# 函数式编程报告

## 1课程内容梳理总结

### 1.1涉及内容

1. 什么是函数式编程、函数式编程语言的特点：引用透明性、高阶函数、惰性求值（延迟计算）与并行、递归调用与尾递归、模式匹配；
2. 介绍了函数式的基本表达式和标准类型（如基础类型unit, int, real, bool, string和表、元组、函数）和声明的作用域、模式和匹配、求值符号的使用、规则说明以及ML中的等式；
3. 程序正确性验证的方法（完全归纳法、结构归纳法等）、近似运行时间分析、时间复杂度的计算、程序执行情况W(n)分析；
4. 以排序算法为例，进行了程序编写、正确性验证和性能分析；
5. 介绍了tree类型，怎么进行类型检测、多态特点、类型推导、多态特征
6. 由函数作为值引出了高阶函数、高阶是函数“多态”的一种体现并举了两个例子：list数据的单独求解问题：map函数和list数据的联合求解问题：foldr，foldl函数

### 1.2课程目标

1. 掌握函数式编程方法
2. 掌握程序书写规范，并采用严格的推导方法证明程序的正确性
3. 掌握串/并行程序的性能分析方法
4. 掌握各种数据结构的特点，学会选择合适的数据结构进行功能设计，提高程序效率

## 2实验分析总结

### 2.1实验一

1. 实验目的

1. 熟悉SML/NJ开发环境及使用
2. 掌握SML基本语法和书写规则
3. SML简单程序设计和程序编写

2. 实验内容

学会使用ML表达式以及标准类型并按照要求编写了简单的功能函数。我举两个例子作介绍。

实验1：完成函数Mult: int list list -> int的编写,该函数调用mult 实现int list list中所有整数乘积的求解。

求解思路：之前已经有mult(L)函数，传入的是一个元素为Int 类型的list，功能是将list中的元素相乘，返回int值。在Mult[]中，传入的参数是一个二维列表即list的元素也是list。当传入的为[]时，应返回值为1，当传入不为空时，先取出list list表头，对表头调用mult函数计算该list的乘积，list list 剩余部分继续递归。

求解代码：fun Mult[] = 1

|Mult(r::R) = mult(r) \* Mult(R);

实验2：编写奇数判断函数oddP: int -> bool，当且仅当该数为奇数时返回true

求解思路：该函数为fn int->bool类型，核心部分是对int值进行是否奇数的判断。依然是进行递归运算。将传入的整型值n不断减2递归到底，总能递归到0或1，0是偶数，则返回false，1为奇数，返回true。

求解代码：fun oddP(0:int):bool = false

|oddP(1) = true

|oddP(n) = oddP(n-2);

### 2.2实验2

1. 实验目的

1. 掌握list结构的ML编程方法和程序性能分析方法
2. 掌握基于树结构的ML编程方法和程序性能分析方法

2. 实验内容

学习了tree的数据类型，并进行了树的建立、遍历、查询。我举几个例子作介绍。

实验1：编写函数reverse和reverse’，要求：

函数类型均为：int list->int list，功能均为实现输出表参数的逆序输出。

求解思路：用到了ML内的追加符，依然是递归，当list为空时，逆序list也为空，其他情况下，取出表头，将其用追加符添加到List尾部，剩余部分依次递归。

求解代码：fun reverse [] = []

|reverse (x::xs) = (reverse xs) @ [x]；

实验2：编写函数listToTree: int list -> tree，将一个表转换成一棵平衡树。

求解思路：首先声明tree的类型。传入为空list时返回空树，其他情况，将list按照长度分为两部分，用ML内置的drop函数取后半部分作为继续递归的右子树及根节点，take取前半部分作为递归的左子树。因此左子树比右子树优先建立。本题我采用的是中序建立。

求解代码：datatype tree = Empty | Br of tree \* int \* tree

fun listToTree [] = Empty

| listToTree xs =

let val k =length xs div 2

val y::ys = List.drop(xs,k)

in Br(listToTree (List.take(xs,k)),y,listToTree ys)

end

实验3：编写函数binarySearch: tree \* int -> bool。当输入参数1为有序树时，如果树中包含值为参数2的节点，则返回true；否则返回false。

求解思路：首先，此题用到了数据类型order（即GREATER | EQUAL | LESS）和ML内置函数Int.Compare。函数传入参数为一个tree和一个int，返回值为bool类型。当传入为空树，则不管int为何，均不能匹配，此处用到了模式匹配符号” \_” ，其他情况，当传入一棵树（默认该树是有序的且由小到大建立），首先将int与其根节点进行比较（调用compare），若返回EQUAL，即刚好找到，返回true，若返回GREATE，表明根节点大于int值，则可能与int相等的值必在左子树，则以左子树调用该函数递归查找，若返回LESS，表明根节点小于int值，则可能与int相等的值必在右子树，则以右子树调用该函数递归查找。

求解代码：fun binarySearch(Empty:tree,\_:int):bool = false

|binarySearch(Br(t1, x, t2),a) = (case Int.compare(x,a) of

GREATER => binarySearch(t1,a)

| EQUAL => true

| LESS => binarySearch(t2,a))；

### 2.3实验3

1. 实验目的

1. 掌握多态类型、option类型和高阶函数的编程方法
2. 利用ML语言求解实际问题

2. 实验内容

学习使用了多态函数、OPTION类型并用高阶函数解决了实际问题。我举几个例子作介绍。

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

求解思路：传入参数为一个函数f和一个多态的list，即对List中每个元素都进行一次f操作，依然是递归。PPT上有个问题是比较函数mapList’和mapList，分析、体会它们有什么不同。我认为前者返回的是一个多态的值，而后者返回的是一个映射。

求解代码：fun mapList(f, []) = []

| mapList (f, x::L) = f(x)::(mapList(f, L))；

实验2：编写函数findOdd，要求：  
 ① 函数类型为: int list -> int option；  
 ② 功能为：如果x为L中的第一个奇数，则返回SOME x；否则返回NONE

求解思路：这里提到了Options类型的概念， option即将空值和一般值包装成同一种类型，我觉得也是为了方便多态。函数实现即，当list为空时，返回NONE，其他情况，取出表头，对其取2余运算，余数为1即为奇数，返回SOME x，为奇数则将剩余部分递归直至找到返回SOME x或者List为空，返回NONE。

求解代码：fun findOdd [] = NONE

| findOdd (x::L) =

if (x mod 2 = 1) then

SOME x

else

findOdd(L)；

实验3：编写函数subsetSumOption: int list \* int -> int list option，要求：

对函数subsetSumOption(L, s)：如果L中存在子集L’，满足其中所有元素之和为s，则结果为SOME L’；否则结果为NONE  
 求解思路：我这里套用了书上的做法，穷举做的。即先找出传入的list的所有子集，然后再计算子集的和，判断满不满足s。其中用到了PPT上的map、folder函数。

求解代码：fun map f [] = []

| map f(x::R) = (f x)::(map f R)

fun sublists [] = [[]]

| sublists (x::R) =

let

val S = sublists R

in

S @ (map (fn A => x::A) S)

end

fun foldr F z [] = z

| foldr F z (x::L) = F(x, foldr F z L)

fun sum L = foldr(op+) 0 L

fun findsub([], a) = NONE

| findsub(x::L:int list list, a) =

if(sum(x) = a)

then SOME(x)

else findsub(L, a)

fun subsetSumOption(L:int list, a:int) =

let val sub = sublists(L)

in findsub(sub, a)

end；

实验4：编写函数：treeFilter: (‘a -> bool) -> ‘a tree -> ‘a option tree

将树中满足条件P（ ‘a -> bool ）的节点封装成option类型保留，否则替换成NONE。  
 求解思路：关系P任意，只要是返回bool的函数即可。做法即遍历书，将采用递归，到某个节点是将其代入关系P判断是否满足，若满足将该点封装成SOME（x），不满足则封装成NONE。

## 3 实验心得

刚上实验一开始比较迷，因为上课也没好好听，有点无从下手，然后就自己看ML程序设计和老师的PPT，才感觉实验基本和上课内容差不多，上课的知识弄懂了就差不多了。几个实验下来，觉得，哎ML语言好方便啊，很多类型直接就定义了，还有许多内置函数，比如数的定义，就特别好懂，甚至和自然语言差不多我觉得。程序设计部分我也觉得主要是算法的设计，然后代码写起来并不太难，另外ML中用递归真的超级好用啊(\*^\_^\*)。

函数式编程老师讲课还是很棒的、深入浅出又不乏幽默，所以后面我都开始认真听课了、、（羞愧），总而言之，这门课还是收获很多滴。