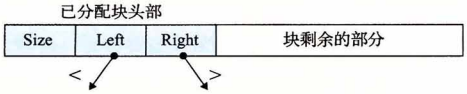

图 9-53　一棵已分配块的平衡树中的左右指针

平衡树方法保证会标记所有从根节点可达的节点，从这个意义上来说它是正确的。这是一个必要的保证，因为应用程序的用户当然不会喜欢把他们的已分配块过早地返回给空闲链表。然而，这种方法从某种意义上而言又是保守的，因为它可能不正确地标记实际上不可达的块，因此它可能不会释放某些垃圾。虽然这并不影响应用程序的正确性，但是这可能导致不必要的外部碎片。

C 程序的 Mark & Sweep 收集器必须是保守的，其根本原因是 C 语言不会用类型信息来标记内存位置。因此，像 int 或者 float 这样的标量可以伪装成指针。例如，假设某个可达的已分配块在它的有效载荷中包含一个 int，其值碰巧对应于某个其他已分配块 b 的有效载荷中的一个地址。对收集器而言，是没有办法推断出这个数据实际上是 int 而不是指针。因此，分配器必须保守地将块 b 标记为可达，尽管事实上它可能是不可达的。

9.11　C 程序中常见的与内存有关的错误

对 C 程序员来说，管理和使用虚拟内存可能是个困难的、容易出错的任务。与内存有关的错误属于那些最令人惊恐的错误，因为它们在时间和空间上，经常在距错误源一段距离之后才表现出来。将错误的数据写到错误的位置，你的程序可能在最终失败之前运行了好几个小时，且使程序中止的位置距离错误的位置已经很远了。我们用一些常见的与内存有关错误的讨论，来结束对虚拟内存的讨论。

9.11.1　间接引用坏指针

正如我们在 9.7.2 节中学到的，在进程的虚拟地址空间中有较大的洞，没有映射到任何有意义的数据。如果我们试图间接引用一个指向这些洞的指针，那么操作系统就会以段异常中止程序。而且，虚拟内存的某些区域是只读的。试图写这些区域将会以保护异常中止这个程序。

间接引用坏指针的一个常见示例是经典的 scanf 错误。假设我们想要使用 scanf 从 stdin 读一个整数到一个变量。正确的方法是传递给 scanf 一个格式串和变量的地址：

```
scanf("%d", &val)
```

然而，对于 C 程序员初学者而言(对有经验者也是如此!)，很容易传递 val 的内容，而不是它的地址：

```
scanf("%d", val)
```

在这种情况下，scanf 将把 val 的内容解释为一个地址，并试图将一个字写到这个位置。在最好的情况下，程序立即以异常终止。在最糟糕的情况下，val 的内容对应于虚拟内存的某个合法的读/写区域，于是我们就覆盖了这块内存，这通常会在相当长的一段时间以后造成灾难性的、令人困惑的后果。

9.11.2　读未初始化的内存

虽然 bss 内存位置(诸如未初始化的全局 C 变量)总是被加载器初始化为零，但是对于堆内存却并不是这样的。一个常见的错误就是假设堆内存被初始化为零：