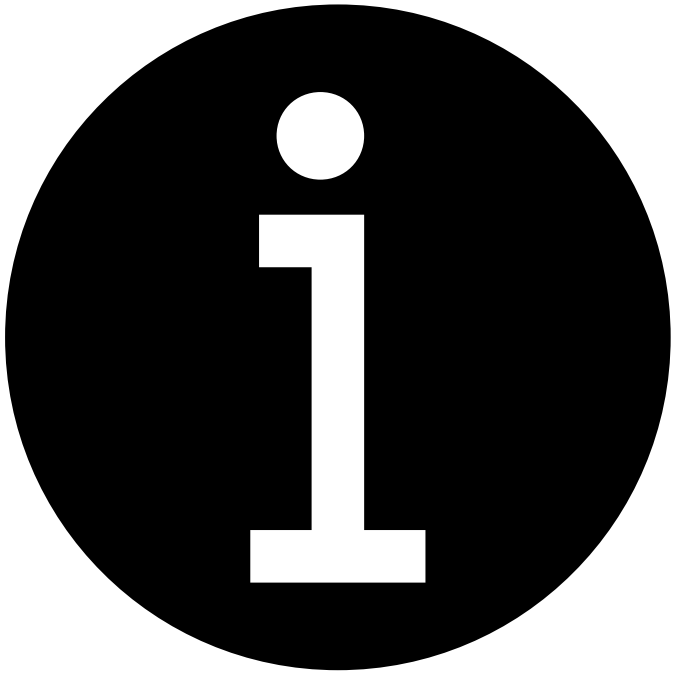


Trabalho AED2223_T2_G64

Trabalho realizado por Tomás Xavier e João Miguel Lourenço

Leitura do Dataset

A leitura dos ficheiros do dataset foi bastante simples.
Primeiramente, antes de tudo, lemos o ficheiro dos airports, pois o número de aeroportos no ficheiro será o número de nodes no nosso grafo, portanto foi necessário ler esse ficheiro em primeiro lugar.
As 3 funções utilizadas para a leitura dos dados dos ficheiros têm algoritmos bastante semelhantes, mas o que difere a função de leitura dos aeroportos e as outras funções é o valor de retorno.
O valor de retorno da função getAirports() é uma queue comprometida por todos os aeroportos fornecidos no ficheiro, que será depois utilizada, após a criação do grafo, para o preenchimento do vetor nodes e, consequentemente, uma unordered_map que mapeia o código de cada aeroporto à sua posição no vetor, para a fácil e rápida obtenção da sua informação.

Algoritmo de leitura

Para a leitura dos dados dos ficheiros da pasta dataset utilizamos sstreams e fstreams.

O algoritmo consiste em ler o ficheiro linha a linha e, por linha, sempre que encontrar uma vírgula, corresponder o valor lido ao seu significado no contexto deste projeto.

Por exemplo, durante a leitura linha a linha do ficheiro flights.csv, o conteúdo antes da primeira vírgula é o aeroporto origem, depois da primeira vírgula e antes da segunda é o aeroporto destino e após a segunda vírgula é a companhia aérea dessa viagem.

Depois disso, os dados lidos são inseridos num mapa que irá corresponder, por exemplo, no caso do airlines.csv, o código da Airline a um objeto Airline com toda a informação sobre essa Airline.

Origem/destino (aeroporto)

Para calcular o caminho com o menor número de voos entre 2 aeroportos utilizamos o método getShortestFilteredPath(). Este método é uma adaptação do algoritmo da BFS lecionado na cadeira de AED, mas com algumas modificações. Uma dessas modificações é o set permittedAirlines, que é passado como parâmetro para o método.

Com o set permittedAirlines conseguimos não só calcular o caminho com o menor número de voos, como também conseguimos filtrar as Airlines.

No caso do user não querer filtrar por Airlines, ou seja, está disposto a utilizar qualquer companhia, o metódo getShortestPath() será chamado, sem qualquer necessidade de set passado o set das Airlines.

Caso o user quiser escolher Airlines específicas, então essas Airlines serão adicionadas ao set permittedAirlines e depois passado como parâmetro para o getShortestFilteredPath().

Origem/destino (cidade e região)

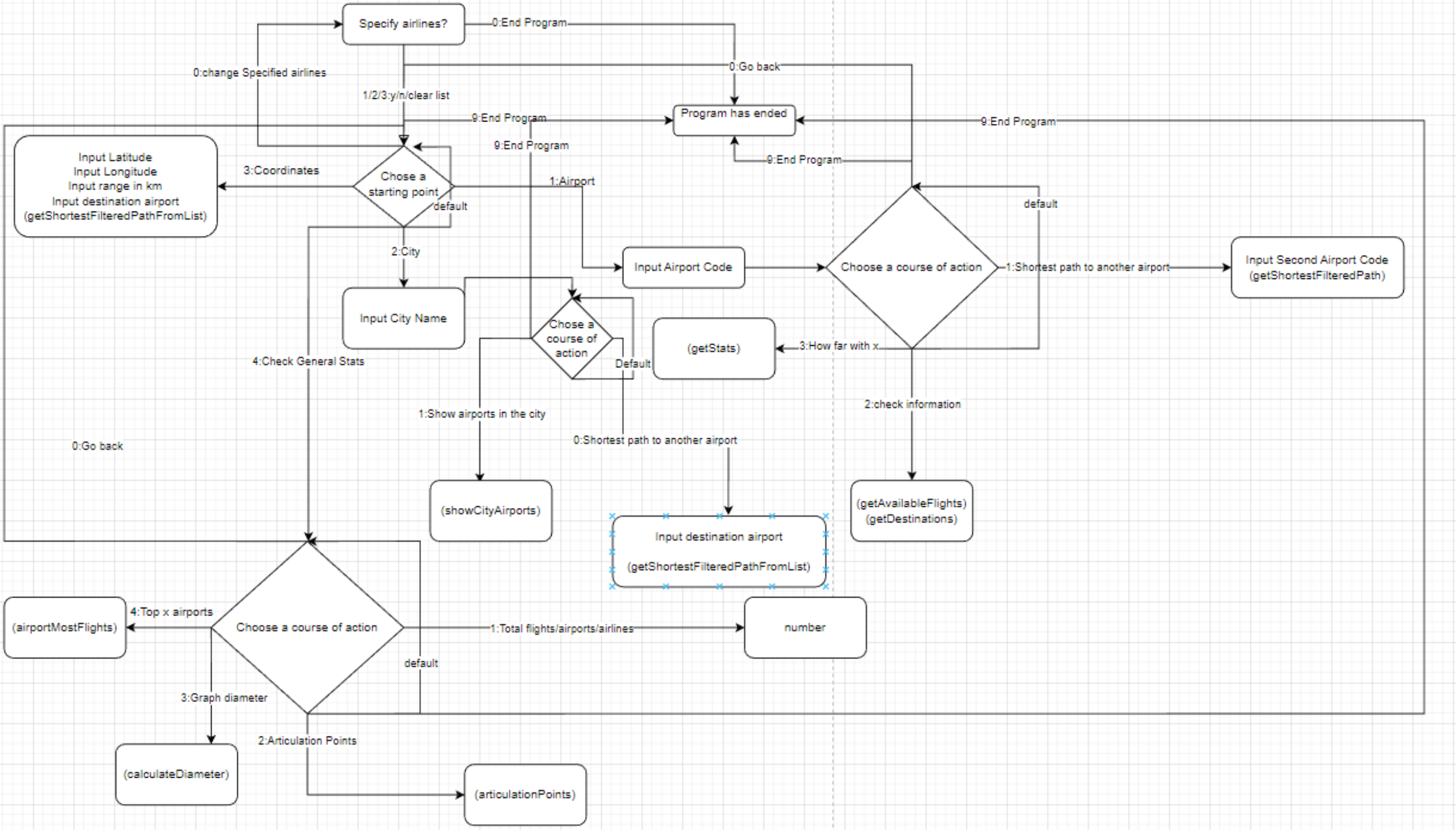
Tal como na implementação só com 1 aeroporto de origem e 1 de destino, tanto na escolha de cidade ou região (a.k.a coordenadas e raio em kilometros) é possível filtrar por companhias aéreas, funcionando da mesma forma - input vazio = método que não filtra é invocado; input não vazio = set com as Airlines inseridas e é chamado o método que filtra as Airlines enquanto percorre o grafo.

Para calcular o caminho com o menor número de voos a partir de uma cidade, foi, primeiramente, necessário criar um unordered_map que mapeia o nome de uma cidade a um vetor com todos os aeroportos.

Quanto à região, foi necessário criar funções de ajuda para ver quais aeroportos, caso haja algum, estão na área com centro nas coordenadas inseridas e raio também dado como input.

Após estas preparações, cada função chama o método getShortestPathFromList()/getShortestFilteredPathFromList(),dependendo se o user quer filtrar ou não as companhias; uma pequena variação do método anterior, que, ao contrário da outra passa o index do aeroporto no vetor nodes como parâmetro, passa uma lista com os indexes dos vários aeroportos na zona desejada (tanto cidade quanto região).


© Nations Online Project
EUROPE
EUROPEAN UNION
EU Member States
EU New Members since 2004
EU New Member 2013
EU Candidates
Possible EU Candidates 2022
EFTA Member States
Edge (represents a flight from an airport to another) contains a list of airlines that offer this flight in particular
Node 1 (airport)
Node 2 (airport)
Node 3 (airport)
Node 4 (airport)
Node 5 (airport)
Node 6 (airport)
Node 7 (airport)
Node 8 (airport)
ICELAND
NORGE NORWAY
SVERIGE SWEDEN
SUOMI FINLAND
РОССИЯ RUSSIA
EESTI ESTONIA
LATVIJA LATVIA
LIETUVA LITHUANIA
БЕЛАРУСЬ BELARUS
ÉIRE IRELAND
UNITED KINGDOM
DANMARK DENMARK
NETHERLANDS
DEUTSCHLAND GERMANY
POLSKA POLAND
ČESKÁ REPUBLIKA CZECH REPUBLIC CZECHIA
SLOVAKIA
УКРАЇНА UKRAINE
FRANCE
Österreich Austria
HUNGARY MAGYARORSZÁG
ROMÂNIA ROMANIA
Switzerland
SLOVENIA
CROATIA
SRBIJA SERBIA
БЪЛГАРИЯ BULGARIA
MONTENEGRO
ITALIA
ESPAÑA SPAIN
ΕΛΛΆΔΑ GREECE
TÜRKIYE TURKEY
ASIA
AFRICA
MALTA
CYPRUS
Atlantic Ocean
North Sea
Norwegian Sea
Baltic Sea
Mediterranean Sea
Black Sea
Aegean Sea
Ionian Sea
Tyrrhenian Sea
Adriatic Sea
250 Km
250 Mi.


Specify airlines?
0:End Program
0:change Specified airlines
0:Go back
1/2/3:y/n/clear list
Program has ended
9:End Program
9:End Program
9:End Program
9:End Program
Input Latitude
Input Longitude
Input range in km
Input destination airport
(getShortestFilteredPathFromList)
3:Coordinates
Chose a starting point
default
1:Airport
2:City
Input Airport Code
Choose a course of action
default
1:Shortest path to another airport
Input Second Airport Code
(getShortestFilteredPath)
Input City Name
Chose a course of action
Default
(getStats)
3:How far with x
4:Check General Stats
1:Show airports in the city
0:Shortest path to another airport
2:check information
0:Go back
(showCityAirports)
Input destination airport
(getShortestFilteredPathFromList)
(getAvailableFlights)
(getDestinations)
(airportMostFlights)
4:Top x airports
Choose a course of action
1:Total flights/airports/airlines
number
default
3:Graph diameter
2:Articulation Points
(calculateDiameter)
(articulationPoints)

Interface

O menu foi feito de forma a tentar minimizar ao máximo prompts repetidos, ou seja, por exemplo, ao invés de estar sempre a perguntar ao user após ele escolher ou aeroporto ou cidade ou região se quer filtrar os voos por companhias aéreas, nós decidimos perguntar no início se o user quer filtrar os seus voos, e, a qualquer altura, caso o user decida que quer adicionar mais uma companhia ou quer deixar de filtrar por companhia, ele tem essa opção, que é bastante mais prático do que ter que inserir todas as companhias aéreas desejadas sempre que se quer ver um caminho entre dois aeroportos.

Destaque de funcionalidade

No início do desenvolvimento deste projeto, quando ainda só calculavamos o caminho mais curto entre 2 aeroportos diretos, sem cidades nem regiões, a função demorava muito tempo a rodar, chegamos a ter que esperar 1 minuto para que a função rodasse e, às vezes, se fossem inseridos 2 aeroportos bastante longes um do outro então o programa não conseguia acabar.
Nós conseguimos reduzir o tempo de execução para que agora seja quase instantâneo e estamos relativamente orgulhosos com isso.

Dificuldades encontradas

Este projeto foi mais tranquilo do que o anterior, não só tivemos mais tempo disponível para trabalhar, como também, em termos de complexidade, é mais acessível. Mesmo assim, houve dificuldades. O algoritmo para calcular os aeroportos, cidades ou países que são atingíveis usando um máximo de Y voos não é muito eficiente, se Y for 3 o programa demora uns 20 segundos a rodar e se Y for maior que 4 parece que não acaba. Também tivemos uma approach bastante naive para o algoritmo do diâmetro do grafo e, por causa disso, ele demora alguns segundos a calcular.