Palindrome Game

Nishant Panchal

2023-01-05

```
#Declaring Global Variables
LG <- matrix(NA, nrow=8, ncol=8)
LG[1,] <- c("r", "l", "q", "s", "t", "z", "c", "a")
LG[2,] <- c("i", "v", "d", "z", "h", "l", "t", "p")
LG[3,] <- c("u", "r", "o", "y", "w", "c", "a", "c")
LG[4,] <- c("x", "r", "f", "n", "d", "p", "g", "v")
LG[5,] <- c("h", "j", "f", "f", "k", "h", "g", "m")
LG[6,] <- c("k", "y", "e", "x", "x", "g", "k", "i")
LG[7,] <- c("l", "q", "e", "q", "f", "u", "e", "b")
LG[8,] <- c("l", "s", "d", "h", "i", "k", "y", "n")

green_letter_pos <- list("6,2","7,3","2,6","3,7")
letter_collection <- c()
green_probability <- 0.95


# -------------- Code for Token/Tracker Movement Starts ------------- #

#Moving token from current position to next position
move_token <- function(currPos.row, currPos.col){
  #Checking if the token is at the edge
  endTok <- ifelse(currPos.row == 1 || currPos.row == 8 || currPos.col == 1 || currPos.col == 8 ,TRUE,F
  new_pos <- list()
  if (endTok){
    new_pos <- move_to_any()
  } else {
    new_pos <- move_to_adj(currPos.row, currPos.col)

    #Preventing the scenario of no movement
    if (new_pos$row == currPos.row && currPos.col == new_pos$col) {
      new_pos <- move_to_adj(currPos.row, currPos.col)
    }
  }
  return (new_pos)
}

#When the token lands on a white square that is not an end square
move_to_adj <- function(currPos.row, currPos.col){
  row_vec <- c(currPos.row-1, currPos.row, currPos.row+1)
  col_vec <- c(currPos.col-1, currPos.col, currPos.col+1)
  adj_row <- sample(row_vec, size=1)
  adj_col <- sample(col_vec, size=1)
```

```
  adj <- list("row" = adj_row,"col" = adj_col)
  return (adj)
}

#When the token lands on a white square that is an end square
move_to_any <- function(){
  row <- sample(1:8, size=1)
  col <- sample(1:8, size=1)
  any_pos <- list("row" = row,"col" = col)
  return (any_pos)
}

# -------------- Code for Token/Tracker Movement Ends ------------- #




# -------- Code for Token landing on Green Square Starts ---------- #

#Checking if square is green
is_square_green <- function(currPos.row, currPos.col){
  position_string <- paste(currPos.row, currPos.col, sep = ",")
  return (position_string %in% green_letter_pos)
}

#Function that decides what to trigger when token lands on green square
green_letter_update <- function(currPos.row, currPos.col){
  #Picking the number 1 or 2 with the given probability to trigger action
  chance_val <- sample(1:2, size = 1, prob=c(green_probability,1-green_probability))
  if(chance_val == 1) {
    replace_collection()
  } else {
    remove_copies_collection(LG[currPos.row, currPos.col])
  }
}

#If action to replace collection is triggered
replace_collection <- function(){
  letter_collection <<- c("f","f","h","k")
}

#If action to remove copies of letter on green square is triggered
remove_copies_collection <- function(letter){
  letter_collection <<- letter_collection[letter_collection != letter]
}

# -------- Code for Token landing on Green Square Ends ---------- #



# -------- Code for controlling the game Starts ------- #

#Low Level Strategy --> Add first 3 elements to your collection,
```

```
#Add 4th element when current letter matches 2nd, skip until we find it
#Add 5th element when current letter matches 1st, skip until we find it
low_level_strategy <- function(startToken.row, startToken.col){
  currToken.row <- startToken.row
  currToken.col <- startToken.col
  currLetter <- LG[startToken.row, startToken.col]
  move_counter <- 1
  letter_collection <<- c(currLetter)

  while(length(letter_collection) < 3) {
    new_pos <- move_token(currToken.row,currToken.col)
    currToken.row <- new_pos$row
    currToken.col <- new_pos$col
    currLetter <- LG[new_pos$row, new_pos$col]
    if(is_square_green(currToken.row,currToken.col)){
      green_letter_update(currToken.row,currToken.col)
    } else {
      letter_collection <<- append(letter_collection, currLetter)
    }
    move_counter <- move_counter+1
  }

  #Handling the case where the token lands on the green letter
  # and the collection changes to F,F,H,K
  if(length(letter_collection) == 4) {
    while(length(letter_collection) < 5){
      new_pos <- move_token(currToken.row,currToken.col)
      currToken.row <- new_pos$row
      currToken.col <- new_pos$col
      currLetter <- LG[new_pos$row, new_pos$col]
      if(is_square_green(currToken.row,currToken.col)){
        green_letter_update(currToken.row,currToken.col)
      } else {
        if (currLetter == "h"){
          letter_collection <<- c("f","h","k","h","f")
        }
        if (currLetter == "k"){
          letter_collection <<- c("f","k","h","k","f")
        }
      }
      move_counter <- move_counter+1
    }
  } else {
    while(currLetter != letter_collection[2] || length(letter_collection) < 4) {
      new_pos <- move_token(currToken.row,currToken.col)
      currToken.row <- new_pos$row
      currToken.col <- new_pos$col
      currLetter <- LG[new_pos$row, new_pos$col]
      if(currLetter == letter_collection[2]){
        letter_collection <<- append(letter_collection, currLetter)
      }
      move_counter <- move_counter+1
    }
```

```
    while(currLetter != letter_collection[1]) {
      new_pos <- move_token(currToken.row,currToken.col)
      currToken.row <- new_pos$row
      currToken.col <- new_pos$col
      currLetter <- LG[new_pos$row, new_pos$col]
      if(currLetter == letter_collection[1]){
        letter_collection <<- append(letter_collection, currLetter)
      }
      move_counter <- move_counter+1
    }
  }


  print(letter_collection)
  print(move_counter)
}
#Modified Strategy --> ,
#Add 4th element when current letter matches 2nd, skip until we find it
#Add 5th element when current letter matches 1st, skip until we find it

# -------- Code for controlling the game Ends ------- #


# -------- Play Game Here ------- #

#move_token(2,6)
low_level_strategy(2,6)
```

```
## [1] "l" "c" "g" "c" "l"
## [1] 11
```