

STAT 527 HW 5

Satoshi Ido (ID: 34788706)

20 February 2023

1. (10 points) Fibonacci sequence. If you put some coefficients in the recursion, like $r_{i+1} = ar_i + br_{i-1}$, where you choose the numbers a and b , then you will have an entirely new sequence. Write a function that takes a , b and the sequence length n as arguments. The function should return the entire sequence. Choose appropriate input arguments and plot the returned sequence.

```
r <- function(a, b, n) {  
  # start with i = 1 and i-1 = 0  
  i <- 0; r1 <- 0; r2 <- 1  
  while (i < n) {  
    print(r1)  
    r3 <- a * r2 + b * r1  
    # update values  
    i <- i + 1  
    r1 <- r2  
    r2 <- r3  
  }  
}  
# Choose a = 1, b = 1, n = 10  
r(1, 1, 10)
```

```
## [1] 0  
## [1] 1  
## [1] 1  
## [1] 2  
## [1] 3  
## [1] 5  
## [1] 8  
## [1] 13  
## [1] 21  
## [1] 34
```

2. (20 points) 0-1 knapsack problem. Given a set of items, each with a weight and a value, determine which items to include in the collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. Mathematically, given a set of n items numbered from 1 up to n , each with a weight w_i and a value v_i , along with a maximum weight capacity W (assumed to be an integer), the 0-1 knapsack problem aims to solve the following optimization problem:

```
wt <- c(10, 20, 30)  
vt <- c(60, 100, 120)  
w <- 50  
n <- length(wt)
```

```

M <- matrix(0, nrow = n + 1, ncol = w)
k <- function(x) {x + 1}

# function (wt = weight, vt = value, w = limit)
Knapsack <- function(wt, vt, w) {
  # to skip the first row, start with 2 to k(n)
  for (i in 2:k(n)) {
    for (j in 1:w) {
      # if the weight of new item is over j-th, use the value at {(i-1), j}
      if (j < wt[i - 1]) {
        M[i, j] <- M[i - 1, j]
      }
      # ifelse, comparing m[i-1,w] & m[i-1,w-wi]+v and pick the bigger one
      else {
        M[i, j] <- max(M[i - 1, j], M[i - 1, j - wt[i - 1]] + vt[i - 1])
      }
    }
  }
  # change the row names
  rownames(M) <- 0:3
  return(M)
}

# wt = {10,20,30}, vt = {60,100,120} and W = 50
M <- Knapsack(wt, vt, w)
M

```

```

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## 0      0      0      0      0      0      0      0      0      0      0      0      0      0
## 1      0      0      0      0      0      0      0      0      0      0      60      60      60
## 2      0      0      0      0      0      0      0      0      0      0      60      60      60
## 3      0      0      0      0      0      0      0      0      0      0      60      60      60
##      [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26] [,27]
## 0          0          0          0          0          0          0          0          0          0
## 1      60      60      60      60      60      60      60      60      60      60      60      60
## 2      60      60      60      60      60      60      100     100     100     100     100     100
## 3      60      60      60      60      60      60      100     100     100     100     100     100
##      [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40]
## 0          0          0          0          0          0          0          0          0          0
## 1      60      60      60      60      60      60      60      60      60      60      60      60
## 2     100     100     100     160     160     160     160     160     160     160     160     160
## 3     100     100     100     160     160     160     160     160     160     160     160     160
##      [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50]
## 0          0          0          0          0          0          0          0          0          0
## 1      60      60      60      60      60      60      60      60      60      60
## 2     160     160     160     160     160     160     160     160     160     160
## 3     180     180     180     180     180     180     180     180     180     180

```