则入OPND 栈，若是运算符则和OPTR栈顶元素比较优先级后执行第三步
3. 若栈顶元素优先级低于运算符，则将运算符放入OPTR栈；若栈顶元素优先级等于运算符，则将栈顶元素弹出并接收下一字符，即脱括号；若栈顶元素优先级高于运算符，则将栈顶元素及操作数的两个元素弹出进行运算，再将运算结果入OPND 栈。循环往复，直到运算完成。

**3、B+树**

* B+树的定义：

B+树一种平衡的多路查找树，是应文件系统所需而出的一种[B-树](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=55765049&ss_c=ssc.citiao.link)的变型树。通常在B+树上有两个头指针，一个指向根结点，一个指向关键字最小的叶子结点。一棵m阶的B+树和m阶的B-树的差异在于：

（1）有n棵子树的结点中含有n个关键字，每个关键字不保存数据，只用来索引，所有数据都保存在叶子节点。

（2）所有的叶子结点中包含了全部关键字的信息，及指向含这些关键字记录的指针，且叶子结点本身依关键字的大小自小而大顺序链接。

（3）所有的非终端结点可以看成是索引部分，结点中仅含其子树（[根结点](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=66940455&ss_c=ssc.citiao.link" \t "_blank)）中的最大（或最小）关键字。

* B+树的查找

对B+树可以进行两种查找运算：

（1）从最小关键字起[顺序查找](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=71748817&ss_c=ssc.citiao.link" \t "_blank)

（2）从根结点开始，进行随机查找

在查找时，若非终端结点上的关键值等于给定值，并不终止，而是继续向下直到叶子结点。因此，在B+树中，不管查找成功与否，每次查找都是走了一条从根到叶子结点的路径。

* B+树的插入

m阶B树的插入操作在叶子结点上进行，若要插入关键值a，找到叶子结点后插入a，算法如下：

①如果当前结点是根结点并且插入后结点关键字数目小于等于m，则算法结束；

②如果当前结点是非根结点并且插入后结点关键字数目小于等于m，则判断若a是新索引值时转步骤④后结束，若a不是新索引值则直接结束；

③如果插入后关键字数目大于m(阶数)，则结点先分裂成两个结点X和Y，并且他们各自所含的关键字个数分别为：u=大于(m+1)/2的最小整数，v=小于(m+1)/2的最大整数；

由于索引值位于结点的最左端或者最右端，不妨假设索引值位于结点最右端，有如下操作：

如果当前分裂成的X和Y结点原来所属的结点是根结点，则从X和Y中取出索引的关键字，将这两个关键字组成新的根结点，并且这个根结点指向X和Y，算法结束；

如果当前分裂成的X和Y结点原来所属的结点是非根结点，依据假设条件判断，如果a成为Y的新索引值，则转步骤④得到Y的双亲结点P，如果a不是Y结点的新索引值，则求出X和Y结点的双亲结点P；然后提取X结点中的新索引值a’，在P中插入关键字a’，从P开始，继续进行插入算法；

④提取结点原来的索引值b，自顶向下，先判断根是否含有b，是则需要先将b替换为a，然后从根结点开始，记录结点地址P，判断P的孩子是否含有索引值b而不含有索引值a，是则先将孩子结点中的b替换为a，然后将P的孩子的地址赋值给P，继续搜索，直到发现P的孩子中已经含有a值时，停止搜索，返回地址P。

* B+树的删除

B+树的删除也仅在叶子结点进行，当叶子结点中的最大关键字被删除时，其在非终端结点中的值可以作为一个“分界关键字”存在。

**4、红黑树**

红黑树是每个节点都带有颜色属性的二叉查找树，颜色或红色或黑色。在二叉查找树强制一般要求以外，对于任何有效的红黑树还有如下的额外要求:

性质1. 节点是红色或黑色。

性质2. 根节点是黑色。

性质3 每个叶节点是黑色的。

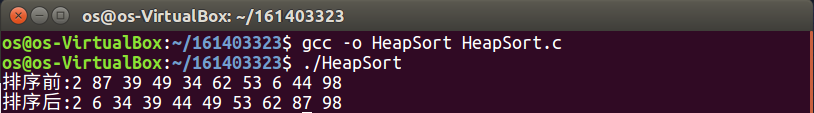
性质4 每个红色节点的两个子节点都是黑色。(从每个叶子到根的所有路径上不能有两个连续的红色节点)

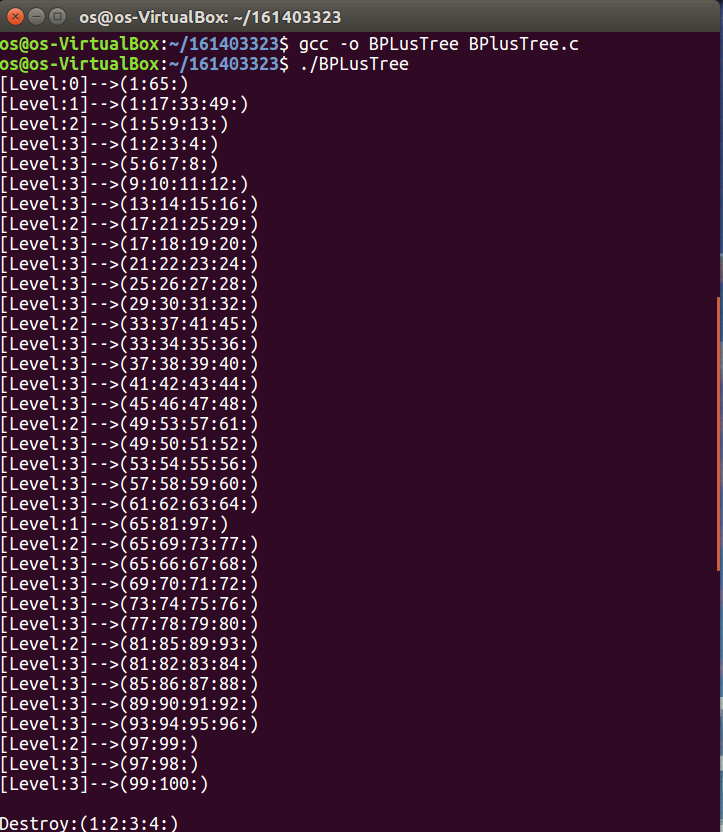
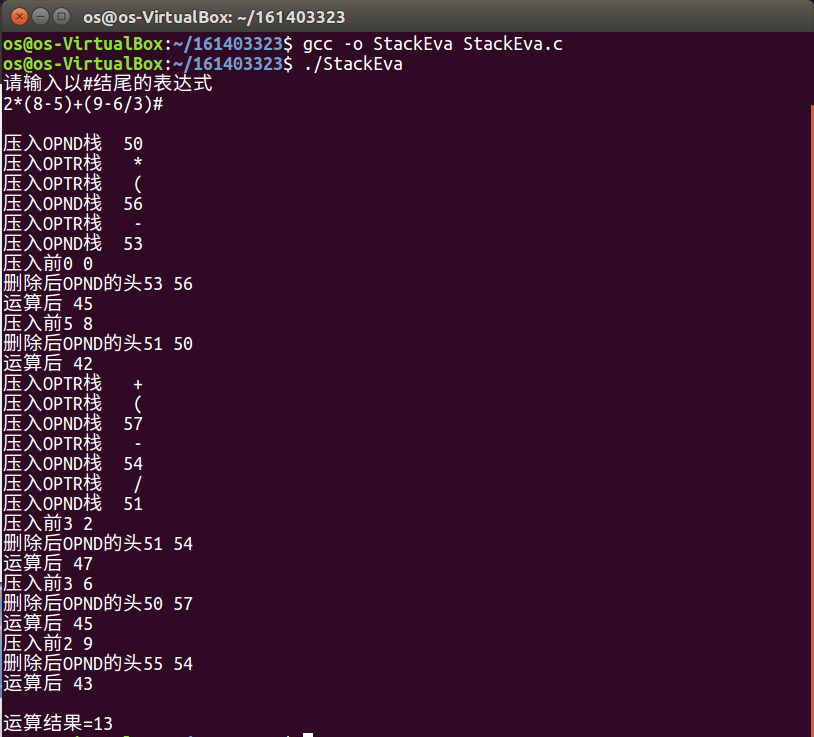
性质5. 从任一节点到其每个叶子的所有路径都包含相同数目的黑色节点。

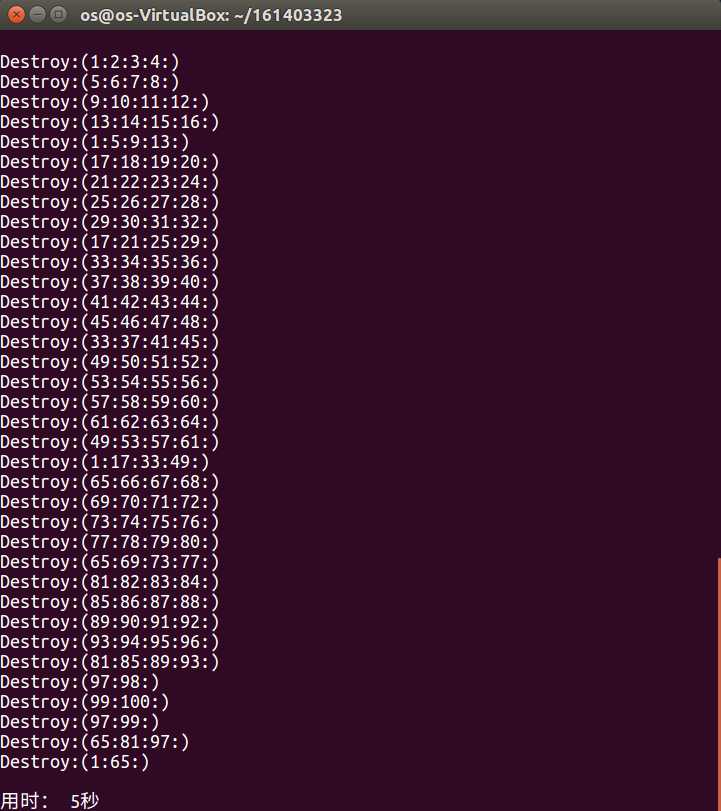
这些约束强制了红黑树的关键性质: 从根到叶子的最长的可能路径不多于最短的可能路径的两倍长。结果是这个树大致上是平衡的。因为操作如插入、删除和查找某个值的最坏情况时间都要求与树的高度成比例，这个在高度上的理论上限允许红黑树在最坏情况下都是高效的，而不同于普通的二叉查找树。

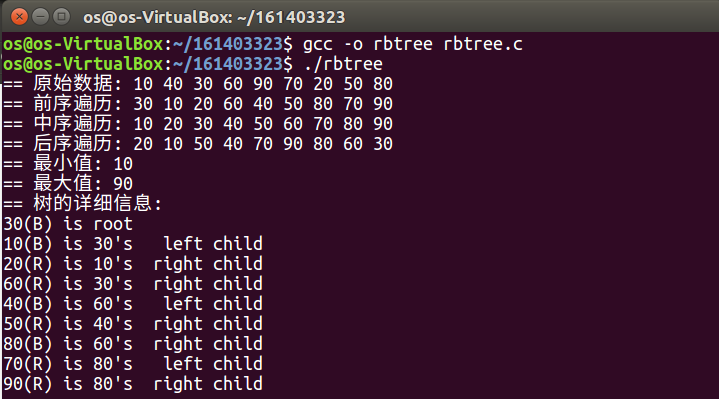
* **实验结果**

通过在命令行执行命令”gcc –o HeapSort HeapSort.c”编译，命令”./HeapSort”执行，其他文件同理，得出堆、栈、B+树和红黑树的结果如下：









* **实验小结**

通过这门Linux实验，我对Linux下的gedit编程及在终端执行C文件的命令更加理解，同时拥有了自己的github账户并学会了如何在Linux下上传项目到github。本次实验并没有选择做项目而是做了数据结构的四个程序，也让我对数据结构有了更深的理解与掌握，更加深层次地理解了这四种程序的算法，收获满满。