```
counter_fn <- function() {

  c <- 1
  function() {
    cat("Optimal Alignment No.", c)
    c <<- c + 1
  }
}

tick <- counter_fn()

# function to detect optimum direction(s) from the current cell while backtracing
choose_directions <- function(updates, possible_sources)
{
  possible_source_values <- c("diagonal" = possible_sources[1], "left" = possible_sources[2],
"up" = possible_sources[3])
  sources <- which(updates == max(updates))
  paste(names(which(possible_source_values[sources] == max(possible_source_values[sources]))),
 collapse = ", ")
}

# function to compute the similarity matrix and the direction matrix
finding_scores <- function(first_sequence, second_sequence, match, mismatch, delete, insert)
{
  m <- nchar(first_sequence); n <- nchar(second_sequence)
  score_matrix <- direction_matrix <- matrix(nrow = (n + 1), ncol = (m + 1))
  colnames(score_matrix) <- c("*", substring(text = first_sequence, first = 1:m, last = 1:m))
  rownames(score_matrix) <- c("*", substring(text = second_sequence, first = 1:n, last = 1:n))

  score_matrix[1, 1] <- 0
  direction_matrix[1, 1] <- "none"
  for(counter in 2:(m + 1))
  {
    score_matrix[1, counter] <- (counter - 1) * delete
    direction_matrix[1, counter] <- "left"
  }
  for(counter in 2:(n + 1))
  {
    score_matrix[counter, 1] <- (counter - 1) * insert
    direction_matrix[counter, 1] <- "up"
  }
  for(row_counter in 2:(n + 1))
  {
    for(column_counter in 2:(m + 1))
    {
      change <- score_matrix[(row_counter - 1), (column_counter - 1)] + ifelse(colnames(score_m
atrix)[column_counter] == rownames(score_matrix)[row_counter], match, mismatch)
      deletion <- score_matrix[row_counter, (column_counter - 1)] + delete
      insertion <- score_matrix[(row_counter - 1), column_counter] + insert
      score_matrix[row_counter, column_counter] <- max(change, deletion, insertion)
      direction_matrix[row_counter, column_counter] <- choose_directions(updates = c(change, de
letion, insertion), possible_sources = c(score_matrix[(row_counter - 1), (column_counter - 1)],
 score_matrix[row_counter, (column_counter - 1)], score_matrix[(row_counter - 1), column_counte
r]))
    }
  }
  return(list("Optimum Scores" = score_matrix, "Optimal Directions" = direction_matrix))
```

```
}

# function to change coordinates of the optimum path(s) to aligned sequences
coordinates_to_alignment <- function(coordinates, Sequence)
{
  characters <- c("*", strsplit(x = Sequence, split = "")[[1]])
  for (counter in (length(coordinates)):2)
  {
    if (coordinates[counter] == coordinates[(counter - 1)])
    {
      coordinates[counter] <- 1
    }
  }
  alignment <- characters[coordinates]
  return(paste(alignment, collapse = ""))
}

# function to backtrace the optimal path(s) and print the alignment(s)
print_optimal_global_alignment <- function(direction_matrix, row_index, column_index, first_coo
rdinates, second_coordinates, first_sequence, second_sequence)
{
  first_coordinates <- c(column_index, first_coordinates)
  second_coordinates <- c(row_index, second_coordinates)
  if(!((row_index == 2) & (column_index == 2)))
  {
    temp <- strsplit(x = direction_matrix[row_index, column_index], split = ", ")[[1]]
    for(direction in temp)
    {
      if(direction == "diagonal")
      {
        column_index_updated <- (column_index - 1)
        row_index_updated <- (row_index - 1)
      } else if(direction == "left")
      {
        column_index_updated <- (column_index - 1)
        row_index_updated <- row_index
      } else if(direction == "up")
      {
        column_index_updated <- column_index
        row_index_updated <- (row_index - 1)
      }
      print_optimal_global_alignment(direction_matrix, row_index_updated, column_index_updated,
 first_coordinates, second_coordinates, first_sequence, second_sequence)
    }
  } else
  {
    tick()
    print(rbind(coordinates_to_alignment(coordinates = first_coordinates, Sequence = first_sequ
ence), coordinates_to_alignment(coordinates = second_coordinates, Sequence = second_sequence)))
    cat("\n")
  }
}

# main function to return optimal global alignments by Needleman Wunsch algorithm
global_alignment <- function(first_sequence, second_sequence, match, mismatch, delete, insert)
{
  scores <- finding_scores(first_sequence = first_sequence, second_sequence = second_sequence,
 match = match, mismatch = mismatch, delete = delete, insert = insert)
```

```
  print_optimal_global_alignment(direction_matrix = scores[[2]], row_index = (nchar(second_sequ
ence) + 1), column_index = (nchar(first_sequence) + 1), first_coordinates = numeric(), second_c
oordinates = numeric(), first_sequence = first_sequence, second_sequence = second_sequence)
}

# example
global_alignment("ACTGTTA", "ACGTAT", 2, -3, -2, -2)
#> Optimal Alignment No. 1     [,1]
#> [1,] "ACTGT*TA"
#> [2,] "AC*GTAT*"
#>
#> Optimal Alignment No. 2     [,1]
#> [1,] "ACTGTTA*"
#> [2,] "AC*GT*AT"
global_alignment("ACTGTTA", "ACGTAT", 2, -3, -2, -2)
#> Optimal Alignment No. 3     [,1]
#> [1,] "ACTGT*TA"
#> [2,] "AC*GTAT*"
#>
#> Optimal Alignment No. 4     [,1]
#> [1,] "ACTGTTA*"
#> [2,] "AC*GT*AT"
```

Created on 2018-08-20 by the reprex package (http://reprex.tidyverse.org) (v0.2.0).