Assignment 3

Hugh Jamieson/jamieson.65

In this assignment, we will design an algorithm that will give us the cheapest flights from source to destination. We will use real flight prices from kayak.com and develop our own DP algorithm that will give us a sequence of flights whose total cost is cheaper than the cheapest (one/multi-stop) flight on kayak.com

First, we will consider a smaller problem. Imagine that there are only 6 airports in the world and only 5 airlines.

NOTE: DO NOT CHANGE THE CODE HERE. ONLY FILL CODE IN FUNCTIONS WHERE IT IS ASKED.

```
airports <- c('BOM', 'NYC', 'DXB', 'LHR', 'FRA', 'DOH')
airlines <- c('AIR_INDIA', 'BRITISH_AIRWAYS', 'EMIRATES',
            'QATAR_AIRWAYS', 'LUFTHANSA')
```

Read data from csv files containing flight prices. Each csv is named after an airport. The prices in that csv correspond to prices for DIRECT FLIGHT, FROM that airport. The columns of the csv represent airline chosen and ROWS represent the DESTINATION

```
#setwd("/Users/jamiesoh/Development/osu/osu-mach-learn/module-3/assignment_3/")
setwd("/Users/hughj/Development/osu/osu-mach-learn/module-3/assignment_3/")

read_csv <- function(file_name) {
  temp <- read.csv(file_name)
  temp2 <- temp[,-1]
  rownames(temp2) <- temp$X
  temp2
}

BOM <- read_csv("BOM.csv")
NYC <- read_csv("NYC.csv")
DXB <- read_csv("DXB.csv")
LHR <- read_csv("LHR.csv")
FRA <- read_csv("FRA.csv")
DOH <- read_csv("DOH.csv")

price_matrix = list(BOM, NYC, DXB, LHR, FRA, DOH) # This is same order as airports
```

Q1: Write a function that returns the lowest cost of direct flight from BOM to NYC (3 points)

Fill the function below

```
lowest_cost_BOM_to_NYC_direct <- function() {
  # Write your code here
  index_of_from <- which(airports=="BOM")[1]
  index_of_to <- which(airports=="NYC")[1]
  min(price_matrix[[index_of_from]][index_of_to,])
}

(lowest_cost_BOM_to_NYC_direct())
```

```
## [1] 1300
```

Q2: Write a function that returns the lowest cost of direct flight from one airport to another (2 points)

Fill the function below

```
lowest_cost_direct_flight <- function(from, to) {
  # Write your code here
  # First get index of FROM airport to check which
  # data frame from price matrix to use
  # Since airports array and price_matrix has same order of airports
  index_of_from <- which(airports==from)[1]
  prices_from <- price_matrix[index_of_from][[1]]

  # Write your code here
  min(prices_from[which(airports==to)[1],])
}

(lowest_cost_direct_flight('BOM', 'NYC'))
```

```
## [1] 1300
```

Q3: Given an array of airports, write a function that outputs the lowest cost to travel from each airport in the array to any airport in the same array. The output should be an NxN matrix where N is length of array of airports. Note that diagonal elements will be 0 (5 points)

Fill the function below

```
lowest_cost_direct_flight_matrix <- function(airports) {
  # Write your code here
  num_ports <- length(airports)
  mx <- matrix(0, num_ports,num_ports)
  rownames(mx) <- airports
  colnames(mx) <- airports
  for (fr in airports){
    for (to in airports){
      mx[fr,to] = lowest_cost_direct_flight(fr,to)
    }
  }
  mx
}
options(scipen=999)
(lowest_cost_direct_flight_matrix(airports))
```

```
##     BOM  NYC       DXB LHR  FRA       DOH
## BOM   0 1300       198 598 1371       925
## NYC 849    0       861 390 2877      1176
## DXB 112 1128         0 725  586 149000000
## LHR 405  392       596   0  198       819
## FRA 975  723       590 206    0       558
## DOH 166 1222 149000000 715  616         0
```

Q4. Here comes the main question. Find the cheapest flight from any airport to any airport which may or maynot be direct flight. (6 points)

Fill the function below

```
lowest_cost_flight_matrix <- function(airports, max_layovers) {
  # Write your code here
  # create the matrix of all possible (fr,to) combinations:

  # cache the direct costs for later
  lcdfm <- lowest_cost_direct_flight_matrix(airports)

  # helper function to calculate the cost.  we keep the total cost for the path in accum.
  # args: fr = from
  #   to = destination
  #   vect = hops visited
  #   hops = number of hops left
  helper <- function(fr, to, vect, hops) {
    #cat(sprintf("helper: fr=%s, to=%s,vect=%s, hops=%d\n", fr, to, paste(vect), hops),'\n')
    # terminal condition: hops==0
    if (hops == 0) {
      accum <- lcdfm[fr, to]
    }
    else {
      # not a direct flight. calculate the costs of all possible previous layovers.
      # we try to employ bellman principal by calculating the tails least cost:
      layovers <-
        vect[!(vect %in% c(fr, to))]  # we should get a vector here

      #cat(paste(c("layovers= ", layovers)), '\n')

      if (length(layovers) < 1)
        stop  # This should never happen if our dim is right!

      # if layover==1, we couls have a-b-e, a-c-e, a-d-e.  we have to add the cost of the
      # tail to the cost of the rest of the flight, then take the minimum.
      min_layover <- 2 ^ 30   # prevent using 0 cost as minimum.
      for (layover in layovers) {
        hop_cost <-
          helper(fr, layover, vect[!(vect %in% layover)], hops - 1) +
          helper(layover, to, vect[!(vect %in% layover)], hops - 1)

        if (hop_cost < min_layover) {
          min_layover <- hop_cost
        }
      }
      accum <- min_layover
    }
    # if the direct flight is cheaper, just use it.
    if (accum < lcdfm[fr,to]) return(accum)
    else return( lcdfm[fr,to])
  }

  msize <- length(airports)
  mx <- matrix(0, msize, msize, dimnames = list(airports, airports))
  for (i in airports) {
    for (j in airports) {
      if (i == j) {
        mx[i, j] <- 0
      }
      else {
        mx[i, j] <- helper(i, j, airports, max_layovers)
      }
    }
  }
  mx
}
(lowest_cost_flight_matrix(c("NYC","BOM"), 0))
```

```
##      NYC BOM
## NYC    0 849
## BOM 1300   0
```

Now lets check the lowest prices when max_layover is 1 and compare them with max_layover = 0 (direct flights).

```
(lowest_cost_flight_matrix(airports,1))
```

```
##     BOM  NYC DXB LHR FRA  DOH
## BOM   0  990 198 598 784  925
## NYC 795    0 861 390 588 1176
## DXB 112 1117   0 710 586 1037
## LHR 405  392 596   0 198  756
## FRA 611  598 590 206   0  558
## DOH 166 1107 364 715 616    0
```

```
(lowest_cost_flight_matrix(airports,0))
```

```
##     BOM  NYC       DXB LHR  FRA       DOH
## BOM   0 1300       198 598 1371       925
## NYC 849    0       861 390 2877      1176
## DXB 112 1128         0 725  586 149000000
## LHR 405  392       596   0  198       819
## FRA 975  723       590 206    0       558
## DOH 166 1222 149000000 715  616         0
```

Lets directly print a dataframe of dollars saved by increasing max_layover. Note that the optimal flight could also be a direct flight.

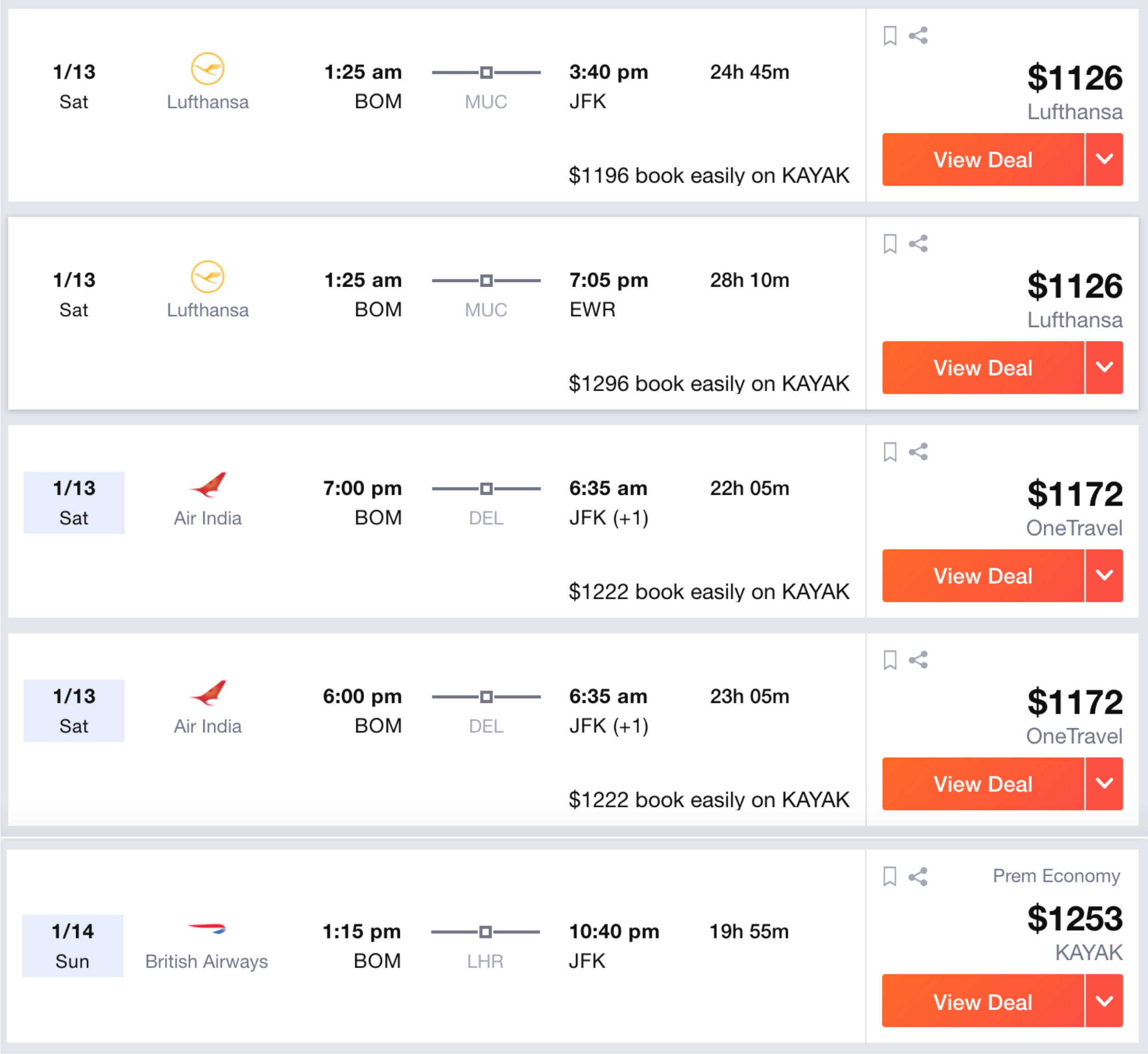
```
(lowest_cost_flight_matrix(airports,0)-lowest_cost_flight_matrix(airports,1))
```

```
##     BOM NYC       DXB LHR  FRA       DOH
## BOM   0 310         0   0  587         0
## NYC  54   0         0   0 2289         0
## DXB   0  11         0  15    0 148998963
## LHR   0   0         0   0    0        63
## FRA 364 125         0   0    0         0
## DOH   0 115 148999636   0    0         0
```

Note that the large numbers in dollars saved are because there was no direct flight but there were one stop flights, so technically you saved the cost of building and flying your own long range Boeing 747

We see that the lowest direct flight from BOM to NYC is $1300 (which is actual price on kayak.com) and one stop flight is $990. Lets see what kayak gives as the cheapest one stop flight of BOM to NYC for same dates.

We see that our algorithm gives much cheaper flights than online websites! Take BOM to LHR by BRITISH_AIRWAYS then take LHR to NYC by AIR_INDIA for a total of just $990.



1/13 Sat — Lufthansa — 1:25 am BOM — MUC — 3:40 pm JFK — 24h 45m — $1196 book easily on KAYAK — $1126 Lufthansa

1/13 Sat — Lufthansa — 1:25 am BOM — MUC — 7:05 pm EWR — 28h 10m — $1296 book easily on KAYAK — $1126 Lufthansa

1/13 Sat — Air India — 7:00 pm BOM — DEL — 6:35 am JFK (+1) — 22h 05m — $1222 book easily on KAYAK — $1172 OneTravel

1/13 Sat — Air India — 6:00 pm BOM — DEL — 6:35 am JFK (+1) — 23h 05m — $1222 book easily on KAYAK — $1172 OneTravel

1/14 Sun — British Airways — 1:15 pm BOM — LHR — 10:40 pm JFK — 19h 55m — Prem Economy — $1253 KAYAK

Try changing max_layovers to 2. You will see a significant increase in runtime! The technique of memoization solves this (Memoization was demonstrated in python tutorial).

Q5. (Bonus Question) Try to use memoization

```r
library("hashmap")
faster_lowest_cost_flight_matrix <- function(airports, max_layovers) {
  # cache the direct costs for later
  lcdfm <- lowest_cost_direct_flight_matrix(airports)
  # well use a hashmap to cache recently computed tails for memoization
  set.seed(13)
  tail_cache <- hashmap(c("xxx"), 0)

  makeFlightName <- function(vectr){
    paste(vectr, collapse = '')
  }
  makeFlightVector <- function(fr,to, v){
    c(fr,v,to)
  }

  # helper function to calculate the cost.  we keep the total cost for the path in accum.
  # args: fr = from
  #   to = destination
  #   vect = hops visited
  #   hops = number of hops left
  helper <- function(path, hops) {
#    cat(sprintf("helper=>path(%s), hops(%d)\n",makeFlightName(path),hops))
    # terminal condition: hops==0
    if (hops == 0) {
      accum <- lcdfm[path[1], path[-1]]
    }
    else {
      # not a direct flight. calculate the costs of all possible previous layovers.
      # we try to employ bellman principal by calculating the tails least cost:
      layovers <- airports[!(airports %in% path)]  # we should get a vector here

      if (length(layovers) < 1)
        stop  # This should never happen if our dim is right!

      # if layover==1, we couls have a-b-e, a-c-e, a-d-e.  we have to add the cost of the
      # tail to the cost of the rest of the flight, then take the minimum.
      min_layover <- 2 ^ 30   # prevent using 0 cost as minimum.
      for (layover in layovers) {
        # insert the layover into the current path
        flight_vector <- c(path[1], layover, path[2:length(path)])
        flight_path <- makeFlightName(flight_vector)
 #       cat(sprintf("flight_path=%s\n", flight_path))
        # if flight has already been costed, use it
        if (tail_cache$has_key(flight_path)) {
#          cat(sprintf("using cache %s\n", flight_path))
          accum <- tail_cache[[flight_path]]
        }else {
          head_cost <- helper(flight_vector[1:2], hops - 1)
          tail_cost <- helper(flight_vector[2:length(flight_vector)], hops - 1)
          hop_cost <- head_cost+tail_cost

          if (hop_cost < min_layover) {
            min_layover <- hop_cost
          }
          # add to cache
          tail_cache$insert(flight_path, min_layover)
        }
      }
      accum <- min_layover
    }
    # if the direct flight is cheaper, just use it.
    if (accum < lcdfm[path[1],path[length(path)]]) return(accum)
    else return( lcdfm[path[1],path[length(path)]])
  }

  msize <- length(airports)
  mx <- matrix(0, msize, msize, dimnames = list(airports, airports))
  for (i in airports) {
    for (j in airports) {
      if (i == j) {
        mx[i, j] <- 0
      }
      else {
        mx[i, j] <- helper(c(i, j), max_layovers)
      }
    }
  }
  mx
}
(faster_lowest_cost_flight_matrix(airports, 5))
```

```
##     BOM  NYC       DXB LHR  FRA       DOH
## BOM   0 1300       198 598 1371       925
## NYC 795    0       861 390 2877      1176
## DXB 112 1128         0 725  586 149000000
## LHR 405  392       596   0  198       819
## FRA 975  723       590 206    0       558
## DOH 166 1222 149000000 715  616         0
```

Q6. (Bonus Question) What will happen if you try to increase number of states? Hint: Read curse of dimensionality in Dynamic Programming

```
faster_lowest_cost_flight_matrix <- function(airports, max_layovers) {

}
```

Now build your own website that offers cheapest flight tickets for patient customers that are willing to wait for their requests!