華中科技大學

函数式编程原理
课程报告

院　　系　计算机学院

专业班级　大数据 2102 班

姓　　名　张钧玮

学　　号　U202115520

指导教师　顾琳

2023 年 10 月 22 日

目　　录

1 函数式语言家族成员调研

1.1 函数式语言简介

函数式语言是一种非冯诺依曼式的计算机编程语言，它将计算过程视为数学函数的求值，避免了状态和可变数据。函数编程语言具有没有变量和副作用,函数为一等公民,适合于并发程序等特点。

1.2 函数式语言家族成员

1.Haskell

Haskell 语言是很多北美名校计算机的入门编程语言,这是一门富有数学魅力的编程语言。

2.Meta Language

本次课程学习使用的编程语言，这门语言是为了帮助在 LCF 定理证明机中寻找证明策略而构想出来的。ML 是兼具一些指令式特征的函数式编程语言。ML 的特征包括：传值调用的求值策略，头等函数，带有垃圾收集的自动内存管理，参数多态，静态类型，类型推论，代数数据类型，模式匹配和异常处理。

3.Lisp

Lisp 是一种基于符号的编程语言，它是第一种被广泛使用的高级语言，也是第一种被广泛使用的函数式编程语言。Lisp 是一种多范式的编程语言，它支持过程式编程，函数式编程，面向对象编程和元编程。Lisp 的语法是由 S 表达式表示的，它使用前缀表示法，它的函数调用和语法规则都是一样的。

4.Java

Java8 通过 lambda 表达式、Stream API引入了函数式编程。事实上很多高级语言例如 C++ 或者 Python 近10年都增加了函数式编程的支持。

2 上机实验心得体会

2.1 实验 3.4 heapify

2.2 heapify 的定义

heapify (最小堆)是一个函数，它将一个无序的数组转换成一个堆。一棵 min-heap 树定义为：

t is Empty;

t is a Node(L, x, R), where R, L are minheaps and values(L), value(R) >= x (value(T) 函数用于获取树T的根节点的值）;

2.2.1 实验要求

编写函数treecompare, SwapDown 和heapify：

treecompare: tree * tree -> order

(* when given two trees, returns a value of type order, based on which tree has a larger value at the root node *)

SwapDown: tree -> tree

(* REQUIRES the subtrees of t are both minheaps)

(ENSURES swapDown(t) = if t is Empty or all of t's immediate children are empty then * just return t, otherwise returns a minheap which contains exactly the elements in t. *)

heapify : tree -> tree

(* given an arbitrary tree t, evaluates to a minheap with exactly the elements of t. *)

2.2.2 实验思路

函数 treecompare 是易于实现的。只需要比较二叉树节点的两个子节点。首先考虑两个节点中存在空节点的情况，如果有一个子节点为空，那么它自然小于另外一个非空子节点，函数值为 LESS 或者 GREATER 。倘若两个子节点都为空，那么函数值应当为 EQUAL 。若两个子节点均为非空，则通过构造函数 br 比较两个子节点的值，返回 LESS 或者 GREATER 。实验代码如下：

```
fun treecompare (Empty, Empty) = EQUAL
 | treecompare (Empty, _) = LESS
 | treecompare (_, Empty) = GREATER
 | treecompare (Br(_, a1, _), Br(_, a2, _)) =
   case Int.compare(a1, a2) of
     GREATER => GREATER
   | EQUAL => EQUAL
   | LESS => LESS;
```

函数 SwapDown 是实现最小堆的关键函数，它的作用对一个最小堆加入一个数并且保持最小堆的性质。首先考虑树为空的情况，这时候直接返回树。然后考虑树的根节点的两个子节点都为空的情况，这时候直接返回树。最后考虑树的根节点的两个子节点都不为空的情况，这时候需要比较根节点和两个子节点的值，如果根节点的值小于两个子节点的值，那么直接返回树。如果根节点的值大于两个子节点的值，那么需要将根节点的值与两个子节点中较小的值交换，然后递归调用 Swap-Down 函数，直到树的根节点的值小于两个子节点的值。实验代码如下：

```
fun SwapDown (Empty : tree) : tree = Empty
 | SwapDown (Br(Empty, a, Empty) : tree) = Br(Empty, a, Empty)
 | SwapDown (Br(t1 as Br(t11, a1, t12), a, Empty) : tree) =
   if treecompare(Br(t1,a,Empty), t1) = GREATER then
     Br(t1, a, Empty)
   else
     Br(Br(t11,a, t12), a1, Empty)
 | SwapDown (Br(Empty, a, t2 as Br(t21, a2, t22)) : tree) =
   if treecompare(Br(Empty, a, t2), t2) = LESS then
     Br(Empty, a,t2)
   else
     Br(Empty, a2, Br(t21,a,t22))
 | SwapDown (Br(t1 as Br(t11, a1, t12), a, t2 as Br(t21, a2, t22)) : tree) =
   if treecompare(Br(t1,a,t2),t1)=LESS andalso treecompare(Br(t1,a,t2),t2)=LESS then
     Br( t1, a,  t2)
```

```
  else if treecompare(Br(t1,a,t2),t1) = GREATER  then
    Br(Br(t11,a,t12), a1, t2)
  else
    Br(t1, a2, Br(t21,a,t22));
```

函数 heapify 是一个简单的函数，它的作业是将一个任意的树转换为最小堆。只需要不断对一个普通二叉树堆化直到每个节点都遍历过为止，返回的便是一个最小堆二叉树。实验代码如下：

```
fun heapify (Empty : tree) : tree = Empty
 | heapify (Br(t1, a, t2) : tree) =
   let
     val swappedT1 = heapify t1
     val swappedT2 = heapify t2
   in
     SwapDown (Br(swappedT1, a, swappedT2))
   end;
```

2.3 实验4.3 mapList

2.3.1 实验要求

编写函数 mapList，要求：① 函数类型为: ((a -> b) * a list) -> b list；② 功能为实现整数集的数学变换(如翻倍、求平方或求阶乘)

编写函数 mapList2，要求：① 函数类型为: (a -> b) -> (a list -> b list)；② 功能为实现整数集的数学变换(如翻倍、求平方或求阶乘)。

2.3.2 实验过程

对于 mapList，因为函数入参是由变换函数 f 和待处理list a 构成的列表，而函数的值是关于结果 b 的list，所以需要对 a list 进行遍历，对每个元素进行函数 f 的运算，然后将结果放入结果 list b中。实验代码如下：

```
fun mapList (f, lst) =
  case lst of
    [] => []  (* 空列表返回空列表 *)
   | x::xs => f x :: mapList (f, xs);
```

对于 mapList2，它接受一个函数 f，该函数将 a 类型映射到 b 类型，以及一个 a list 类型的整数列表。它直接返回一个 b list 类型的新列表。它和 maplist的区别就在于它们接收的参数不同， maplist 接收映射关系和 a list构成的元组而 maplist2 直接接收映射关系和 a list。实验代码如下：

```
fun mapList2 f =
  let
    fun mapFn [] = []
     | mapFn (x::xs) = f x :: mapFn xs;
  in
    mapFn
  end;
```

2.4 实验4.4 exists

2.4.1 任务描述

编写函数： exists: (int -> bool) -> int list -> bool 要求：对函数p: int -> bool, 整数集L: int list,有：

exist p L =>* true if there is an x in L such that p x=true;

exits p L =>* false otherwise.

2.4.2 实验过程

根据题意函数 exists接受一个谓词函数 p 和一个整数列表 L，并检查列表中是否存在某个元素满足谓词 p，如果是，则返回 true，否则返回 false。我们可以通过递归遍历 list 得到函数的值，因此代码如下：

```
fun exists p [] = false  (* 如果列表为空，返回 false *)
 | exists p (x::xs) =
   if p x then true  (* 如果 p x 为 true，返回 true *)
   else exists p xs;  (* 否则继续在剩余的列表中查找 *)
```

3　课程建议和意见

老师很活泼上课互动很积极，课堂氛围很好，满分捏。