**最小化堆**

1. **实验目的和要求**

设有一个100\*100的稀疏矩阵，其中约有1%的非0元素。每个非0元素以一个三元组表示（行号，列好，元素值）。欲将此矩阵中的非0元素存放在一个如代码清单9-2所实现的闭散列表中。试设计散列表的长度、散列表中元素的类型以及元素到关键字值的转换函数。（不能使用书上的散列表类）。

**二、实验内容**

1. 实验准备
   1. 理论知识介绍
      1. 树的定义:
         1. 树形结构的特点是一个数据就够可以有很多个直接后继，但只有一个直接前驱，而线性结构中的每个数据元素至多只有一个直接前驱和一个直接后继。
      2. 二叉树
         1. 二叉树是结点的有限集合，它或者为空，或者由一个根结点及两颗互不相交的左、右子树构成，而其左、右子树又都是二叉树。
         2. 如果一颗二叉树中任意一层的结点个数都达到了最大值，那么这颗二叉树称为满二叉树或丰满树。
         3. 完全二叉树是在满二叉树的最底层自右至左依次（不能跳过任何一个结点）去掉若干个结点。
         4. 二叉树的链接实现
            1. 二叉树的主要储存方式是链接储存，用链接储存结构储存二叉树是很自然的，二叉树的链接储存就是用指针指出父子关系。大多数树的应用只需要找结点的儿子，因此每个结点只需要指出它的子结点储存在哪里。但也有一些应用需要找结点的父亲，因此还必须指出它的父结点储存在哪里。这样就星辰了两种链接储存结构：标准储存结构和广义的标准储存结构。
            2. 标准储存结构也称为二叉链表。在二叉链表中，每个储存结点由三个字段组成，储存结构数据元素值的数据字段以及指向左、右儿子的指针字段。data储存二叉树节点中的数据元素，left和right存放左、右儿子的地址。left和right又是也被称为左、右指针。
      3. 优先级队列
         1. 进入队列中的结点具有优先级，结点优先级越高出队越早，优先级越低出队越晚，优先级相同者，则按先进先出的原则处理。节点之间的关系是由结点的优先级决定的，而不是由入队的先后次序决定的。这样的队列称为优先级队列。在优先级队列中，优先级最高的是队头元素，优先级最低的是队尾元素。
         2. 二叉堆
            1. 二叉堆是满足两个基本的特性：结构性和有序性的二叉树。
            2. 二叉堆最好是一颗满二叉树，至少也应该是一颗完全二叉树。这就是二叉堆的结构性。
            3. 允许快速完成入队和出队的操作的特性是由堆的有序性保证的。如果想要快速找到最小的元素，可以让最小的元素位于根的位置。如果认为任何子树也应该是一个堆（递归地），那么人和节点都应该小于它的虽有子孙。这就是堆的有序性。当根结点是最小元素时，称为最小化堆。
2. 实验项目
   1. 分析
      1. 根据最小化堆的特性，最小化堆的最小的元素一定是根结点。第二小的元素一定在根结点的其中的一个子结点当中。但是第三小的元素不能确定准确的位置。
      2. 虽然第三小的元素不能确定准确的位置，但是我们可以确定第三小的元素所在的区域。假设最小化堆为a,b,c,d,e,f,g…,其中b是第二小的元素，那么我们可以判定，第三小的元素只可能出现在c,d,e这三个位置中的一个。
      3. 如果第三小的元素出现在位置c，那么我们就不需要调整二叉堆的位置。但如果二叉堆的第三小的元素出现在位置d,e,那么我们就要将d,e调整到原来的c的位置，但此时我们就无法确定新的c的位置了，因为c有可能要比d,e的子结点中的元素要大。
      4. 这时候我们就可以模仿最小化堆的出队时所用的向下过滤的方法，先在d,e的位置放置一个空结点，利用向下过滤的方法找到新的c的位置。
   2. 方案
      1. 先找到第二小的元素的位置，以及第二层的另外一个元素的位置。
      2. 然后判断另外一个元素是不是第三小的元素，不是第三小的元素的话就要在第二小的元素的子结点中找第三小的元素。
      3. 若第二层的另外一个元素不是第三小的元素，找到第三小的元素之后交换位置，然后利用已经构造好的向下过滤的方法调整整个二叉堆的有序性。
   3. 测试数据

int main() {

int a[16] = { 0,

1,

2, 9,

3, 4, 10, 11,

5, 6, 7, 8, 12, 13, 14, 15

};

priorityQueue<int> pq(a, 15);

pq.adjustOrder();

pq.printHeap();

return 0;

}

* 1. 测试结果

1 2 3 5 4 10 11 9 6 7 8 12 13 14 15

* 1. 出错情况

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Severity | Code | Description | Project | File | Line | Suppression State |
| Error | LNK2019 | unresolved external symbol "public: void \_\_thiscall priorityQueue<int>::adjustOrder(void)" (?adjustOrder@?$priorityQueue@H@@QAEXXZ) referenced in function \_main | Heap | C:\Users\93137\source\repos\Heap\Heap\main.obj | 1 |  |
| Error | LNK2019 | unresolved external symbol "public: void \_\_thiscall priorityQueue<int>::printHeap(void)" (?printHeap@?$priorityQueue@H@@QAEXXZ) referenced in function \_main | Heap | C:\Users\93137\source\repos\Heap\Heap\main.obj | 1 |  |
| Error | LNK1120 | 2 unresolved externals | Heap | C:\Users\93137\source\repos\Heap\Debug\Heap.exe | 1 |  |

将main函数放到main.cpp中的时候会发生这样的错误，如果将main函数放到heap.cpp中错误就解决了，目前不太清楚为什么会发生这样的错误。

**三、实验小结**

1. 重点

二叉堆的实现以及二叉堆的应用。

1. 难点

如何有效利用二叉堆的特点（比如左子结点下标是右结点下标的两倍）来有效地判断各个元素的位置。以及利用原先已有的用来排序的向下过滤的函数来保证现有函数的有序性。

1. 收获、体会

由于最小化堆是基于二叉堆的，二叉堆又是基于二叉树的，所以最小化堆的性质有很多都是基于二叉树的性质。但是最小化堆又有自己独特的有序性，我们可以利用最小化堆的特性推出最小化堆中的最小的三个元素的大概位置,交换第二层元素和第三小元素的位置和然后再利用向下过滤的方法最终得到结果。