1999 年 12 月
第 29 卷第 6 期
西北大学学报(自然科学版)
Journal of Northwest University(Natural Science Edition)
Dec. 1999
Vol. 29 No. 6

在 Java 语言中实现动态数据结构

赵文静

(西安建筑科技大学 自动控制系,陕西 西安 710055)

摘要:就如何在摒弃了指针的 Java 语言中实现动态数据结构提出了两种解决方案:用 vector 类和用数组实现之。以二叉树为例分别给出了两种方法的实现算法,讨论了它们各自的优缺点。

关 键 词:Java 语言;C++语言;动态数据结构;指针;数组;静态;二叉链表
中图分类号:TP36 **文献标识码**:A **文章编号**:1000-274X(1999)06-0485-03

Java 是一种完全面向对象的语言,它是由 C++衍生而来的。其语言风格类似于 C++,但又摒弃了 C++中的一些不必要的功能,并取消了指针,从而消除了用户复写内存单元或破坏有用数据的可能性,因此,Java 比 C++更容易学习且具有很高的稳定性和安全性。

动态数据结构是一种十分灵活的数据结构,它是在程序运行时,根据需要用指针申请(扩充)或释放(缩减)其数据元素所占内存单元的结构。这对于某个特定平台进行底层程序设计,非常有用,故在 Pascal,C,C++等广为流行的语言中是最重要的数据类型之一。但是,由于指针直接访问内存,会导致代码混乱,甚至影响程序的安全性。对于像 Java 这样的网络编程语言而言,安全性是至关重要的,因此 Java 摒弃了指针。

为了在去除指针的 Java 语言中实现 C++中的动态数据结构,笔者提出了两种解决方案:① 用 vector 类实现动态数据结构;② 用数组类型模拟动态数据结构,即由用户管理数组空间(建立可用空间栈)实现之。

1 用 vector 类实现动态数据结构

向量 vector 是 Java 提供的一种功能很强的数据结构,在向量中可以存放不同类型的对象,且向量的容量可以扩充。这也就是说,向量类型提供了一种与"动态数组"相近的概念(但却不是 C/C++意义上的动态数组)。下面我们以常用的数据结构——二叉树为例,说明其实现方法。

1.1 二叉树的定义

二叉树是一个集合 T:它可以是空集,也可以是一个由结点组成的有限集。同时,集合 T 具有下列的性质:① 如果 T 是空集,则称 T 是空的二叉树;② 如果 T 是有限集,则 T 有一个特定的,称之为根的结点,以及称为该根的两个互不相交的左子树和右子树构成,同时这两棵子树亦是二叉树。很明显,这是一个递归定义。一般情况下,人们常用多重链表来表示一棵二叉树。其结点包含一个数据元素和分别指向其左、右子树的两个分支,这表示二叉树的链表中的结点至少包含数据域和左、右指针域 3 个域。

1.2 用 vector 类实现任意二叉树类的方法

首先定义一个 node 类来表示二叉树的结点,该类的结构如图 1 所示。

图中 data 域用于存放结点数据,lchild 域和 rchild 域分别存放该结点左、右孩子的地址(即左、右孩子在存储二叉树结点的向量中的 index 值)。

lchild	data	rchild

图 1 二叉树结点结构

Fig. 1 Binarytree node structure

然后,为二叉树创建一个向量实例 BT,实际上就是在系统内部创建一个专用数组。向量中存放的对象就是二叉树中结点 node 类实例。每当向二叉树中插入一个结点时便对该数组进行动态调整。

建立任意二叉树,用先根递归方法是最简单自然

收稿日期:1999-02-02
基金项目:原冶金部基础研究课题资助项目(96-自-01)
作者简介:赵文静(1949-),女,浙江诸暨人,西安建筑科技大学副教授,从事数据库系统应用研究。

的方法。即:① 建立根结点;② (递归)建立根结点的左子树;③ (递归)建立根结点的右子树。

在 C++中,二叉树结点的左、右孩子域定义为指针,该算法的递归实现非常简单,直接将左、右孩子的内存地址填入相应的孩子域即可。然而,在我们的向量实现方案中则较麻烦一些,需要先用 addElement 方法将根结点存入向量,得到其存储位置(index 值),递归建立根结点的左、右子树中各结点。之后,还需将逐层递归返回的左、右孩子的存储位置回填到根结点的左、右孩子域,并用 setElementAt 方法存回原处。

我们给出的二叉树类的定义如下:

```
class BT   // 二叉树类
{ int root;   // 二叉树根结点指针
vector bt;   // 存放二叉树结点的向量
string s;int index;
public BT();// 构造方法
public int crt _ BT();//递归建立二叉树
public int Search _ BT(int r,char elem);//在二叉树中检索元素 elem
public void preOrder(int i) //先根遍历二叉树
public void inOrder(int i) //中根遍历二叉树
}
```

递归建立二叉树的方法的源码如下:

```
public int crt _ BT()
{ node node;
int l,r,i=0;
char x= '0 ';
x=s.charAt(index++);
if(x! = '0 '){
node = new node(x);// 建立二叉树结点 x
bt.addElement(node);// 将结点插入向量
i=bt.size();
l = crt _ BT()-1;r = crt _ BT()-1;
// 递归建立结点 x 的左右子树
node.lchild=l;// 填入结点 x 的左孩子地址
node.rchild=r;// 填入结点 x 的右孩子地址
bt.setElementAt(node,i-1);//重置结点 x
}
return i;
}
```

用 crt _ BT()方法,递归建立的一棵二叉树如图 2 所示,结点左面标示的数据是结点在向量中的位置(index),lchild/rchild 域填-1 表示"空"。

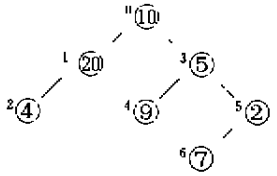

图 2 一般二叉树

Fig. 2 Binary tree

1.3 讨 论

vector(向量)类和数组存在许多类似之处,也可以说是一种扩展的数组,但数组只能保存固定大小的同一类型数据,必须一次申请所有的存储单元。引入向量的概念提供对动态数组的支持,此外还在 vector 类中封装了增、删、改、检索等各项操作。因此,向量是 Java 语言提供的一种高级数据结构。

对于本文所举的二叉树而言,由于结点对象间是非线性关系且二叉树是一种递归结构,因此必须在结点对象中存放其后继结点对象的地址(本文用 lchild,rchild 存放后继对象在 vector 中的下标)。这样,一旦要删除二叉树中某结点对象,由于向量的删除操作会自动将其后继对象依次前移,原存储下标大于被删对象下标的对象,其存储下标均被减 1,使得留下的结点中有些后继结点的地址发生了变化,而要将这种变化反映到其前趋结点(父结点)的 lchild,rchild 域,所需进行的维护操作太复杂。因此,用 vector 类实现的二叉树结构只适用于不需要删除操作的一些应用,如果需频繁进行插入、删除操作的应用,则应采用下述方法实现之。

2 用数组类型模拟动态数据结构

用数组类型模拟 C++的动态数据结构,其方法是:为数组申请足够的存储空间,将申请到的所有的存储单元建立成可用空间栈,由用户管理数组空间,每当插入结点时从可用空间栈取得一个结点存储单元,删除结点时将存储单元归还可用空间栈。现以应用很广的,需频繁进行插入、删除操作的二叉排序树为例,讨论具体实现方法,并分析该实现方案存在的优点及不足之处。

2.1 二叉排序树的定义

如果一棵二叉树的每个结点对应于一个关键码,整个二叉树各结点对应的关键码组成一个关键码集合,并且此关键码集合中的各个关键码在二叉树中是按一定次序排列的,这时我们称此二叉树为二叉排序树。

二叉排序树或者是一棵空树,或者是具有下列

性质的二叉树：① 若它的左子树不空，则左子树上所有结点的值均小于它的根结点的值；② 若它的右子树不空，则右子树上所有结点的值均大于它的根结点的值；③ 它的左右子树也分别是二叉排序树。图 3 所示为一棵关键码是整数的二叉排序树。

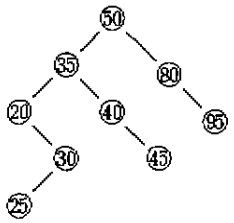

图 3　二叉排序树

Fig. 3　Binary sort tree

2.2　用数组实现二叉排序树的方法

(1) 定义二叉排序树结点类

```
class node
    { int data; // 结点数据域
    int lchild, rchild;
    public node(int d)
    }
```

(2) 定义二叉排序树结构

```
    public class BST
    { final int maxnode = 100;
    // 二叉排序树中最大结点数
    public int root;// 根结点指针
    node elements[];
    // 存放二叉排序树结点的数组
    int nodeStack[];// 二叉排序树结点空间栈
    int avail; // 可用空间栈的栈顶指针
    int father=-1; // 某结点的父结点指针
    public BST();// 构造方法
    int getNode(int x);
    void releaseNode(int p);
    public int Search _ BST(int elem);
    public void ins _ BST(int elem);
    public boolean del _ BST(int elem);
    }
```

(3) 从可用空间栈取得及释放结点的方法的具体实现

```
    int getNode(int x)
    { int p;
    p=avail; avail=nodeStack[avail];
    elements[p] = new node(x);
    return p;
    }
    void releaseNode(int p)
    { nodeStack[p]=avail; avail=p; }
```

2.3　讨　论

采用数组类型模拟 C++ 的动态数据结构实际上是用“静态二叉链表”实现动态数据结构。其优点是效率较高，可完全实现 C++ 中对二叉排序树的各项维护操作。不足之处是由用户管理数组空间，每当插入结点时从可用空间栈取得一个结点存储单元，删除结点时将存储单元归还可用空间栈，对程序员要求较高。然而，vector 实现方案中是由 Java 类库中的 vector 类提供了各项操作方法，用户不必管理存储单元。

参考文献：

[1]　廖卫东，陈　梅. Java 程序设计实用指南[M]. 北京：机械工业出版社，1996.

[2]　赵文静. 数据结构[M]. 西安：西安交通大学出版社，1999.

(编　辑　曹大刚)

Implementation of dynamic data structure in Java

ZHAO Wen-jing

(Department of Automatic, Xi'an University of Architecture & Technology Xi'an 710055, China)

Abstract: Dynamic data structure is the most important data type in Pascal, C, C++ in which dynamic data structure is implemented with pointer; How to implement dynamic data structure in Java is discussed, which does not have pointer, is discussed. Two techniques are proposed to implement with vector class and array. An example of binary tree implemented using the above two methods is given with the comparing of the methods.

Key words: Java C++; pointer; vector static binary linklist