hw1 solution

Statistical Computing, Jieun Shin

Autumn 2022

문제 1.

먼저 R의 함수를 사용해보자.

```
0.1 == (3-2.9)            # 정확히 성립하는가
```

```
## [1] FALSE
```

```
all.equal(0.1, (3-2.9))   # 근사적으로 성립하는가 (오차범위 내에서 비교)
```

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

그리고 파이썬의 함수를 사용해보자.

```
import math

0.1 == (3-2.9)            # 정확히 성립하는가
```

```
## False
```

```
math.isclose(0.1, 3-2.9)  # 근사적으로 성립하는가 (오차범위 내에서 비교)
```

```
## True
```

두 값이 정확히 정확히 성립하는지 판단하는 == 연산자를 사용했을 때는 FALSE, 근사적으로 비교하는 all.equal (파이썬에서는 math.isclose)을 사용했을 때는 TRUE라는 결과가 나왔다.

사용자가 실수값을 정확히 입력도 컴퓨터는 부동소수점 방식을 사용하기 때문에 변환 과정에서 약간의 오차가 발생한다. 따라서 약간의 차이를 인정하는 all.equal (혹은 math.isclose)를 사용하여 비교하는 것이 좋다.

문제 2.

```
# exp(-1) 기본 계산
exp(-1)
```

```
## [1] 0.3678794
```

```
# Rmpfr에서 -1을 120비트로 설정한 후 계산
library(Rmpfr)

x = mpfr(-1, precBits = 120)
exp(x)
```

```
## 1 'mpfr' number of precision  120   bits
## [1] 0.36787944117144232159552377016146086741
```

Rmpfr을 사용하면 더 정밀한 결과를 출력함을 확인할 수 있다.

문제 3.

먼저 함수를 정의한다.

```
# 재귀 프로그램
fiborecursive = function(i){
  if (i <= 2){
    value = 1
  } else{
    return(fiborecursive(i-1) + fiborecursive(i-2))
  }
}

# 반복 프로그램
fiboiterative = function(i){
  if (i <= 2){
    value = 1
  } else{
    value1 = 1
    value2 = 1
    for(j in 3:i){
      value = value1 + value2
      value1 = value2
      value2 = value
    }
  }
  return(value)
}


# 메모 프로그램
fibomemo = local({
  memo = c(1, 1, rep(NA, 100))
  f = function(x) {
  if(x == 0) return(0)
  if(x < 0) return(NA)
  if(x > length(memo))
  stop("'x' too big for implementation")
  if(!is.na(memo[x])) return(memo[x])
  ans = f(x-2) + f(x-1)
  memo[x] <<- ans
  ans
  }
})
```

R에서 system.time함수를 이용하여 실행속도를 비교해보자.

```
iter = c(10, 30, 50)
time = matrix(0, 3, 3)    # 기록할 곳 저장

t = 0
for(i in iter){
  t = t + 1
  time[t, 1] = system.time(fiborecursive(i))[3]
  time[t, 2] = system.time(fiboiterative(i))[3]
```

```
  time[t, 3] = system.time(fibomemo(i))[3]
}

colnames(time) = c("재귀", "반복", "메모")
rownames(time) = iter

time
```

```
##         재귀 반복 메모
## 10      0.00    0    0
## 30      0.73    0    0
## 50 10739.01    0    0
```

실행 시간을 보면 재귀 프로그램의 시간이 가장 오래 걸리는 것을 알 수 있다.

파이썬에서도 동일하게 실행하여도 재귀프로그램의 시간이 가장 오래 걸리는 것을 확인할 수 있다.

```
def fib(n):
  if n < 2:
    return n
  else:
    return fib(n-1) + fib(n-2)

def fib_loop(n):
  n1 = 0
  n2 = 1
  if n == 1:
    return 0
  if n == 2:
    return 1
  for i in range(2, n):
    sum = n1 + n2
    n1 = n2
    n2 = sum
  return n2

memo = {0:0, 1:1}

def fib_memo(n, memo):
  if n in memo:
    return memo[n]
  memo[n] = fib_memo(n-1) + fib_memo(n-2)
  return memo[n]
```