IPS Instituto Politécnico de Setúbal

VOOS DOMÉSTICOS AMERICANOS

Algoritmos e Tipos Abstratos de Dados

Nome: Tomás Ramos Nº 202100310
David Ganhão Nº 202100314
Eric Silva Nº 202100798
André Caetano Nº 202100270
Turma: 1ºL_EI-01/ 1ºL_EI-02

IPS Instituto Politécnico de Setúbal

Índice

IPS Instituto Politécnico de Setúbal

ATDs Utilizados

Neste projeto foi utilizado o *ADT List* com uma estrutura de um *ArrayList*, para guardar os voos,porque operação *listGet* é usada com muita frequência para aceder aos voos. Como também foi usado *ADT Map* com uma estrutura de *LinkedList,* para guardar os aeroportos, sendo indiferente escolher entre *ArrayList* e *LinkedList* pois a operação de pesquisa e as restantes têm complexidade O(n), no entanto queremos a ordem não é importante e queremos garantir que ao criarmos o Map não queremos pares repetidos.

Algoritmos e Complexidades

```
Algorithm SHOWF
input:  flights - list of flights
        airport - string
BEGIN

IF flights = null THEN RETURN END IF

IF flights.isEmpty()  THEN RETURN END IF

newList <- getFlights(flights, airport) // flights of the airport O(n)

size <- newList.size()
arrFlight  <- flights.toArray     // array of newList O(n)

timeInfoFlightArrival(arrFlight,0,size)  // O(n)
END
O(n) varia com o tamanho de flights
```

Figura 1 - Função SHOWF

```
Algorithm LISTAR
input: flights -  list of Flights
        airlines - array of Airlines
IF flights = null THEN RETURN END IF

IF flights.isEmpty() THEN RETURN END IF

flightSize <- flights.size()

airlineSize <- airlines.size()

print LIST OF AIRLINES:

FOR i<-0 TO airlineSize DO
    FOR j<-0 TO flightSize DO

        IF strcmp(airlines[i].iatacode, flight[j].airline) = 0 THEN
            print Airline airline[I].name
            break;
        END IF

    END FOR
END FOR
END
O(n2) varia com o tamanho de flight e de airlines
```

Figura 2 - Função LISTAR

```
Algorithm LISTAP
input: flights -  list of Flights
        airports - map of Airports

IF flights = null or airports = null THEN RETURN END IF

IF flights.isEmpty() or airports.isEmpty() THEN RETURN END IF

flightSize <- flights.size()

airportSize <- airports.size()

keys <- airports.keys()    //array of keys

print LIST OF AIRPORTS:

FOR i<-0 TO airportSize DO
    airport <- aiports.getValue(keys[i])

    FOR j<-0 TO flightSize DO
        flight <- flights[j]

        IF strcmp(flight.originAirport, key[i].code) = 0  or strcmp(flight.destinationAirport, key[i].code) = 0 THEN
            print airport.iatacode + airport.airport + airport.city  + airport.state
            break;
        END IF

    END FOR
END FOR
END
O(n2) varia com o tamanho de flights e de airports
```

Figura 3 – Função LISTAP

```
Algorithm ONTIME
input: flights -  list of Flights
        airlines - array of Airports
print — ON TIME MENU ————
print Insert Tolerance (between 0 and 30):
read number

IF number < 0 or number > 30 THEN
    print NUMBER INSERTED INVALID
    return;
END IF

print   Airline      OT_Departures      OT_Arrivals

FOR i<-0 TO airline.size() DO

    print airlines[I].iatacode + onTimeDeparture(flights, airlines[I],number) + onTimeArrival(flights, airlines[I],
number)

    END FOR
print END
END
O(n) varia com o tamanho de flights
```

Figura 4 – Função ONTIME

```
Algorithm AIRPORT_S
input: airports - map of the airports

condition <- true
choice <- -1
size <- airports.size
airportsTemp <- airports.toArray()
WHILE condition DO
    print      AIRPORTS_S Menu
    print  1. Sort by City Ascending
    print  2. Sort by City Descending
    print  3. Sort by Latitude from N to S
    print  4. Sort by Longitude from E to W
    print  5. Return
    read choice
    SWITCH
            CASE 1  :  airportsOrderedCity(airports, size, true,airportsTemp)
                       printAirportsInfo(airportsTemp,0,size)
                       break;
            CASE 2  :  airportsOrderedCity(airports, size, false,airportsTemp)
                       printAirportsInfo(airportsTemp,0,size)
                       break;
            CASE 3  :  airportsOrderedLatitude(airports, size,airportsTemp)
                       printAirportsInfo(airportsTemp,0,size)
                       break;
            CASE 4  :  airportsOrderedLongitude(airports, size,airportsTemp)
                       printAirportsInfo(airportsTemp,0,size)
                       break;
            CASE 5  :  condition <- false
                       break;
            DEFAULT: print Unknown command
                       break;
    END SWITCH
END WHILE
END          O(n) varia com o tamanho de airports
```

Figura 5 – Função AIRPORT_S

IPS Instituto Politécnico de Setúbal

Conclusão

Neste projeto, ao todo, os alunos conseguiram a alcançar os desafios propostos. No princípio, o grupo teve alguns problemas em relação ao trabalho proposto, contudo esses problemas foram resolvidos através do estudo e várias pesquisas na internet.

Relativamente aos fatores limitantes a função TCP não consegue considerar as horas de partida e chegada, todavia consegue encontrar o trajeto mais rápido.

Em conclusão este projeto ajudou a construir um maior espírito de equipa, ajudando assim numa maior fluidez na realização de trabalhos futuros.