Relatório Projeto Final - Teoria dos Grafos

Lucas Eduardo Rosa de Freitas[1] **Yuri Andreas May Henrique**[1]

[1]Universidade do Estado de Santa Catarina (UDESC)
Centro de Ciências Tecnológicas – Caixa Postal 15.064 – Joinville – SC – Brasil

lucas.freitas@edu.udesc.br

m.yuri@outlook.com

1. Introdução

Este é um trabalho de conclusão da matéria Teoria dos Grafos, do curso de ciência da computação, na qual tem como objetivo testar e explorar o conhecimento dos alunos. Neste trabalho, foi testado se os alunos aprenderam todos os principais conceitos e algoritmos abordados em sala de aula. A proposta desse projeto, era de resolver o problema dos torcedores do Coringa, um grupo de 150 pessoas que queriam ir assistir ao jogo do time em Natal, partindo de São Paulo. O grande problema se resumia em esquematizar a rota na qual eles fariam para chegar ao destino com o menor custo possível. Para isso, foi nos passadas todas as informações com o quadro de voos da empresa e as restrições exigidas.

Este artigo está divido em um total de 3 capítulos, onde o capítulo 2 trata do desenvolvimento completo do código e o capítulo 3 trata das dificuldades que tivemos para desenvolver esse projeto .

2. Desenvolvimento

De acordo com o plano de voo, mostrado na figura 1, sabemos todas as variáveis necessárias para a resolução do problema. Assim, utilizamos a ferramenta Graphviz para podermos visualizar o grafo de forma mais didática, as informações do trecho nas arestas.

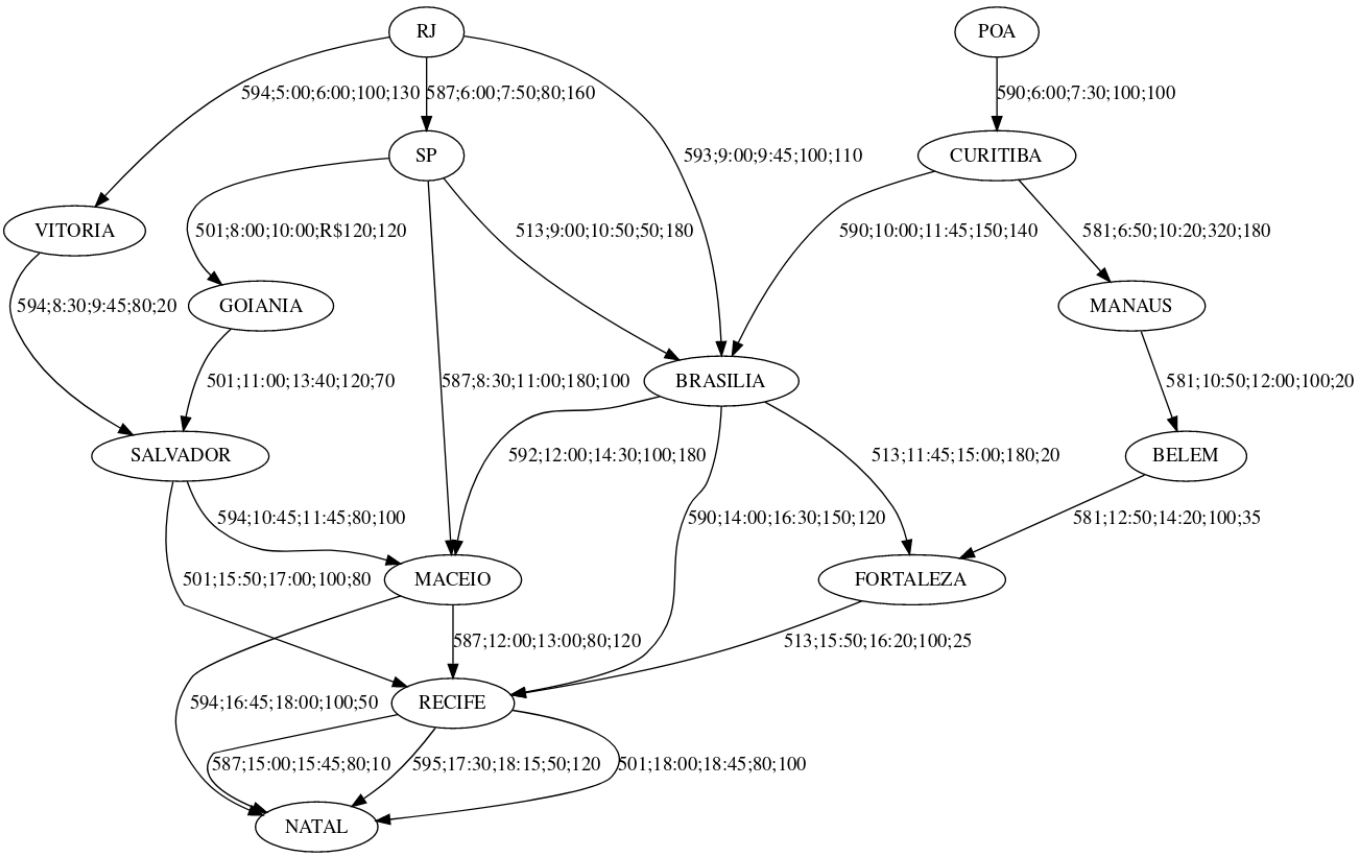

Figura 1. Grafo do plano de voo

O primeiro passo para resolvermos o problema, foi criar duas estruturas, na qual uma seria os trechos, nessa estrutura está contida todas as informações que devem ser guardadas, são elas: número de voo, hora de chegada, hora de partida, origem, destino, preço e vagas. A variável inteira vagas, é baseada nas reservas, como no quadro de voos nos informa que tem por exemplo 120 reservas, logo tem apenas 80 vagas, achamos que dessa forma ficaria mais simplificado o desenrolar do problema.

A outra estrutura, chamamos de voos, nela está contido um numero de voo e um trecho, variável do tipo trechos, assim temos que cada voo tem o seu ”código ”e um trecho.

Após nossas duas estruturas estarem bem definidas, começamos a parte de ir resolvendo o problema com algumas funções. A primeira função que foi implementada foi chamada de menorCaminho, com um nome bem intuitivo, ela é chamada recursivamente, na função do *Dijkstra*, pegando o menor caminho.

A nossa ideia para resolver o problema, foi de utilizar o algoritmo de *Dijkstra*, concebido em 1959, esse algoritmo soluciona o problema do caminho mais curto num grafo dirigido (que é o nosso caso) e com grafo não dirigido. Pensamos na utilização dele, pois o objetivo principal é minimizar a despesa total de deslocamento do grupo, utilizando o preço da passagem o algoritmo de *Dijkstra* nós dá o caminho mais ”barato”. A implementação do algoritmo foi realizada sem modificações, no caso isso representa que as restrições serão tratadas em outro momento.

A ideia desenvolvida, foi de utilizar uma matriz de adjacência e a matriz ser baseada no preço da passagem, assim o algoritmo de *Dijkstra*, já iria percorrer pelo menor. Pegando essa aresta de menor peso, tem-se a verificação das restrições, que são elas: chegar a Natal no máximo as 19:00H, não possuir mais que cinco trechos entre cidades, tempo de conexão entre as cidades não pode ser inferior a 1:00H, cada voo tem capacidade para 200 passageiros, com isso temos que cuidar com a quantidade de vagas que temos disponíveis.

Na função main, ocorre por primeiro a inicialização e em seguida a adição de novos trechos de forma manual.

A inserção de novos trechos, se dá pela seguinte forma, primeiramente é declarado uma variável t do tipo trechos, com tamanho de 24, que seria todo o plano de voo. Após isso, para cada posição, ou seja do 0 ao 23, adicionamos as informações:
t[0].numero_voo = 501;
t[0].horaChegada= 1000;
t[0].horaPartida= 800;
t[0].origem = ”SAO”;
t[0].destino = ”GOI”;
t[0].preco=120;
t[0].vagas=80;
Com isso, um novo trecho é criado.

Apos ser realizada toda a inserção de dados, eu preencho a matriz de adjacência tornando como base dela o preço. Como já mencionado, pensamos dessa maneira, pois assim o algoritmo já iniciaria pegando o menor preço, bastando verificar as restrições.

3. Dificuldades

No decorrer do projeto, tivemos algumas dificuldades. Uma das principais foi em como tratar as restrições, em qual momento e de que forma seria possível fazer essas verificações.

Outra dificuldade que tivemos, foi com o apontamento de valores dos vértices instanciados para fazer a aplicação das restrições.