hw 1 solution

Statistical Computing, Jieun Shin

Autumn 2021

문제 1.

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

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

```
all.equal(0.1 , (3 - 2.9))    # 근사적으로 성립하는가(오차 범위내에서 비교)
```

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

두 값이 정확히 성립하는지 판단하는 == 연산자를 사용했을 때는 FALSE, 근사적으로 비교하는 all.equal을 사용했을 때는 TRUE 라는 결과가 나옴을 확인하였다.

이러한 차이는 컴퓨터 연산에서 실수를 표현할 때는 부동소수점 방식을 사용하기 때문에 나타난다. 따라서 소수 자릿수를 비교할 때는 ==을 이용하지 않고 두 값의 차이가 오차보다 작은지를 판단해야 한다.

문제 2.

```
library(Rmpfr)
```

```
## Warning: 패키지 'gmp'는 R 버전 4.1.2에서 작성되었습니다
```

```
x = mpfr(-20, precBits = 100)
exp(-20)
```

```
## [1] 2.061154e-09
```

```
exp(x)
```

```
## 1 'mpfr' number of precision  100   bits
## [1] 2.0611536224385578279659403801559e-9
```

Rmpfr을 사용한 결과가 더 정밀한 결과를 출력함을 확인할 수 있다.

문제 3.

```
# (1) 재귀 프로그램
fiborecursive = function(i){
  if(i == 0){return(0)}
  if (i <= 2){
    value = 1
    return(value)
  }
  else{
    return(fiborecursive(i-1) + fiborecursive(i-2))
```

```
  }
}

# (2) 반복 프로그램

fiboiterative = function(i){
  if(i == 0){return(0)}
  if (i <= 2){
    value = 1
    return(value)
  }
  else{
    value1 = 1
    value2 = 1
    for(i in 3:i){
      value = value1 + value2
      value1 = value2
      value2 = value
    }
  }
  return(value)
}

# (3) 메모기능
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
}
})


# 시간 측정
for(i in c(10, 20, 30)){
  cat("i" = i , "recursive = ", system.time(fiborecursive(i))[3],
      ", iterative = ", system.time(fiboiterative(i))[3],
      ", memo = ", system.time(fibomemo(i))[3], "\n")}
```

```
## 10 recursive =  0 , iterative =  0 , memo =  0
## 20 recursive =  0 , iterative =  0.02 , memo =  0
## 30 recursive =  0.92 , iterative =  0 , memo =  0
```

결과를 보면 재귀 프로그램의 시간이 가장 오래 걸리는 것을 확인할 수 있다.