ESS212-HW1

Yiting Yu

2024-01-15

Problem 1

Yiting's Github repository url

Problem 2

```
language <- "R"
cat(sprintf("%s says: Hello, World!\n", language))
```

```
## R says: Hello, World!
```

Please see the GitHub repository for Python, Julia, and Matlab code for question 2.

Problem 3

$$S_1(n) \equiv 1 + 2 + 3 + ... + n = \frac{n(n+1)}{2}$$
$$O_1(n) \equiv 1 + 3 + 5 + ... + (2n-1) = n^2$$
$$S_2(n) \equiv 1^2 + 2^2 + 3^2 + ... + n^2 = \frac{n(n+1)(n+2)}{6}$$

```
# S_1(n)
# iterative algorithm
S1_iter <- function(n) {
  S <- 0
  for (i in  1:n) {
    S <- S + i
    }
  return(S)
  }

# recursive algorithm
S1_rec <- function(n) {
  if (n > 1) {
    S = n + S1_rec(n-1)
    }
  else{
    S = 1
    }
  return(S)
  }
```

Verify

```
n <- c(1, 10, 100)
(r_s1 <- data.frame(
  n = n,
  iter_result = sapply(n, S1_iter),
  rec_result = sapply(n, S1_rec),
  real_result = n*(n+1)/2
))
```

```
##     n iter_result rec_result real_result
## 1   1           1          1           1
## 2  10          55         55          55
## 3 100        5050       5050        5050
```

Describe the internal state of the programs during the computational process.

The iterative algorithm initializes a sum variable S to 0 and iterates from 1 to n, accumulating the sum of numbers in each iteration. Once it reaches n, the function returns S, the total sum of the first n positive integers.
The recursive algorithm for summing integers from 1 to n decrements n and accumulates values until n is 1. As recursion unwinds, it adds each n to the total, returning the sum of integers from 1 to the original n.

Using either the recursive algorithm or the iterative algorithm, write and test computer programs to evaluate O1(n) and S2(n).

```
# O_1(n)
# iterative algorithm
O1_iter <- function(n) {
  S <- 0
  for (i in 1:n) {
    S <- S + 2*i-1
  }
  return(S)
}

# recursive algorithm
O1_rec <- function(n) {
  if (n > 1) {
    S = 2*n-1 + O1_rec(n-1)
  } else {
    S = 1
  }
}

# verify
n <- c(1, 10, 100)
(r_o1 <- data.frame(
  n = n,
  iter_result = sapply(n, O1_iter),
  rec_result = sapply(n, O1_rec),
  real_result = n^2
))
```

```
##     n iter_result rec_result real_result
```

```
## 1   1           1          1            1
## 2  10         100        100          100
## 3 100       10000      10000        10000
```

```
# S_2(n)
## iterative algorithm
S2_iter <- function(n) {
  S <- 0
  for (i in 1:n) {
    S <- S + i^2
  }
  return(S)
}
## recursive algorithm
S2_rec <- function(n) {
  if (n > 1) {
    S = n^2 + S2_rec(n-1)
    }
  else{
    S = 1
    }
  return(S)
  }

## verify
n <- c(1, 10, 100)
(r_s2 <- data.frame(
  n = n,
  iter_result = sapply(n, S2_iter),
  rec_result = sapply(n, S2_rec),
  real_result = (n*(n+1)*(2*n+1))/6
))
```

```
##     n iter_result rec_result real_result
## 1   1           1          1           1
## 2  10         385        385         385
## 3 100      338350     338350      338350
```

Please see the GitHub repository for Python, Julia, and Matlab code for question 3.

Problem 4

$$C(n,k) = C(n-1,k-1) + C(n-1,k)$$

Write a procedure in your favorite computer language that uses Equation (7) evaluates $C(n,k)$ using a recursive algorithm.

```
bino_coef <- function(n,k){
  if (k < n && k > 0) {
    C = choose(n-1, k-1) + choose(n-1, k)
  }
  else{
    if (k > n){
      C = 0
```

```
    }
    if (k == 0 || k == n){
      C = 1
    }
  }
  return(C)
}

# verify
n <- 180
k <- 23
(r_bino_coef <- data.frame(
  n = n, k = k,
  r1 = bino_coef(n, k),
  r2 = choose(n, k)
))
```

```
##     n  k           r1           r2
## 1 180 23 6.625138e+28 6.625138e+28
```

Write a procedure in your favorite computer language that takes an integer n as input and prints out the n-th row of Pascal's triangle.

```
Pt_nrow <- function(n) {
  row <- numeric(2*n-1)
  row[seq(1, 2*n-1, by=2)] <- sapply(0:(n-1), function(k) choose(n-1, k))

  # Print the row
  cat(row, "\n")
}

# verify:
Pt_nrow(9)
```

```
## 1 0 8 0 28 0 56 0 70 0 56 0 28 0 8 0 1
```

```
Pt_nrow(10)
```

```
## 1 0 9 0 36 0 84 0 126 0 126 0 84 0 36 0 9 0 1
```

Problem 5

$$S_n = \frac{a(1-r^n)}{1-r}$$

```
S_n <- function(a, n, r) {
  if (n > 1) {
    Sn = a*(r^(n-1)+S_n(a,n-1,r))
  }
  else {
    Sn = a
  }
  return(Sn)
}
```

Write unit tests to demonstrate that your function works. Your tests should include cases where $r < 0, r = 0, 0 < r < 1, r = 1, and r > 1$.

```
library(testthat)
a <- 1
n <- 10
# unit tests
test_that("Test S_n function with various r values", {
  # r < 0
  expect_equal(S_n(a, n, -1), a*(1-(-1)^n) / (1-(-1)))
  # r = 0
  expect_equal(S_n(a, n, 0), a)
  # 0 < r < 1
  expect_equal(S_n(a, n, 0.5), a*(1-(0.5)^n) / (1-0.5))
  # r = 1
  expect_equal(S_n(a, n, 1), a*n)
  # r > 1
  expect_equal(S_n(a, n, 1.5), a*(1-1.5^n) / (1-1.5))
})
```

```
## Test passed
```

For what values of r does the limit $\lim_{n\to\infty} S_n$ exist? What is the limiting value?

$$S_n = \frac{a(1-r^n)}{1-r}$$

When $|r| < 1$, the limit $\lim_{n\to\infty} S_n$ exist, the limiting value is $\frac{a}{1-r}$.
When $|r| \geq 1$, the limit $\lim_{n\to\infty} S_n$ doesn't exist.