山东大学 软件 学院

数据结构 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201900302030 | 姓名：邵嘉明 | | 班级： 数字媒体技术 |
| 实验题目：实验六 堆和搜索树 | | | |
| 实验学时：2 | | 实验日期：2020.11.24 | |
| 实验目的：  掌握堆和搜索树的基本概念，插入、删除方法。 | | | |
| 硬件环境：PC | | | |
| 软件环境：Visual Studio 2019 | | | |
| 实验步骤与内容：   1. 输入一系列不为零的正整数（最多不超过20个），遇到0代表输入结束（不包含0）。 2. 根据上面输入的数据序列，用初始化方法创建最大堆（不要用节点依次插入的办法创建最大堆），然后输出最大堆的层次序列。 3. 输出用堆排序后的排序结果。 4. 根据上面输入的数据，创建二叉搜索树（关键字不允许重复，如遇重复，则不重复插入该关键字），输出二叉搜索树的前序序列、中序序列（分行输出）。 | | | |
| 结论分析与体会：  了解并掌握了堆和搜索树的基本概念，插入、删除方法 | | | |

**本次实验主要代码：**

#include <stdio.h>

#include <stdlib.h>

#include <iostream>

#include <string>

#include <vector>

#include <algorithm>

#include <iomanip>

using namespace std;

const int n\_max=10000;

int a[n\_max], n;

int b[n\_max],previous[n\_max], in[n\_max], pren\_01, inn\_01;

struct tree\_Node

{

int data;

struct tree\_Node \*leftChi, \*rightChi;

};

tree\_Node \*nodeInsert(tree\_Node \*result, int value)

{

if (result == NULL)

{

result = (tree\_Node\*)malloc(sizeof(tree\_Node));

result->data = value;

result->leftChi = result->rightChi = NULL;

}

else if (value == result->data) return result;

else if (value < result->data) result->leftChi = nodeInsert(result->leftChi, value);

else result->rightChi = nodeInsert(result->rightChi, value);

return result;

}

void Previous(tree\_Node \*result)

{

if (result == NULL) return;

previous[pren\_01++] = result->data;

Previous(result->leftChi);

Previous(result->rightChi);

}

void plugIn(tree\_Node \*result)

{

if (result == NULL) return;

plugIn(result->leftChi);

in[inn\_01++] = result->data;

plugIn(result->rightChi);

}

void nodeShift()

{

int temp, update;

while (1)

{

update = 0;

for (int i = n / 2; i >= 1; --i)

{

int rightSon = 2 \* i + 1;

if (rightSon <= n && a[rightSon] > a[i])

{

temp = a[i];

a[i] = a[rightSon];

a[rightSon] = temp;

update = 1;

}

int leftSon = 2 \* i;

if (a[leftSon] > a[i] && leftSon <= n)

{

temp = a[i];

a[i] = a[leftSon];

a[leftSon] = temp;

update = 1;

}

}

if (!update) break;

}

}

int main()

{

cout<<"Input"<<endl;

while (cin>>a[n+1] && a[n+1])++n;

cout<<"Output"<<endl;

tree\_Node \*result = NULL;

for (int i = 1; i <= n; ++i)

{

result = nodeInsert(result, a[i]);

}

nodeShift();

for (int j = 1; j <= n; ++j)

{

cout<<a[j]<<(j == n ? '\n' : ',');

}

int count = 0;

while (n > 0)

{

b[++count] = a[1];

a[1] = a[n];

--n;

nodeShift();

}

for (int i = count; i >= 1; --i)

{

cout<<b[i]<<(i == 1 ? '\n' : ',');

}

Previous(result);

plugIn(result);

for (int i = 0; i < pren\_01; i++)

{

cout<<previous[i]<<(i == pren\_01 - 1 ? '\n' : ',');

}

for (int i = 0; i < inn\_01; i++)

{

cout<<in[i]<<(i == inn\_01 - 1 ? '\n' : ',');

}

cout<<"End"<<endl;

return 0;

}