Stats102A, Summer 2023 - Homework 4

Luke Villanueva - 206039397

08/06/23

1: Dealing with Large Numbers

```r
source("206039397_stats102a_hw4.R")
```

Constructor Function and Generic Function Demos

```r
# create objects
o1 <- pqnumber(1,3,4,1:8)
o2 <- pqnumber(1,6,0,c(3,9,5,1,4,1,3))
o3 <- pqnumber(-1,5,1,c(2,8,2,8,1,7,2))

# demonstrating is__pqnumber()
is_pqnumber(o1)
```

```
## [1] TRUE
```

```r
is_pqnumber(o2)
```

```
## [1] TRUE
```

```r
is_pqnumber(o3)
```

```
## [1] TRUE
```

```r
# demonstrating print.pqnumber()
print(o1)
```

```
## [1] sign = 1
## [1] p = 3
## [1] q = 4
## [1] nums =
## [1] 1 2 3 4 5 6 7 8
```

```r
print(o1,DEC = T)
```

```
## [1] 87654.321
```

```r
print(o2)
```

```
## [1] sign = 1
## [1] p = 6
## [1] q = 0
## [1] nums =
## [1] 3 9 5 1 4 1 3
```

```r
print(o2,DEC = T)
```

```
## [1] 3.141593
```

```r
print(o3)
```

```
## [1] sign = -1
## [1] p = 5
## [1] q = 1
## [1] nums =
## [1] 2 8 2 8 1 7 2
```

```r
print(o3,DEC = T)
```

```
## [1] -27.18282
```

```r
# demonstrating as_pqnumber
demo <- as_pqnumber(c(0,4,1,3,0,0,0,0),3,4)
print(demo,DEC = T)
```

```
## [1] 3.14
```

```r
# demonstrating as_numeric
as_numeric(demo)
```

```
## [1] 0 4 1 3 0 0 0 0
```

Addition and Subtraction

Algorithms

```
FUNCTION carry-over(vec)

  SET overflow = 0
```

```
  SET indx = vector of nums along vec

  FOR i in indx
    SET vec[i] = vec[i] +overflow

    IF vec[i] > 9
      SET overflow = 1
      SET vec[i] = vec[i] - 10
    END IF
    ELSE
      SET overflow = 0
    END ELSE


    IF i == indx[last indx] & overflow == 1
      SET temp = l
      SET l = vec length + 1 of l
      FOR j in vec along l
        SET l[j] = temp[j]
      END FOR

      l[i+1] <- 1

    END IF

  END FOR

  RETURN l

END FUNCTION


FUNCTION borrowing(v)

  SET indx = vec along v

  WHILE if any of v < 0

    FOR i in indx
      IF v[i] < 0
        SET v[i+1] = v[i+1] -1
        SET v[i] = v[i] + 10
      END IF
    END FOR

  END WHILE

END FUNCTION


FUNCTION add(x,y)

 DECLARE res_sign, rs
```

```
 SET xs = nums vec in x
 SET ys = nums vec in y
 SET svals = vec -p to q of x

 IF x sign == 1 & y sign == 1
  SET rs = xs + ys
  SET res_sign = 1
 END IF

 IF x sign == 1 & y sign == -1
  IF x > y
    SET rs = xs - ys
    SET res_sign = 1
  END IF
  ELSE
    SET rs = ys - xs
    SET res_sign = -1
  END ELSE
 END IF

IF x sign == -1 & y sign == 1
  IF x > y
    SET rs = xs - ys
    SET res_sign = -1
  END IF
  ELSE
    SET rs = ys - xs
    SET res_sign = 1
  END ELSE
 END IF

 IF x sign == -1 & y sign == -1
  SET rs = xs + ys
  SET res_sign = -1
 END IF

 WHILE if any rs > 9 or rs < 0
  SET rs = carry_over(rs)
  SET rs = borrowing(rs)
 END WHILE

  WHILE svals length is < rs length
    SET svals = append (svals + (last val of svals + 1))
  END WHILE

  RETURN res_sign * sum(rs * 10^svals)

END FUNCTION


FUNCTION subtract

 DECLARE res_sign, rs
```

```
 SET xs = nums vec in x
 SET ys = nums vec in y
 SET svals = vec -p to q of x

 IF x sign == 1 & y sign == 1
  IF x > y
    SET rs = xs - ys
    SET res_sign = 1
  END IF
  ELSE
    SET rs = ys - xs
    SET res_sign = -1
  END ELSE
 END IF

 IF x sign == 1 & y sign == -1
  SET rs = xs + ys
  SET res_sign = 1
 END IF

IF x sign == -1 & y sign == 1
    SET rs = xs + ys
    SET res_sign = -1
 END IF

 IF x sign == -1 & y sign == -1
  IF x > y
    SET rs = xs - ys
    SET res_sign = -1
  END IF
  ELSE
    SET rs = ys - xs
    SET res_sign = 1
  END ELSE
 END IF

 WHILE if any rs > 9 or rs < 0
  SET rs = carry_over(rs)
  SET rs = borrowing(rs)
 END WHILE

  WHILE svals length is < rs length
    SET svals = append (svals + (last val of svals + 1))
  END WHILE

  RETURN res_sign * sum(rs * 10^svals)

END FUNCTION
```

Demonstrations

```r
o2 <- pqnumber(-1,3,4,c(2,8,1,7,2,0,0,0))

add(o1,o2)
```

```
## [1] 87627.139
```

```r
# check accuracy
print(o1,DEC = T) + print(o2,DEC = T)
```

```
## [1] 87654.321
## [1] -27.182

## [1] 87627.14
```

```r
add(o2,o1)
```

```
## [1] 87627.139
```

```r
# check accuracy
print(o1,DEC = T) + print(o2,DEC = T)
```

```
## [1] 87654.321
## [1] -27.182

## [1] 87627.14
```

```r
subtract(o1,o2)
```

```
## [1] 87681.503
```

```r
# check accuracy
print(o1,DEC = T) - print(o2,DEC = T)
```

```
## [1] 87654.321
## [1] -27.182

## [1] 87681.5
```

```r
subtract(o2, o1)
```

```
## [1] -87681.503
```

```r
# check accuracy
print(o2,DEC = T) - print(o1,DEC = T)
```

```
## [1] -27.182
## [1] 87654.321

## [1] -87681.5
```

Problem 2: Root-Finding Problem

1.

```r
bisection <- function(a,b,f,tol)
{
  mid <- (a+b)/2
  f_mid <- f(mid)
  while(abs(f_mid) - 0 > tol)
  {
    # if f mid is < 0, then 0 is on right interval
    if(f_mid < 0)
    {
      a <- mid
    }
    else
    {
      b <- mid
    }

    mid <- (a+b)/2
    f_mid <- f(mid)
  }

  return(mid)

}

f1 <- function(x)
{x**3 + 23}

f2 <- function(x)
{x**x - 18}

f3 <- function(x)
{exp(-x**(2)) - (1/10)}
```

The formula for the minimum number of iterations is: $n >= \frac{log(b-a)-log(tol)}{log(2)}$

```r
bisection(-5,5,f1,10**-8)
```

```
## [1] -2.843867
```

[-5,5], estimated iterations: 30

```r
bisection(-5,5,f2,10**-8)
```

```
## [1] 2.803663
```

[-5,5], estimated iterations: 30

```
bisection(-5,5,f3,10**-8)
```

```
## [1] -1.517427
```

[-5,5], estimated iterations: 30

2.

Fixed point algorithm:

As long as the difference between g_res and x is more than the tolerance, the code will repeatedly set x = g(x) and then find g(x), such that g is f but rearranged to be equal to x.

```
g <- function(x)
{18/log(x)}

fixed_point <- function(x,g,tol)
{
  g_res <- g(x)
  while(abs(g_res - x) > tol)
  {
    x <- g_res
    g_res <- g(x)
  }

  return(g_res)

}

fixed_point(2,g,10**-8)
```

```
## [1] 8.439243
```

Newton's method algorithm:

As long as the $x_1 - x_0$ is greater than the tolerance, set $x_1 = x_0 - \frac{f(x)}{f'(x)}$ and $x0 = x1$.

```
f <- function(x)
{x**x - 18}

f_deriv <- function(x)
{(x**x) * (log(x) + 1)}

newton <- function(x0,f,fd,tol)
{
  x1 <- x0 - (f(x0)/fd(x0))
  while(abs(x1-x0) > tol)
  {
    x0 <- x1
    x1 <- x0 - (f(x0)/fd(x0))
  }
```

```
  return(x1)

}

newton(2,f,f_deriv,10**-8)
```

```
## [1] 2.803663
```