函数式语言程序设计课程实验

2015010370 寇植

算法思路

基于问题中的猜数字游戏，猜测返回的结果二元组，既$Score(G,C)$有这样的性质，我们假设C为设定好的数字，G为某一次的猜测值，那么在给定的数字取值范围$[a_1..a_n]$中，若$a_i = C$，那么一定有$Score(a_i,C) = Score(G,C)$，同时$Score$函数具有交换性，既$Score(G,C) = Score(C,G)$，那么对于给定的数字取值范围中，可以通过每次得到的返回结果，通过$Score$函数，筛出符合要求的a_i，即满足$Score(a_i,C) = Score(G,C)$的$a_i$，将它们作为下一次猜测的取值范围，通过一次次筛选，最终得出C的取值

算法实现

自定义数据结构——游戏状态
其中$guessNum$表示最近的一次猜测值，$guessEval$表示最近的一次返回结果，$currentRange$表示当前的取值范围

```
data GameState = State {
    guessNum :: Int,
    guessEval :: (Int, Int),
    currentRange :: [Int]
} deriving (Show)
```

$guess$函数每次从当前的取值范围中取出一个值作为猜测值，选择的函数为自定义的$select$函数，这里为取出第一个元素

```
guess ::  RandomGen g => g -> GameState -> (Int, GameState)
guess g s = (y, s) where
  y = select $ currentRange s
```

每次$refine$函数更新状态，新的取值范围通过筛选函数$prune$得到，同时将得到的结果值更新到$guessEval$

```
refine :: (Int, GameState) -> (Int, Int) -> GameState
refine (y, s) (a, b) = s' where
  newRange = prune y (a, b) (currentRange s)
  s' = State { guessNum = y, guessEval = (a,b), currentRange = newRange}
```

筛选函数$prune$，遍历输入的取值范围，根据前面提到的算法去掉不符合条件的值，得到新的取值范围，其中的$judge$函数是判断$Score(s,x)$是否等于(a,b)

```
prune :: Int -> (Int, Int) -> [Int] -> [Int]
prune s (a, b) [x] = judge s (a,b) x
prune s (a, b) (x:xs) = judge s (a, b) x ++ (prune s (a,b) xs)
```

实验结果

在目录下执行 `stack test`
$C = 487$

```
you guess 123 -> (1, 1)
you guess 456 -> (2, 2)
you guess 478 -> (4, 2)
you guess 487 -> (4, 4)
Bingo!
it takes 4 guesses
```

$C = 4762$

```
you guess 123 -> (1, 0)
you guess 1456 -> (2, 0)
you guess 2547 -> (3, 0)
you guess 4075 -> (2, 1)
you guess 4762 -> (4, 4)
Bingo!
it takes 5 guesses
```

$C = 9527$

```
you guess 123 -> (1, 1)
you guess 456 -> (1, 0)
you guess 4178 -> (1, 0)
you guess 5729 -> (4, 1)
you guess 7925 -> (4, 1)
you guess 9527 -> (4, 4)
Bingo!
it takes 6 guesses
```