

第二次作业

1 请用归纳法证明 isort()函数的正确性 (通过 ins()函数来证明)

代码如下:

```
fun ins x [] = [x]
  | ins x (y::ys) =
    if x < y then x::y::ys
    else y :: (ins x ys)

fun isort [] = []
  | isort (x::L) = ins x (isort L)
```

ins 函数的作用是将一个数(第一个参数)插入到一个升序数组中(第二个参数)
能匹配到两种情况:

1. 第二个参数是空数组, 则直接插入即可
2. 第二个参数有元素, 则先得到第一个元素(y)和剩余元素(ys)
 1. 若 $x < y$, 则将 x, y, ys 依次连接并返回
 2. 若 $x \geq y$, 则尝试将 x 插入到 ys 中

分析:

1. 初始条件(上述第一点)成立
2. 每次迭代会使数组长度减一, 最终一定能终止
3. 迭代条件(上述第二点)能保证数组一定是升序的

综上, 此程序是正确的

2 分析下列表达式的类型

1 fun all

- 类型: $fn : int * string\ list \rightarrow string\ list$

- 作用: 将"are belong to us"这个字符串重复第一个参数这么多次, 与第二个参数连接, 返回连接后的数组

```
fun all(your, base)=
  case your of
    0 => base
  | _ => "are belong to us"::all(your-1, base)
```

2 fun funny

- 类型: `fn : ('a * int -> int) * 'a list -> int`
- 作用: 实现 reduce 的功能, 将第二个参数的每一项和当前 reduce 的值(初始为 0)依次传入第一个参数这个函数, 返回下一次 reduce 的值

```
fun funny (f, []) = 0
  | funny(f, x::xs)=f(x, funny(f, xs))
```

3 Hello, World

- 类型: `fn : ANY -> string`
- 作用: 该表达式的值是一个函数, 函数接受一个参数, 但是不管传入什么参数都会返回字符串 "Hello, World!"

```
(fn x => (fn y => x)) "Hello, World!"
```

3 分析下列函数的执行性能

```
fun fib n =
  if n<=2 then 1
  else fib(n-1) + fib(n-2);
```

(* 这里改成 1 似乎合理一些? 从 1 开始计数, 和 fib 函数保持统一 *)

```
fun fibber (1: int) : int * int = (1, 1)
  | fibber (n: int) : int * int =
    let val (x: int, y: int) = fibber (n-1)
    in (y, x + y)
    end
```

1. fib 函数使用递归的方法求解 Fibonacci 数列, 自顶向下, 但是会进行多次重复计算, 复杂度为 $O(2^n)$
2. fibber 函数使用的是迭代的做法, 自底向上, 没有重复计算, 复杂度为 $O(n)$

