Airline flights cheaper than Kayak

In this project, 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 dynamic programming algorithm that will give us a sequence of flights whose total cost is cheaper than the cheapest (one or 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.

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

Data from csv files containing flight prices will be read into RStudio. 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 airlines chosen and ROWS represent the DESTINATION

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

BOM <- read_csv("/Users/brian/Desktop/BOM.csv")
NYC <- read_csv("/Users/brian/Desktop/NYC.csv")
DXB <- read_csv("/Users/brian/Desktop/DXB.csv")
LHR <- read_csv("/Users/brian/Desktop/LHR.csv")
FRA <- read_csv("/Users/brian/Desktop/FRA.csv")
DOH <- read_csv("/Users/brian/Desktop/DOH.csv")

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

We will begin by writing a function that returns the lowest cost of direct flight from BOM to NYC.

Fill the function below

```
lowest_cost_BOM_to_NYC_direct <- function() {
 min(BOM['NYC',])

}

(lowest_cost_BOM_to_NYC_direct())
```

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

We will now write a function that returns the lowest cost of direct flight from one airport to another. We will get the index of the FROM airport to check which data frame from price matrix to use since airports array and price_matrix has same order of airports.

```
lowest_cost_direct_flight <- function(from, to) {

  index_of_from <- which(airports==from)[1]
  prices_from <- price_matrix[index_of_from][[1]]
  min(prices_from[to,])

}

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

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

We will write a function that outputs the lowest cost to travel from each airport in the array to any airport in the same array, given an array of airports. The output should be an NxN matrix where N is length of the array of airports. The diagonal elements should be 0 within the matrix.

```
lowest_cost_direct_flight_matrix <- function(airports) {
  N <- length(airports)
lowest_cost_matrix <- data.frame(matrix(nrow=N, ncol=N))
colnames(lowest_cost_matrix) <- airports
rownames(lowest_cost_matrix) <- airports
  for (from in airports){
    for (to in airports){
      lowest_cost_matrix[from,to] <- lowest_cost_direct_flight(from, to)
      }
    }
  lowest_cost_matrix

}

(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
```

Now, we must find the cheapest flight from any airport to any airport which may or maynot be direct flight.

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

   N <- length(airports)
   lowest_cost_matrix <- data.frame(matrix(nrow=N, ncol=N))
   colnames(lowest_cost_matrix) <- airports
   rownames(lowest_cost_matrix) <- airports

   direct_flight_matrix <- lowest_cost_direct_flight_matrix(airports)

  if (max_layovers==0){
    return(lowest_cost_direct_flight_matrix(airports))
  }

  for (final_destination in airports){
    for (current_airport in airports){

      optimal_price_matrix_from_next_airport <-
        lowest_cost_flight_matrix(airports, max_layovers-1)

      min_cost_to_reach_next_airports <-
        direct_flight_matrix[current_airport,]

      minimum_cost_from_current_to_final <-
        min(min_cost_to_reach_next_airports +
              optimal_price_matrix_from_next_airport[,final_destination])

      lowest_cost_matrix[current_airport,final_destination] <-
        minimum_cost_from_current_to_final

      }
  }

  lowest_cost_matrix
}
```

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
```

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

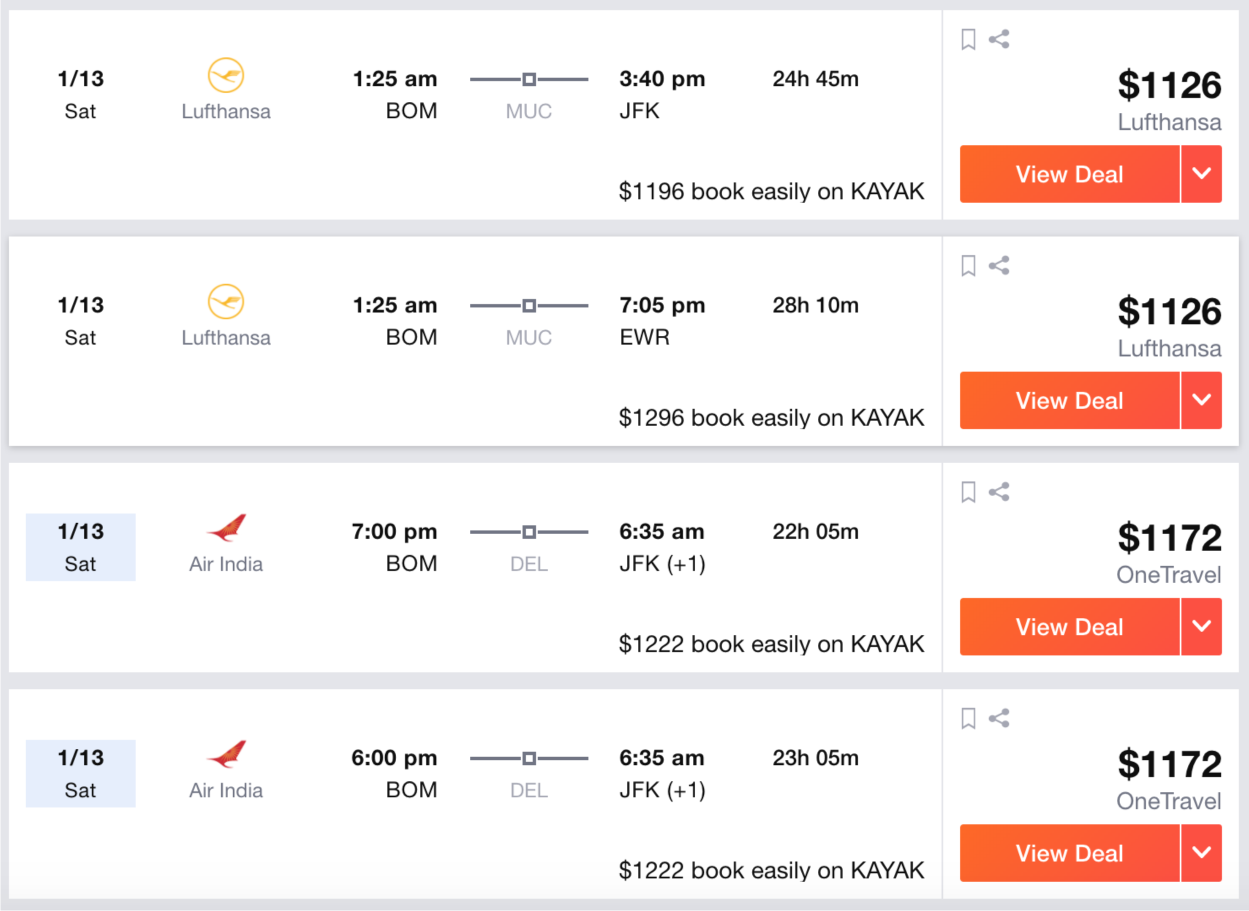
```
(lowest_cost_flight_matrix(airports,0)-lowest_cost_flight_matrix(airports,1))
```

```
##      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 we 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. Let's see what kayak gives as the cheapest one stop flight from 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.

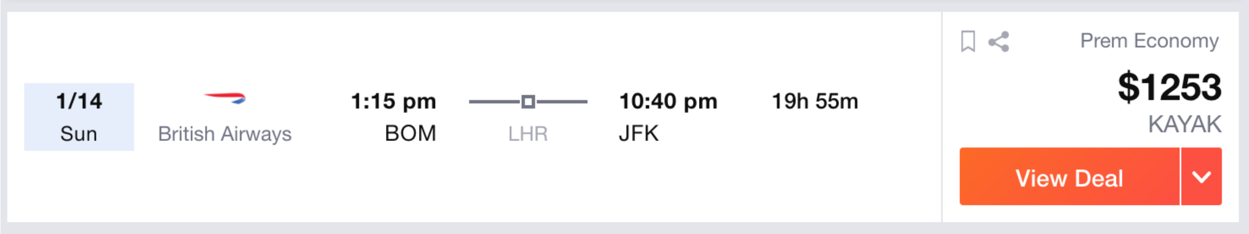


1/13 Sat
Lufthansa
1:25 am BOM — MUC — 3:40 pm JFK
24h 45m
$1196 book easily on KAYAK
$1126
Lufthansa

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

Max_layovers will be changed to 2. There will be a significant increase in runtime! The technique of memoization solves this.

```
# This code will be slow
(lowest_cost_flight_matrix(airports,2))
```

```
##      BOM  NYC DXB LHR FRA  DOH
## BOM    0  990 198 598 784  925
## NYC 795    0 861 390 588 1146
## DXB 112 1102   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
```

Memoization will be employed for faster run times

```
faster_lowest_cost_flight_matrix <- function(airports, max_layovers) {
  N <- length(airports)
    lowest_cost_matrix <- data.frame(matrix(nrow=N, ncol=N))
    memoized_lowest_cost_matrix <- list()
    colnames(lowest_cost_matrix) <- airports
    rownames(lowest_cost_matrix) <- airports

    direct_flight_matrix <- lowest_cost_direct_flight_matrix(airports)

    memoized_lowest_cost_matrix[[1]] =
      lowest_cost_direct_flight_matrix(airports)

    for (layover in 1:max_layovers+1){
      optimal_price_matrix_from_next_airport <-
      memoized_lowest_cost_matrix[[layover-1]]
        for (final_destination in airports){
          for (current_airport in airports){
              min_cost_to_reach_next_airports <-
                direct_flight_matrix[current_airport,]
                minimum_cost_from_current_to_final <- min(min_cost_to_reach_next_airports + optimal_p

        lowest_cost_matrix[current_airport,final_destination] <- minimum_cost_from_current_to_final

        }
    }
  memoized_lowest_cost_matrix[[layover]] = lowest_cost_matrix
    }

  memoized_lowest_cost_matrix[[max_layovers+1]]
}
```

```
(faster_lowest_cost_flight_matrix(airports,2))
```

```
##     BOM  NYC DXB LHR FRA  DOH
## BOM   0  990 198 598 784  925
## NYC 795    0 861 390 588 1146
## DXB 112 1102   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
```

Now, a website can be built that offers cheapest flight tickets for patient customers that are willing to wait for their requests!