第10次作业

## 第7章习题

7.3构造有20个元素的二分查找的判定树，并求解下列问题：

（1）各元素的查找长度最大是多少？

（2）查找长度为1、2、3、4、5的元素各有多少？具体是哪些元素？（假设下标从0开始）

（3）查找第13个元素依次要比较哪些元素？

7.5设计算法以构造有n个元素（下标范围从1到n）的二分查找判定树。

7.8设计算法，对给定的二叉排序树，求出在等概论情况下的平均查找长度。

7.12已知一棵二叉排序树如下，其各结点的值虽然未知，但其中序序列为1,2,3,4,5,6,7,8,9。请标注各结点的值。

图7-1 题7.12图

7.13已知散列表地址区间为0～9,散列函数为H(k)=k % 7,采用线性探测法处理冲突。将关键字序列11,22,35,48,53,62,71,85依次存储到散列表中，试构造出该散列表，并求出在等概论情况下的平均查找长度。

7.17已知散列表的地址区间为0～10，散列函数为H(k)=k % 11，采用线性探测法处理冲突。设计算法在其中查找值为x的元素，若查找成功，返回其下标，否则返回－1。

7.3

（1）各元素的查找长度最大是多少？

答：各元素的查找长度最大是该元素的深度h。

（2）查找长度为1、2、3、4、5的元素各有多少？具体是哪些元素？（假设下标从0开始）

答：查找长度为1、2、3、4、5的元素各有1、2、4、8、5。

具体是：（9）、（4、14）、（1、6、11、17）、（0、2、5、7、10、12、15、18）、（3、8、13、16、19）号元素

（3）查找第13个元素依次要比较哪些元素？

答：需要比较第9、14、11、12、13号元素。

7.5

#include<bits/stdc++.h>

using namespace std;

//定义二分查找判定树的节点

struct TreeNode {

int val;

TreeNode\* left;

TreeNode\* right;

TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

};

// 递归构造二分查找判定树

TreeNode\* buildBST(int l, int r) {

if (l > r) {

return nullptr;

}

int m = l + (r - l) / 2;

TreeNode\* root = new TreeNode(m);

root->left = buildBST(l, m - 1);

root->right = buildBST(m + 1, r);

return root;

}

//搜索元素

bool search(TreeNode\* root, int target) {

if (root == nullptr) {

return false;

}

if (root->val == target) {

return true;

}

if (root->val > target) {

return search(root->left, target);

}

else {

return search(root->right, target);

}

}

int main() {

int n;

cout << "默认n不小于5，每次搜索5个数据" << endl;

cout << "请输入元素个数n:";

cin >> n;

TreeNode\* root = buildBST(1, n);

cout << "请输入想要查找的五个数据(不超过n)" << endl;

// 测试搜索元素

int a, b, c, d, e;

cin >> a >> b >> c >> d >> e;

vector<int> values = { a, b, c, d, e };

for (int value : values) {

cout << "Search " << value << ": " << search(root, value) << endl;

}

return 0;

}

7.8

#include <bits/stdc++.h>

using namespace std;

// 定义二叉排序树的结点

struct TreeNode {

int val;

TreeNode\* left;

TreeNode\* right;

TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

};

// 在二叉排序树中插入新元素

void insert(TreeNode\*& root, int val) {

if (root == nullptr) {

root = new TreeNode(val);

}

else if (val < root->val) {

insert(root->left, val);

}

else {

insert(root->right, val);

}

}

// 计算树高和所有节点的深度之和

int dfs(TreeNode\* root, int depth, int& sum) {

if (root == nullptr) {

return 0;

}

int L = dfs(root->left, depth + 1, sum);

int R = dfs(root->right, depth + 1, sum);

sum += depth;

return 1 + L + R;

}

// 求二叉排序树的平均查找长度

double getASL(TreeNode\* root) {

int depthSum = 0, nodeNum = dfs(root, 1, depthSum);

return static\_cast<double>(depthSum) / nodeNum;

}

int main() {

// 测试

int n;

cout << "请输入二叉排序树结点的个数:";

cin >> n;

TreeNode\* root = nullptr;

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

insert(root, i);

}

double asl = getASL(root);

printf("ASL: %.2f\n", asl);

return 0;

}

7.12

图7-1 题7.12图

**4**

**1**

**3**

**2**

**8**

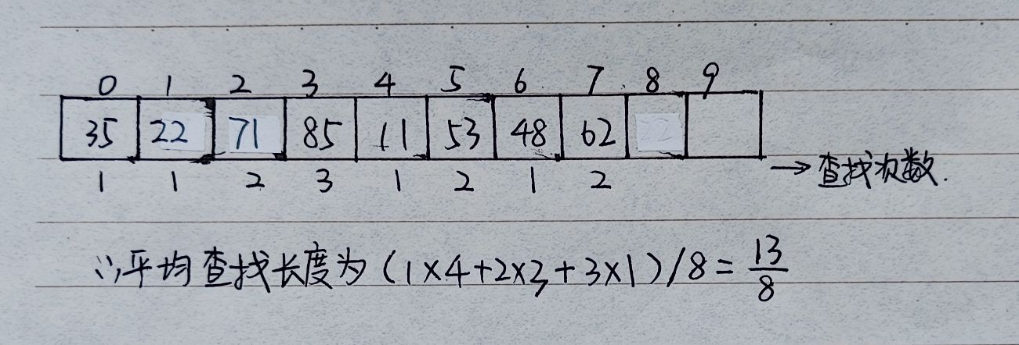
**6**

**5**

**9**

**7**

7.13



7.17

int findList(int A[], int x) {

int x1 = x % 11; //计算若存在的散列地址

while (A[x1]) { //在同义子表中顺序查找

if (A[x1] == x)

return x1; //如果找到，直接返回下标

else

x1++; //没找到，遍历下一个

}

return -1;

}

【时间分析】时间几乎都耗费在循环上，同义子表有几个元素就循环几次，所以O(n)。