猜数字游戏实验报告

2015080121 软工52 李在弦

算法设计

1. 在initialize函数里对可能集（S）进行过滤出现重复数字的数的操作。
2. 通过guess函数随机得到可能集（S）中的一个元素
3. 在refine函数里对可能集（S）中的所有元素（sElem）检查它是否符合(P,Q)的条件

   3.1 先判断这个元素（数字）是否满足P的条件。假设我们称从步骤2随机得到的元素为A，称可能集（S）中的某一个元素为B。这时候，为了更容易判断它是否满足P的条件，把A和B的类型换成字符串，然后对A[0],A[1],A[2],A[3]做 A[i] `elem` B 的判断。如果其中n个数字满足P条件，这时候，如果（n == P），那么我们认为B满足P的条件，所以把B保留在可能集（S）中；如果（n /= P），那么我们认为B不满足P的条件，所以从可能集（S）中删除B。

   3.2 接着判断这个元素（数字）是否满足Q的条件。这个过程比判断P条件的更简单，我们检查A[i]和B[i]是否相等。如果n次满足（A[i] == B[i]）的条件，而且（n == Q），那么我们认为B满足Q的条件，所以把B保留在可能集（S）中；如果（n /= Q），那么我们认为B不满足Q的条件，所以从可能集（S）中删除B。

直到可能集（S）中只剩一个元素，反复步骤2和3。这样完成猜数字。

实验结果

对所有“没有出现重复数字”而“在（0100≤n≤9999）范围内”的数字的总共猜测次数为27633

平均猜测次数约为 5.48

函数说明

```
initialize :: (Int -> Bool) -> GameState
initialize _ = filter checkValid [1..9999]
```

进行过滤出现重复数字的数

```
checkValid :: Int -> Bool
checkValid n = if ((n `div` 1000) /= 0)
               then 100 <= n && n <= 9999 && (length (Data.List.nub (show n)) == 4)
               else 100 <= n && n <= 9999 && (length (Data.List.nub ("0" ++ show n)) == 4)
```

实现过滤函数

```
guess :: RandomGen g => g -> GameState -> (Int, GameState)
guess g s = ((s!!) $ fst $ randomR (0000,  length s - 1) g, s)
```

随机得到可能集（S）中的一个元素

```
refine :: (Int, GameState) -> (Int, Int) -> GameState
refine (x, s) (p, q) = filter (checkEqualPQ x p q) s
```

对可能集（S）中的所有元素检查它是否符合(P,Q)的条件

```
checkEqualPQ :: Int -> Int -> Int -> Int -> Bool
checkEqualPQ x p q sElem = ((checkEqualP (makeString x) (makeString sElem)) == p) && ((checkEqualQ (makeString x) (makeString sElem)) == q)
```

判断当前元素是否符合(P,Q)的条件

```
makeString :: Int -> String
makeString n = let str = (show n)
               in if (length str == 3)
                  then "0" ++ str
                  else str
```

把数字转换成字符串

```
checkEqualP :: String -> String -> Int
checkEqualP xs sElem = if (length xs == 0) then 0
                       else if ((head xs) `elem` sElem) then 1 + checkEqualP (tail xs) sElem
                       else checkEqualP (tail xs) sElem
```

判断当前元素是否满足P的条件

```
checkEqualQ :: String -> String -> Int
checkEqualQ xs sElem = let a = if ((xs !! 0) == (sElem !! 0)) then 1 else 0
                           b = if ((xs !! 1) == (sElem !! 1)) then 1 else 0
                           c = if ((xs !! 2) == (sElem !! 2)) then 1 else 0
                           d = if ((xs !! 3) == (sElem !! 3)) then 1 else 0
                       in (a+b+c+d)
```

判断当前元素是否满足Q的条件