ソフトウェアサイエンス実験 S8 課題 1-3

200911434 青木大祐

平成 24 年 11 月 11 日

1.3.1 最大公約数

ユークリッドの互除法を用いて最大公約数を計算する。引数のどちらかが負数だった場合は絶対値を引数にして計算を行う。

```
let rec gcd (x, y) =
  if x <= 0 || y <= 0 then gcd (abs x, abs y)
  else if x = y then x
  else if x > y then gcd(x-y, y)
  else gcd(x, y-x);;

gcd(10, -25);;
gcd(1287192390, 1293749908);;
```

リスト 1: 実行結果

```
#   - : int = 5
# - : int = 22
```

正しく計算できていることが分かる。

1.3.2 フィボナッチ数

以下の関数はフィボナッチ数列の、引数に指定した n 番目の数字を出力する。

```
let rec fib n =
  match n with
      1 -> 1
    | 2 -> 1
    | n -> fib(n - 2) + fib(n - 1);;

fib 90000;;
```

リスト 2: 実行結果

```
   #   - : int = 55
   # - : int = 6765
```

正しく計算できていることが分かる。

1.3.3 クイックソート

再帰を用いてクイックソートを実装した。リストの先頭を pivot に指定した List.partition で左右に分割し、それぞれに対して再帰的にクイックソートを適用していく。

```
let rec quicksort l =
  match l with
    | pivot::rest -> let less, greater = List.partition (fun x -> x < pivot) rest in
                     quicksort less @ [pivot] @ quicksort greater
    | _ -> [];;

quicksort [14;5;13;5;7;3;1;32;52;7];;
```

リスト 3: 実行結果

```
   #   - : int list = [1; 3; 5; 5; 7; 7; 13; 14; 32; 52]
```

これより正しくソートされていることがわかる。